Universidade do Minho
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2023/2024

Calendário de Eventos na cidade de Braga

Diogo Coelho da Silva (a100092), Pedro Miguel Ramôa Oliveira (a97686)

Janeiro, 2024

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Calendário de Eventos na Cidade de Braga

Diogo Coelho da Silva (a100092), Pedro Miguel Ramôa Oliveira (a97686)

Janeiro, 2024

Resumo

Este relatório foi realizado no âmbito da unidade curricular de Bases de Dados, que tinha como objetivo principal criar um sistema de gestão de base de dados com o maior foco a incidir na análise, no desenho, modelação, arquitetura e implementação de um sistema deste tipo.
O projeto que nos vamos basear é um projeto fictício que foi criado e imaginado derivando de muitas situações em que muitas empresas ficaram depois da pandemia que assolou o país.

O projeto em questão surgiu como resposta ao declínio das atividades festivas e culturais em Braga, provocada pela pandemia global de 2019, bem como pelo encerramento de empresas voltadas para a gestão de eventos na cidade. Com o propósito de reavivar o espírito festivo de Braga e apoiar a sua economia local, um grupo de 3 jovens liderado por Martim Santos, entregou-nos a tarefa e o desafio de criar um Sistema de Bases de Dados para a Gestão de Eventos em Braga.

Os objetivos primordiais deste SBDG consistem em promover eventos, festividades e atrações turísticas na cidade, bem como em facilitar o acesso dos turistas a informação detalhadas sobre esses eventos. A administração da cidade também é beneficiada, pois este sistema auxilia na gestão eficaz de eventos e coleta dados essenciais sobre a participação de turistas e a população de eventos e atrações.

Numa primeira fase deste projeto, foram estabelecidos os requisitos que a nossa base de dados deveria comportar. Após a recolha dos requisitos e uma análise cuidada aos mesmos, foi altura de começar a realizar um modelo conceptual, em que foram estabelecidas entidades, derivadas dos requisitos anteriormente recolhidos, e os relacionamentos existentes entre si. Foi feita no fim destes dois passos, a validação do nosso modelo conceptual, e assim foi obtido o aval para avançar para a próxima etapa da criação do SGBD.

Após uma primeira avaliação do trabalho prático, foi retomado o mesmo com a conversão do modelo conceptual, anteriormente realizado, num modelo lógico. Neste passo foram revistos todos os relacionamentos e a criação das respetivas tabelas. De seguida, o mesmo modelo lógico foi traduzido para o sistema de gestão de bases de dados que foi escolhido, no nosso caso, o *MySQL*. Nesta fase foram feitas interrogações à nossa base de dados para verificar que

se verificavam os resultados pretendidos e foi feito também um plano de segurança e recuperação de dados.

Área de Aplicação: Desenho, arquitetura, implementação e manipulação de um Sistema de Base de Dados.

Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Recolha de Requisitos, Análise de Requisitos, Modelo Conceptual, Entidades, Relacionamentos, Atributos, *MySQL*, Segurança e Recuperação de dados, Sistema de Gestão de base de dados, Modelo Lógico, Implementação Física, Interrogações.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

A cidade em questão é um importante destino turístico localizado numa região de beleza natural exuberante. Com uma população de cerca de aproximadamente 193 mil habitantes, esta cidade atrai visitantes de todo o mundo devido à sua rica cultura, património histórico e eventos variados. A sua importância turística reflete-se na contribuição significativa para a economia local, através da indústria do turismo, que abrange setores como a hotelaria, alimentação, lazer e comércio.

A cidade de Braga há muito é reconhecida pela sua juventude e pelo empenho constante em promover uma série de eventos culturais e festivos ao longo do ano. Contudo, esse ímpeto festivo foi interrompido em 2019, quando o mundo foi assolado por uma pandemia. Durante quase dois anos, a cidade viu-se privada de eventos, o que culminou num desinteresse por parte das entidades organizadoras em retomar tais atividades. Além disso, diversas empresas incumbidas da gestão de eventos e festividades na cidade acabaram por encerrar as suas operações, em virtude das dificuldades enfrentadas durante esse período desafiador.

Diante deste quadro, um trio de jovens decidiu que esta maré não podia continuar, e assim, uniram-se nos esforços de reavivar o espírito festivo de Braga. Martim Santos, nascido em 15 de março de 1996, natural da própria cidade, foi o principal impulsionador da iniciativa. Motivado pelo falecimento do seu avô, Albertino Faria, que figurava entre os pioneiros na organização destas festividades na cidade, comprometeu-se a reerguer o fervor festivo da localidade. Para tal desígnio, convidou os seus amigos de longa data, Francisco Ferreira, de 28 anos, e Ana Rodrigues, de 27 anos, a se juntarem a ele nesta aventura. Nasceu assim a empresa “Bracara Eventos”.

1.2. Fundamentação

A justificação para a implementação deste sistema de base de dados é baseada numa série de desafios enfrentados nos últimos anos, que afetaram a capacidade de organizar eventos e festas na cidade de forma eficaz. Primeiramente, a interrupção desses eventos levou ao declínio gradual do hábito de organizá-los, tornando a retoma da normalidade uma tarefa mais complicada. Além disso, ao longo do tempo, grupos e empresas envolvidos na organização

destes eventos enfrentaram dificuldades para acompanhar os avanços tecnológicos que poderiam proporcionar maior eficiência e facilidade na gestão dos mesmos.

O aumento da população na cidade, juntamente com o crescente número de turistas que a visitam, aumentou a complexidade de organizar e divulgar os eventos de maneira satisfatória. Consequentemente, a tarefa de coordenar e comunicar eficazmente essas atividades tornou-se praticamente impossível.

Para abordar essas situações e recuperar o tempo perdido, o grupo decidiu realizar a implementação de um sistema de gestão bases de dados. Este sistema permitirá que os mesmos armazenem antecipadamente as informações necessárias, possibilitando um maior controlo sobre cada evento festivo e turístico na cidade.

1.3. Objetivos

O grupo, comandado pela liderança, experiência e conhecimento adquiridos pelo Martim ao longo da sua vida com o seu falecido avô, estabeleceu um conjunto de objetivos que pretendem alcançar com o SGBD que estão prestes a criar, dos quais se destacam:

1. Reestruturar e aprimorar o modelo de gestão desses eventos, visando melhorar substancialmente a sua capacidade de organização e gerenciamento.
2. Aprimorar a eficácia e a otimização da gestão de cada evento.
3. Aumentar a eficácia na divulgação e promoção desses eventos na cidade.
4. Facilitar a pesquisa de eventos com base em diversos critérios, tais como data, tipo de evento, localização, etc…
5. Aumentar receitas a partir de bilhetes vendidos.
6. Registar os eventos de forma detalhada
7. Fornecer informações relevantes sobre cada evento, como programação, preço, disponibilidade de ingressos, etc…
8. Oferecer uma interface *user friendly* para os utilizadores terem acesso calendário de eventos e interagirem com as informações disponíveis.

Estes objetivos visam aprimorar a gestão e a visibilidade dos eventos da cidade, proporcionando um serviço mais eficiente e acessível tanto para os residentes, como para os turistas e para todos os intervenientes na realização destes eventos.

1.4. Viabilidade

A viabilidade deste projeto é prometedora, considerando o potencial económico que a cidade oferece. O grupo está confiante de que ao implementar um sistema mais eficiente e moderno para a gestão de eventos na cidade de Braga, poderá alcançar os seguintes benefícios:

1. Recuperar aproximadamente 50% das perdas decorrentes do período de pandemia e confinamento. Este objetivo não apenas cobrirá os custos associados à criação deste Sistema de Gestão de Bases de Dados, mas também gerará lucro adicional.
2. Manter um acompanhamento constante do número de bilhetes vendidos para cada evento pago, bem como o lucro associado a cada um deles.
3. Monitorizar o fluxo de pessoas em cada evento, o que contribuirá para uma gestão mais eficiente e segura.
4. Identificar e compreender as tendências e preferências dos residentes da cidade e dos turistas em relação aos eventos, permitindo a adaptação das atividades à demanda e ao gosto do público.

A viabilidade deste projeto é prometedora, considerando o potencial económico que a cidade oferece.

1.5. Recursos

A equipa de desenvolvimento deste SGBD delineou e apresentou, juntamente com o grupo, um conjunto de recursos que pensaram serem necessários para a implementação da mesma. Estes recursos foram divididos em dois sub-recursos: recursos materiais e recursos humanos.

Recursos Humanos:

- O grupo Bracara Eventos, constituído por Martim Santos, Francisco Ferreira e Ana Rodrigues
- Um Engenheiro de Bases de Dados
- Um Arquiteto de Bases de Dados
- Um especialista em segurança de redes
- Um Administrador de Bases de Dados
- 1 Equipa de 2 analistas
- Desenvolvedor de *Software*
- Artistas
- Staff
- Cidadãos
- Câmara Municipal de Braga

Recursos Materiais:

- *Hardware*:
    - 1 Servidor

    - 6 Postos de Venda de Bilhetes
    - 2 Posto de Informação

- *Software*:
    - SGBD
    - Aplicação para vendas de bilhetes e informação sobre os eventos
    - BrModelo
    - *SQL Workbench*

1.6. Equipa de Trabalho

A equipa de trabalho vai ser divida em três categorias: Pessoal Interno, Pessoal Externo e Outros. O pessoal interno a este projeto vão ser os intervenientes responsáveis pela idealização e criação do mesmo. O pessoal externo vão ser todas as pessoas que vão ser contratadas para a realização deste projeto. Dito isto podemos distribuir a equipa da seguinte forma:

Pessoal Interno:

- Martim Santos
- Francisco Ferreira
- Ana Rodrigues

Pessoal Externo:

- Arquiteto de Bases de Dados
- Engenheiro de Bases de Dados
- Administrador de Bases de Dados
- Desenvolvedor *Software*
- Equipa de *Marketing*
- Analistas:
    - Diogo Coelho da Silva
    - Pedro Miguel Ramôa Oliveira

Outros:

- Cidadãos e turistas selecionados
- Camara Municipal de Braga

Podemos, agora, explicitar um pouco sobre o que cada categoria da equipa de trabalho vai trabalhar sobre.

O pessoal interno vai ser responsável pela gestão e organização dos eventos, como por exemplo contratar artistas, fazer o atendimento a novos clientes, validação de serviços, entre outros tópicos relacionados com esta área.

O pessoal externo vai ser responsável por implementar os pedidos do pessoal interno. O Arquiteto de Bases de Dados vai analisar os requisitos recolhidos anteriormente e esboçar um modelo conceptual do SBGB, que mais tarde irá ser utilizado pelo Engenheiro de Bases de Dados para converter este esboço conceptual para um modelo físico. O Administrador de Bases de Dados vai realizar a manutenção regular necessária para um projeto desta dimensão. O desenvolvedor de software vai criar uma aplicação, para que os cidadãos e turistas possam muito resumidamente consultar informações sobre os eventos e também realizar um conjunto de operações, como por exemplo comprar bilhetes para eventos selecionados.

A adicionar, vai também ser contratada uma equipa de marketing de forma a divulgar e impulsionar o lançamento deste novo sistema.

A equipa de analista vai ser responsável pela recolha e organização de requerimentos.

A última categoria, “Outros”, vai incluir cidadãos e turistas selecionados para a realização de inquéritos de opinião e também a CMB, de forma a conseguir a aceder a documentos de informação arquivados, para a recolha informação que possa ser transformada em requisitos.

1.7. Plano de Execução

O plano de execução do trabalho seguirá um diagrama de GANTT detalhado, abrangendo as fases de definição do sistema, definição de requisitos e modelação concetual. O lançamento do sistema está previsto para o dia 1/1/2024.

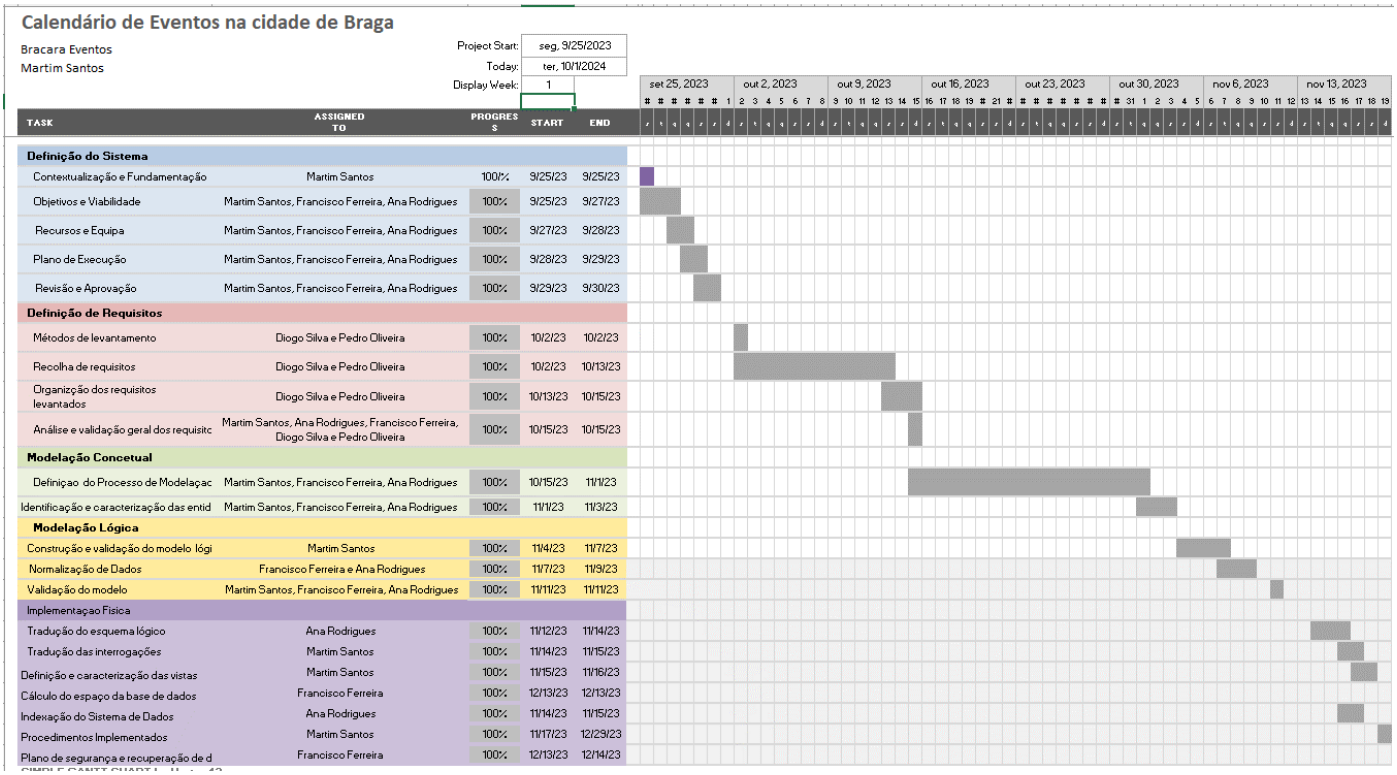


Figura 1 - Diagrama GANTT

Após ter previsto as datas necessárias para a implementação dos passos, equipa sentiu a necessidade de adicionar uma semana para a realização do modelo conceptual.

1.8. Revisão e Aprovação

Após o término dos trabalhos da primeira fase deste projeto, no dia 28/9/2023, foi convocada uma reunião com a empresa Bracara Eventos, sendo o propósito da mesma uma análise da definição do sistema, bem como a fundamentação, os objetivos e a viabilidade o mesmo. Compareceu também nesta reunião o Engenheiro de Bases de Dados, o Arquiteto de Bases de Dados, a equipa de analistas, o Administrador de Bases de Dados, todos estes contratados para a realização deste projeto. Em conjunto todos os pormenores foram revistos e validaram tudo o que tinha sido definido anteriormente, assim como também foi aprovado o plano de trabalhos, descrito no diagrama de GANTT produzido. O grupo decidiu, então avançar com o processo de desenvolvimento e implementação do SGBD.

2. Definição de Requisitos

2.1. Método de levantamento e de análise de requisitos adotado

Os requisitos foram recolhidos por meio de um processo abrangente. Esta coleta de requisitos resultou de uma combinação de entrevistas com *stackholders*-chave, ou por outras palavras, pessoas e organizações que podem ser afetadas por este projeto, pesquisas de campo e análises de documentos relacionados com eventos passados e relacionados com o mercado turísticos da cidade, estes últimos facultados pela CMB.

As entrevistas foram realizadas junto de organizadores de eventos locais, autoridades municipais e representantes da indústria do turismo nesta região, para tentar captar e perceber as suas necessidades e expectativas em relação ao SGBD.

As pesquisas de campo serviram para ter uma melhor compreensão das dinâmicas que os eventos têm e das preferências dos cidadãos e dos visitantes. Os documentos serviram para ter um *insight view* adicional.

Com o avanço dos trabalhos em relação ao último ponto, todos os requisitos que foram levantados, foram anotados num documento de recolha (documento de requisitos). Neste documento de requisitos, para cada requisito recolhido, os analistas registaram: a área da aplicação, quem fez o levantamento, quem forneceu o requisito, o texto de requisito, a data e hora do seu levantamento.

Número	Data e Hora	Descrição	Área	Fonte	Analista
1	2/10/2023 10:54	Todos os eventos que se vão realizar na cidade devem ser registados, com a atribuição de um número único.	Evento	Martim Santos	Diogo Silva
2.3	3/11/2023 17:34	Cada evento vai ter um ID do evento, ou seja, um número único para esse evento, um nome, uma pequena descrição, deverá designar se é pago ou não, se sim deverá incluir um preço, uma data e hora, uma localização onde o evento se irá realizar, deverá listar todas as atividades que possam ocorrer no evento, os artistas que no evento participem, e estará implícito qual foi o custo de realizar o mesmo.	Evento	Martim Santos	Diogo Silva
3.2	2/11/2023 11:04	Um evento poderá ser pago ou não.	Evento	Martim Santos	Pedro Oliveira
4.1	3/10/2023 13:58	A localização de um evento deverá ter a rua onde se realizará o evento, uma descrição do local e o código postal.	Evento	Martim Santos	Pedro Oliveira
5	2/10/2023 11:32	Um evento tem sempre uma atividade pelo menos, senão não é considerado um evento. Pode ter mais do que uma atividade.	Evento	Francisco Ferreira	Diogo Silva

6	4/10/2023 15:23	Uma atividade só pode ser atividade de um evento	Atividade	Francisco Ferreira	Pedro Oliveira
7	4/10/2023 15:34	Cada atividade vai ter um ID da atividade, um nome, uma pequena descrição, um preço, uma data e hora (horário), uma localização, e um artista, poderá ser paga ou não, e tem de estar implícito o custo de realizar uma atividade.	Atividade	Francisco Ferreira	Diogo Silva
8.2	15/11/2023 14:26	É possível ter um registo imediato, anexado aos eventos e às atividades do valor total de ganhos relativos ao evento ou atividade, da quantidade de bilhetes vendidos, bem como a quantidade de bilhetes ainda disponível para o evento ou atividade em questão.	Atividade	Francisco Ferreira	Diogo Silva
9	2/10/2023 11:39	Cada artista poderá ser individual ou não (ex.: banda, grupo…)	Artista	Ana Rodrigues	Pedro Oliveira
10	2/10/2023 11:45	A localização de uma atividade vai ter a rua onde se realizará a atividade, uma descrição do local e o código postal.	Atividade	Martim Santos	Diogo Silva
11	2/10/2023 11:47	Um artista (vamos considerar um artista como sendo um elemento individual ou um grupo), tem um número único de artista para o identificar, um nome, uma descrição e é possível aceder aos dados do seu agente, e ainda tem o custo associado a contratar esse artista.	Artista	Ana Rodrigues	Diogo Silva
12	2/10/2023 11:43	Um agente tem um ID de agente único, um nome, um email e o seu número de telefone.	Agente	Francisco Ferreira	Pedro Miguel
13	2/10/2023 14:15	É possível registar a afluência de cada evento na cidade quando o evento é pago.	Gestão	Martim Santos	Diogo Silva
14	7/10/2023 18:42	Quando o evento não é pago, vai ser impossível controlar a afluência de pessoas, pelo menos no início deste projeto, uma vez que não temos capacidade de ter os meios necessários para tal.	Gestão	Martim Santos	Diogo Silva
15	2/10/2023 14:18	É possível ver o total de bilhetes vendidos para cada evento pago.	Venda	Martim Santos	Pedro Oliveira
16.1	2/10/2023 14:20	Um evento pode ser grátis, mas ter atividades pagas.	Venda	Martim Santos	Diogo Silva
17	2/10/2023 14:21	Quando se paga o bilhete de um evento, todas as atividades que nele existem estão consideradas pagas.	Venda	Martim Santos	Diogo Silva
18	2/10/2023 14:25	É possível ver o total de bilhetes vendidos para cada atividade paga.	Venda	Ana Rodrigues	Pedro Oliveira
19	5/10/2023 11:06	Vai ser possível pesquisar cada evento por data, se é pago ou não, se for pago pelo preço de bilhete.	Evento	Martim Santos	Diogo Silva
20.3	5/10/2023 11:14	Vai ser possível verificar o total de receitas de cada evento, com o total de bilhetes vendidos.	Venda	Martim Santos	Diogo Silva
21	5/10/2023 11:17	Cada evento pago vai ter um número máximo de bilhetes para serem vendidos, ou seja, a capacidade de cada atividade	Venda	Ana Rodrigues	Pedro Oliveira
22	5/10/2023 14:53	O sistema deverá estar operacional durante 24 horas.	SGBD	Francisco Ferreira	Pedro Oliveira
23.5	5/10/2023 14:57	O sistema poderá ser acedido pelo grupo Bracara Eventos de forma parcial, e de forma total pelo gestor do sistema de bases de dados. O sistema poderá ser acedido pelos utilizadores para	SGBD	Francisco Ferreira	Diogo Silva

		apenas verem detalhes dos eventos e das atividades. O grupo apenas tem acesso a operações de consulta nas tabelas da base de dados.			
24	5/10/2023 16:30	Ao final de cada dia o sistema deverá realizar um relatório com os seguintes dados: o número total de bilhetes vendidos para cada evento, o valor faturado nesse dia com as vendas, o número de bilhetes comprados.	Gestão	Martim Santos	Diogo Silva
25	5/10/2023 16:34	A cada momento é possível ver a listagem de todos eventos na cidade ordenados pela data.	Gestão	Martim Santos	Pedro Oliveira
26	5/10/2023 16:40	A cada momento é possível filtrar quais são os eventos pagos ou não	Gestão	Martim Santos	Pedro Oliveira
27	5/10/2023 17:04	Os bilhetes vendidos deverão ter uma estrutura *default* para todo o tipo de eventos que sejam pagos.	Bilhete	Ana Rodrigues	Pedro Oliveira
28	5/10/2023 17:08	Todos os bilhetes vão ter um registo próprio com a data da venda do bilhete, para melhor controlo sobre a venda dos mesmos.	Bilhete	Ana Rodrigues	Pedro Oliveira
29	5/10/2023 17:10	Os bilhetes têm um número de bilhete, o nome do respetivo evento, o nome da respetiva atividade e um preço.	Bilhete	Ana Rodrigues	Pedro Oliveira
30	13/10/2023 10:53	Cada evento tem sempre um ou mais elemento de staff para ajudar na organização e manutenção do mesmo.	Staff	Ana Rodrigues	Pedro Oliveira
31	13/10/2023 10:55	Cada elemento do staff vai ter um identificador do staff, um nome, a função que cada elemento tem e a lista de números de telefone do elemento.	Staff	Ana Rodrigues	Pedro Oliveira
32	13/10/2023 12:55	Cada elemento do staff pode ajudar em mais do que um evento.	Staff	Martim Santos	Diogo Silva
33	13/10/2023 12:57	Um evento tem sempre pelo menos um elemento do staff.	Staff	Martim Santos	Diogo Silva
34	20/11/2023 11:11	O administrador tem de fazer backups diárias, em mais do que uma localização de modo a não perder dados.	SGBD	Ana Rodrigues	Diogo Silva
35	20/11/2023 11:15	É possível contar quantos eventos se realizaram num ano civil.	SGBD	Ana Rodrigues	Diogo Silva
36	20/11/2023 16:11	É possível obter o número de bilhetes vendidos para um evento e para as atividades em questão.	SGBD	Ana Rodrigues	Diogo Silva
37	20/11/2023 18:00	É possível comprar bilhetes para um evento, não sendo obrigatório, pois o evento pode ser gratuito.	Bilhete	Ana Rodrigues	Pedro Oliveira
38	20/11/2023 18:11	É possível comprar bilhetes para uma atividade, não sendo obrigatório, pois a atividade pode ser gratuita.	Bilhete	Ana Rodrigues	Diogo Silva
39	20/11/2023 18:12	Uma atividade pode ou não ter um artista associado, por exemplo se a atividade for uma prova de comida. Por outro lado, um artista pode participar em 1 ou mais atividades.	Atividade	Ana Rodrigues	Pedro Oliveira
40	21/11/2023 19:23	A cada momento é possível aceder a uma listagem dos eventos que se realizam num mês.	Gestão	Ana Rodrigues	Pedro Oliveira
41	21/11/2023 19:32	A cada momento é possível aceder a uma listagem dos 5 eventos que mais faturaram.	Gestão	Ana Rodrigues	Pedro Oliveira

Tabela 1 - Requisitos

2.2. Análise e Organização

Quando os analistas verificaram que a maioria dos processos operacionais em torno da organização de eventos já tinham sido estudados e os seus requisitos levantados, começaram a fazer a análise e a organização dos requisitos obtidos. Nesta segunda fase, os analistas iriam verificar a ocorrência de erros, inconsistências, redundâncias, entre outros problemas que podiam ter aparecido anteriormente. Estes requisitos recolhidos anteriormente foram analisados um a um, e, tendo em conta as vistas de utilização definidas, começaram a organizá-los de acordo com as três vertentes do trabalho de dados do futuro sistema, nomeadamente:

- Descrição: Para acolher os requisitos que referiam a criação de objetos na base de dados (tabelas, atributos, domínios, restrições, etc., …);
- Manipulação: Para incluir tudo aquilo que referisse o povoamento ou exploração de dados, quer fosse através de simples interrogações à base de dados (*queries)*, ou fosse através de procedimentos, funções, utilização na aplicação, etc…
- Controlo: Para saber como é que iriam gerenciar a base de dados e a sua utilização

Estes novos documentos têm uma estrutura semelhante ao documento de requisitos inicial, porem os requisitos têm uma numeração própria e o seu texto já foi revisto e analisado pelos analistas do sistema.

Número	Data e Hora	Descrição	Área	Fonte	Analista
RD1	02/10/2023 10:54	Todos os eventos que se vão realizar na cidade devem ser registados, com a atribuição de um número único.	Evento	Martim Santos	Diogo Silva
RD2.1	03/10/2023 13:56	Cada evento vai ter um ID do evento, ou seja, um número único para esse evento, um nome, uma pequena descrição, deverá designar se é pago ou não, se sim deverá incluir um preço, uma data e hora, uma localização onde o evento se irá realizar, deverá listar todas as atividades que possam ocorrer no evento, os artistas que no evento participem, e estará implícito qual foi o custo de realizar o mesmo.	Evento	Martim Santos	Diogo Silva
RD3	02/10/2023 10:59	A localização de um evento deverá ter a rua onde se realizará o evento, uma descrição do local e o código postal.	Evento	Francisco Ferreira	Diogo Silva
RD4	02/10/2023 11:04	Um evento tem sempre uma atividade pelo menos, senão não é considerado um evento. Pode ter mais do que uma atividade.	Evento	Martim Santos	Pedro Oliveira
RD5	02/10/2023 11:32	Uma atividade só pode ser atividade de um evento	Atividade	Martim Santos	Pedro Oliveira
RD5.1	03/10/2023 13:58	Cada atividade vai ter um ID da atividade, um nome, uma pequena descrição, um preço, uma data e hora (horário),	Atividade	Martim Santos	Pedro Oliveira

		uma localização, e um artista, poderá ser paga ou não, e tem de estar implícito o custo de realizar uma atividade.			
RD6	02/10/2023 11:32	É possível ter um registo imediato, anexado aos eventos e às atividades do valor total de ganhos relativos ao evento ou atividade, da quantidade de bilhetes vendidos, bem como a quantidade de bilhetes ainda disponível para o evento ou atividade em questão.	Gestão	Francisco Ferreira	Diogo Silva
RD7.2	04/10/2023 15:34	Cada artista poderá ser individual ou não (ex.: banda, grupo…).	Artista	Francisco Ferreira	Diogo Silva
RD8	02/10/2023 11:39	A localização de uma atividade vai ter a rua onde se realizará a atividade, uma descrição do local e o código postal.	Artista	Ana Rodrigues	Pedro Oliveira
RD9	02/10/2023 11:45	Um artista (vamos considerar um artista como sendo um elemento individual ou um grupo), tem um número único de artista para o identificar, um nome, uma descrição e é possível aceder aos dados do seu agente, e ainda tem o custo associado a contratar esse artista.	Artista	Martim Santos	Diogo Silva
RD10	02/10/2023 11:47	Um agente tem um ID de agente único, um nome, um email e o seu número de telefone.	Agente	Ana Rodrigues	Diogo Silva
RD11.1	02/10/2023 11:43	Um evento pode ser grátis, mas ter atividades pagas.	Evento	Francisco Ferreira	Pedro Oliveira
RD12	02/10/2023 14:20	Quando se paga o bilhete de um evento, todas as atividades que nele existem estão consideradas pagas.	Bilhete	Martim Santos	Diogo Silva
RD13	02/10/2023 14:21	Cada evento pago vai ter um número máximo de bilhetes para serem vendidos, ou seja, a capacidade de cada atividade	Venda	Martim Santos	Diogo Silva
RD14	02/10/2023 14:30	Os bilhetes vendidos deverão ter uma estrutura *default* para todo o tipo de eventos que sejam pagos.	Venda	Francisco Ferreira	Pedro Oliveira
RD15	02/10/2023 14:50	Todos os bilhetes vão ter um registo próprio com a data da venda do bilhete, para melhor controlo sobre a venda dos mesmos.	Bilhete	Martim Santos	Diogo Silva
RD16	02/10/2023 14:54	Os bilhetes têm um número de bilhete, o nome do respetivo evento, o nome da respetiva atividade e um preço.	Bilhete	Martim Santos	Diogo Silva
RD17	02/10/2023 14:55	Cada evento tem sempre um ou mais elemento de staff para ajudar na organização e manutenção do mesmo.	Staff	Francisco Ferreira	Diogo Silva
RD18.1	02/10/2023 15:24	Cada elemento do staff vai ter um identificador do staff, um nome, a função que cada elemento tem e a lista de números de telefone do elemento.	Staff	Francisco Ferreira	Diogo Silva
RD19	13/10/2023 12:55	Cada elemento do staff pode ajudar em mais do que um evento.	Staff	Martim Santos	Diogo Silva
RD20	13/10/2023 12:57	Um evento tem sempre pelo menos um elemento do staff.	Staff	Martim Santos	Diogo Silva
RD21	16/10/2023 13:00	É possível comprar bilhetes para um evento, não sendo obrigatório, pois o evento pode ser gratuito.	Venda	Francisco Ferreira	Pedro Oliveira
RD21	16/10/2023 13:00	É possível comprar bilhetes para uma atividade, não sendo obrigatório, pois a atividade pode ser gratuita.	Venda	Francisco Ferreira	Pedro Oliveira

RD22	17/10/2023 12:56	Uma atividade pode ou não ter um artista associado, por exemplo se a atividade for uma prova de comida. Por outro lado, um artista pode participar em 1 ou mais atividades.	Atividade	Francisco Ferreira	Pedro Oliveira

Tabela 2 - Requisitos Descrição

Número	Data e Hora	Descrição	Área	Fonte	Analista
RM1	02/10/2023 14:15	É possível registar a afluência de cada evento na cidade quando o evento é pago.	Gestão	Martim Santos	Diogo Silva
RM2	07/10/2023 18:42	Quando o evento não é pago, vai ser impossível controlar a afluência de pessoas, pelo menos no início deste projeto, uma vez que não temos capacidade de ter os meios necessários para tal.	Gestão	Martim Santos	Diogo Silva
RM3	02/10/2023 14:20	É possível ver o total de bilhetes vendidos para cada evento pago.	Venda	Martim Santos	Pedro Oliveira
RM4	02/10/2023 14:25	É possível ver o total de bilhetes vendidos para cada atividade paga.	Venda	Ana Rodrigues	Pedro Oliveira
RM5	02/10/2023 14:28	É possível ver o total de bilhetes vendidos para cada atividade paga.	Venda	Francisco Ferreira	Pedro Oliveira
RM6	03/10/2023 10:53	Vai ser possível pesquisar cada evento por data, se é pago ou não, se for pago pelo preço de bilhete.	Gestão	Martim Santos	Pedro Oliveira
RM7	03/10/2023 10:55	Vai ser possível verificar o total de receitas de cada evento, com o total de bilhetes vendidos.	Gestão	Ana Rodrigues	Pedro Oliveira
RM8	03/10/2023 10:56	Ao final de cada dia o sistema deverá realizar um relatório com os seguintes dados: o número total de bilhetes vendidos para cada evento, o valor faturado nesse dia com as vendas, o número de bilhetes comprados.	Gestão	Ana Rodrigues	Pedro Oliveira
RM9	03/10/2023 10:59	A cada momento é possível ver a listagem de todos eventos na cidade ordenados pela data.	Gestão	Martim Santos	Pedro Oliveira
RM10	05/10/2023 11:06	A cada momento é possível filtrar quais são os eventos pagos ou não	Evento	Martim Santos	Diogo Silva
RM11	05/10/2023 11:14	É possível contar quantos eventos se realizaram num ano civil.	Gestão	Martim Santos	Diogo Silva
RM12	05/10/2023 16:30	É possível obter o número de bilhetes vendidos para um evento e para as atividades em questão.	Gestão	Martim Santos	Diogo Silva
RM13	21/11/2023 19:23	A cada momento é possível aceder a uma listagem dos eventos que se realizam num mês.	Gestão	Ana Rodrigues	Pedro Oliveira
RM14	21/11/2023 19:32	A cada momento é possível aceder a uma listagem dos 5 eventos que mais faturaram.	Gestão	Ana Rodrigues	Pedro Oliveira

Tabela 3 - Requisitos Manipulação

Número	Data e Hora	Descrição	Área	Fonte	Analista
RC1	05/10/2023 14:53	O sistema deverá estar operacional durante 24 horas.	SGBD	Francisco Ferreira	Pedro Oliveira
RC2	05/10/2023 14:57	O sistema poderá ser acedido pelo grupo Bracara Eventos de forma, forma parcial, e de forma total pelo gestor do sistema de bases de dados. O sistema poderá ser acedido pelos utilizadores para apenas verem detalhes dos eventos e das atividades. O grupo apenas tem acesso a operações de consulta nas tabelas da base de dados	SGBD	Francisco Ferreira	Diogo Silva
RC3	05/10/2023 16:53	O administrador tem de fazer backups diárias, em mais do que uma localização de modo a não perder dados.	SGBD	Francisco Ferreira	Diogo Silva
RC4	05/11/2023 16:53	O administrador da base de dados tem permissão total para toda a base de dados.	SGBD	Francisco Ferreira	Diogo Silva

Tabela 4 - Requisitos Controlo

2.3. Análise e validação geral dos requisitos

Após todos os requisitos terem sido revistos pela equipa de analistas, realizou-se uma reunião com os restantes intervenientes do projeto para fazer a validação dos mesmos.

Quando se discutia sobre os requisitos em questão, especialmente os requisitos de controlo, surgiu uma grande dúvida. Como todos os presentes na reunião sabiam e entendiam o quão valiosos são os dados de uma pessoa nos dias de hoje, surgiu a necessidade de criar um requisito unicamente utilizado para garantir aos utilizadores que os seus dados estão sobre o seu controlo, ou seja, os mesmos poderiam acedê-los e eliminá-los a qualquer momento.

Outro dos requisitos que decidiram que era crucial implementar foi a realização de backups regulares, pois, num sistema desta dimensão, é impensável e quase proibido perder qualquer dado que seja.

Por último, foram discutidos detalhes relativos á implementação de um sistema de subscrição, que permite aos utilizadores terem regalias adicionais.

No final, e após se limarem alguns detalhes e novos requisitos propostos durante a revisão, deu-se a aprovação dos mesmos, e o aval para começar a próxima etapa do projeto.

3. Modelação Concetual

3.1. Apresentação da abordagem de modelação realizada

O processo da modelação conceptual é o primeiro passo, ou melhor, o primeiro esquema base de uma base de dados. O modelo conceptual que foi apresentado, é um esquema abstrato e de alto nível, representando as várias entidades envolvidas e descritas na definição dos requisitos, os relacionamentos entre as mesmas e os diversos atributos que caracterizam as entidades e os relacionamentos.

Para este processo decorrer de forma natural e sem qualquer tipo de surpresa pelo caminho, é importante assegurar que todos os requisitos que foram previamente levantados conseguem definir e atingir todos os objetivos propostos para este projeto. Com base neles identificamos as principais entidades principais da futura base de dados de projeto, bem como os seus relacionamentos e atributos.

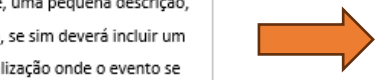
A criação do nosso modelo conceptual foi feita com recurso ao software “BrModelo”, utilizando para a implementação das entidades, relacionamentos e atributos, a notação de Peter Chen. O software vai nos permitir criar um Diagrama ER, que é um tipo de fluxograma que ilustra como entidades, por exemplo, pessoas, objetos ou conceitos, que se relacionam entre si dentro de um sistema. Portanto, o primeiro passo para criarmos o nosso diagrama, é identificar as entidades, e após isso determinar como é que as entidades estão relacionadas.

3.2. Identificação e caracterização das entidades

As entidades presentes no nosso modelo conceptual derivaram todas de requisitos de descrição recolhidos anteriormente. Foram identificadas seis entidades a partir dos mesmos. Para as seguintes entidades, vamos corresponder um requisito que lhe deu origem.

Entidade Evento:

RD2.1	03/10/2023 13:56	Cada evento vai ter um ID do evento, ou seja, um número único para esse evento, um nome, uma pequena descrição, deverá designar se é pago ou não, se sim deverá incluir um preço, uma data e hora, uma localização onde o evento se irá realizar, deverá listar todas as atividades que possam ocorrer no evento, os artistas que no evento participem, e estará implícito qual foi o custo de realizar o mesmo.



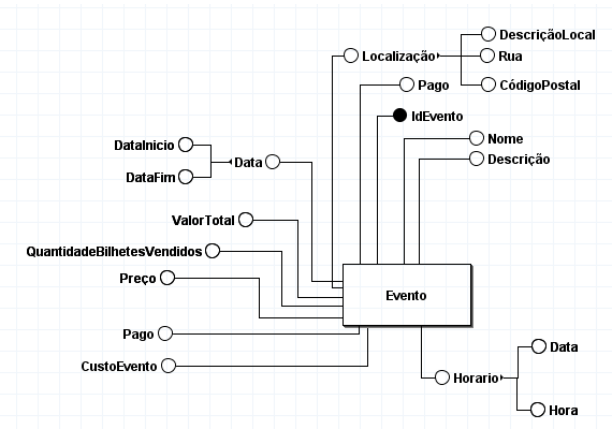


Figura 2- Conversão Requisito para Entidade "Evento"

Entidade Atividade:

RD5.1	03/10/2023 13:58	Cada atividade vai ter um ID da atividade, um nome, uma pequena descrição, um preço, uma data e hora (horário), uma localização, e um artista, poderá ser paga ou não, e tem de estar implícito o custo de realizar uma atividade.

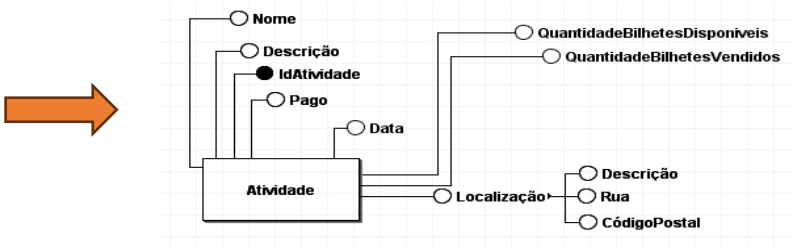


Figura 3 - Conversão Requisito para Entidade "Atividade"

Entidade Staff:

RD18.1	02/10/2023 15:24	Cada elemento do staff vai ter um identificador do staff, um nome, a função que cada elemento tem e a lista de números de telefone do elemento.

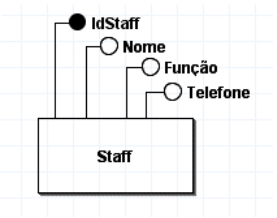


Figura 4 - Conversão Requisito para Entidade "Staff"

Entidade Bilhete:

RD14	02/10/2023 14:30	Os bilhetes vendidos deverão ter uma estrutura default para todo o tipo de eventos que sejam pagos.
RD16	02/10/2023 14:54	Os bilhetes têm um número de bilhete, o nome do respetivo evento, o nome da respetiva atividade e um preço.

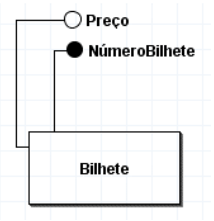


Figura 5 - Conversão Requisito para Entidade "Bilhete"

Entidade Artista:

RD10	02/10/2023 11:47	Um agente tem um ID de agente único, um nome, um email e o seu número de telefone.

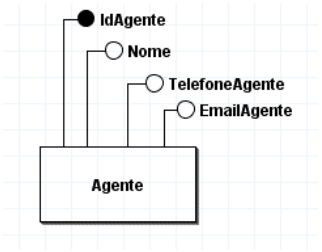


Figura 6 - Conversão Requisito para Entidade "Artista"

Entidade Agente:

RD9	02/10/2023 11:45	Um artista (vamos considerar um artista como sendo um elemento individual ou um grupo), tem um número único de artista para o identificar, um nome, uma descrição e é possível acessar aos dados do seu agente, e ainda tem o custo associado a contratar esse artista.

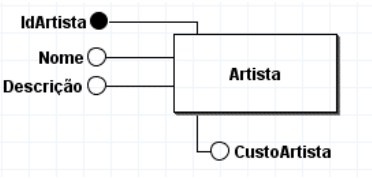


Figura 7 - Conversão Requisito para Entidade "Agente"

De seguida, foi criada uma tabela para abrigar cada uma das entidades e descrever de uma forma sucinta o que cada uma vai representar.

Tabela 5 - Entidades Modelo Conceptual

Designação	Descrição	Sinónimos	Ocorrência
Evento	Representa um evento que irá se realizar na cidade, acolhendo o registo de informações importantes para a realização do mesmo, bem como algumas informações financeiras também sobre o mesmo.	Acontecimento	Cada evento vai ter um número próprio, único e sequencial, que é atribuído quando o evento é registado na base de dados.
Atividade	Representa uma atividade dentro de um evento, acolhendo o registo de informações importantes para a realização do mesmo.	---	Cada atividade vai ter um número próprio, único e sequencial, que é atribuído quando a atividade é registada na base de dados.
Staff	Informações sobre as pessoas que vão trabalhar num evento.	Voluntário	Cada indivíduo é identificado com um nº único e sequencial, que é obrigatório para se registar como staff.
Bilhete	Entidade com informações básicas e essenciais que irão também ser apresentadas nos próprios bilhetes.	Ingresso	Todos os bilhetes vão ter um número de bilhete único.
Artista	Representa a informação sobre um artista que vai atuar numa atividade de um evento. Quando utilizamos a palavra artista estamos a considerar que este pode ser individual ou pode ser uma banda.	Banda	Cada artista vai estar identificado na base de dados com um número identificador único.
Agente	Representa a informação sobre um agente que trabalha para uma banda. Contém informação básica para definir um agente.	*Manager*	Cada agente vai estar identificado na base de dados com um número identificador único.

3.3. Identificação e caracterização dos relacionamentos

Temos um total de cinco relacionamentos entre entidades, sendo derivados os mesmos de requisitos anteriormente definidos que permitem interligar as entidades entre si.

Relacionamento Evento-Staff:

RD19	13/10/2023 12:55	Cada elemento do staff pode ajudar em mais do que um evento.
RD20	13/10/2023 12:57	Um evento tem sempre pelo menos um elemento do staff.

Figura 8 - Requisitos Descrição referentes à criação do relacionamento entre as entidades Evento e Staff

A partir destes dois requisitos de descrição é possível claramente definir um relacionamento entre as entidades “Evento” e “Staff”. Os requisitos acusam ainda, que este relacionamento terá cardinalidade 1: N, devido a ser um relacionamento 1 para muitos. É possível ler este

relacionamento da seguinte maneira: Um Evento tem 1 ou mais elementos do *Staff*, e um elemento do staff pode ajudar num ou mais eventos.

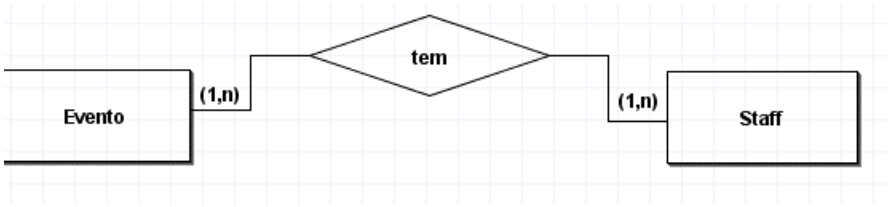


Figura 9 - Relacionamento "tem"

Relacionamento Evento-Atividade:

RD4	02/10/2023 11:04	Um evento tem sempre uma atividade pelo menos, senão não é considerado um evento. Pode ter mais do que uma atividade.
RD5	02/10/2023 11:32	Uma atividade só pode ser atividade de um evento

Figura 10 - Requisitos Descrição referentes à criação do relacionamento entre as entidades Evento e Atividade

A partir dos requisitos, é possível definir um relacionamento entre as entidades “Evento” e “Atividade” com cardinalidade 1 para N. É possível traduzir o relacionamento da seguinte forma, “Um evento tem pelo menos uma atividade podendo ter mais, mas uma atividade só é atividade desse evento”.

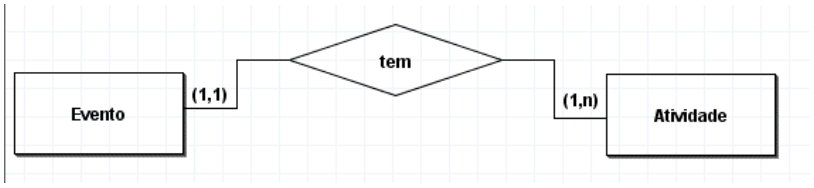


Figura 11 - Relacionamento "tem"

Da mesma forma é possível definir os restantes relacionamentos:
Evento-Bilhete e Atividade-Bilhete

RD21	16/10/2023 13:00	É possível comprar bilhetes para um evento, não sendo obrigatório, pois o evento pode ser gratuito.
RD21	16/10/2023 13:00	É possível comprar bilhetes para uma atividade, não sendo obrigatório, pois a atividade pode ser gratuita.

Figura 12 - Requisitos Descrição referentes à criação do relacionamento entre as entidades Evento, Bilhete e Atividade

Como existem eventos e atividades gratuitas, a obrigatoriedade de comprar bilhete não existe, então foi decidido utilizar a cardinalidade 1: N que se pode traduzir da seguinte forma para ambos os relacionamentos: Um evento pode ou não vender bilhetes. Cada bilhete é vendido mais do que uma vez para um evento.

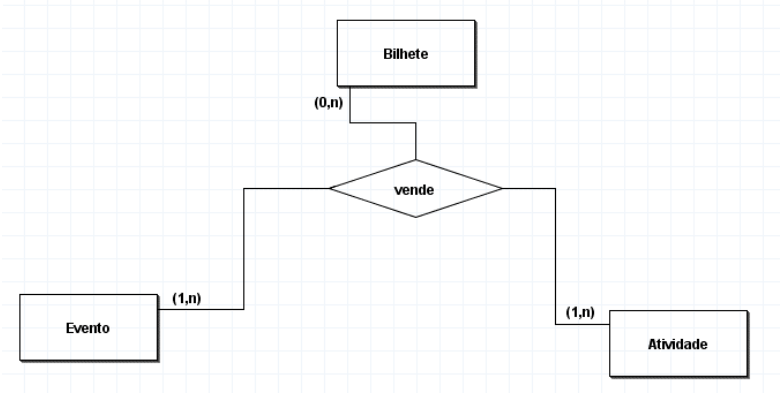


Figura 13 - Relacionamento “Vende”

Atividade-Artista:

RD22	17/10/2023 12:56	Uma atividade pode ou não ter um artista associado, por exemplo se a atividade for uma prova de comida. Por outro lado, um artista pode participar em 1 ou mais atividades.

Figura 14 - Requisitos Descrição referentes à criação do relacionamento entre as entidades Atividade e Artista

Este relacionamento vai ter cardinalidade 1: N – 0:1 e pode-se ler da seguinte forma: Uma atividade pode ou não ter um artista, sendo que se tiver é apenas 1. Um artista pode ser artista dessa atividade ou de outras.

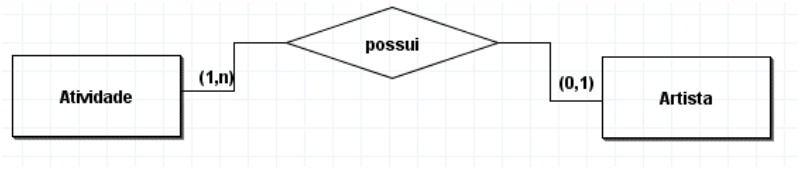


Figura 15 - Relacionamento “possui”

Artista – Agente

RD9	02/10/2023 11:45	Um artista (vamos considerar um artista como sendo um elemento individual ou um grupo), tem um número único de artista para o identificar, um nome, uma descrição e é possível acessar aos dados do seu agente, e ainda tem o custo associado a contratar esse artista.

Figura 16 - Requisitos Descrição referentes à criação do relacionamento entre as entidades Artista e Agente

Embora não esteja visível à primeira vista, é possível retirar um relacionamento entre “Artista” e “Agente”, a partir do momento que diz que é possível aceder aos dados do agente. A cardinalidade deste relacionamento é 1: N – 1:1 e pode ser traduzida da seguinte forma: Um artista tem um e um só agente. Um agente agencia um ou mais artistas.

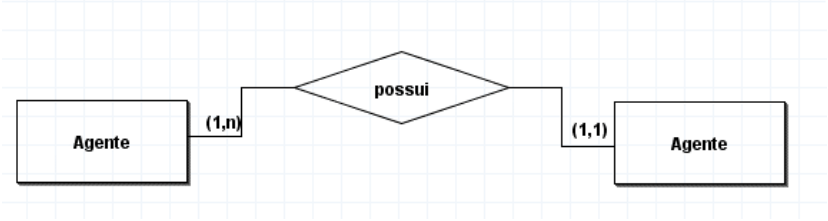


Figura 17 - Relacionamento "possui"

Ficam agora resumidas os relacionamentos em formato tabela.

Tabela 6 - Relacionamentos Modelo Conceptual

Entidade	Relacionamento	Cardinalidade	Entidade	Explicação
Evento	tem	1:1 – 1: N	Atividade	Um evento pode ter uma ou várias atividades associadas a ele. Cada atividade pertence a um evento.
Evento	tem	1: N- 1: N	Staff	Um evento tem um ou mais elementos do staff. Um elemento do staff faz parte desse e pode fazer parte de mais eventos.
Evento	vende	1, N:0, N	Bilhete	Um evento pode ou não vender bilhetes. Cada bilhete é vendido mais do que uma vez para um evento.
Atividade	vende	1, N:0, N	Bilhete	Uma atividade pode ou não vender bilhetes. Cada bilhete é vendido mais do que uma vez para uma atividade.
Atividade	possui	1: N – 0:1	Artista	Uma atividade pode ou não ter um artista. Se tiver um artista, será apenas um. Um artista pode estar associado a várias atividades em diferentes eventos.
Artista	tem	1: N- 1:1	Agente	Cada artista possui um único agente que pode representar vários artistas.

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

Com base nos requisitos levantados, e para cada entidade e relacionamento definido, foram associados os seguintes atributos:

- Entidades:
  - **Evento** – ID, Pago, Nome, Descrição, Localização (Descrição, Rua, CódigoPostal), Data (Data_Inicio, Data_Fim), ValorTotal, QuantidadeBilhetesVendidos, Preço, CustoEvento

- **Atividade** – Nome, Descrição, IdAtividade, Pago, Data, Localização (Descrição, Rua, CódigoPostal) e Lotação, Preço, CustoAtividade, QuantidadeBilhetesDisponveis, QuantidadeBilhetesVendidos
- **Staff** – IdStaff, Nome, Função e Telefone
- **Bilhete** –NúmeroBilhete e Preço
- **Artista** –IdArtista, Nome, Descrição, CustoArtista
- **Agente** – IdAgente, Nome, TelefoneAgente e eMailAgente

Em baixo está apresentada uma tabela que tem como intuito descrever de forma muito resumida o papel de cada atributo neste modelo conceptual da base de dados.

Entidade	Atributo	Descrição	Domínio e Tamanho	Nulo	Exemplo
Evento	IdEvento	Identificador do Evento	INTEGER	N	1
	Pago	Valor booleano para dizer se o evento é pago ou não	BOOLEAN ou TINYINT(1)	N	True
	Nome	Nome do Evento	VARCHAR(50)	N	Braga Authentica
	Descrição	Pequena descrição sobre o evento	TEXT	S	"Festival de música onde irão estar presentes os maiores artistas do mundo, …"
	Localização (	Localização do Evento			
	Descrição	Descrição do local do evento	VARCHAR(100)	N	Altice Fórum
	Rua	Nome da Rua	VARCHAR(75)	N	R. Monsenhor Airosa
	CódigoPostal)	Código Postal	VARCHAR(20)	N	4705-002
	Data				
	Data_Inicio	Data que marca o início do evento	DATETIME	N	17/12/2023
	Data_Fim	Data que marca o fim do evento	DATETIME	N	21/12/2023
	ValorTotal	Valor Arrecadado	INTEGER	S	15000
	QuantidadeBilhetesVendidos	Quantos bilhetes foram vendidos	INTEGER	S	500
	Preço	Preço derivado do bilhete	DECIMAL(8,2)	S	50.00
	CustoEvento		DECIMAL(8,2)	S	50000.00

		Quanto custa realizar o evento			
Atividade	IdAtividade	Identificador da atividade	INTEGER	N	1
	Nome	Nome da Atividade	VARCHAR(50)	N	Concerto Placebo
	Descrição	Pequena descrição sobre a atividade	TEXT	S	“Placebo voltam aos palcos no festival “Braga Authentica…”
	Pago	Valor booleano para dizer se a atividade é paga ou não	BOOLEAN ou TINYINT(1)	N	True
	Data	Data que se irá realizar uma atividade	DATETIME	N	20/12/2023
	Localização	Localização onde decorrerá uma atividade			
	Descrição	Descrição do local da atividade	VARCHAR(100)	N	Altice Fórum
	Rua	Nome da Rua	VARCHAR(75)	N	R.Monsenhor Airosa
	CódigoPostal	Código Postal	VARCHAR(20)	N	4705-002
	Lotação	Lotação máxima de espectadores numa atividade	INTEGER	N	50000
	Preço	Custo da atividade	DECIMAL(8,2)	S	50.00
	CustoAtividade	Quanto custa realizar a atividade	DECIMAL(8,2)	S	50000.00
	QuantidadeBilhetesDisponiveis	Nrº Bilhetes Disponiveis	INTEGER	S	1000
	QuantidadeBilhetesVendidos	Nrº Bilhetes Vendidos	INTEGER	S	500
Staff	IdStaff	Identificador de um elemento do Staff	INTEGER	N	1

	Nome	Nome do elemento do Staff	VARCHAR(75)	N	Manuel Afonsino
	Funçao	Função que vai exercer no evento	VARCHAR(50)	N	Auxiliar de limpeza
	Telefone	Número Telefone	VARCHAR(15)	S	+351 992 488 223
Bilhete	NúmeroBilhete	Identificador de um bilhete	INTEGER	N	1
	Preço	Preço do bilhete	INTEGER	N	100
Artista	IdArtista	Identificador do artista	INTEGER	N	1
	Nome	Nome ou designação do artista (consideramos que um artista pode ser individual, ou então um grupo)	VARCHAR (75)	N	Kurt Cobain, The Script
	Descrição	Pequena descrição adicional sobre o artista	TEXT	S	"Esta banda foi inaugurada no dia 24 de setembro de 1998…"
	CustoArtista	Quanto custa o artista	DECIMAL(8,2)	S	1000.00
Agente	IdAgente	Identificador de um agente	INTEGER	N	1
	Nome	Nome do agente	VARCHAR(75)	N	Nicolino Andrade Vieira
	TelefoneAgente	Telefone Agente	VARCHAR(15)	N	+351 912 241 224
	eMailAgente	Email do agente	VARCHAR(50)	N	nicolino_vieira23@outlook.pt

Tabela 7 - Atributos Modelo Conceptual

Quanto a alguns tipos de atributos presentes nas tabelas, foram discutidas diferentes maneiras de representar o valor pretendido.

Por exemplo, nas entidades "Evento" e "Atividade" e no respetivo atributo "Pago", comum às duas entidades, foi escolhido o tipo `BOOLEAN`. Esta escolha prende-se no facto de querermos que este atributo apenas represente os valores `True` e `False`, de modo a exprimir se um evento é pago ou não, e se uma atividade é paga ou não. Na linguagem *SQL*, não existe um tipo especifico `BOOLEAN`, mas existe este tipo `BOOLEAN` como um sinónimo associado ao tipo `TINYINT(1)`, que apenas apresenta dois valores: 1 para `True` e 0 para `False`. Para facilitar a

leitura dos tipos dos atributos foi determinado utilizar o tipo *BOOLEAN*, sabendo no entanto que o tipo certo para se utilizar nestas situações é o tipo *TINYINT*(1).
Foi decidido também utilizar um atributo composto “Data” na entidade “Evento”, com os atributos “Data_Inicio” e “Data_Fim”, para marcar o dia em que começa um evento e o dia em que termina o mesmo evento.

3.5. Apresentação e explicação do diagrama ER produzido

O modelo conceptual finalizado contém 6 entidades e 5 relacionamentos.
Das 6 entidades, a que tem maior grau (número de atributos que a constituem) ,16, é a entidade “Evento”, sendo a de menor grau a entidade “Bilhete” com grau 2.
Dos 5 relacionamentos, todos são relacionamentos binários, ou seja, ligam apenas 2 entidades, à exceção do relacionamento “vende” que liga 3 entidades e assim é dito ternário.
Este modelo conceptual foi realizado com o objetivo de otimizar a simplificação na criação de tabelas e cumprir o maior número de requisitos possível.

Em baixo, é possível observar como ficou definido o modelo conceptual final, depois de analisados todos os pormenores, todos as entidades, relacionamentos e atributos. O mesmo está disponível nos anexos, junto com o relatório.

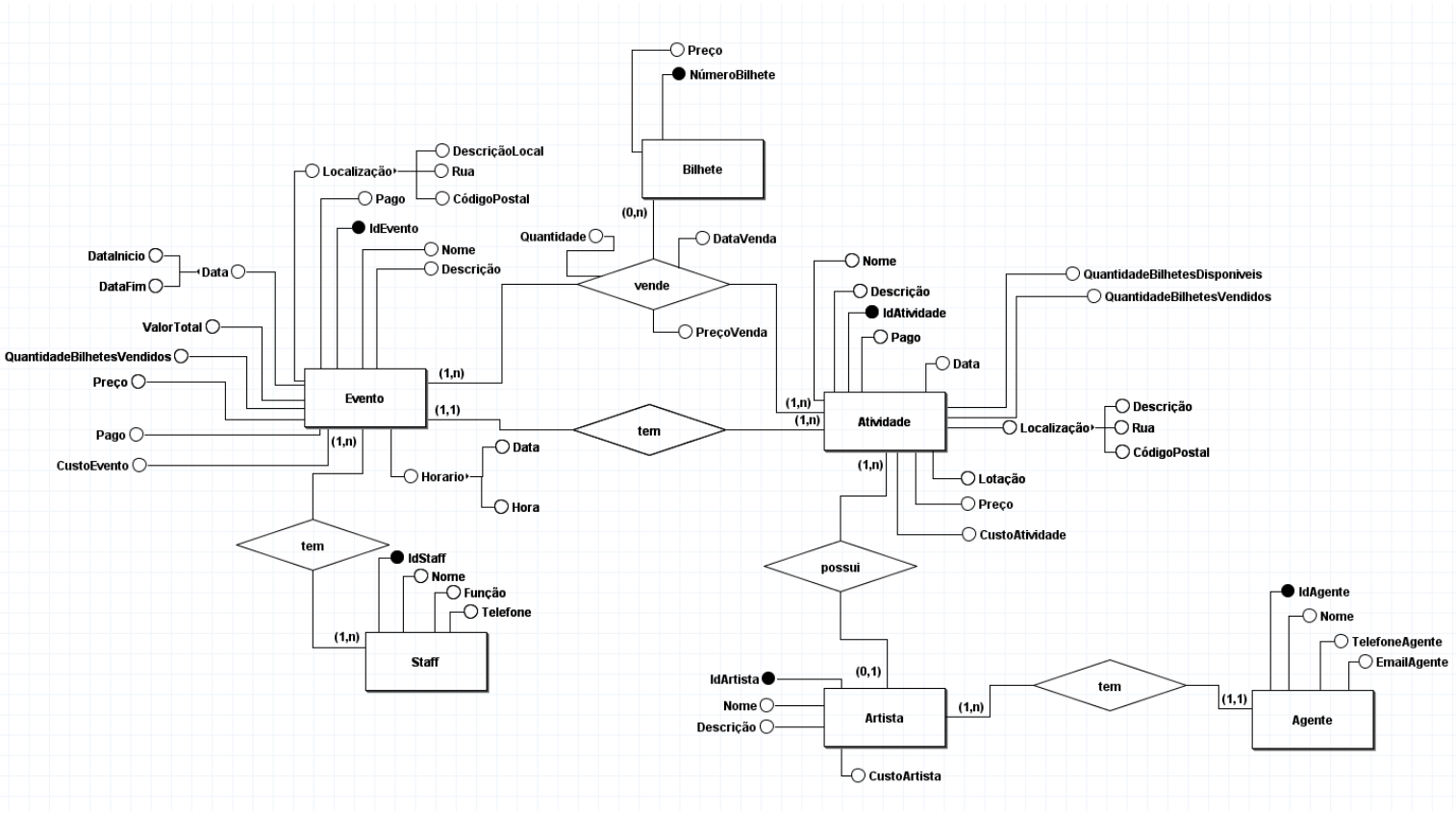


Figura 18 - Diagrama ER Final

Desta forma, e após ter uma melhor noção do modelo que pretendemos implementar na nossa base de dados de gestão de um calendário de eventos numa cidade turística, mais especificamente em Braga, é possível avançar para a conceção de uma nova modelação, lógica, em que o objetivo final é chegar a uma implementação física.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Nesta etapa do trabalho, foram realizadas diversas operações para além da construção do modelo lógico. Nomeadamente, operações como a validação do nosso esquema através da normalização de dados, verificação das restrições de integridade e a revisão do esquema lógico final produzido com os seus utilizadores. Nesta última operação, foi realizada a verificação se a nossa base de dados vai ser capaz de responder às necessidades dos mesmos.

O modelo lógico infra apresentado resulta da conversão das entidades e relacionamentos do nosso modelo conceptual. Esta conversão foi realizada a partir de uma linha definida de pontos que passamos a apresentar:

- Uma entidade no esquema conceptual corresponde a uma tabela no modelo lógico.
- Um atributo composto é representado na sua tabela base, apenas pelos seus atributos que o compõem.
- Um atributo multi-valor dá origem a uma nova tabela, com um relacionamento 1: N com a sua tabela de referência.
- Um relacionamento binário 1: N (um para muitos) é implementando com uma chave estrangeira do lado “N” com referência à tabela do lado do “1”.
- Um relacionamento binário N: M (muitos para muitos) dá origem a uma nova tabela, contendo na sua chave primária duas chaves estrangeiras, cada uma delas com referência a uma das tabelas envolvidas no relacionamento.
- Os identificadores de uma entidade passam a designar-se *Primary Key* ou *Foreign Key,* caso em que o identificador de uma entidade esteja presente noutra tabela.

Passou-se então à realização da conversão. As tabelas que foram criadas foram as seguintes:

1. **Evento**
    - 1.1 ***Primary Key:*** IdEvento INT
    - 1.2 **Atributos:** IdEvento INT, Nome VARCHAR(70), Descrição TEXT, DataInicio DATETIME, DataFim DATETIME, CodPostal VARCHAR(50), DescriçãoLocal VARCHAR(50), Rua VARCHAR(70), Pago TINYINT(1), ValorTotal INT, QuantidadeBilhetesVendidos INT, Preço DECIMAL(8,2), CustoEvento DECIMAL(8,2)
    - 1.3 ***Foreing Key:*** Não tem

A tabela “Evento” é uma derivação da entidade “Evento” no nosso modelo conceptual. Como o relacionamento é 1: N entre a entidade “Evento” e “Atividade”, não foram adicionadas

nenhuma chave estrangeira do lado da tabela “Evento”. A chave primária foi obtida ao converter o identificador da entidade.

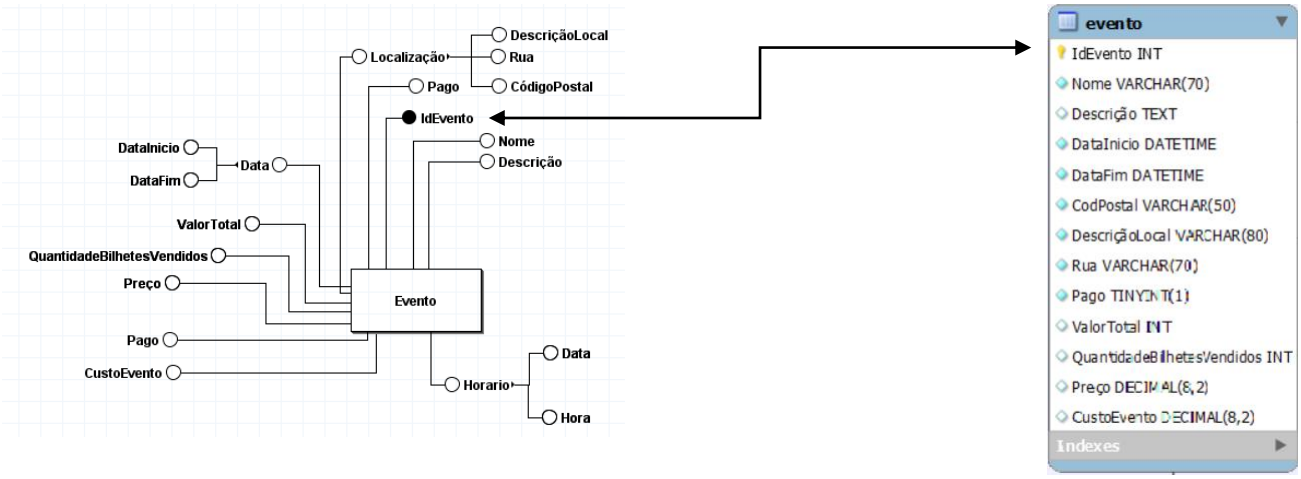


Figura 19 - Conversão tabela "Evento" para tabela "Evento"

2. **Atividade**

   2.1 Primary Key: IdAtividade INT

   2.2 Atributos: IdAtividade INT, Nome VARCHAR (70), Descrição TEXT, Pago TINYINT(1), CodPostal VARCHAR(50), DescriçãoLocal VARCHAR(50), Rua VARCHAR(70),Data DATETIME, Lotação INT, IdEvento INT, IdArtista INT, ValorTotal INT, QuantidadeBilhetesVendidos INT, BilhetesDisponiveis INT, Preço DECIMAL (8,2), CustoAtividade DECIMAL(8,2)

   2.3 Foreing Key: IdEvento INT e IdArtista INT

A tabela “Atividade” é resultante de uma derivação da entidade “Atividade” no nosso modelo conceptual. Como o relacionamento “tem” entre a entidade “Evento” e “Atividade” tem cardinalidade 1: N, então do lado da atividade foi incluída a chave estrangeira “IdEvento”. Como o relacionamento “possui” entre “Atividade” e “Artista” tem cardinalidade (1,N)-(0,1), foi adicionada uma chave estrangeira “IdArtista” na tabela “Atividade”. A chave primária foi obtida ao converter o identificador da entidade.

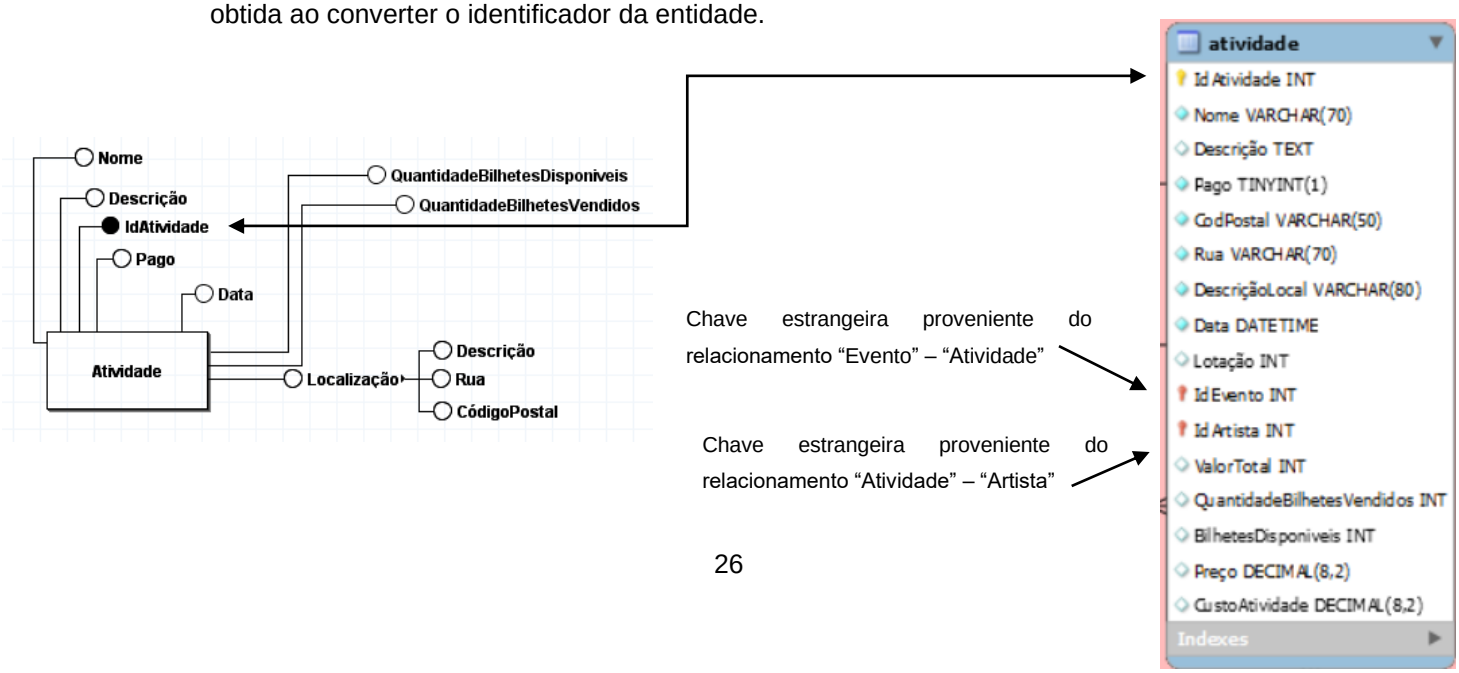


Figura 20 - Conversão entidade "Atividade" para tabela "Atividade"

3. **Bilhete**
   - ***3.1 Primary Key:*** NúmeroBilhete INT
   - ***3.2* Atributos:** NúmeroBilhete INT, Preço DECIMAL(8,2), IdEvento INT, IdArtista INT
   - ***3.3 Foreing Key:*** IdEvento INT e IdArtista INT

A tabela “Bilhete” é uma derivação da entidade “Bilhete” do nosso modelo conceptual. Como existe um relacionamento entre “Atividade” – “Bilhete” e “Evento” – “Bilhete”, ambos com cardinalidade 1: N, foram adicionadas duas chaves estrangeiras “IdEvento” e “IdAtividade”, ambas provenientes, respetivamente, das entidades “Evento” e “Atividade”.

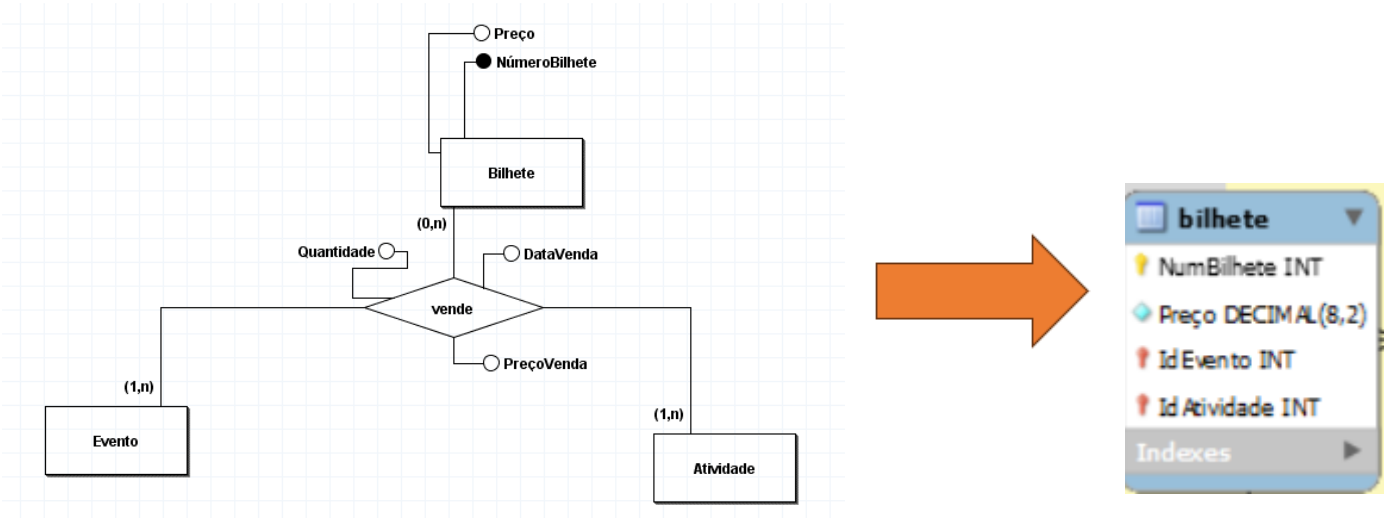


Figura 21 - Conversão relacionamento entre "Evento", "Atividade" e "Bilhete" para a tabela "Bilhete"

4. **BilheteVendidos**
   - ***4.1 Primary Key:*** IdBilheteVendidos INT
   - ***4.2* Atributos:** IdBilheteVendido INT, IdEvento INT, IdAtividade INT, Quantidade INT, DataVenda DATETIME, PreçoVenda DECIMAL(8,2), IdBilhete INT
   - ***4.3 Foreing Key:*** IdEvento INT, IdArtista INT, IdBilhete INT

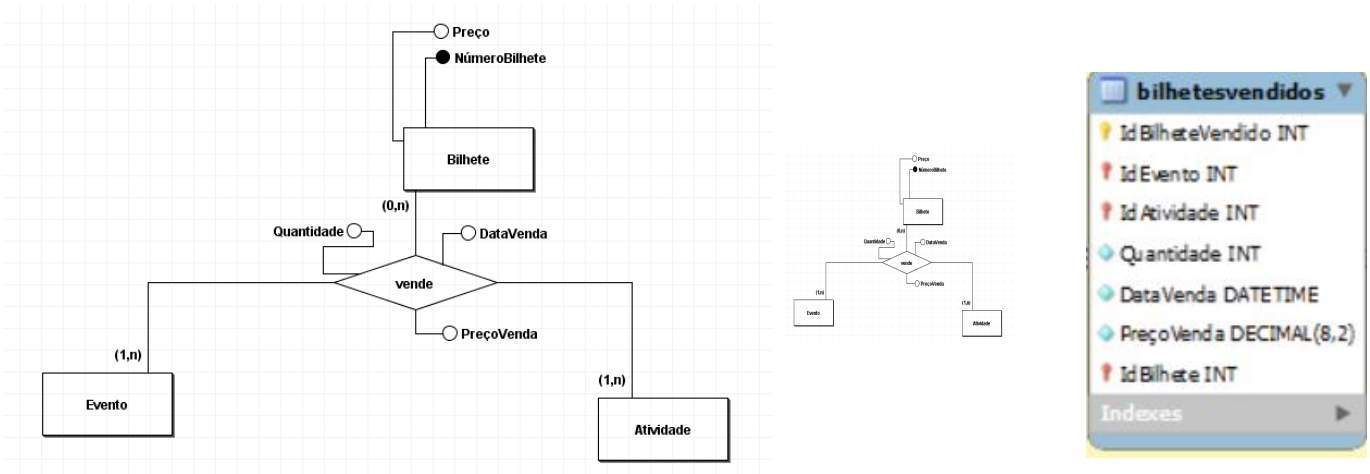


Figura 22 - Conversão relacionamento entre "Bilhete", "Evento" e "Atividade" para a tabela "BilhetesVendidos"

Esta tabela foi criada, posteriormente à realização do nosso modelo conceptual, com o objetivo de manter registo de todos os bilhetes vendidos, um por um, associados a um evento, atividade e a um tipo de bilhete criado para este evento ou atividade. Esta tabela resultou do relacionamento "vende", entre "Evento", "Atividade" e "Bilhete". As chaves estrangeiras presentes nesta tabela, relacionam-se respetivamente com as chaves primárias das tabelas "Evento", "Atividade" e "Bilhete". Além das chaves estrangeiras que foram adicionadas, foram também adicionados à tabela os campos DataVenda, Quantidade e PreçoVenda, provenientes dos atributos presentes no relacionamento "Vende".

5. **Artista**
   - ***5. 1 Primary Key:*** IdArtista INT
   - **5.2 Atributos:** IdArtista INT, Nome VARCHAR (70), Descrição TEXT, IdAgente INT, CustoArtista DECIMAL(8,2)
   - **5.3 *Foreing Key:*** IdAgente INT

Esta tabela é uma derivação direta da entidade "Artista" e dos seus atributos. Como existe um relacionamento 1: N entre "Artista" e "Agente", foi adicionada uma chave estrangeira IdAgente à tabela "Artista".

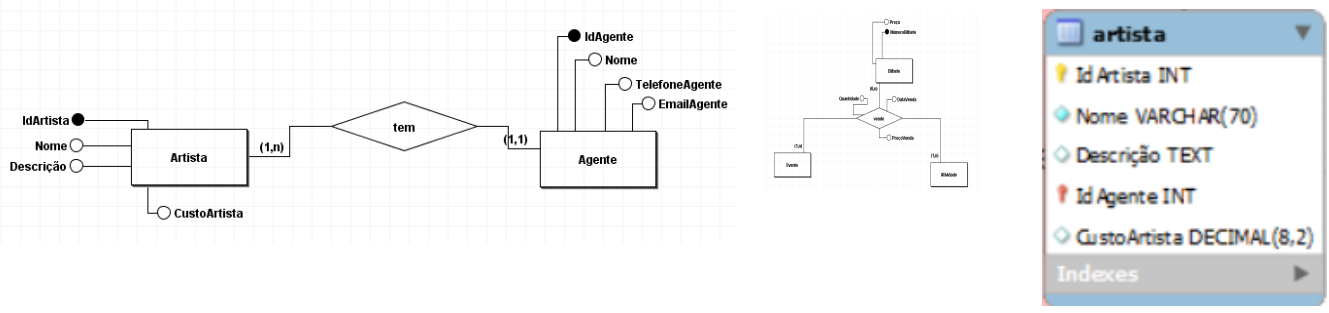


Figura 23 - Conversão relacionamento entre "Artista" e "Agente" para a tabela "Artista"

6. **Agente**
   - ***6.1 Primary Key:*** IdAgente INT
   - **6.2 Atributos:** IdAgente INT, Nome VARCHAR (70), TelefoneAgente VARCHAR (20) , EmailAgente VARCHAR(70)
   - **6.3 *Foreing Key:*** Não tem

Esta tabela é uma derivação direta da entidade “Agente” e dos seus atributos.

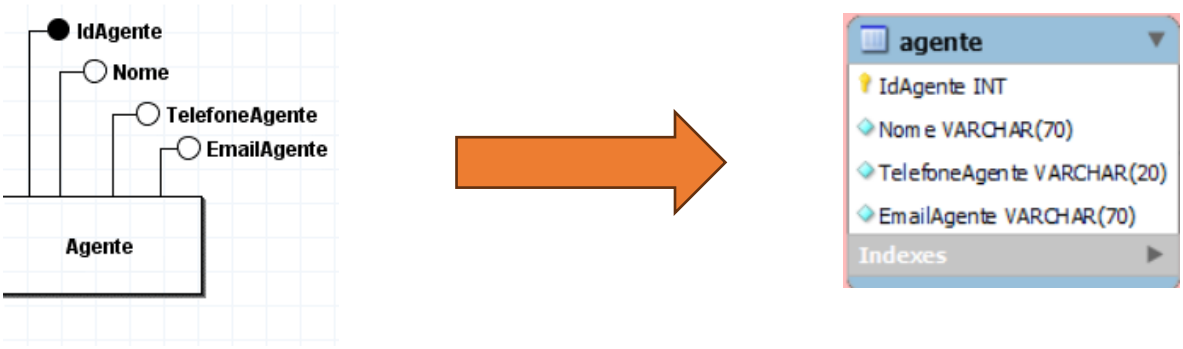


Figura 24 - Conversão da entidade "Agente" para a tabela "Agente"

7. **Staff**
   - ***7.1 Primary Key:*** IdStaff INT
   - **7.2 Atributos:** IdStaff INT, Nome VARCHAR (70), Telefone VARCHAR (20) , Função VARCHAR(70)
   - **7.3 *Foreing Key:*** Não tem

Esta tabela é derivada diretamente a partir da entidade “Staff”

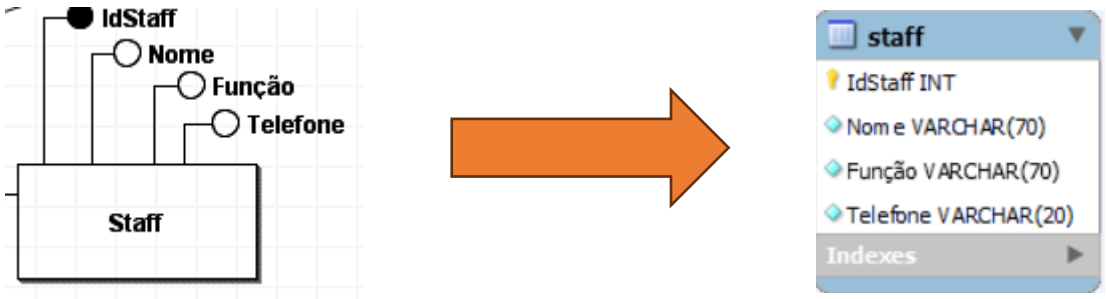


Figura 25 - Conversão da entidade "Staff" para a tabela "Staff"

8. **StaffEvento**
   - ***8.1 Primary Key:*** Staff INT , Evento INT
   - **8.2 Atributos:** Staff INT, Evento INT
   - **8.3 *Foreing Key:*** *Staff INT, Evento INT*

Neste caso, e como existe um relacionamento N: N entre as entidades “Evento” e “Staff”, foi criada esta tabela “StaffEvento”, que aporta ambas as chaves primárias da tabela “Evento” e “Staff”.

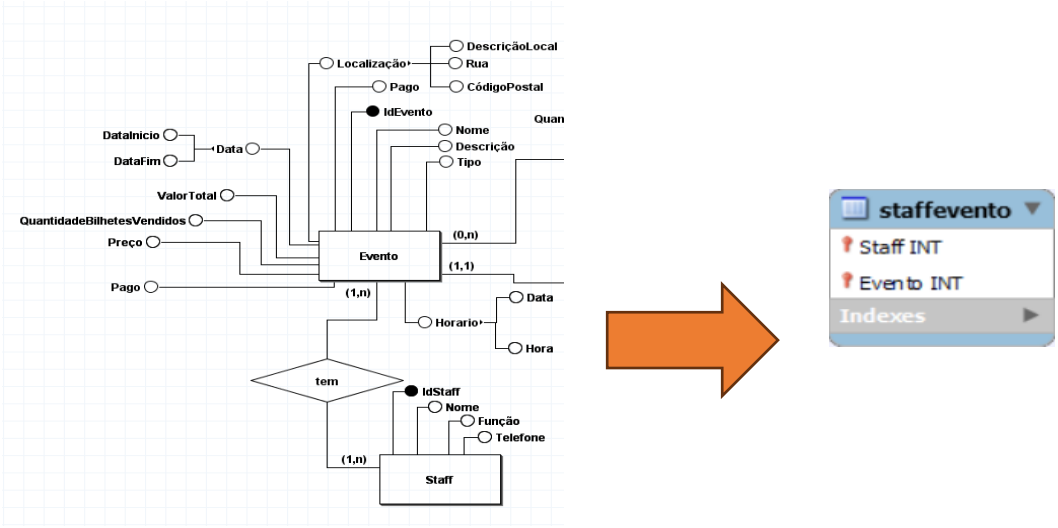


Figura 26 - Conversão do relacionamento entre as entidades "Evento" e "Staff" para a tabela "StaffEvento"

Assim, e após o término da nossa conversão podemos observar que foram obtidas 8 tabelas, das quais 2 foram originadas a partir de relacionamentos.

4.2. Normalização de Dados

Nesta etapa do trabalho, e já com o modelo lógico praticamente finalizado, é altura de o validar através da normalização dos seus dados. A normalização de dados tem como objetivo evitar a redundância dos dados, permitindo assim otimizar da melhor forma a nossa base de dados e permite mais tarde, evitar erros que possam existir na inserção, remoção e alteração dos registos na base de dados.

Para efetuar uma validação preliminar através da normalização dos dados, foi necessário verificar se o nosso modelo cumpre todas as formas normais até à terceira.

A Primeira Forma Normal (1ªFN) estipula que os atributos de uma tabela devem ser atómicos, isto é, as tabelas não podem conter registos duplicados. Ao analisar as tabelas geradas na conversão para o modelo lógico, foi verificado que este critério é cumprido, confirmando que o nosso modelo satisfaz a 1ªFN.

A Segunda Forma Normal (2ªFN) requer que a 1ªFN seja satisfeita e que todos os atributos não-chave das tabelas do nosso modelo lógico dependam unicamente da chave primária dessa tabela. Ao reexaminar o modelo, foi constatado que todos os atributos das tabelas dependem da chave primária das mesmas.

A Terceira Forma Normal (3ªFN) implica que, além da satisfação da 1ªFN e da 2ªFN, todos os atributos não-chave não dependem de outros atributos não-chave. Mais uma vez, após revisão das tabelas, observamos que este requisito é cumprido.

Portanto, tendo as três formas normais satisfeitas, foi possível validar o nosso modelo lógico através da normalização dos dados.

4.3. Apresentação e explicação do modelo lógico produzido

O modelo lógico produzido, a partir da conversão do modelo conceptual foi o seguinte.

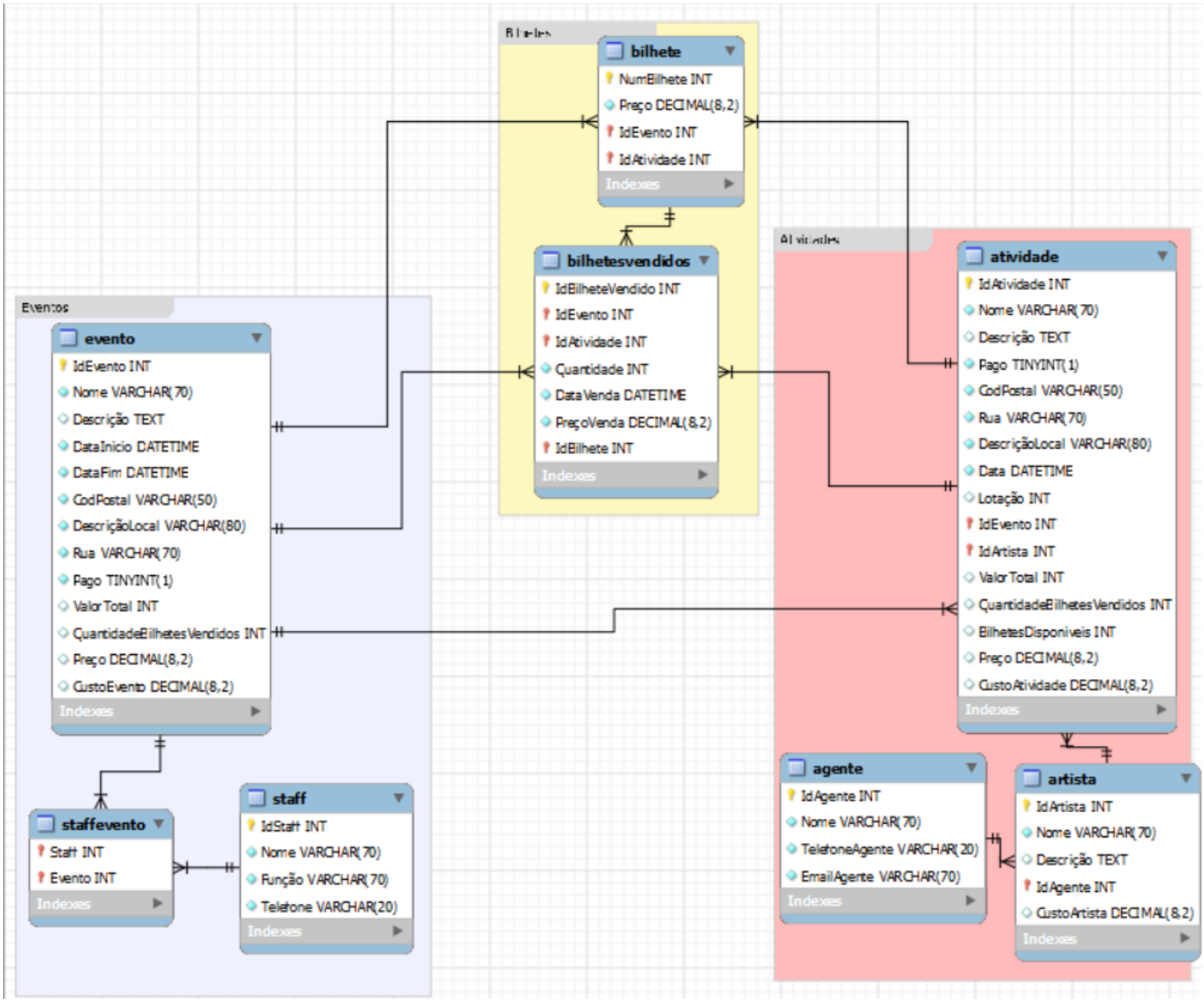


Figura 27 - Diagrama das tabelas do modelo lógico

À medida o modelo lógico foi construído, foram realizadas algumas alterações que não estavam previstas no modelo conceptual, como por exemplo a inserção dos atributos “Quantidade”, “DataVenda” e “PreçoVenda”, na tabela “bilhetesVendidos”, e posteriormente também foi revisto o modelo conceptual, para aportar estas alterações. Estas melhorias foram sempre feitas, respeitando os requisitos base do projeto, e visando melhorar a performance e otimização da base de dados, sem nunca comprometer a normalização dos mesmos. Toda esta conversão do modelo conceptual para o lógico foi explicada ao detalhe no ponto 4.1 deste relatório, e revista mais do que uma vez para respeitar todas as regras pré-definidas, e disponíveis também nesse

mesmo ponto. Este modelo lógico permite uma gestão eficaz dos eventos, e também a apresentação de detalhes sobre os mesmos, assim como fácil gestão da venda dos bilhetes e organização do staff e dos artistas.

4.4. Validação do modelo com interrogações do utilizador

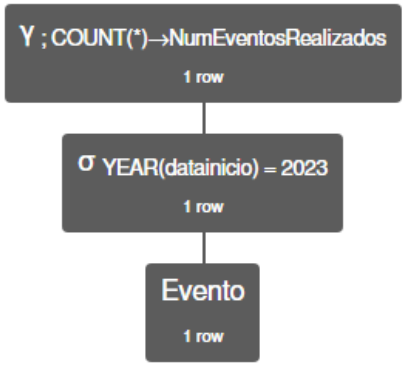
De modo a validar o nosso modelo lógico e entender se o mesmo consegue responder às necessidades dos utilizadores, e dos pré-requisitos base que foram escolhidos, foram selecionadas algumas interrogações dos requisitos de exploração e, recorrendo à álgebra relacional, pretendemos observar se as mesmas podem ser respondidas de forma satisfatória.

a) Contabilizar o número de eventos que foram realizados num ano:

$$\sigma_{\text{YEAR(datainicio)} = 2023}\ (\ \text{Evento}\ )$$

Figura 28 - Árvore de derivação da álgebra relacional e respetiva álgebra

A primeira operação a realizar é uma seleção (σ), que é usada para escolher todas as linhas da tabela “Evento”, que satisfazem a condição que diz que o ano é 2023.

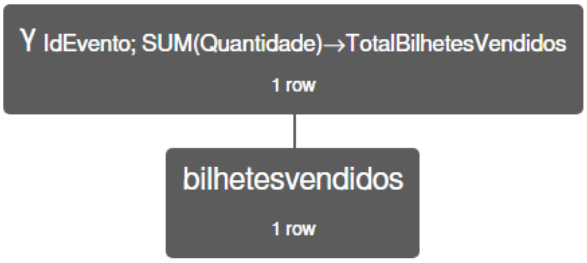


$$\gamma\ ;\ \text{COUNT}(*)\rightarrow\text{NumEventosRealizados}\ (\ \sigma_{\text{YEAR(datainicio)} = 2023}\ (\ \text{Evento}\ )\ )$$

Figura 29 - Árvore de derivação da álgebra relacional e respetiva álgebra

De seguida foi utilizada uma operação de agregação de dados (γ), para ser possível realizar a contagem de todos os Eventos, cujo ano é de 2023, ou seja, dos valores que foram filtrados anteriormente. Por fim foi realizada uma operação de renomeação (→).

b) Obter o número de bilhetes vendidos por evento, ordenados por quantidade de bilhetes vendidos

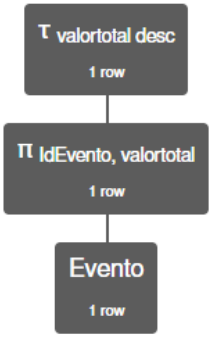


γ IdEvento; SUM(Quantidade)→TotalBilhetesVendidos (bilhetesvendidos)

Figura 30 - Árvore de derivação da álgebra relacional e respetiva álgebra

Não há muito que saber acerca do processo de formação desta fórmula. Foi agrupada a soma de bilhetes comprados para cada registo “IdEvento” presente na tabela “BilhetesVendidos”, e após isso foi renomeada a fórmula para “TotalBilhetesVendidos”.

c) Calcular o total obtido por evento, ordenado pelo valor total em ordem crescente



τ valortotal desc (π IdEvento, valortotal (Evento))

Figura 31 - Árvore de derivação da álgebra relacional e respetiva álgebra

Para responder a esta interrogação, basta projetar as colunas “IdEvento” e “valortotal” da tabela “Evento” e por fim ordenar por ordem decrescente a coluna “valortotal”.

d) Listar todos os eventos grátis

$$\pi_{\text{IdEvento, nome}} (\sigma_{\text{pago = false}} (\text{Evento}))$$

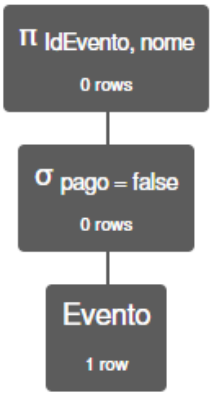
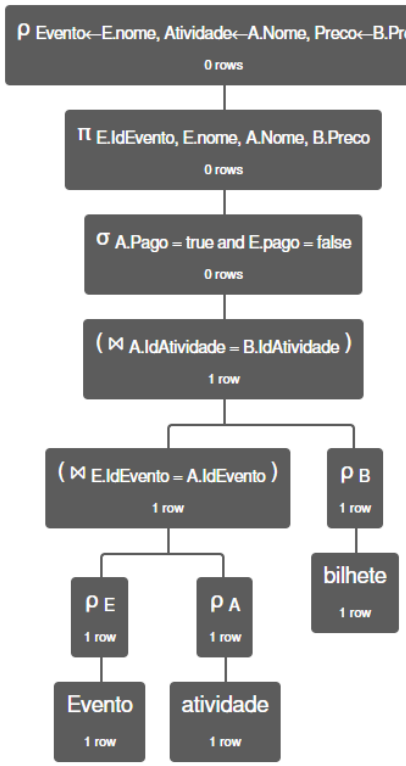


Figura 32 - Árvore de derivação da álgebra relacional e respetiva álgebra

São projetadas as colunas “IdEvento” e “nome”, da tabela “Evento”, onde o atributo “pago” é falso, ou seja, o evento é gratuito.

e) Selecionar os eventos grátis que tenham atividades pagas



$$\rho_{\text{Evento←E.nome, Atividade←A.Nome, Preco←B.Preco}} \pi_{\text{E.IdEvento, E.nome, A.Nome, B.Preco}} \sigma_{\text{A.Pago = true and E.pago = false}} ((\rho_{\text{E}} \text{Evento} \bowtie_{\text{E.IdEvento = A.IdEvento}} \rho_{\text{A}} \text{atividade}) \bowtie_{\text{A.IdAtividade = B.IdAtividade}} \rho_{\text{B}} \text{bilhete})$$

Figura 33- Árvore de derivação da álgebra relacional e respetiva álgebra

Esta expressão começa por realizar operações de mudança de nome, e após isso, une as tabelas “Evento”, designada por E na expressão, “Atividade” e “Bilhete”, a primeira junção com base nos atributos “IdEvento” de ambas as tabelas “Evento” e “Atividade”, a junção desta com base nos atributos “IdAtividade” das tabelas “Atividade” e “Bilhete”. Após a junção, a condição “`σ A.Pago = true and E.Pago = false`” filtra as linhas onde o evento é gratuito, mas tem atividades pagas. Em último lugar, são projetadas as colunas desejadas.

5. Implementação Física

Esta parte do relatório é referente à implementação física. O sistema de gestão de base de dados que foi utilizado é o *MySQL*. Para desenvolver a base de dados, foi utilizado o *software MySQL Workbench*, e respetiva linguagem padrão, o *SQL* (*Structured Query Language*).

5.1. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido

O esquema da base de dados da mercearia foi implementado em *MySQL* utilizando o *script SQL* "202324-UM-LCC-BD-G04-CriaçãoBD", que foi desenvolvido para este efeito.
Tudo começa com uma simples instrução para criar a nossa base de dados.

```
-- **Criação da Base de Dados**
CREATE DATABASE Eventos;
```

De seguida, foi utilizado o seguinte comando para explicitar ao *MySQL Server* qual é a base de dados em que queremos trabalhar.

```
-- Instrução para assinalar qual a base de dados em que pretendemos trabalhar
USE Eventos;
```

Após este comando, foi iniciada a implementação das nossas tabelas na base de dados "Eventos".
O comando para criar uma tabela na linguagem *SQL* é o seguinte: `CREATE TABLE`
Para o caso da tabela "Evento", foi explicitado ao comando em cima referido, qual é o nome da tabela que pretendíamos criar, assim como os nomes e tipos das suas colunas. Por fim, é possível ou não explicitar qual é o tipo de motor de criação das tabelas que pretendemos utilizar. No nosso caso foi escolhido utilizar o motor *InnoDB*. Este motor é o motor padrão do *MySQL*, e é frequentemente utilizado devido ao seu alto desempenho e também às suas funcionalidades, como por exemplo o suporte ao padrão de transações *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade). Foi utilizada também em certos atributos, a sintaxe *NOT NULL*, que não permite inserir dados na tabela, sem preencher estes atributos.

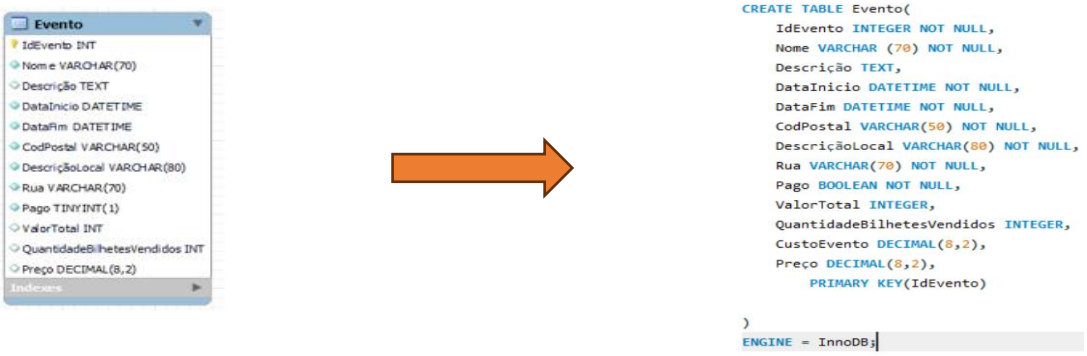


Figura 34 - Implementação física da tabela "Evento", na linguagem *SQL*

No *script SQL* para a criação da BD, foram utilizadas sintaxes específicas para identificar as chaves primárias e as chaves estrangeiras.

Observemos agora a seguinte tabela para exemplificar este ponto:

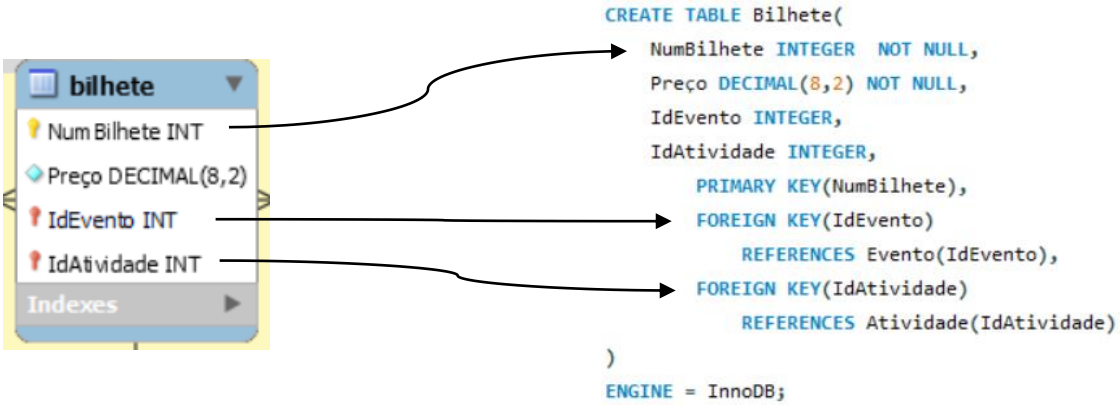


Figura 35 Implementação física da tabela "Bilhete", na linguagem *SQL*

Neste exemplo, a chave primária foi identificada com a sintaxe *PRIMARY KEY* e as chaves estrangeiras com a sintaxe *FOREIGN KEY REFERENCES*, esta última identificando a chave estrangeira, e respetivamente a tabela onde a chave primária está inserida.
Da mesma forma, foram implementas as restantes tabelas.

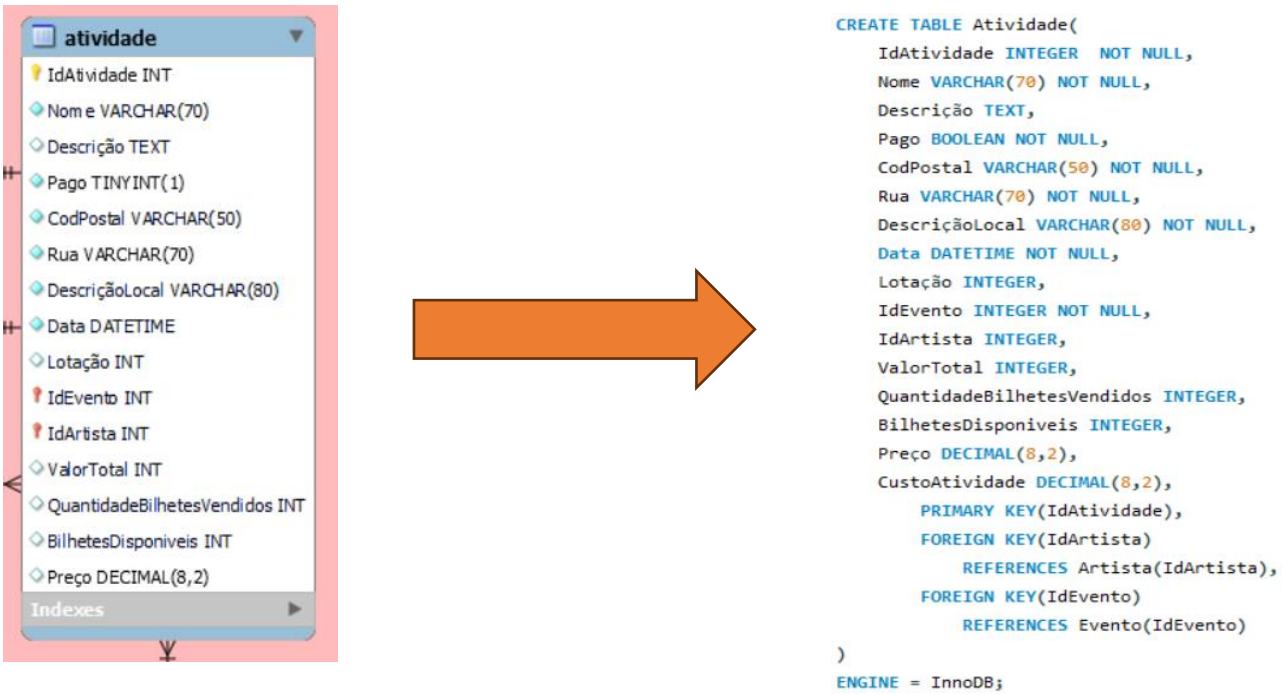
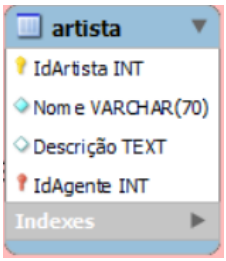
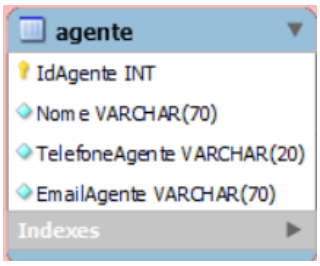


Figura 36 - Implementação física da tabela "Atividade", na linguagem *SQL*



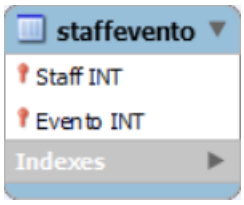

```sql
CREATE TABLE Artista(
    IdArtista INTEGER  NOT NULL,
    Nome VARCHAR(70) NOT NULL,
    Descrição TEXT,
    IdAgente INTEGER NOT NULL,
    CustoArtista DECIMAL(8,2),
        PRIMARY KEY(IdArtista),
        FOREIGN KEY(IdAgente)
            REFERENCES Agente(IdAgente)
)
ENGINE = InnoDB;
```

Figura 37 Implementação física da tabela "Artista", na linguagem *SQL*



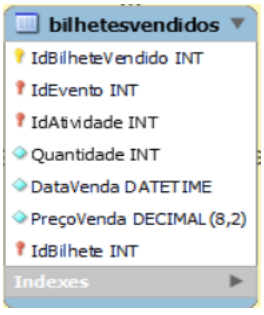

```sql
CREATE TABLE Agente(
    IdAgente INTEGER  NOT NULL,
    Nome VARCHAR(70) NOT NULL,
    TelefoneAgente VARCHAR(20) NOT NULL,
    EmailAgente VARCHAR(70) NOT NULL,
        PRIMARY KEY(IdAgente)
)
ENGINE = InnoDB;
```

Figura 38 Implementação física da tabela "Agente", na linguagem *SQL*




```sql
CREATE TABLE StaffEvento(
    Staff INTEGER NOT NULL,
    Evento INTEGER NOT NULL,
    FOREIGN KEY(Staff)
            REFERENCES Staff(IdStaff),
    FOREIGN KEY(Evento)
            REFERENCES Evento(IdEvento)
)
ENGINE = InnoDB;
```

Figura 39 Implementação física da tabela "StaffEvento", na linguagem *SQL*




```sql
CREATE TABLE BilhetesVendidos (
    IdBilheteVendido INT NOT NULL AUTO_INCREMENT,
    IdEvento INT,
    IdAtividade INT,
    Quantidade INTEGER NOT NULL,
    DataVenda DATETIME NOT NULL,
    PreçoVenda DECIMAL(8,2) NOT NULL,
    IdBilhete INT,
    PRIMARY KEY (IdBilheteVendido),
    FOREIGN KEY (IdEvento)
            REFERENCES Evento(IdEvento),
    FOREIGN KEY (IdAtividade)
            REFERENCES Atividade(IdAtividade),
    FOREIGN KEY (IdBilhete)
            REFERENCES Bilhete(NumBilhete)
)
ENGINE = InnoDB;
```

Figura 40 Implementação física da tabela "BilhetesVendidos", na linguagem *SQL*

5.2. Tradução das interrogações do utilizador para SQL

Anteriormente foram desenvolvidas algumas interrogações, recolhidas com base nos requisitos de manipulação, de modo a validar o nosso modelo lógico e se o mesmo seria capaz de responder a interrogações por parte dos utilizadores, foram utilizadas as mesmas interrogações, agora para comprovar a capacidade da base de dados em as satisfazer.

***Queries* de consulta:**

a) Contabilizar o número de eventos que foram realizados num ano:

Para que possamos realizar esta *querie* de forma satisfatória e para todos os anos que forem pedidos, recorremos a um procedimento (*stored procedure)*, de modo que, dado um determinado ano, o mesmo possa fazer a *querie* pretendida para esse mesmo ano.

Para criar um procedimento foi utilizada uma sintaxe específica. Primeiro, foi delimitado o procedimento com dois símbolos iguais à escolha. Por conveniência e para evitar possíveis conflitos, foi selecionado que para este trabalho que os símbolos vão ser "$$". Começamos por colocar o comando *"DELIMITER $$"* .Após este comando, foi assinalado que vai ser criado um procedimento, e todo o código inerente à *querie*, ficou contido dentro do mesmo. Como o procedimento recebe um ano como argumento de entrada, o mesmo foi identificado com a sintaxe correta. O código completo do procedimento é o seguinte.

```
DELIMITER $$
CREATE PROCEDURE spContaEventosAno(
    IN Ano INT
)
BEGIN
    SELECT COUNT(*) AS NumEventosRealizados
    FROM Evento
    WHERE YEAR(DATAINICIO) = Ano;
END $$
```

```
CALL spContaEventosAno(2024);
```

Resultado

	NumEventosRealizados
▶	2

Figura 41 - Realização da *querie*, que contabiliza o número de eventos realizados num ano

b) Obter o número de bilhetes vendidos por evento, ordenados por quantidade de bilhetes vendidos

Para responder a esta interrogação na linguagem *SQL*, foram seguidos alguns passos. Primeiramente, foi selecionada a coluna "IdEvento" e a soma de todas as quantidades de cada evento, a partir da tabela "BilhetesVendidos". De seguida, foram agrupadas todos as somas por evento, com recurso ao comando "*GROUP BY*". Este comando permite que não haja linhas repetidas com informações redundantes do mesmo evento. Por último foi utilizado o comando "*ORDER BY*" para ordenar todos os registos do total de bilhetes vendidos por ordem decrescente.

```
SELECT IdEvento, SUM(Quantidade) AS TotalBilhetesVendidos
    FROM BilhetesVendidos
    GROUP BY (IdEvento)
    ORDER BY (TotalBilhetesVendidos) DESC;
```

Resultado

	IdEvento	TotalBilhetesVendidos
▶	1	10
	3	5

Figura 42 - Realização da *querie*, que devolve a quantidade de bilhetes vendidos para um evento

c) Calcular o total obtido por evento, ordenado pelo valor total em ordem crescente

Da forma como está desenhada a nossa base de dados, esta interrogação torna-se bastante fácil de se realizar. Como existe um campo "ValorTotal" na nossa tabela "Eventos" que é atualizado, a partir de uma *stored procedure*, que é responsável por adicionar bilhetes comprados à nossa base de dados. Muito resumidamente, o procedimento incrementa a este campo o valor dos bilhetes multiplicado pela quantidade que foi comprada dos mesmos. Então, para realizar a *querie*, basta selecionar as colunas "IdEvento" para identificar qual é o evento, e a coluna "ValorTotal" para exibir o total obtido pelo respetivo evento. Por fim foi utilizado um comando de ordenação, para ordenar os resultados por ordem decrescente.

```
SELECT IdEvento, ValorTotal FROM Evento
    ORDER BY(ValorTotal) DESC;
```

Resultado

	IdEvento	ValorTotal
▶	1	350
	3	250
	2	0
*	NULL	NULL

Figura 43 - Realização da querie que calcula o total obtido por evento

d) Selecionar os agentes que agenciam mais do que 2 artistas e quem agenciam

Para realizar esta interrogação recorremos ao uso de *sub-queries*, que muito resumidamente consistem em executar uma *querie*, e após a execução da mesma, utilizar o resultado para realizar a segunda *querie*. Primeiramente foi executada a *querie* interior, que filtra dos agentes, aqueles que agenciam mais do que um artista. Foi realizada uma operação de junção de tabelas, com base nos atributos "IdAgente" da tabela "Agente" e "IdAgente" da tabela "Artista". Feito isto temos acesso aos artistas que são agenciados pelo um agente específico. Após isso, os resultados foram agrupados por agente, e realizada uma operação de filtragem, para selecionar apenas os agentes que têm 2 ou mais artistas agenciados. Na *querie* exterior, foram projetados os nomes do agente e do artista respetivamente, e foi usada novamente uma operação de junção para selecionar os artistas que são agenciados por um determinado agente.

```
SELECT A.Nome AS NomeAgente, AR.Nome AS NomeArtista
    FROM Agente AS A
    INNER JOIN Artista AS AR
        ON A.IdAgente = AR.IdAgente
    WHERE A.IdAgente IN (
        SELECT A.IdAgente
        FROM Agente AS A
        INNER JOIN Artista AS AR
            ON A.IdAgente = AR.IdAgente
        GROUP BY A.IdAgente
        HAVING COUNT(AR.IdAgente) > 2
)
```

Resultado

	NomeAgente	NomeArtista
▶	Haley Rollins	Foo Fighters
	Haley Rollins	Nirvana
	Haley Rollins	Madonna
	Tiago Rodrigues	Carlos do Carmo
	Ana Costa	Mão Morta
	Ana Costa	Xutos & Pontapés
	Ana Costa	Gisela João
	Ana Costa	Capitão Fausto
	Tiago Rodrigues	HMB
	Tiago Rodrigues	The Gift

Figura 44 - Realização da querie que indica quais os agentes que agenciam 2 ou mais artistas

e) Fazer um relatório diário da receita gerada por cada evento. (por ano)

Para realizar esta interrogação recorremos a um recurso do *MySQL* chamado *Event Scheduler*, que permite agendar comandos *SQL*, ou uma *querie* neste caso, para um horário em específico e em repetição. Como pretendíamos obter um relatório diário da receita gerada, foi criado um *scheduler*, que todos os dias à meia-noite, adiciona numa tabela "RelatorioDiário", que foi criada, o relatório diário pretendido com informações da data, a quantidade de bilhetes vendidos para cada evento e o valor faturado.

```
DELIMITER $$
CREATE EVENT RelatorioDiario
    ON SCHEDULE
        EVERY 1 DAY
        STARTS CURRENT_DATE + INTERVAL 1 DAY
    DO
    BEGIN
        DECLARE dataVenda DATE;
        SET dataVenda = CURDATE();

        INSERT INTO RelatorioDiario(IdRelatorio,Data,IdEvento,BilhetesVendidos,QuantidadeBilhetes,ValorFaturado)
        SELECT
            NULL,
            dataVenda,
            IdEvento,
            COUNT(IdBilheteVendido) AS BilhetesVendidos,
            SUM(Quantidade) AS QuantidadeBilhetes,
            SUM(Quantidade * PreçoVenda) AS ValorFaturado
        FROM BilhetesVendidos
        WHERE DATE(DataVenda) = dataVenda
        GROUP BY IdEvento;

END;
$$
```

Figura 45 - Realização de uma *querie* que faz um relatório diário com recurso a funcionalidade de *scheduler*

f) Selecionar todos os eventos gratuitos

Esta *querie* é muito simples, mas cumpre um dos requisitos de manipulação, que diz que é possível filtrar por eventos pagos e gratuitos. De modo análogo podemos realizar esta

```
SELECT IdEvento, Nome FROM Evento
    WHERE Pago = false
```

Resultado

IdEvento	Nome
2	Mes Gastronómico em Braga
NULL	NULL

interrogação

para o caso de pretendermos selecionar todos os eventos pagos, sendo apenas necessário trocar o valor de "Pago" para "`True`"

Figura 46 - *Querie* que seleciona todos os eventos gratuitos

g) Selecionar os eventos grátis que tenham atividades pagas

De certo modo, esta interrogação é semelhante à anterior, mas mais uma vez cumpre um dos requisitos de descrição e manipulação, que dizem respetivamente, que existem eventos gratuitos com atividades pagas, e que é possível visualizar quais são, e quais são as respetivas atividades. A *querie* começa com a seleção das colunas "IdEvento" e "Nome" da tabela "Evento", da coluna "nome" da tabela "Atividade" e da coluna" Preço" da tabela "Bilhete". De seguida, e de modo a conseguir recolher estes dados realiza duas operações de junção interligadas para aceder à tabela "Atividade" e de seguida à tabela "Bilhete". Por fim escolhe os resultados onde para onde a coluna "Pago" nas tabelas "Evento" e "Atividade", tem valores "`false`" e "`true`" respetivamente, ou seja, os valores onde os eventos são gratuitos e as atividades pagas.

```
SELECT E.IdEvento, E.Nome AS Evento, A.Nome AS Atividade, B.Preço AS Preco
    FROM Evento AS E
    INNER JOIN Atividade AS A
        ON E.IdEvento = A.IdEvento
        INNER JOIN Bilhete AS B
        ON A.IdAtividade = B.IdAtividade
    WHERE A.Pago = true AND E.Pago = false
```

Resultado

	IdEvento	Evento	Atividade	Preco
▶	2	Mes Gastronómico em Braga	Festa da Francesinha	12.50

Figura 47 - Realização da *querie* que seleciona todos os eventos gratuitos, mas que tenham atividades pagas

5.3. Definição e caracterização das vistas de utilização em SQL

Durante a definição de requisitos, nomeadamente na parte dos requisitos de controlo, foram definidas uma série de "regras" para a parte de controlo da nossa base de dados. Por outras palavras, foram definidas horas de operação da nossa base de dados, quem podia trabalhar na mesma, assim como as permissões que os utilizadores teriam. Desta forma, e a partir dos nossos requisitos, foram desenvolvidas uma série de instruções *SQL*, para implementar os mesmos.
Antes de ser desenvolvido o script de permissões para os utilizadores, foram criados os utilizadores para a base de dados.
Primeiro foi criado um utilizador *admin*, que é o responsável por administrar e fazer a manutenção da base de dados.

```
-- Criação de um utilizador da base de dados para o administrador
CREATE USER 'admin'@'localhost';
SET PASSWORD FOR 'admin'@'localhost' = 'admin';
```

De seguida, foi criado um utilizador comum para o grupo Bracara Eventos.

```
-- Criação de um utilizador da base de dados para o grupo Bracara Eventos
CREATE USER 'bracaraevento'@'localhost';
SET PASSWORD FOR 'bracaraevento'@'localhost' = 'bracara';
```

Por último, foi criado um utilizador para representar um utilizador diário da nossa base de dados.

```
CREATE USER 'genericuser'@'localhost';
SET PASSWORD FOR 'genericuser'@'localhost' = 'user';
```

Com os utilizadores criados, prosseguiu-se com a definição dos utilizadores que cada utilizador tem na base de dados.

Requisitos de controlo:

RC2- "O sistema poderá ser acedido pelo grupo Bracara Eventos de forma, forma parcial, e de forma total pelo gestor do sistema de bases de dados. O sistema poderá ser acedido pelos utilizadores para apenas verem detalhes dos eventos e das atividades. O grupo apenas tem acesso a operações de consulta nas tabelas da base de dados."

Para este efeito, foram dadas permissões de consulta e visualização de dados ao utilizador "bracaraevento", para todas as tabelas da nossa base de dados.

```
-- Definição de alguns previlégios para o utilizador 'bracaraevento'.
-- Permissão para a execução de instruções SELECT
GRANT SELECT ON  Eventos.* TO 'bracaraevento'@'localhost';
```

RC4- "O administrador da base de dados tem permissão total para toda a base de dados."

Para cumprir este requisito, foram concedidas todas as permissões possíveis para o administrador na base de dados "Eventos".

```
GRANT ALL ON Eventos.* TO 'admin'@'localhost';
```

RC2- """O sistema poderá ser acedido pelo grupo Bracara Eventos de forma, forma parcial, e de forma total pelo gestor do sistema de bases de dados. O sistema poderá ser acedido pelos utilizadores para apenas verem detalhes dos eventos e das atividades. O grupo apenas tem acesso a operações de consulta nas tabelas da base de dados."

Para cumprir este requisito foram criadas duas vistas de utilização de dados, para exibir detalhes dos eventos e das atividades. A vista "viewEvento" expõe apenas os detalhes necessários dos eventos, e a vista "viewAtividade" mostra apenas os detalhes necessários das atividades.

```
CREATE VIEW viewAtividade AS
    SELECT
        A.Nome AS Nome,
        A.Descrição AS Descrição,
        A.Pago AS éPaga,
        CONCAT(A.Rua, ',', A.CodPostal, ',', A.DescriçãoLocal) AS Morada,
        A.Data AS Data,
        A.Lotação AS LotaçãoMáxima,
        AR.Nome AS NomeArtista,
        E.Nome AS NomeEvento,
        B.Preço AS PreçoAtividade
    FROM Atividade AS A
    LEFT JOIN Artista AS AR
        ON A.IdArtista = AR.IdArtista
    LEFT JOIN Evento AS E
        ON A.IdEvento = E.IdEvento
    LEFT JOIN Bilhete AS B
        ON A.IdAtividade = B.IdAtividade;
```

```
CREATE VIEW viewEvento AS
    SELECT
        E.Nome AS Nome,
        E.Descrição AS Descrição,
        E.Pago AS éPaga,
        CONCAT(E.Rua, ',', E.CodPostal, ',', E.DescriçãoLocal) AS Morada,
        E.DataInicio AS Data
    FROM Evento AS E;
```

Figura 48 - Criação de duas vistas de utilização

Com as duas vistas implementadas, foram dadas as permissões necessárias para os utilizadores da base de dados conseguirem visualizar estes dados.

```
GRANT SELECT ON Eventos.viewatividade TO 'genericuser'@'localhost';
GRANT SELECT ON Eventos.viewevento TO 'genericuser'@'localhost';
```

5.4. Cálculo do espaço da base de dados

Tendo em conta os nossos conhecimentos prévios e pesquisas relativamente ao espaço que cada tipo de dado ocupa, foi calculado o espaço ocupado por cada atributo pertencente a cada tabela, de forma a estimar qual será a ocupação em espaço da nossa base de dados. Para ajudar no cálculo, foi assumido o tipo *TEXT* como sendo um tipo *VARCHAR (300)*.

Evento

Atributos	Tipo	Tamanho (Bytes)
IdEvento	INTEGER	4
Nome	VARCHAR(50)	52 (50+2)
Descrição	TEXT	302
CustoEvento	DECIMAL(8,2)	5 (4+1)
Pago	BOOLEAN	1
DescriçãoLocal	VARCHAR(100)	102
Rua	VARCHAR (75)	77
CódigoPostal	VARCHAR (20)	22
DataInicio	DATETIME	8
DataFim	DATETIME	8
ValorTotal	INTEGER	4
QuantidadeBilhetesVendidos	INTEGER	4
Preço	DECIMAL(8,2)	5
TOTAL	-	**594**

Tabela 8 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Evento"

Atividade

Atributos	Tipo	Tamanho (Bytes)
IdAtividade	INTEGER	4
Nome	VARCHAR(50)	52
Descrição	TEXT	302
Pago	BOOLEAN	1
CodPostal	VARCHAR(50)	52
DescriçãoLocal	VARCHAR(80)	82
Rua	VARCHAR(70)	72
Data	DATETIME	8

Lotação	INTEGER	4
IdEvento	**INTEGER**	**4**
IdArtista	INTEGER	4
ValorTotal	INTEGER	4
QuantidadeBilhetesVendidos	INTEGER	4
BilhetesDisponiveis	INTEGER	4
Preço	DECIMAL(8,2)	5
CustoAtividade	DECIMAL(8,2)	5
TOTAL	-	**607**

Tabela 9 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Atividade"

Artista

Atributos	**Tipo**	**Tamanho (Bytes)**
IdArtista	INTEGER	4
Nome	VARCHAR(70)	72
Descrição	TEXT	302
IdAgente	INTEGER	4
CustoArtista	DECIMAL(8,2)	5
TOTAL	**--------**	**387**

Tabela 10 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Artista"

Agente

Atributos	**Tipo**	**Tamanho (Bytes)**
IdAgente	INTEGER	4
Nome	VARCHAR(70)	72
TelefoneAgente	VARCHAR(20)	22
EmailAgente	VARCHAR(70)	72
TOTAL	**------**	**170**

Tabela 11 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Agente"

Staff

Atributos	**Tipo**	**Tamanho (Bytes)**
IdStaff	INTEGER	4
Nome	VARCHAR(70)	72
Função	VARCHAR(20)	22
Telefone	VARCHAR(70)	72
TOTAL	**------**	**170**

Tabela 12 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Staff"

Bilhete

Atributos	Tipo	Tamanho (Bytes)
NumBilhete	INTEGER	4
Preço	DECIMAL(8,2)	5
IdEvento	INTEGER	4
IdAtividade	INTEGER	4
TOTAL	**------**	**17**

Tabela 13 - Tabela que determina o tamanho de cada tipo de dados para a tabela "Bilhete"

StaffEvento

Atributos	Tipo	Tamanho (Bytes)
Staff	INTEGER	4
Evento	INTEGER	4
TOTAL	------	8

Tabela 14 - Tabela que determina o tamanho de cada tipo de dados para a tabela "StaffEvento"

BilhetesVendidos

Atributos	Tipo	Tamanho (Bytes)
IdBilheteVendido	INTEGER	4
IdEvento	INTEGER	4
IdAtividade	INTEGER	4
Quantidade	INTEGER	4
DataVenda	DATETIME	8
PreçoVenda	DECIMAL(8,2)	5
IdBilhete	INTEGER	4
TOTAL	------	33

Tabela 15 - Tabela que determina o tamanho de cada tipo de dados para a tabela "BilhetesEventos"

Sem povoamento, o tamanho total ocupado pela nossa base de dados é 594+607+387+170+170+17+8+33 = 1986 *bytes* ≈ 1,9 kB

Com o povoamento inicial a nossa base de dados ocupa:

594*8 + 607*51 + 387*61 + 170*70 + 170*7 + 17*15 + 8*7 + 33*100 (este valor é alterado regularmente por isso vamos supor 100) = 75 847 *bytes* ≈ 74,1 kB. Estes valores são uma estimativa sobre o primeiro mês de utilização. Se contarmos com um acréscimo de 1000% em relação aos bilhetes vendidos vamos ter um número de entradas na tabela “BilhetesVendidos” a rondar a casa dos 100*1000 = 100 000 bilhetes vendidos. Se contarmos com um acréscimo de 50 % nos valores de atividades e eventos, teríamos a cada mês um total de mais 4 eventos e 20 atividades. No fim de um ano, são 48 eventos e 240 atividades. Podemos contar com um acréscimo de 20% nos artistas e nos respetivos agentes, o que dará um total de 73 artistas e 84 agentes registados na base de dados. Ora com estes dados, e contando que teremos cerca de 100 elementos do staff registados para ajudar nos eventos, a nossa base de dados ao fim de um

ano ocuparia: 56 * 594 + 607 * 291 + 387 * 73 + 170 * 84 + 170*100 + 17* 100 (100 tipos de bilhetes) + 8*100 + 33 * 100000 = 3 571 932 *bytes* ≈ 3 488,2 kB ≈ 3,4 MB

Precisamos de ter em conta que esta é meramente uma estimativa do tamanho de ocupação da base de dados, com base em cálculos bastante rudimentares.

5.5. Indexação do Sistema de Dados

A vantagem de definir os nossos próprios índices na base de dados, em adição aos já criados automaticamente é que estes aceleram operações de consulta e seleção. Devemos apenas criar índices para tabelas que são consultadas com elevada frequência. Devemos também ter em atenção a quantidade de entradas nas tabelas, pois quantas mais entradas uma tabela tiver, mais eficiente será o índice. Podemos desde já então excluir índices para as tabelas “Staff” e “Agente”.
Na tabela “BilhetesVendidos”, como são efetuadas muitas operações de consulta e seleção, e irá ter inevitavelmente muitas entradas, é uma boa candidata a ter índices. Portanto foi criado um índice para a data de venda dos bilhetes. Pode ser útil procurar os bilhetes vendidos numa dada data.

```
CREATE INDEX DataVenda ON BilhetesVendidos (DataVenda);
```

Figura 49 - Criação do índice "DataVenda"

A certo ponto da nossa base de dados, a tabela “Eventos” também poderá ser afetada com o elevado número de entradas na mesma. Pode ser útil fazer pesquisas por nome e data de começo de um evento, e por isso foram criados estes dois índices.

```
CREATE INDEX NomeEvento ON Evento(Nome);
CREATE INDEX DataInicioEvento ON Evento(DataInicio);
```

Figura 50 - Criação dos índices "NomeEvento" e "DataInicioEvento"

Do mesmo modo podemos, e a pensar no crescimento da nossa base de dados, criar dois índices iguais aos anteriores, mas desta vez para a tabela “Atividade”.

```
CREATE INDEX NomeAtividade ON Atividade(Nome);
CREATE INDEX DataInicioAtividade ON Atividade(Data);
```

Figura 51 - Criação dos índices "NomeAtividade" e "DataInicioAtividade"

5.6. Procedimentos Implementados

De forma a conseguir simular a venda de bilhetes, quer para eventos, quer para atividades de eventos, decidimos implementar procedimentos com operações de transações para inserir a quantidades de bilhetes para um certo identificador de evento ou de atividade à tabela “BilhetesVendidos”. À medida que desenvolvemos estes procedimentos, foram realizadas algumas alterações às tabelas “Eventos” e “Atividades”, que já foram mostradas anteriormente no trabalho, de modo a aportar informações relevantes, como por exemplo o valor total que cada evento ou atividade arrecadou com a venda dos bilhetes, a quantidade de bilhetes disponíveis e até mesmo o preço dos bilhetes para as respetivas tabelas. De modo a explicar os procedimentos da melhor forma vamos por etapas tentar esclarecer o raciocínio por detrás da lógica dos mesmos.

Os procedimentos são os seguintes:

spVendaBilhetesEventos

```
DELIMITER $$
CREATE PROCEDURE spVendaBilhetesEventos(
    IN QuantidadeBilhetes INTEGER,   -- Argumento de entrada relativa à quantidade de bilhetes a comprar.
    IN Evento_Id INTEGER,            -- Argumento de entrada que diz qual o evento que queremos comprar o bilhete.
    OUT Resultado VARCHAR(150)       -- Argumento de saída com informações relativas ao controlo de operações durante esta transação.
)
AtualizarBilhetesEvento:BEGIN
    -- Declaração de variáveis locais
    DECLARE idx INT DEFAULT 0;          -- Varíavel de iteração para o ciclo While.
    DECLARE PreçoBilhete DECIMAL(8,2);  -- Varíavel que guarda o preço do bilhete relativo ao evento selecionado.
    DECLARE EventoPago BOOLEAN;         -- Varíavel que guarda a informação que permite saber se um evento é pago ou não

    -- Declaração de variáveis de controlo e de um handler para deteção da ocorrência de exceções SQL
    DECLARE vErro INT DEFAULT 0;
    DECLARE CONTINUE HANDLER
        FOR SQLEXCEPTION
            SET vErro = 1;
    -- Inicio da transação da venda
    START TRANSACTION;
    -- Guarda na variável EventoPago a informação que diz se um evento é ou não pago
    SELECT Pago INTO EventoPago
    FROM Evento
    WHERE IdEvento = Evento_Id;

    -- Condição que testa se o evento é pago (=1) ou não (=0)
    IF EventoPago = 0 THEN  -- **(Quando EventoPago = 0)**Neste caso atualizamos resultado com uma mensagem de saída e fazemos rollback de todas as operações feitas até ao momento
        SET Resultado = 'Este evento é gratuito. Não é possível adicionar bilhetes.';
        ROLLBACK;
    ELSE
    -- **(Quando EventoPago = 1)**Neste caso é guardado o preço na variável PreçoBilhete, preço este que é retirado a partir da tabela "Bilhete"
        SELECT Preço INTO PreçoBilhete
        FROM Bilhete
        WHERE IdEvento = Evento_Id
```

Figura 52 - Excerto procedimento "spVendaBilhetesEventos"

e **spVendaBilhetesAtividades**

```
DELIMITER $$
CREATE PROCEDURE spVendaBilhetesAtividades(
    IN QuantidadeBilhetes INTEGER,
    IN Atividade_Id INTEGER,
    OUT Resultado VARCHAR(150)
)
AtualizarBilhetesAtividades:BEGIN
    DECLARE idx INT DEFAULT 0;
    DECLARE PreçoBilhete DECIMAL(8,2);
    DECLARE AtividadePaga BOOLEAN;
    DECLARE Evento_Id INT; -- Declare uma variável para armazenar o IdEvento

    -- Declaração de variável de controle de erro
    DECLARE vErro INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        SET vErro = 1;

    START TRANSACTION;
    -- Obtenha o IdEvento relacionado à IdAtividade
    SELECT IdEvento INTO Evento_Id
    FROM Atividade
    WHERE IdAtividade = Atividade_Id;

    -- Verifica se a atividade é paga ou gratuita
    SELECT Pago INTO AtividadePaga
    FROM Atividade
    WHERE IdAtividade = Atividade_Id;

    -- Se a atividade for gratuita, exibe uma mensagem de erro e sai do procedimento
    IF AtividadePaga = 0 THEN
        SET Resultado = 'Esta atividade é gratuita. Não é possível adicionar bilhetes.';
```

Figura 53 - Excerto procedimento "spVendaBilhetesAtividades"

Estes são apenas excertos do código dos mesmos, uma vez que seria impraticável ter o código neste documento. O mesmo está disponível na pasta “Scripts SQL” com o nome 202324-UM-LCC-BD-G04-StoredProcedures.sql.

Começando pelo primeiro procedimento.

```
DELIMITER $$
CREATE PROCEDURE spVendaBilhetesAtividades(
    IN QuantidadeBilhetes INTEGER,
    IN Atividade_Id INTEGER,
    OUT Resultado VARCHAR(150)
)
AtualizarBilhetesAtividades:BEGIN
    DECLARE idx INT DEFAULT 0;
    DECLARE PreçoBilhete DECIMAL(8,2);
    DECLARE AtividadePaga BOOLEAN;
    DECLARE Evento_Id INT; -- Declare uma variável para armazenar o IdEvento

    -- Declaração de variável de controle de erro
    DECLARE vErro INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        SET vErro = 1;
```

Foi declarada a assinatura do procedimento com a definição dos argumentos de entrada e saída do procedimento. A seguir a isso, foram declaradas uma série de variáveis que vão ser utilizadas ao longo do mesmo, assim como variáveis de controlo de erros.

```
START TRANSACTION;
-- Obtenha o IdEvento relacionado à IdAtividade
SELECT IdEvento INTO Evento_Id
FROM Atividade
WHERE IdAtividade = Atividade_Id;

-- Verifica se a atividade é paga ou gratuita
SELECT Pago INTO AtividadePaga
FROM Atividade
WHERE IdAtividade = Atividade_Id;
```

De seguida foi indicado que a transação começa neste ponto. Foi utilizada uma transação para assegurar a integridade dos dados que vão ser introduzidos. Na vertente de vendas é comumente utilizado este recurso por estas utilizarem o padrão ACID, ou seja, garantem a atomicidade de dados, a consistência, o isolamento e durabilidade.

Após este passo, foram atribuídas às variáveis anteriormente criadas informações relativas ao identificador do evento e se o mesmo é pago ou não.

```
IF EventoPago = 0 THEN  -- **(Quando EventoPago = 0)**Neste caso atualizamos resultado com uma mensagem de saída e fazemos rollback de todas as operações feitas até ao momento
    SET Resultado = 'Este evento é gratuito. Não é possível adicionar bilhetes.';
    ROLLBACK;
ELSE
-- **(Quando EventoPago = 1)**Neste caso é guardado o preço na variável PreçoBilhete, preço este que é retirado a partir da tabela "Bilhete"
    SELECT Preço INTO PreçoBilhete
    FROM Bilhete
    WHERE IdEvento = Evento_Id
    LIMIT 1;
    -- Atualização da tabela "Evento" já com as informação dos bilhetes que foram vendidos
    UPDATE Evento
    SET ValorTotal = IFNULL(ValorTotal, 0) + (PreçoBilhete * QuantidadeBilhetes),
        QuantidadeBilhetesVendidos = IFNULL(QuantidadeBilhetesVendidos, 0) + QuantidadeBilhetes,
        Preço = PreçoBilhete
    WHERE IdEvento = Evento_Id;
    -- Atualização da tabela "Atividade" já com as informação dos bilhetes que foram vendidos
    UPDATE Atividade
    SET ValorTotal = IFNULL(ValorTotal, 0) + (PreçoBilhete * QuantidadeBilhetes),
        QuantidadeBilhetesVendidos = IFNULL(QuantidadeBilhetesVendidos, 0) + QuantidadeBilhetes,
        BilhetesDisponiveis = Lotação - QuantidadeBilhetesVendidos
    WHERE IdEvento = Evento_Id;
    -- Ciclo que a cada iteração insere na tabela "BilhetesVendidos" informações relativas a cada bilhete
    -- Enquanto que o iterador for menor do que a quantidade de bilhetes pretendida, é adicionado a tabela um bilhete de cada vez.
    WHILE idx < QuantidadeBilhetes DO
        INSERT INTO BilhetesVendidos (IdEvento,Quantidade,DataVenda,PreçoVenda)
        VALUES (Evento_Id,1,NOW(),PreçoBilhete);
        SET idx = idx + 1;
    END WHILE;
```

De seguida, é verificado se o evento não é pago. Se não o for, é apresentada uma mensagem de erro e feito *rollback* às operações até aí realizadas, ou seja, desfaz as operações. Se o evento for pago, é prosseguida a venda dos bilhetes. É atualizada a tabela “Evento”, de modo que a quantidade de bilhetes vendidos aumente e o preço na tabela seja atualizado para o preço do bilhete correspondente ao evento. É também atualizada a tabela “Atividade”, pois comprando bilhete para um evento, todas as atividades ficam pagas. Nesta tabela é atualizado também o campo de bilhetes disponíveis, que decresce com a quantidade de bilhetes vendidos. A forma que arranjamos para inserir, um a um os bilhetes na tabela “BilhetesVendidos”, foi criar um ciclo que vai começar em 0 e vai um a um até ao número de bilhetes pretendidos para venda. A cada iteração deste ciclo são adicionadas as informações à tabela “BilhetesVendidos”.

Esta pode não ser a melhor forma de fazer isto pois se a quantidade de bilhetes de uma só vez for muito elevada perdemos performance e eficácia na nossa base de dados.

```
        -- Interrupção da transação da venda
        IF vErro = 1 THEN
            SET Resultado = 'Transação abortada. Ocorreu um erro durante as operações.';
            ROLLBACK;
            LEAVE AtualizarBilhetesEvento;
        ELSE
            COMMIT; -- Confirmação da transação
            SET Resultado = 'Transação concluída com sucesso!';
        END IF;
    END IF;
END$$
```

Se não existirem mais erros é feita a confirmação da transação e mostrado o resultado com mensagem de sucesso.
A explicação do procedimento para a venda de bilhetes para as atividades é análoga à anterior.

Foram também desenvolvidos procedimentos para realizar a limpeza da base de dados, tendo estes que ser utilizados de forma responsável e segura. Como apenas o administrador tem acesso a estes procedimentos, à partida não existe qualquer problema. Os procedimentos são:

```
DELIMITER $$
-- REINICIA A CONTAGEM DA QUANTIDADE E DO VALOR TOTAL
CREATE PROCEDURE spResetValoresEventos(
    IN Evento_Id INTEGER
)
BEGIN
    UPDATE Evento
    SET ValorTotal = 0,
        QuantidadeBilhetesVendidos = 0,
        Preço = 0
    WHERE IdEvento = Evento_Id;
END$$
```

Figura 54 - Excerto procedimento "spResetValoresEventos"

Que dado um identificador de evento, faz um *reset* (coloca os valores pretendidos a 0) aos valores da tabela do mesmo.

```
DELIMITER $$
-- REINICIA A CONTAGEM DA QUANTIDADE E DO VALOR TOTAL
CREATE PROCEDURE spResetValoresAtividades(
    IN Atividade_Id INTEGER
)
BEGIN
    SET SQL_SAFE_UPDATES = 0;
    UPDATE Atividade
    SET ValorTotal = 0,
        QuantidadeBilhetesVendidos = 0,
        Preço = 0,
        BilhetesDisponiveis = Lotação
    WHERE IdAtividade = IdAtividade;
    SET SQL_SAFE_UPDATES = 1;
END$$
```

Figura 55 - Excerto procedimento "spResetValoresAtividades"

Que dado um identificador de atividade, faz um *reset* aos valores da tabela da mesma.
Estes dois últimos procedimentos, embora possam ser executados individualmente, foram desenvolvidos para serem chamados por um procedimento “spResetAll”, que como o nome indica faz uma limpeza geral. Faz um *reset* aos valores das tabelas “Evento” e “Atividade” e apaga todas as entradas da tabela “BilhetesVendidos”. Este procedimento recebe dois argumentos. O ID do evento até onde queremos que sejam reinicializados os valores e o mesmo para o ID da atividade. São utilizados dois ciclos, um para percorrer desde 0 até ao número do último ID do evento, e a cada iteração, executa uma vez o procedimento “spResetValoresEvento” para o ID em que a iteração está, e o mesmo se verifica para o ciclo responsável por chamar o procedimento “spResetValoresAtividades”. Por fim apaga da tabela “BilhetesVendidos” todas os registos.

```
DELIMITER $$
CREATE PROCEDURE spResetAll(
    IN LastEvento INTEGER,
    IN LastAtividade INTEGER
)
BEGIN
    DECLARE idx INT DEFAULT 1;
    SET SQL_SAFE_UPDATES = 0;
    -- Reseta os valores dos eventos de 1 a LastEvento
    WHILE idx <= LastEvento DO
        CALL spResetValoresEventos(idx);
        SET idx = idx + 1;
    END WHILE;

    SET idx = 1; -- Reinicializa o contador

    -- Reseta os valores das atividades de 1 a LastAtividade
    WHILE idx <= LastAtividade DO
        CALL spResetValoresAtividades(idx);
        SET idx = idx + 1;
    END WHILE;

    -- Remove todas as entradas da tabela BilhetesVendidos
    DELETE FROM BilhetesVendidos WHERE IdBilheteVendido > 0;
    ALTER TABLE BilhetesVendidos AUTO_INCREMENT = 1;
    SET SQL_SAFE_UPDATES = 1;
END$$
```

Figura 56 - Excerto procedimento "spResetAll"

Posteriormente foram criados também dois procedimentos: spTop5Receitas e spListaEventos, que pretendem responder a requisitos de manipulação recolhidos anteriormente.

A cada momento é possível aceder a uma listagem dos eventos que se realizam num mês.
A cada momento é possível aceder a uma listagem dos 5 eventos que mais faturaram.

A partir destes requisitos resultaram os seguintes procedimentos:

```
DELIMITER $$
CREATE PROCEDURE spTop5Receitas()
    BEGIN
        SELECT IdEvento, Nome , QuantidadeBilhetesVendidos, ValorTotal
        FROM Evento
        ORDER BY(ValorTotal) DESC
        LIMIT 5;
    END$$
```

IdEvento	Nome	QuantidadeBilhetesVendidos	ValorTotal
1	Star Wars: O Concerto	15	1600
3	Braga Authentica	5	325
4	Xmas Night Music	5	250
2	Mes Gastronómico em Braga	0	0
5	Hopen – Braga Beer Festival	0	0

Figura 57 - Excerto procedimento "spTop5Receitas"

```
DELIMITER $$
CREATE PROCEDURE spListaEventos(
    IN Mes INTEGER -- 1 corresponde a Janeiro, 2 a Fevereiro, etc...
)

BEGIN
        SELECT
            E.Nome AS NomeEvento,
            E.Descrição AS DescriçãoEvento,
            CONCAT(E.DataInicio, '-', E.DataFim) AS DataEvento,
            A.Nome AS NomeAtividade,
            A.Descrição AS DescriçãoAtividade,
            A.Data AS DataAtividade
        FROM Evento AS E
        INNER JOIN Atividade AS A ON E.IdEvento = A.IdEvento
        WHERE MONTH(E.DataInicio) = Mes;
END$$
CALL spListaEventos(10);
```

```
CALL spListaEventos(10);
```

NomeEvento	DescriçãoEvento	DataEvento	NomeAtividade	DescriçãoAtividade	DataAtividade
Star Wars: O Concerto	Concerto épico sobre este incrível universo	2024-10-11 21:30:00-2024-10-11 21:30:00	Concerto Dark Side, Ato 1	NULL	2024-10-11 21:30:00
Star Wars: O Concerto	Concerto épico sobre este incrível universo	2024-10-11 21:30:00-2024-10-11 21:30:00	Concerto Dark Side, Ato 2	NULL	2024-10-11 22:30:00
Star Wars: O Concerto	Concerto épico sobre este incrível universo	2024-10-11 21:30:00-2024-10-11 21:30:00	May the Force be with you, Ato Final	NULL	2024-10-11 23:30:00

Figura 58 -Excerto procedimento "spListaEventos"

Foram também criados dois *triggers* e uma função. Os *triggers* são utilizados para atualizar o custo de um evento, sempre que é adicionada uma atividade correspondente ao mesmo e para não permitir a inserção de datas incorretas na tabela evento.

```
DELIMITER $$
CREATE TRIGGER trAtualizaCustoEvento
    AFTER INSERT ON Atividade
    FOR EACH ROW
BEGIN
        UPDATE Evento
        SET CustoEvento = CustoEvento + NEW.CustoEvento
        WHERE IdEvento = NEW.IdEvento;
END $$


DELIMITER $$
CREATE TRIGGER trAsseguraDatas
    BEFORE INSERT ON Evento
    FOR EACH ROW
    BEGIN
        IF NEW.DataInicio > NEW.DataFim THEN
            SIGNAL SQLSTATE '45000' -- MENSAGEM DE ERRO PREDIFINIDA EM SQL
            SET MESSAGE_TEXT = "Data Inválida";
        END IF;
    END$$
```

Figura 59- Excerto da implementação dos gatilhos "trAtualizaCustoEvento" e "trAsseguraDatas"

A função desenvolvida recebe um recebe um ano, e devolve o total de gastos por ano, de todos os eventos juntos.

```
DELIMITER $$
CREATE FUNCTION fuTotalGastoAno(Ano INTEGER)
    RETURNS DECIMAL(10,2)
    DETERMINISTIC
BEGIN
    DECLARE totalGasto DECIMAL(10,2);
    SELECT
        SUM(CustoEvento)
    INTO totalGasto
    FROM Evento AS E
    WHERE YEAR(DataInicio) = Ano;
    RETURN totalGasto;
END$$

SELECT fuTotalGastoAno(2023)
SELECT CustoEvento FROM Evento
```

Figura 60 – Excerto da função "fuTotalGastoAno"

5.7. Plano de segurança e recuperação de dados

A implementação de um plano é um passo tão importante quanto a implementação bem-sucedida de uma base de dados. Por isso, foi decidido fazer este plano para garantir a segurança e integridade dos dados armazenados.

O primeiro passo deste plano é assegurar a segurança dos dados e caso o primeiro passo falhe, assegurar a recuperação dos dados. O plano de segurança dos dados passa por estes pontos:

1. **Realização de backups regulares**: Serão realizados backups diários em horários de menor tráfico de dados na nossa base de dados em mais do que 3 localizações, para reduzir o risco da perda total dos dados em caso de uma ocorrência inesperada. As 3 localizações serão: um servidor privado para armazenar a base de dados e os seus backups, um computador na posse do administrador da base de dados e por último um backup numa plataforma de clouding.
2. **Contratação de um engenheiro de Segurança de Redes**: De forma a prevenir ataques e identificar possíveis ameças à nossa base de dados, será contratado um especialista nesta área que irá implementar medias proativas e reativas, mantendo este plano de segurança o mais atualizado possível.
3. **Controlo do acesso à base de dados**: Foram definidas regras muito específicas sobre quem pode ou não aceder à base de dados e quais os privilégios que têm sobre a mesma.

No caso de uma quebra neste plano, foram implementados mais 2 pontos que permitem a recuperação dos dados que foram comprometidos:

1. **Procedimentos pré-definidos de recuperação dos dados:** No caso de falha do sistema ou perca dos dados, foram definidos uma série de passos a seguir para restaurar os dados a partir dos backups.
2. **Testes de recuperação frequentes:** Vão existir frequentes simulações de perda de dados, para otimizar o processo de restauro dos mesmos.

6. Conclusões e Trabalho Futuro

O presente projeto teve como objetivo a criação de uma base de dados para a gestão de um calendário de eventos. Uma vez realizadas todas as etapas para a produção de uma base de dados completamente operacional, é necessário analisar a prestação de cada etapa, sendo possível retirar conclusões.

Este projeto tem por base a criação de uma "história" para suportar toda a ideia e a necessidade da criação de uma base de dados. Uma vez realizada esta parte, prosseguiu-se para a listagem dos objetivos, análise da viabilidade do projeto e, ainda, a elaboração de um diagrama de GANTT, representando esta última um desafio para a realização deste projeto. A elaboração do mesmo, embora importante para o tipo de trabalhos com prazos, não foi bem-sucedida no decorrer da realização deste trabalho, talvez pela falta de compreensão do funcionamento dos mesmos. É por isso, um ponto de destaque a melhorar para futuros trabalhos.

A segunda etapa a ser realizada foi a criação de um modelo conceptual a partir dos requisitos recolhidos, que viria a servir de base para mais tarde. A criação de entidades e respetivos relacionamentos foi algo pensado e concretizado com sucesso. Considerando agora em retrospetiva, poderia ter sido introduzida uma maior complexidade à base de dados, que se poderia ter traduzido num maior número de ações a realizar com a mesma.

A partir do modelo conceptual foi possível realizar a conversão para o modelo lógico de acordo com regras pré-definidas, e que, portanto, decorreu também com bastante facilidade. A segunda dificuldade surgiu quando ao validar o modelo, foi necessária a utilização do software “RelaX” para a criação de árvores de álgebras relacionais. Apesar das inúmeras tentativas para a criação destas árvores, o software apresentava erros incoerentes, provavelmente pela inexperiência dos elementos do grupo na utilização da plataforma. Por este motivo, surgiu a necessidade de reduzir o nível das *queries* à base de dados. Este é o segundo ponto a melhorar em trabalhos futuros.

Por último, foi realizado um *script* de criação da base de dados na linguagem *SQL*. Este último passo tratou de converter para algo funcional tudo o que tinha sido realizado até ao momento. Foi talvez a parte mais entusiasmante do trabalho. A realização de *queries*, procedimentos e todos os métodos que foram implementados permitiu ver a base de dados em ação, tendo sido bastante motivador.

Sem dúvida que o projeto em questão abriu os horizontes a este paradigma da tecnologia e foi importante para perceber as saídas que existem nesta área.

Relativamente a trabalhos futuros, o objetivo é não só melhorar os dois pontos supracitados, como também aprimorar a técnica nesta área. Para além disso, seria interessante fora do contexto desta disciplina, proceder à criação de uma aplicação que confira uma interface visual à nossa base de dados.

Por fim, pode concluir-se que o sistema criado encontra-se apto para o objetivo proposto inicialmente, a gestão de um calendário de eventos.

Referências

- Lucidchart O que é um diagrama entidade relacionamento? Lucidchart. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento.
- Connolly, T., Begg, C., Database Systems: A Practical Approach to Design, Implementation, and Management, Addison-Wesley, Global Edition, 26 Sep 2014. ISBN-10: 1292061189, ISBN-13: 978-1292061184.
- MySQL :: MySQL 8.0 Reference Manual :: 11.7 Data Type Storage Requirements. (2024). Mysql.com. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html
- MySQL :: MySQL 8.0 Reference Manual :: 13.1.13 CREATE EVENT Statement. (2024). Mysql.com. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/create-event.html
- Events in MySQL. (2023, January 17). GeeksforGeeks; GeeksforGeeks. Disponível em: https://www.geeksforgeeks.org/events-in-mysql/
- Belo,O., Sistemas de Bases de Dados, A Mercearia da D. Acácia, Um Caso de Estudo, Universidade do Minho, 2022
- Belo,O., Sistemas de Bases de Dados, A Linguagem SQL, Notas de Leitura, Universidade do Minho, 2022
- Belo, O., Normalização de Dados, Sistemas de Bases de Dados, Notas de Leitura, Universidade do Minho, 2020
- Belo, O., A Álgebra Relacional, Notas de Leitura, Universidade do Minho, 2020

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema Gestão Base de Dados
CMB	Câmara Municipal de Braga
ER	Entidade Relação
NIF	Número de Identificação Fiscal
SQL	*Structured Querie Language*
ID	Identificador
ACID	Atomicidade, Consistência, Isolamento, Durabilidade
MB	Megabyte
kB	Kilobyte