



Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2025/2026

Bela Vista

Débora Caetano, Enrico Prazeres, Gonçalo Fernandes

Dezembro, 2025

BD

Data da Recepção	
Responsável	
Avaliação	
Observações	

Bela Vista

Débora Caetano, Enrico Prazeres, Gonçalo Fernandes

Dezembro, 2025

Resumo

O núcleo de estudantes CeBELUM requistou aos estudantes de engenharia informática da Universidade do Minho o desenvolvimento de uma base de dados relacional para a gestão das suas viagens. Este documento descreve o processo de desenvolvimento conduzido pelo grupo de estudantes.

Numa fase inicial foi justificada ao grupo a necessidade da base de dados e definiu o contexto operacional, reunindo-se com a equipa informática para discussão do plano de execução.

Após a definição do método de levantamento de requisitos, estes foram identificados junto do núcleo e posteriormente organizados. Seguiu-se a construção e documentação de um diagrama Entidade-Relacionamento (ER), que foi convertido num modelo lógico relacional, normalizado até à terceira forma normal (3FN) e validado através da implementação de interrogações com recurso à álgebra relacional.

Este modelo foi convertido para um modelo físico no Sistema de Gestão de Bases de Dados (SGBD) MySQL.

O sistema resultante tem como objetivo a melhoria da gestão de viagens do núcleo, abrangendo o registo de participantes, destinos, transportes, alojamentos e custos associados.

Área de Aplicação: Desenho e Arquitetura de Sistemas de Bases de Dados.

Palavras-Chave: Sistema de Base de Dados, Bela Vista, Modelação Conceptual, Modelação Lógica, Viagem, Implementação Física

Índice

1. Definição do Sistema	1
1.1. Contexto de aplicação	1
1.2. Motivação e Objetivos do Trabalho	1
1.3. Análise da Viabilidade do Projeto	2
1.4. Recursos e Equipa de Trabalho	3
1.5. Sistema atual	3
1.5.1. Criação do roteiro	3
1.5.2. Marcação da viagem	4
1.5.3. Registo pós viagem	4
1.6. Plano de Execução do Projeto	5
1.7. Revisão e Aprovação	6
2. Levantamento e Análise dos Requisitos	8
2.1. Método de Levantamento e de Análise de Requisitos Adotado	8
2.1.1. Levantamento	8
2.1.1.1. Vistas de Utilização	9
2.1.1.2. Documento de Requisitos	9
2.1.2. Análise	9
2.2. Organização dos Requisitos Levantados	10
2.3. Análise e Validação Geral dos Requisitos	13
3. Modelação Conceptual	14
3.1. Apresentação da Abordagem de Modelação Realizada	14
3.2. Identificação e Caracterização das Entidades	14
3.2.1. Viagem	15
3.2.2. Dia de Viagem	15
3.2.3. Ponto de Interesse	16
3.2.4. Utilizador	16
3.2.5. Organização	17
3.3. Identificação e Caracterização dos Relacionamentos	17
3.3.1. Viagem → Dia de Viagem : 1 → n	17
3.3.2. Viagem → Utilizador : n → n	17
3.3.3. Viagem → Organização : n → n	17
3.3.4. Utilizador → Organização : n → n	18
3.3.5. Dia de Viagem → Ponto de interesse : 1 → n	18
3.4. Critérios de Definição de Domínios	18
3.5. Identificação e Caracterização dos Atributos das Entidades e dos Relacionamentos	19
3.5.1. Viagem	19
3.5.2. Dia de Viagem	19
3.5.3. Ponto de Interesse	19
3.5.4. Utilizador	19
3.5.5. Organização	20
3.6. Apresentação e Explicação do Diagrama ER Produzido	20
3.7. Validação do esquema	21
3.7.1. Revisão Interna	21
3.7.2. Validação do CeBELUM	21
3.7.3. Aprovação Final	22

4. Modelação Lógica	23
4.1. Construção e Validação do Modelo de Dados Lógico	23
4.2. Apresentação e Explicação do Modelo Lógico Produzido	23
4.2.1. Derivação de Relações do Modelo de Dados Local	23
4.2.1.1. Utilizador	23
4.2.1.2. Viagem	24
4.2.1.3. ViagemRealizadaUtilizador	24
4.2.1.4. Organização	25
4.2.1.5. ViagemMencionaOrganização	25
4.2.1.6. UtilizadorPertenceOrganização	25
4.2.1.7. DiaDeViagem	26
4.2.1.8. Ponto de Interesse	27
4.2.2. Esquema Lógico Final	27
4.3. Normalização de Dados	29
4.4. Validação do Modelo com Interrogações do Utilizador	30
4.5. Validação do Esquema Lógico Final	38
5. Implementação Física	39
5.1. Apresentação e explicação da base de dados implementada	39
5.1.1. Tabela Utilizador	39
5.1.2. Tabela Organização	39
5.1.3. Tabela UtilizadorPertenceOrganização	40
5.1.4. Tabela Viagem	40
5.1.5. Tabela ViagemRealizadaUtilizador	40
5.1.6. Tabela ViagemMencionaOrganização	41
5.1.7. Tabela DiaDeViagem	41
5.1.8. Tabela PontoDelInteresse	42
5.1.9. Tabela Fotografias	42
5.1.10. Ordem de Criação das Tabelas	42
5.1.11. Políticas de Integridade Referencial	43
5.2. Criação de utilizadores da base de dados	43
5.3. Povoamento da base de dados	44
5.3.1. Tabela “Utilizador”	45
5.3.2. Tabela “Organização”	45
5.3.3. Tabela “UtilizadorPertenceOrganização”	45
5.3.4. Tabela “Viagem”	45
5.3.5. Tabela “ViagemMencionaOrganização”	45
5.3.6. Tabela “DiaDeViagem”	45
5.3.7. Tabela “PontoDelInteresse”	46
5.3.8. Tabela “Fotografias”	46
5.3.9. Tabela “ViagemRealizadaUtilizador”	46
5.4. Cálculo do espaço da base de dados (inicial e taxa de crescimento anual)	47
5.5. Definição e caracterização de vistas de utilização em SQL	51
5.5.1. Custo Viagens	51
5.5.2. <i>Pontos de Interesse da Viagem</i>	51
5.5.3. Pontos de Interesse e Fotografias	52
5.5.4. Logística do Dia de Viagem	52
5.5.5. Utilizadores	52
5.5.6. Organizadores	52
5.5.7. Organizações das Viagens	53
5.5.8. Utilizadores e suas Organizações	53
5.5.9. Avaliações da Viagem	53
5.5.10. Participantes da Viagem	53
5.5.11. Viagem com Itinerário	53
5.6. Tradução das interrogações do utilizador para SQL	54
5.7. Indexação do Sistema de Dados	58
5.7.1. Tabela “Viagem”	58
5.7.2. Tabela “DiaDeViagem”	59

5.7.3. Tabela “ViagemRealizadaUtilizador”	59
5.7.4. Tabela “Utilizador”	59
5.7.5. Tabela “UtilizadorPertenceOrganizacao”	59
5.8. Implementação de procedimentos, funções e gatilhos	59
5.8.1. <i>Function</i> - calculateTripRating (RM29)	59
5.8.2. <i>Function</i> - isAdmin	60
5.8.3. <i>Function</i> - calculateTripCost (RM12)	60
5.8.4. <i>Procedure</i> - getTripByLoc (RM34)	60
5.8.5. <i>Procedure</i> - getUserTrips (RM24)	61
5.8.6. <i>Procedure</i> - getTripsByOrgId (RM31)	62
5.8.7. <i>Procedure</i> - getTripsByCost (RM22)	62
5.8.8. <i>Procedure</i> - getTripsByDates (RM23)	62
5.9. Implementação de transações	63
5.9.1. <i>Procedure</i> - createTrip (RM35)	63
5.9.2. <i>Procedure</i> - editTrip (RM38)	64
5.9.3. <i>Procedure</i> - deleteTrip (RM38)	65
5.9.4. <i>Procedure</i> - createTripDay (RM39)	65
5.9.5. <i>Procedure</i> - createInterestPoint (RM40)	66
5.9.6. <i>Procedure</i> - reviewTrip (RM41)	67
5.9.7. <i>Procedure</i> - editTripReview (RM42)	68
5.9.8. <i>Procedure</i> - deleteTripReview (RM42)	69
5.10. Exportação de informação	70
6. Conclusão	72
6.1. Análise do Diagrama de Gantt Planeado x Real	72
6.2. Trabalho futuro	72
6.3. Reflexão Final	73
7. Bibliografia	74
Lista de Siglas e Acrónimos	75
Anexos	76
I. Diagrama de Gantt - Planeamento de Tarefas	76
II. Diagrama de Gantt - Tarefas Realizadas	78
III. Requisitos Levantados	79
IV. Requisitos de Descrição	84
V. Requisitos de Manipulação	86
VI. Requisitos de Controlo	88
VII. Modelo Físico	89
VIII. Povoamento da base de dados	90
IX. Índices criados	91
X. Procedimentos, funções e gatilhos	91
XI. <i>Script</i> de criação de utilizadores	93
XII. <i>Script</i> calculadora RelaX	95
XIII. Criação de vistas	96
XIV. Interrogações do Utilizador	101
XV. Transações	102

Lista de Figuras

Figura 1	Roteiro no Sistema Antigo	4
Figura 2	Registo do hotel	4
Figura 3	Avaliação posterior à viagem	5
Figura 4	Diagrama de Gantt	6
Figura 5	Modelo Conceptual final	21
Figura 6	Conversão da Entidade Utilizador - Modelo Lógico	24
Figura 7	Conversão da Entidade Viagem - Modelo Lógico	24
Figura 8	Conversão do Relacionamento Realizada - Modelo Lógico	24
Figura 9	Conversão da Entidade Organização - Modelo Lógico	25
Figura 10	Conversão do Relacionamento Menciona - Modelo Lógico	25
Figura 11	Conversão do Relacionamento Pertence - Modelo Lógico	26
Figura 12	Conversão da Entidade Dia de Viagem - Modelo Lógico	26
Figura 13	Conversão da Entidade Ponto de Interesse - Modelo Lógico	27
Figura 14	Modelo lógico da base de dados	28
Figura 15	Modelo lógico da base de dados	28
Figura 16	RM5 - Árvore álgebra relacional	31
Figura 17	RM5 - Expressão álgebra relacional	31
Figura 18	RM11 - Árvore Álgebra Relacional	32
Figura 19	RM11 - Expressão álgebra relacional	32
Figura 20	RM12 - Árvore Álgebra Relacional	33
Figura 21	RM12 - Expressão Álgebra Relacional	33
Figura 22	RM21 - Árvore Álgebra Relacional	34
Figura 23	RM21 - Árvore Álgebra Relacional	34
Figura 24	RM28 - Árvore Álgebra Relacional	35
Figura 25	RM28 - Expressão Álgebra Relacional	35
Figura 26	RM30 - Árvore Álgebra Relacional	36
Figura 27	RM30 - Expressão Álgebra Relacional	36
Figura 28	RM32 - Árvore Álgebra Relacional	37
Figura 29	RM32 - Expressão Álgebra Relacional	37

Lista de Tabelas

Tabela 1	Requisitos de Descrição	10
Tabela 2	Requisitos de Manipulação	11
Tabela 3	Requisitos de Controlo	12
Tabela 4	Caracterização das Entidades	15
Tabela 5	Caracterização atributos da tabela Viagem	15
Tabela 6	Caracterização atributos da tabela Dia de Viagem	16
Tabela 7	Caracterização atributos da tabela Ponto de Interesse	16
Tabela 8	Caracterização atributos da tabela Utilizador	17
Tabela 9	Caracterização atributos da tabela Organização	17
Tabela 10	Relacionamentos identificados	18
Tabela 11	Mapeamento de domínios entre MySQL e RelaX	30
Tabela 12	Espaço ocupado pelos tipos de dados de MySQL“	47
Tabela 13	Espaço ocupado DECIMAL	47
Tabela 14	Espaço ocupado pelos atributos de cada registo da tabela “Utilizador”	48
Tabela 15	Espaço ocupado pelos atributos de cada registo da tabela “Organização”	48
Tabela 16	Espaço ocupado pelos atributos de cada registo da tabela “UtilizadorPertenceOrganização”	48
Tabela 17	Espaço ocupado pelos atributos de cada registo da tabela “Viagem”	48
Tabela 18	Espaço ocupado pelos atributos de cada registo da tabela “ViagemRealizadaUtilizador”	48
Tabela 19	Espaço ocupado pelos atributos de cada registo da tabela “ViagemMencionaOrganização” ..	49
Tabela 20	Espaço ocupado pelos atributos de cada registo da tabela “DiaDeViagem”	49
Tabela 21	Espaço ocupado pelos atributos de cada registo da tabela “PontoDeInteresse”	49
Tabela 22	Espaço ocupado pelos atributos de cada registo da tabela “Fotografias”	49
Tabela 23	Tamanho ocupado por cada tabela na base de dados BelaVista	50
Tabela 24	Evolução do tamanho da base de dados BelaVista	50
Tabela 25	Aumento da base de dados BelaVista avaliado em 3 anos	50

1. Definição do Sistema

1.1. Contexto de aplicação

No verão de 2025, o CeBELUM (Centro de Estudantes de Biotecnologia e Engenharia de Laboratório da Universidade do Minho) preparava-se para realizar mais uma das suas tradicionais viagens. Este evento, aguardado por todos os membros, representa uma oportunidade para fortalecer laços, descontrair após períodos intensos de avaliações e explorar novos destinos em conjunto.

Nesta edição, reuniram-se numa tarde de Maio, no espaço do CeBELUM, para organizar a aventura. A ambição era explorar o litoral português. O plano incluía a definição do itinerário, seleção de destinos, reserva de alojamentos para o grupo, identificação de atividades de interesse que pudessem agradar todos os participantes e muitos outros detalhes pertinentes.

O processo criativo decorreu utilizando métodos comuns e acessíveis à direção. Para a gestão de orçamentos, da qual faziam parte gastos no alojamento e atividades, optaram pela criação de folhas de cálculo. Já para a elaboração de rotas, e armazenamento de links e novas ideias, eram utilizados *chats* de mensagens online, notas nos telemóveis, entre outros meios.

Cada organizador assumiu, naturalmente, diferentes funções. Destas faziam parte a pesquisa de alojamentos, pontos de interesse culturais e gastronómicos, coordenação logística de transportes e horários, entre as demais. A coordenação foi feita através da combinação destas várias ferramentas digitais, com atualizações frequentes à medida que novas informações eram descobertas.

No final, a viagem ficou registada na memória dos participantes e dispersa pelos diversos suportes digitais utilizados: fotografias nos telemóveis, mensagens com recomendações trocadas e as várias versões do plano que foram sendo criadas e atualizadas ao longo do processo de organização.

1.2. Motivação e Objetivos do Trabalho

A experiência deste núcleo ilustra um problema comum no planeamento de viagens: a dependência de ferramentas digitais genéricas sem integração específica. A combinação de folhas Excel, sucessivas versões de roteiros (como o inevitável “Retiro CeBELUM 2025 (definitivo) v5 (AGORA É A SÉRIO.xls”), notas dispersas no telemóvel e links partilhados no WhatsApp revelou rapidamente as suas limitações. Esta fragmentação resultou na perda de informações críticas, como datas, reservas (como o “Airbnb com vista deslumbrante” que afinal era um T0 em Gualtar com vista para o Lidl), pontos de interesse e outros detalhes essenciais para a realização da viagem.

Problemas encontrados

A utilização de múltiplas aplicações dispersa dados essenciais por diferentes plataformas, dificultando o acesso consolidado e gerando confusão pela coexistência de versões conflituosas do plano. Após a viagem, a tentativa de armazenar ou partilhar a experiência com outros membros do CeBELUM enfrentou novamente a dispersão da informação, exigindo um esforço significativo de consolidação manual para consultas posteriores ou reutilização da planificação em edições futuras.

A dificuldade em comparar viagens anteriores, avaliar destinos ou identificar atividades bem-sucedidas revelou a ausência de uma visão global e histórico estruturado. A falta de normalização nos registo e as análises manuais aumentavam a probabilidade de erros. Além demais, os custos de impressão de materiais físicos e criação de álbuns, embora preservem memórias, implicam gastos adicionais e requerem espaço de armazenamento, tornando-se pouco prático a longo prazo.

Necessidade de mudança

Este desafio não é exclusivo do CeBELUM, uma vez que é enfrentado por outros núcleos de estudantes, grupos de amigos e viajantes em geral. Torna-se evidente a necessidade de uma solução integrada que centralize toda a informação relativa a viagens, assegurando consistência de dados, acesso em tempo real e partilha eficaz da experiência.

A digitalização surge assim como uma alternativa acessível e duradoura, permitindo guardar memórias de forma segura e organizada num único sistema através da integração de uma base de dados.

A partir destas limitações, o núcleo de estudantes definiu os seguintes objetivos para o sistema de bases de dados do projeto Bela Vista:

- **Centralizar a informação da viagem:**

Reunir todos os dados num único sistema acessível e estruturado, eliminando a dispersão por múltiplas plataformas.

- **Facilitar o planeamento e registo de viagens:**

Proporcionar uma gestão simples e intuitiva que permita criar, editar e organizar viagens sem a complexidade de coordenar diferentes aplicações. O sistema deve tornar o processo de planeamento mais ágil, reduzindo o tempo dedicado a tarefas administrativas e logísticas.

- **Preservar memórias e experiências:**

Substituir os tradicionais registos físicos por um formato digital de fácil consulta e armazenamento permanente. Ao digitalizar fotografias, notas e recordações, elimina-se a necessidade de espaço físico e os custos associados à impressão de materiais, mantendo as memórias acessíveis e protegidas ao longo do tempo.

- **Promover a colaboração entre utilizadores:**

Possibilitar a partilha de informações e atualização conjunta dos roteiros em tempo real. Todos os utilizadores podem contribuir com sugestões, adicionar pontos de interesse e manter-se sincronizados quanto às decisões tomadas, evitando mal-entendidos e conflitos de versões.

- **Melhorar a eficiência da organização:**

Reducir erros, duplicações e dispersão de dados através de uma estrutura normalizada e consistente. A validação automática de informações e a eliminação de redundâncias minimizam o risco de perda de dados críticos e garantem a fiabilidade do sistema.

- **Garantir uma experiência agradável e moderna:**

Tornar o processo de planear e recordar viagens mais envolvente e acessível.

1.3. Análise da Viabilidade do Projeto

A equipa do projeto Bela Vista acredita firmemente que a implementação do novo sistema de base de dados trará diversos benefícios para a organização e partilha de experiências de viagem, que por sua vez, terão retornos operacionais e financeiros, rápidos e significativos a curto e médio prazo.

Com a centralização normalizada dos dados, espera-se:

Viabilidade Financeira:

- Eliminação de custos com registos físicos (papel, impressões, arquivos), face à substituição dos mesmos por um sistema digital, com uma redução estimada de 70-80% nos custos administrativos associados à gestão de documentação. Ademais, este permitirá ainda garantir a preservação de memórias e dados a

longo prazo, de forma segura e acessível, reduzindo desta forma possíveis custos com trabalho causado por dados inconsistentes ou perdidos, o que pode representar uma poupança de 15 a 20 horas de trabalho por viagem planeada;

- Baixo investimento inicial em ferramentas de *software* através da adoção de uma *stack* tecnológica *open-source*, eliminando custos de licenças e reduzindo o investimento tecnológico em aproximadamente 80-90%;

Viabilidade Operacional:

- Possibilidade de implementação incremental, permitindo ajustes contínuos com base no *feedback* real dos utilizadores e na evolução das necessidades operacionais;
- Potencialização de expansão futura com investimento mínimo adicional, graças à arquitetura relacional do projeto, que permite integrar novas funcionalidades sem necessidade de reestruturação completa, e com apenas 20-25% do esforço que seria necessário.

Posto isto, a criação de uma base de dados mostrou-se viável e capaz de resolver o problema inicial, enfrentado não só por este núcleo, mas como também por empresas e particulares. Apenas administradores poderão eliminar utilizadores, viagens ou organizações. O sistema deverá permitir gerir os perfis de acesso (administrador, utilizador normal). Deverá existir cópia de segurança periódica da base de dados. O administrador poderá consultar estatísticas de utilização (número de viagens, avaliações, etc.).

1.4. Recursos e Equipa de Trabalho

Na implementação do sistema da base de dados do projeto Bela Vista, serão necessários mobilizar recursos humanos e materiais, de forma a garantir o funcionamento, manutenção e segurança do sistema. Por conseguinte, estão previstos:

Recursos Humanos

Internos - A vice-presidente Milena Salomé e o secretário Júlio César são a fonte principal de informação e requisitos, pois são os responsáveis pelo projeto e utilizadores finais.

Externos - Débora Caetano, Enrico Prazeres e Gonçalo Fernandes são os responsáveis pelo desenvolvimento técnico desde a Contextualização até à Implementação Física do sistema de base de dados.

Recursos Materiais

- Equipamento suficiente para executar *BrModelo* e *MySQL Workbench* que serão as ferramentas mais custosas computacionalmente.
- Acesso às ferramentas citadas no ponto anterior para a realização da Modelação e Implementação do sistema.
- Servidores capazes de comportar o sistema, físicos ou em *cloud*.

1.5. Sistema atual

Atualmente, como já foi referido, os membros organizam as viagens em folhas de cálculo e grupos em plataformas virtuais para armazenar roteiros das viagens que irão fazer. Adicionalmente, sistemas de armazenamento em nuvem para guardar as recordações das viagens que já realizaram, o que tem causado conflitos e confusões entre os membros.

Abaixo descrevemos todo o processo de marcação de uma viagem, desde a criação do roteiro da viagem até o registo e avaliação pós viagem.

1.5.1. Criação do roteiro

Para chegar a um consenso estes realizam um pequeno *brainstorming* sobre os monumentos e pontos de interesse que gostariam de conhecer ao longo da viagem. Para isso utiliza-se uma folha de cálculo para fazer uma recolha e filtrar os monumentos que todos acham interessante.

O roteiro é idealizado e registado atualmente da seguinte forma:

	A Dia	B Atividade 1	C Atividade 2	D Atividade 3	E Atividade 4
1					
2	Dia 1 - Faro	Cidade Velha (Vila Adentro)	Sé de Faro	Marina e pôr do sol	Jantar com frutos do mar
3	Dia 2 - Ria Formosa	Passeio de barco pelas ilhas Deserta, Farol e Culatra	Praia e almoço com peixe fresco	Retorno a Faro	
4	Dia 3 - Tavira & Cacela Velha	Castelo de Tavira	Ilha de Tavira (barco)	Vila de Cacela Velha (vista incrível)	
5	Dia 4 - Lagos	Grutas da Ponta da Piedade	Praia do Camilo e Dona Ana	Centro histórico de Lagos	
6	Dia 5 - Sagres	Fortaleza de Sagres	Cabo de São Vicente (pôr do sol)	Praia do Beliche	
7	Dia 6 - Costa Vicentina / Aljezur	Praias: Amado, Bordaia, Arrifana	Castelo de Aljezur	Almoço típico (peixe grelhado)	
8	Dia 7 - Vila Nova de Milfontes	Praia das Furnas	Passeio pelo centro	Almoço à beira do rio Mira	

Figura 1: Roteiro no Sistema Antigo

1.5.2. Marcação da viagem

A recolha e partilha das informações de alojamento eram realizadas de forma informal, através de um grupo de WhatsApp. Nesse canal eram trocadas mensagens com detalhes sobre o hotel escolhido, incluindo datas, preços e condições da reserva.

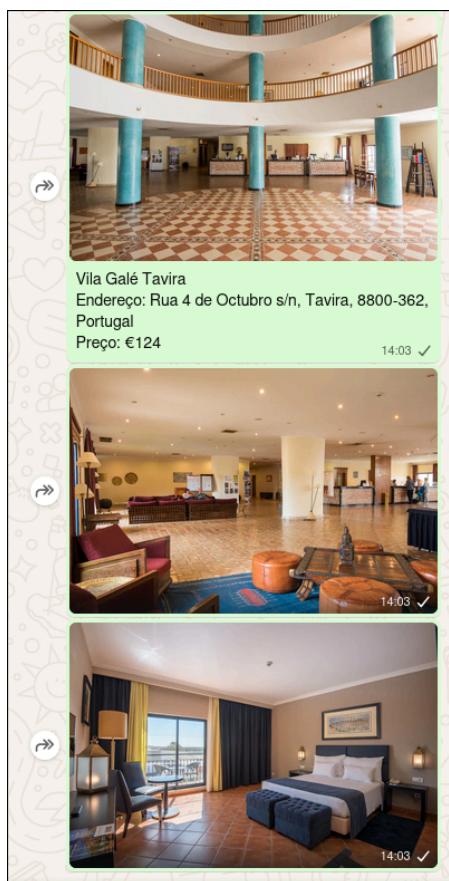


Figura 2: Registo do hotel

1.5.3. Registo pós viagem

No final de cada viagem, procura-se saber a opinião de cada participante, recorrendo-se à ferramenta *Notion*.

Evaluation System

The screenshot shows a web-based evaluation system. At the top, there's a header with the title 'Evaluation System' and a 'New' button. Below the header is a table titled 'Evaluation Details' with columns for Escopo (Scope), Sítio (Site), Avaliador (Reviewer), Data (Date), and Avaliação (Evaluation). The table contains 7 rows, each representing a day of travel:

Escopo	Sítio	Avaliador	Data	Avaliação
Dia 1	Faro	Enrico Prazeres	July 16, 2025	3
Dia 2	Ria Formosa			
Dia 3	Tavira & Cacela Velha			
Dia 4	Lagos			
Dia 5	Sagres			
Dia 6	Costa Vicentina / Aljezur			
Dia 7	Vila Nova de Milfontes			

Below the table, there are buttons for '+ New page', 'COUNT 7', 'UNIQUE 1', and 'LATEST July 16, 2025'. A note at the bottom says 'Write, press 'space' for AI, '/' for commands...'. The main area below the table is a large text input field.

On the right side of the interface, there is a 'Avaliação' section with a red asterisk icon. It displays five colored bars representing different sentiment scores from 1 to 5, each with a corresponding number and a short quote:

- 5: Localidade incrível. Comunidade local incrível. Serviço de hotel impecável.
- 4: Sítio muito agradável. Comunidade local cu serviço de hotel não tanto.
- 3: Localidade interessante. Habitantes locais neutros. Serviço de hotel satisfatório.
- 2: Não é tão agradável. Muitos problemas de logística e organização no hotel.
- 1: Não gostei nada. Nunca voltaria.

Figura 3: Avaliação posterior à viagem

1.6. Plano de Execução do Projeto

Para que o desenvolvimento e a implementação do sistema de bases de dados do projeto Bela Vista seja eficaz, a equipa de profissionais, juntamente de um dos responsáveis do projeto, definiu um plano de trabalhos.

Este plano inclui os períodos de execução de cada uma das fases do ciclo de vida de um Sistema de Base de Dados, junto dos responsáveis por cada etapa.

BELA VISION

Implementação do Sistema de Base de Dados

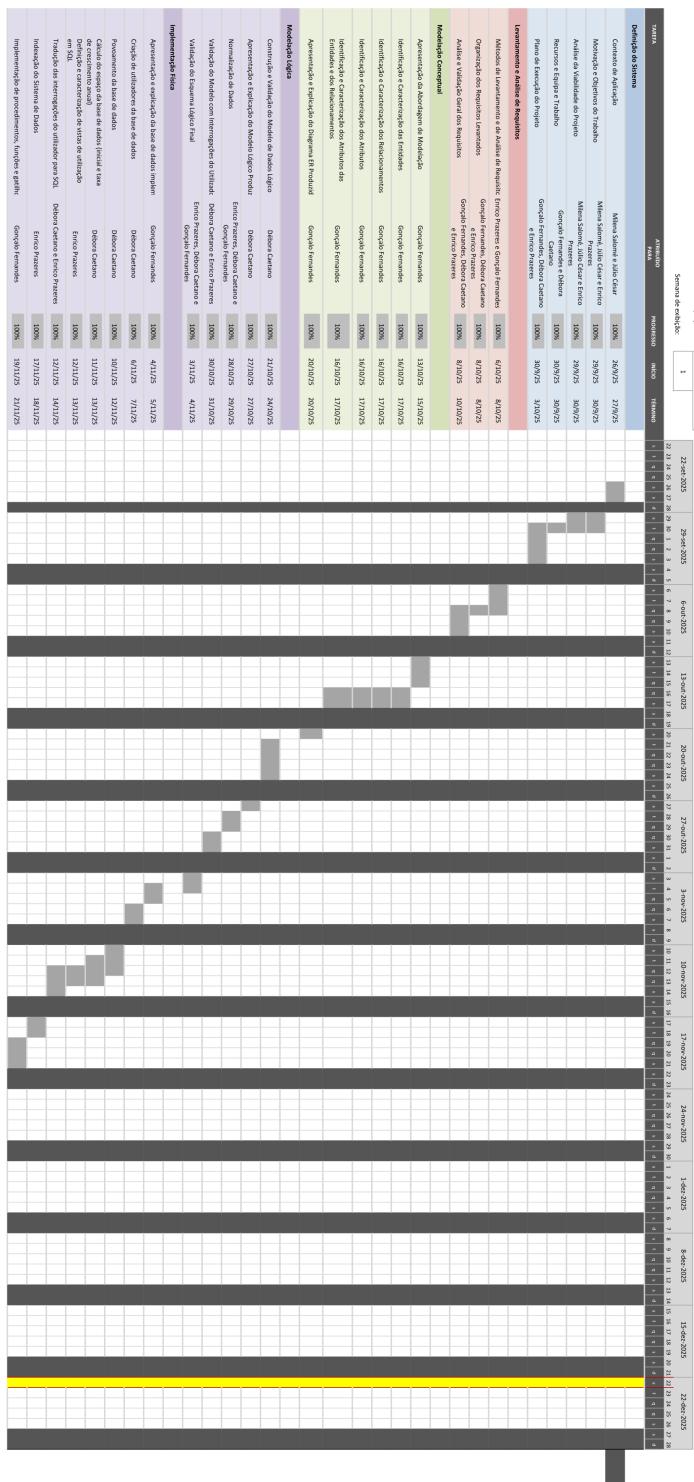


Figura 4: Diagrama de Gantt

1.7. Revisão e Aprovação

A equipa de desenvolvimento reuniu-se com os responsáveis pelo projeto para analisar o plano de trabalhos proposto. Durante a reunião, foi avaliada a planificação das tarefas com base no diagrama de Gantt

elaborado pela equipa. Após a análise, os três idealizadores manifestaram a sua satisfação e aprovaram o plano apresentado.

Uma vez obtida a validação e aprovação de todos os participantes, o projeto prossegue para a fase seguinte, dedicada à recolha e especificação detalhada dos requisitos necessários ao desenvolvimento da base de dados e das funcionalidades do sistema.

2. Levantamento e Análise dos Requisitos

2.1. Método de Levantamento e de Análise de Requisitos Adotado

Na implementação do sistema de base de dados do projeto Bela Vista, será necessário um processo de levantamento de requisitos detalhados.

Reuniões:

- Reunião interna: Discutir, entre os membros da presidência do CeBELUM, a necessidade da criação de um sistemas de bases de dados.
- Reunião de primeiro contacto: Entender se será viável iniciar o projeto com a equipa de desenvolvimento.
- Reunião interna do CeBELUM: Recolha de *feedback* sobre a experiência dos membros.
- Reunião de diagnóstico e planeamento: Discutir as necessidades e desafios operacionais com a equipa de desenvolvimento.

Análise da Documentação:

- Análise dos recursos que contém a organização dos roteiros.
- Análise das imagens para entender a forma como as marcações dos hotéis/pousadas são geridos.
- Análise dos métodos de avaliação de viagens utilizados previamente.

Observação dos Processos:

- Observação da criação da viagem do núcleo para conhecer os processos.
- Observação da gestão da marcação do alojamento.
- Identificação dos pontos onde há possibilidade de contribuição de diferentes membros responsáveis pela organização da viagem.

Inquéritos:

- Inquéritos aos membros do CeBELUM (utilizadores finais) para avaliar a satisfação com o processo de organização de toda a viagem, desde a organização de cada dia da viagem ao método de avaliação posterior.

Pesquisa de outras soluções:

- Pesquisa sobre plataformas de gestão utilizados por outros centros de estudantes para analisar as funcionalidades.

2.1.1. Levantamento

Para o desenvolvimento do sistema de base de dados para o projeto Bela Vista foi feito um levantamento de requisitos.

Este levantamento, como esperado, inclui a organização e validação dos requisitos por áreas do sistema. Para garantir um levantamento preciso dos requisitos, foram utilizadas diversas abordagens:

- Viagens: Contém os detalhes principais sobre a viagem, tendo um título e uma descrição.
- Dias de viagem: Constitui uma viagem e contém detalhes de um dia de viagem, incluindo datas, custo e possivelmente o transporte e alojamento para o dia.
- Pontos de Interesse: Constitui um dia de viagem e possui um nome, descrição, localização, horário e uma ou mais fotografias.

- Utilizador: Possui nome, idade, contacto (email ou telefone) e pode pertencer a uma organização. Um Utilizador pode fazer uma viagem enquanto organizador ou participante, bem como pode fazer uma avaliação da viagem.
- Organização: Possui um nome e tem relação com Viagem e Utilizador.

No contexto do sistema de base de dados, estas áreas representam diferentes perspetivas de utilização, correspondentes à forma como os utilizadores acedem e manipulam a informação necessária ao desempenho das suas funções.

Para compreender melhor o processo de organização de uma viagem, a equipa realizou uma análise detalhada da documentação de anos anteriores, com especial foco nas avaliações das viagens feitas pelas equipas organizadoras e participantes.

Dessa análise foram identificados alguns pontos de maior criticidade:

- Organização de transportes e alojamento para cada dia;
- Horários de visita dos pontos de interesse;
- Avaliação posterior da viagem.

Toda a documentação recolhida foi arquivada e associada aos requisitos identificados, garantindo uma referência clara e consistente para as fases seguintes do projeto.

Ao longo do levantamento, os requisitos foram sendo documentados num Documento de Requisitos, assegurando assim que todas as necessidades identificadas sejam devidamente consideradas na conceção do sistema, sendo estes dados confirmados a partir das reuniões com os responsáveis do núcleo, Milena Salomé e Júlio César. Durante essas reuniões, foram discutidas as necessidades operacionais e os objetivos do sistema.

2.1.1.1. Vistas de Utilização

O desenvolvimento do sistema seguirá uma abordagem modular, na qual cada área será analisada e estruturada de forma autónoma. Esta metodologia permitirá compreender de forma aprofundada as necessidades específicas de cada funcionalidade antes da sua integração no sistema final.

Depois de desenvolvidas as diferentes componentes, proceder-se-á à integração entre todas as vistas de utilização, assegurando que o sistema funcione de forma coerente, consistente e eficiente.

À medida que cada vista é aperfeiçoada individualmente, torna-se fundamental garantir a sua compatibilidade, de forma a que o sistema de base de dados consiga unificar todas as perspetivas de utilização num único esquema global.

Com esta abordagem, a CeBELUM beneficiará de um sistema de base de dados alinhado com a organização das suas viagens, promovendo uma gestão mais eficiente e uma experiência superior tanto para os organizadores como para os participantes.

2.1.1.2. Documento de Requisitos

Todos os requisitos recolhidos foram posteriormente registados num documento específico para os mesmos, incluindo a data e hora do levantamento, uma descrição detalhada do requisito em si, a vista de utilização correspondente, a fonte de informação e o analista responsável pelo levantamento do requisito.

2.1.2. Análise

Depois de recolhidos os requisitos é fundamental avaliar a sua viabilidade e realizar uma revisão detalhada, de forma a garantir que os requisitos garantem as necessidades da base de dados e que se encontram corretos. Este processo permite identificar e corrigir possíveis erros, repetições ou contradições, confirmando que o projeto avance com uma estrutura bem definida. Essa revisão é crucial para garantir que o sistema de base de dados seja desenvolvido sobre uma fundação sólida e consistente.

2.2. Organização dos Requisitos Levantados

À medida que o processo de levantamento de requisitos terminava, foi necessária a distribuição destes por categorias, nomeadamente:

- **Descrição:** engloba os requisitos relativos à estrutura e organização dos dados na base de dados, incluindo atributos, domínios, restrições e outros objetos de dados;
- **Exploração:** contém os requisitos associados à manipulação e utilização dos dados, como *queries*, relatórios e procedimentos que permitem o povoamento e análise da informação;
- **Administração:** refere-se aos requisitos de gestão do sistema de base de dados, incluindo o controlo de acessos, monitorização, a segurança e a definição de permissão dos utilizadores.

Para cada uma das categorias foi criado um documento em *Excel* específico:

- Documento de Requisitos de Descrição:

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos
Documento de Requisitos de Descrição

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RD1	06/10/2025 08:15	Um utilizador é caracterizado por um identificador único, nome, idade, contacto (telefone e email) e se é admin ou não.	Pessoas	Milena Salomé	Enrico
RD2	06/10/2025 08:30	Uma organização é caracterizada por um identificador único e nome.	Pessoas	Júlio Cesar	Débora
RD3	06/10/2025 08:45	Um utilizador pode pertencer a zero ou mais organizações.	Pessoas	Milena Salomé	Gonçalo
RD4	06/10/2025 08:53	Uma organização deve ter pelo menos um utilizador associado.	Pessoas	Milena Salomé	Enrico
RD5	06/10/2025 09:00	Uma viagem é caracterizada por um identificador único, título e descrição.	Viagens	Júlio Cesar	Enrico
RD6	06/10/2025 09:10	A data de ida de uma viagem corresponde à data mais antiga dos seus dias de viagem.	Viagens	Júlio Cesar	Débora
RD7	06/10/2025 09:10	A data de regresso de uma viagem corresponde à data mais recente dos seus dias de viagem.	Viagens	Júlio Cesar	Gonçalo
RD8	06/10/2025 09:13	O custo total de uma viagem corresponde à soma dos custos dos seus dias de viagem.	Viagens	Milena Salomé	Débora
RD9	06/10/2025 09:17	Uma viagem é constituída por um ou mais dias de viagem.	Viagens	Júlio Cesar	Débora
RD10	06/10/2025 09:30	Um dia de viagem é caracterizado por uma data, custo, e opcionalmente por alojamento e transporte.	Viagens	Milena Salomé	Enrico
RD11	06/10/2025 09:50	Um ponto de interesse é caracterizado por um identificador único, nome, localização, descrição e horário.	Conteúdo	Júlio Cesar	Gonçalo

Tabela 1: Requisitos de Descrição

- Documento de Requisitos de Manipulação:

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos

Documento de Requisitos de Manipulação

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RM1	07/10/2025 09:30	O sistema deve permitir registar um novo utilizador.	Pessoas	Júlio Cesar	Débora
RM2	07/10/2025 09:33	O sistema deve permitir atualizar os dados de um utilizador.	Pessoas	Júlio Cesar	Enrico
RM3	07/10/2025 09:36	O sistema deve permitir eliminar um utilizador.	Pessoas	Júlio Cesar	Enrico
RM4	07/10/2025 09:45	O sistema deve permitir criar uma nova organização.	Pessoas	Milena Salomé	Débora
RM5	07/10/2025 09:52	O sistema deve listar todos os utilizadores pertencentes a uma organização.	Pessoas	Júlio Cesar	Gonçalo
RM6	07/10/2025 09:58	O sistema deve permitir associar utilizadores a organizações.	Pessoas	Milena Salomé	Enrico
RM7	07/10/2025 10:30	O sistema deve permitir remover utilizadores de organizações.	Pessoas	Milena Salomé	Gonçalo
RM8	07/10/2025 10:35	O sistema deve permitir criar uma nova viagem.	Viagens	Júlio Cesar	Enrico
RM9	07/10/2025 10:38	O sistema deve permitir editar os dados de uma viagem existente.	Viagens	Júlio Cesar	Débora
RM10	07/10/2025 10:40	O sistema deve permitir eliminar uma viagem.	Viagens	Milena Salomé	Enrico
RM11	07/10/2025 11:00	O sistema deve calcular as datas de ida e regresso das viagens com base nos seus dias.	Viagens	Júlio Cesar	Gonçalo
RM12	07/10/2025 11:30	O sistema deve calcular o custo total das viagens com base nos custos dos seus dias.	Viagens	Milena Salomé	Débora
RM13	07/10/2025 11:33	O sistema deve permitir criar um dia de viagem associado a uma viagem.	Viagens	Milena Salomé	Gonçalo

Tabela 2: Requisitos de Manipulação

- Documento de Requisitos de Controlo:

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos

Documento de Requisitos de Controlo

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RC1	08/10/2025 14:34	O sistema deve implementar dois tipos de perfis: Gestor e Utilizador Normal.	-	Milena Salomé	Enrico
RC2	08/10/2025 14:57	Administradores têm permissões de leitura, escrita e em todas as tabelas.	-	Milena Salomé	Gonçalo
RC3	08/10/2025 15:20	Administradores podem consultar estatísticas globais do sistema.	-	Júlio Cesar	Enrico
RC4	08/10/2025 15:24	Utilizadores normais têm permissões de leitura, escrita e em todas as tabelas exceto a tabela Organização.	-	Júlio Cesar	Gonçalo

Tabela 3: Requisitos de Controlo

2.3. Análise e Validação Geral dos Requisitos

Para validar os requisitos levantados foi realizada uma reunião com todos os intervenientes do projeto. Cada requisito foi então analisado e discutido, permitindo a identificação de possíveis detalhes que pudessem surgir. No final, todos os requisitos são aprovados permitindo uma precisão e alinhamento com as necessidades do Bela Vista.

3. Modelação Conceptual

3.1. Apresentação da Abordagem de Modelação Realizada

O desenvolvimento do projeto Bela Vista depende de uma etapa fundamental: a elaboração de um esquema conceptual. Durante esta etapa, define-se a estrutura da base de dados, onde estão incluídas as entidades principais, os seus relacionamentos e os respetivos atributos.

O processo de modelação conceptual adota a notação de Chen em conjunto com a notação de Min-Max de Abrial. A implementação desta abordagem foi realizada através da ferramenta BR Modelo, que possibilita a visualização do modelo em tempo real, com representações gráficas das entidades, relacionamentos e atributos.

3.2. Identificação e Caracterização das Entidades

O sistema é composto por cinco entidades principais, responsáveis por estruturar e organizar as informações relacionadas ao planeamento e gestão das viagens:

- Viagem
- Dia de Viagem
- Ponto de Interesse
- Utilizador
- Organização

A seguir apresenta-se a tabela de caracterização das entidades, contendo para cada uma:

- uma breve definição
- sinónimos ou termos alternativos
- a respetiva ocorrência no sistema

Designação	Descrição	Sinónimos	Ocorrência
Viagem	A viagem é a deslocação realizada pelos utilizadores.	Expedição, Excursão	Cada viagem possui um identificador único e é criada quando um utilizador a planeia. Uma viagem possui as informações principais, como título e descrição.
Dia de Viagem	O dia de viagem agrupa elementos como data, custo e possivelmente alojamento e transporte.	Segmento da Viagem	O dia de viagem é criado quando o utilizador o adiciona ao sistema. É identificado pela sua data.
Ponto de Interesse	Local relevante, atrativo ou significativo que pode ser visitado durante um dia de viagem, catalogado no sistema com informações descritivas.	Atração, Local	O ponto de interesse tem um identificador único e sequencial que é atribuído quando um novo local é adicionado ao catálogo do sistema.
Utilizador	Pessoa registada no sistema que planeia ou participa em viagens.	User, Viajante, Turista	Cada utilizador representa uma pessoa registada no sistema. O utilizador tem um identificador único e é atribuído quando a pessoa se regista.
Organização	Empresas ou instituições envolvidas.	Grupo	Cada organização é identificada pelo seu identificador único.

Tabela 4: Caracterização das Entidades

3.2.1. Viagem

A entidade **Viagem** representa o núcleo central do projeto Bela Vista, agregando todas as informações relacionadas com uma experiência de viagem.

Segundo o requisito de descrição **RD5**, uma viagem é composta pelos seguintes atributos:

- **ID** - identificador único da viagem;
- **Título** - nome ou título atribuído à viagem;
- **Descrição** - texto descritivo da viagem, incluindo objetivos e características gerais;

Segue-se um exemplo fictício de uma viagem registada no sistema Bela Vista. Viagem com o identificador numérico “42”, intitulada “Verão no Litoral Português 2025”, com a descrição “Exploração da costa portuguesa com foco em gastronomia local e praias”.

Atributo	Descrição	Domínio	Nulo	Derivado	Composto	Multivalorado	Requisito	Exemplo
Id	Identificador da viagem.	INT	N	N	N	N	RD5	653
Título	Título atribuído à viagem.	VARCHAR(150)	N	N	N	N	RD5	Verão no litoral português.
Descrição	Texto descritivo da viagem.	TEXT	N	N	N	N	RD5	Exploração da costa portuguesa.

Tabela 5: Caracterização atributos da tabela Viagem

3.2.2. Dia de Viagem

A entidade **Dia de Viagem** representa um dia singular de viagem, estando relacionado com um ou mais pontos de interesse.

Segundo o requisito de descrição **RD10**, um dia de viagem é composto pelos seguintes atributos:

- **Data** - data em que este dia ocorreu;
- **Custo** - custo específico do dia;
- **Alojamento** - possível alojamento para o dia;
- **Transporte** - possível transporte para o dia;

Segue-se o exemplo fictício de um dia de viagem registado no sistema. Dia de viagem datado a 7/16/2024, com custo de €45.00, com alojamento na Pousada de São Jonas e sem transporte destacado.

Atributo	Descrição	Domínio	Nulo	Derivado	Composto	Multivalorado	Requisito	Exemplo
Data	Data em que este dia ocorreu.	DATE	N	N	N	N	RD10	7/16/2024
Custo	Custo específico do dia.	DECIMAL(8,2)	N	N	N	N	RD10	45.00
Alojamento	Possível alojamento para o dia.	VARCHAR(150)	S	N	N	N	RD10	Pousada de São Jonas
Transporte	Possível transporte para o dia.	VARCHAR(100)	S	N	N	N	RD10	NULL

Tabela 6: Caracterização atributos da tabela Dia de Viagem

3.2.3. Ponto de Interesse

A entidade **Ponto de Interesse** representa locais, atividades ou destinos relevantes. Ademais, a necessidade de armazenar fotografias específicas para cada visita (mesmo quando o local físico é o mesmo) reforça a existência de “Ponto de Interesse” como entidade independente, evitando a partilha não intencional de conteúdo fotográfico entre diferentes utilizadores ou viagens.

Segundo o requisito de descrição **RD11** e **RD14**, um ponto de interesse é composto pelos seguintes atributos:

- **ID** - identificador único de um ponto de interesse;
- **Nome** - designação do local ou atividade;
- **Localização** - morada ou coordenadas geográficas do ponto de interesse.
- **Descrição** - descrição do ponto de interesse;
- **Horário** - hora de visita ao ponto de interesse;
- **Fotografias** - imagens associadas ao ponto de interesse;

Segue-se o exemplo fictício de um ponto de interesse registado no sistema. Ponto de interesse com o identificador numérico “89”, denominado “Torre Eiffel”, visitado às 14:30, localizado em Paris, França.

Atributo	Descrição	Domínio	Nulo	Derivado	Composto	Multivalorado	Requisito	Exemplo
Id	Identificador do ponto de interesse.	INT	N	N	N	N	RD11	89
Nome	Designação do local ou atividade	VARCHAR(150)	N	N	N	N	RD11	Torre Eiffel
Descrição	Descrição do ponto de interesse	TEXT	N	N	N	N	RD11	Monumento icónico construído em 1889, oferece vista panorâmica sobre Paris
Horário	Hora de visita ao ponto de interesse	TIME	N	N	N	N	RD11	14:30
Localização	Morada ou coordenadas geográficas do ponto de interesse.	VARCHAR(200)	N	N	N	N	RD11	Paris, França; 41.1579° N, 8.6291° W
Fotografias	Imagens associadas ao ponto de interesse	-	N	N	N	S	RD14	foto1.jpg

Tabela 7: Caracterização atributos da tabela Ponto de Interesse

3.2.4. Utilizador

A entidade **Utilizador** representa os indivíduos que utilizam o sistema Bela Vista para registar e partilhar as suas experiências de viagem.

Segundo o requisito de descrição **RD1**, um utilizador é composto pelos seguintes atributos:

- **ID** - identificador único do utilizador;
- **Nome** - nome completo do utilizador;
- **Idade** - idade do utilizador;
- **Contacto** - composto por Email e Telefone do utilizador.
- **Admin** - determina se um utilizador é administrador

Segue-se um exemplo fictício de um utilizador registado no sistema Bela Vista. Utilizador com o identificador numérico “128”, chamado Jonas Prazeres Fernandes, com 22 anos de idade, o número telefónico 912 345 678 e o endereço de email jonas.praz@email.pt.

Atributo	Descrição	Domínio	Nulo	Derivado	Composto	Composto por	Multivalorado	Requisito	Exemplo
Id	Identificador do utilizador	INT	N	N	N	-	N	RD1	128
Nome	Nome do utilizador.	VARCHAR(100)	N	N	N	-	N	RD1	Jonas Prazeres Fernandes
Idade	Idade do utilizador	INT	S	N	N	-	N	RD1	22
Contacto	Forma de contacto.	-	N	N	S	Telefone,Email	-	RD1	-
Telefone	Contacto telefónico do utilizador.	VARCHAR(15)	N	N	-	-	N	RD1	912345678
Email	Contacto de email do utilizador.	VARCHAR(150)	N	N	-	-	N	RD1	jonas.praz@email.pt

Tabela 8: Caracterização atributos da tabela Utilizador

3.2.5. Organização

A entidade **Organização** pode representar grupos oficiais, como núcleos de estudantes, empresas, entre outras entidades.

Segundo o requisito de descrição **RD2**, uma organização é composta pelos seguintes atributos:

- **ID** - identificador único da organização;
- **Nome** - designação oficial da organização;

Segue-se o exemplo fictício de uma organização registada no sistema. Organização com o identificador numérico “201”, denominada “CeBELUM”.

Atributo	Descrição	Domínio	Nulo	Derivado	Composto	Multivalorado	Requisito	Exemplo
Id	Identificador da organização.	INT	N	N	N	N	RD2	201
Nome	Nome da organização.	VARCHAR(150)	N	N	N	N	RD2	CeBELUM

Tabela 9: Caracterização atributos da tabela Organização

3.3. Identificação e Caracterização dos Relacionamentos

3.3.1. Viagem → Dia de Viagem : 1 → n

Segundo o **RD9** e o **RD25**, cada viagem é constituída por um ou mais dias de viagem, por outro lado, um dia de viagem constitui uma e apenas uma viagem. Esta estrutura possibilita a especificação pormenorizada de todas as atividades, custos, alojamentos e transportes utilizados em cada dia, facilitando o planeamento e a consulta posterior da experiência de viagem.

3.3.2. Viagem → Utilizador : n → n

Segundo o **RD16** e o **RD19**, cada viagem pode ser realizada por um ou mais utilizadores, e por sua vez, cada utilizador pode realizar zero ou mais viagens. Este relacionamento inclui atributos adicionais como o Cargo do utilizador na viagem (ex: organizador, participante), a Avaliação Quantitativa (classificação numérica da experiência, numa escala de 0 a 5) e a Avaliação Descritiva (comentário textual sobre a viagem), permitindo registar o papel e a opinião de cada participante em cada viagem.

3.3.3. Viagem → Organização : n → n

Segundo o **RD22** e o **RD21**, cada viagem pode mencionar zero ou mais organizações envolvidas (como grupos de estudantes, empresas, associações parceiras ou outras entidades relevantes), e cada organização pode ser mencionada em zero ou mais viagens. Esta relação não representa filiação, mas sim participação, apoio ou simples referência a uma organização no contexto da viagem. Assim, permite registrar entidades que foram visitadas, que apoiaram a viagem ou que tiveram algum papel relevante no seu contexto, facilitando a identificação de viagens associadas a determinadas organizações e o acompanhamento das experiências relacionadas.

3.3.4. Utilizador → Organização : n → n

Segundo o **RD3** e o **RD4**, cada utilizador pode pertencer a zero ou mais organizações, e cada organização deve ter pelo menos um utilizador associado. Ao contrário da relação anterior, esta relação representa filiação ou vínculo, indicando a que grupos, empresas ou núcleos cada utilizador pertence. Desta forma, é possível gerir viagens realizadas ou organizadas por esses grupos, bem como identificar os utilizadores associados a cada entidade.

3.3.5. Dia de Viagem → Ponto de interesse : 1 → n

Segundo o **RD13** e o **RD12**, cada dia de viagem pode incluir a visita a zero ou mais pontos de interesse, e cada ponto de interesse está associado a exatamente um dia de viagem específico. Esta estrutura garante que as fotografias e descrições de cada ponto de interesse permaneçam únicas para cada experiência de viagem. Por exemplo, se dois utilizadores diferentes visitarem a Torre Eiffel, cada um terá o seu próprio registo de “Ponto de Interesse” com as suas fotografias específicas, mesmo referindo-se ao mesmo local físico, evitando assim conflitos de dados e garantindo a privacidade das experiências individuais.

Entidade	Relacionamento	Cardinalidade	Participação	Entidade
Dia de Viagem	Constitui	N:1	T:T	Viagem
Viagem	Realizada	N:N	P:T	Utilizador
Viagem	Menciona	N:N	P:P	Organização
Utilizador	Pertence	N:N	T:P	Organização
Ponto de Interesse	Visitado	N:1	P:T	Dia de Viagem

Tabela 10: Relacionamentos identificados.

3.4. Critérios de Definição de Domínios

Antes da caracterização detalhada dos atributos de cada entidade, é relevante apresentar o processo de tomada de decisões que levaram à escolha dos domínios dos atributos:

- Para atributos onde será armazenado um valor inteiro, optou-se por atribuir o tipo de dados `INT`. À medida que se caracterizavam os atributos, julgou-se que nenhum inteiro precisaria de assumir valores superiores ao máximo que este tipo suporta (cerca de dois mil milhões).
- Para guardar valores decimais (custos), optou-se pelo tipo `DECIMAL`, que assume valores decimais exatos, ao contrário dos tipos `FLOAT`. Sabendo que custos de viagem raramente ultrapassam os 10000€, foi escolhido o tipo `DECIMAL(10, 2)`, onde o número dois representa as duas casas decimais para armazenar céntimos.
- Para guardar datas, como “Data” da entidade “Dia de Viagem”, foi trivial a escolha do tipo `Date`, construído para o armazenamento de datas.
- Para guardar referências textuais, utilizou-se o tipo `VARCHAR`. Os critérios de definição de comprimento máximo procuraram manter consistência entre atributos de diferentes entidades. Alguns exemplos incluem `VARCHAR(150)` para títulos e designações, `VARCHAR(100)` para nomes de utilizadores, e `VARCHAR(255)` para endereços eletrónicos.
- Para o armazenamento de campos textuais longos e detalhados, utilizou-se o tipo `TEXT`, que permite textos de comprimentos até 2^{16} caracteres. São exemplos de atributos com este domínio “Descrição” de “Viagem” ou “Descrição” de “Ponto de Interesse”.
- Para o atributo multivvalorado “Fotografias”, considerou-se `TEXT` para armazenar caminhos ou URLs das imagens, sendo a multiplicidade $(0,n)$ expressa através da notação de atributo multivvalorado no diagrama ER.

3.5. Identificação e Caracterização dos Atributos das Entidades e dos Relacionamentos

3.5.1. Viagem

Segundo o requisito **RD5**, uma viagem é composta pelos seguintes atributos:

- **ID** - Identificador único - Número atribuído na criação da viagem no sistema. Permite identificar cada viagem de forma única. Exemplo: 42, `INT`;
- **Título** - Nome ou título atribuído à viagem. É utilizado para a identificação rápida da viagem. Exemplo: Verão no Litoral Português 2025, `VARCHAR(150)`;
- **Descrição** - Texto descritivo da viagem. Fornece informação sobre objetivos, características gerais e expectativas da experiência. Exemplo: Exploração da costa portuguesa com foco em gastronomia local e praias, `TEXT`;

3.5.2. Dia de Viagem

Segundo o requisito **RD10**, cada dia de viagem é composto pelos seguintes atributos:

- **Data** - Data em que este dia específico ocorreu. Permite a organização cronológica dos eventos da viagem. Exemplo: 16/07/2025, `DATE`;
- **Custo** - Valor específico gasto neste dia. Facilita o controlo detalhado das despesas diárias. Exemplo: 45.00, `DECIMAL(8, 2)`;
- **Alojamento** - Designação do local de estadia para este dia. Regista onde o viajante pernoitou. Exemplo: Pousada de São Jonas, `VARCHAR(150)`;
- **Transporte** - Meio de transporte utilizado neste dia. Documenta a mobilidade durante a viagem. Exemplo: Comboio Regional, Automóvel próprio, `VARCHAR(100)`.

3.5.3. Ponto de Interesse

Segundo o requisito **RD11** e **RD14**, cada ponto de interesse é composto pelos seguintes atributos:

- **ID** - Identificador único - Número atribuído na criação do ponto de interesse no sistema. Permite identificar cada ponto de interesse de forma única. Exemplo: 89, `INT`;
- **Nome** - Designação do local ou atividade. É utilizado para a identificação do ponto de interesse. Exemplo: Torre Eiffel, Praia da Rocha, `VARCHAR(150)`;
- **Descrição** - Texto descritivo do ponto de interesse. Fornece informação sobre características, história ou impressões pessoais do local. Exemplo: Monumento icónico construído em 1889, oferece vista panorâmica sobre Paris, `TEXT`;
- **Horário** - Hora da visita ao ponto de interesse. Regista o momento específico da experiência. Exemplo: 09:00, 14:30, `TIME`;
- **Localização** - Morada ou coordenadas geográficas do ponto de interesse. Permite localizar geograficamente o local visitado. Exemplo: Paris, França; 41.1579° N, 8.6291° W, `VARCHAR(200)`;
- **Fotografias** - Conjunto de imagens associadas ao ponto de interesse. Documenta visualmente a experiência no local. Este atributo é multivlorado, permitindo múltiplas fotografias por ponto de interesse. Exemplo: foto1.jpg, foto2.jpg, `TEXT (0,n)`.

3.5.4. Utilizador

Segundo o requisito **RD1**, cada utilizador é composto pelos seguintes atributos:

- **ID** - Identificador único - Número atribuído na criação da conta do utilizador no sistema. Permite identificar cada utilizador de forma única. Exemplo: 128, `INT`;
- **Nome** - Nome completo do utilizador. É utilizado para a identificação do utilizador no sistema. Exemplo: Jonas Prazeres Fernandes, `VARCHAR(100)`;
- **Idade** - Idade do utilizador. Fornece informação demográfica sobre o utilizador. Exemplo: 22, `INT`;
- **Contacto** - Fornece os contactos do utilizador e é composto por:
 - **Telefone** - O contacto telefónico do utilizador. Permite contacto direto com o utilizador. Exemplo: +351 912 345 678, `VARCHAR(15)`.

- **Email** - O endereço de email do utilizador. É utilizado para comunicação e autenticação no sistema.
Exemplo: jonas.praz@email.pt, VARCHAR(150);
- **Admin** - Tipo de permissões do utilizador. É utilizado para determinar o nível de acesso e privilégios no sistema. Utilizadores com Admin TRUE têm permissões elevadas, podendo aceder e consultar informações de todos os utilizadores e viagens do sistema, independentemente da sua participação. Utilizadores comuns (Admin FALSE) têm acesso restrito apenas aos seus próprios dados e às viagens em que participaram. Este atributo é fundamental para o controlo de acessos e segurança do sistema. Exemplo: 0, BOOLEAN;

3.5.5. Organização

Segundo o requisito RD2, cada organização é composta pelos seguintes atributos:

- **ID** - Identificador único - Número atribuído na criação da organização no sistema. Permite identificar cada organização de forma única. Exemplo: 201, INT;
- **Nome** - Designação oficial da organização. É utilizado para a identificação da entidade coletiva. Exemplo: CeBELUM, Núcleo de Estudantes de Engenharia, VARCHAR(150).

3.6. Apresentação e Explicação do Diagrama ER Produzido

Após a identificação e caracterização de entidades, relacionamentos e os seus atributos, a equipa de desenvolvimento procedeu à esquematização de um diagrama Entidade-Relacionamento. Este processo foi realizado com recurso ao software BR Modelo, para a geração de um diagrama conceptual em formato digital. Além de mais, o diagrama foi construído na íntegra de uma só vez, considerando-se portanto, todas as vistas de utilização em simultâneo.

O processo de construção do diagrama ER foi dividido em quatro etapas principais:

1^a Inserção de entidades e atributos: Foram, numa primeira fase, inseridas todas as entidades identificadas, juntamente com os seus respetivos atributos, fazendo uso da simbologia adequada da notação de Chen. Nesta fase, foram também adicionados ao modelo os domínios de todos os atributos, bem como a informação sobre quais constituíam identificadores de cada entidade e quais eram opcionais (a tracejado). Os atributos compostos foram representados com ligações aos seus componentes, e os atributos multivariados foram assinalados com a notação de acordo com os requisitos.

2^a Adição de relacionamentos: Foram adicionados todos os relacionamentos entre entidades, representados por losangos, bem como as suas cardinalidades e participações utilizando a notação Min-Max de Abrial. Esta notação, expressa como (*min*, *max*) junto a cada entidade participante no relacionamento, permite especificar de forma precisa os limites de participação. Por exemplo, no relacionamento “Constitui” entre “Viagem” e “Dia de Viagem”, a notação (1,n) junto a “Dia de Viagem” indica que cada viagem é constituída por um ou mais dias, enquanto (1,1) junto a “Viagem” indica que cada dia pertence a exatamente uma viagem.

3^a Verificação de requisitos: Todos os requisitos de descrição foram relidos, verificando-se que o modelo desenvolvido não impedia o cumprimento de nenhum deles. Esta validação cruzada entre requisitos e modelo conceptual foi essencial para garantir a completude do diagrama.

4^a Organização gráfica: Por último, ajustou-se o posicionamento espacial dos elementos gráficos do diagrama, procurando minimizar o cruzamento de linhas e maximizar a legibilidade. A entidade mais central ao domínio (“Viagem”) foi posicionada de forma proeminente, com as entidades relacionadas distribuídas à sua volta de modo lógico.

Segue-se o modelo conceptual final produzido:

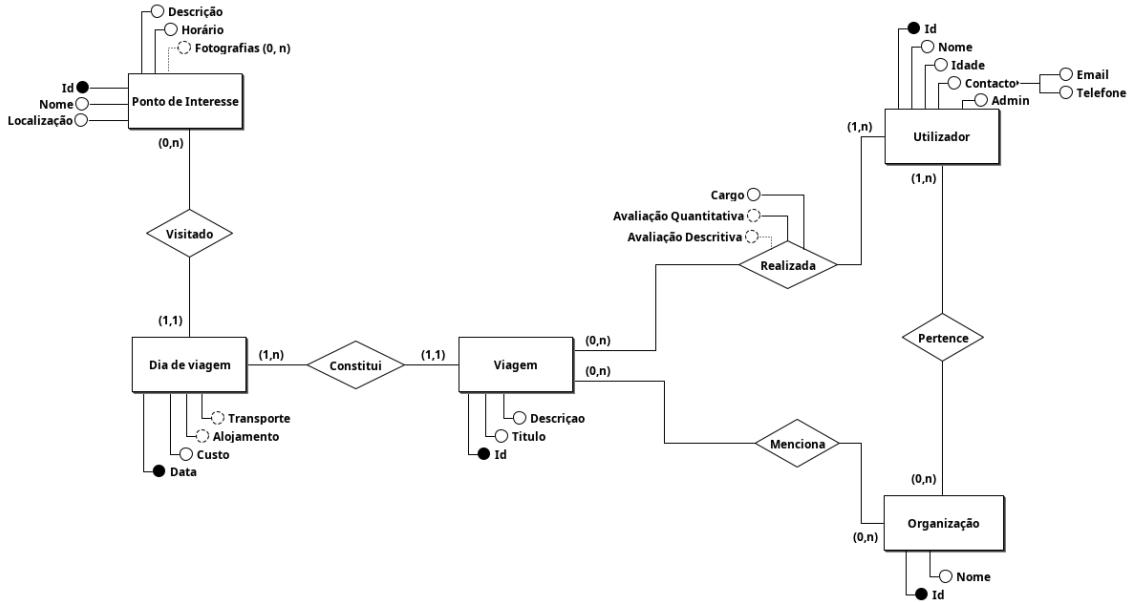


Figura 5: Modelo Conceptual final

3.7. Validação do esquema

Após terminar a elaboração do modelo conceptual e da sua devida documentação, a equipa de desenvolvimento realizou um processo de validação envolvendo uma revisão interna e, posteriormente, a apresentação ao grupo CeBELUM.

3.7.1. Revisão Interna

Numa primeira fase, a equipa procedeu a uma revisão sistemática do modelo, verificando:

- **Completude:** Confirmou-se que todos os requisitos de descrição levantados encontravam representação adequada no modelo, seja através de entidades, relacionamentos ou atributos.
- **Correção das Cardinalidades:** Analisaram-se cuidadosamente todas as cardinalidades e participações, garantindo que refletiam corretamente os requisitos. Foi dada particular atenção à distinção entre participação total e parcial, assegurando que, por exemplo, um “Ponto de Interesse” não pode existir sem estar associado a um “Dia de Viagem” (participação total), mas uma “Organização” pode existir sem estar mencionada em nenhuma viagem (participação parcial).

3.7.2. Validação do CeBELUM

Após a aprovação da revisão interna, o modelo foi apresentado ao CeBELUM. Esta apresentação incluiu:

- Explicação detalhada de cada entidade, relacionamento e respetivos atributos;
- Demonstração de como o modelo suporta os requisitos levantados;
- Exemplos práticos de instâncias do modelo, usando os dados fictícios previamente definidos (como a viagem “Verão no Litoral Português 2025”).

Durante esta validação, foram identificados alguns pontos que necessitavam de clarificação:

Atributo “Fotografias”: Inicialmente havia alguma ambiguidade sobre se as fotografias deveriam ser uma entidade separada ou um atributo multivalorado de “Ponto de Interesse”. Após discussão, confirmou-se que a modelação como atributo multivalorado era adequada para os requisitos atuais do sistema, mantendo a simplicidade do modelo sem comprometer a funcionalidade.

Cardinalidade do relacionamento “Pertence”: Foi discutida a cardinalidade mínima de “Utilizador” no relacionamento “Pertence”. A especificação (1, n) no diagrama sugeria que todo o utilizador deveria pertencer

a pelo menos uma organização. Após análise dos requisitos, confirmou-se que a cardinalidade correta deveria ser $(0,n)$, permitindo utilizadores individuais sem filiação organizacional. Esta correção foi imediatamente incorporada no modelo.

Relacionamento “Visitado”: O centro de estudantes (CeBELUM) validou a decisão de modelar cada visita a um local como um “Ponto de Interesse” distinto, mesmo quando múltiplas viagens visitam o mesmo local físico, garantindo assim a independência das fotografias e descrições de cada experiência.

3.7.3. Aprovação Final

Após a incorporação das correções identificadas e uma segunda revisão do modelo atualizado, o modelo conceptual foi formalmente aprovado pelos responsáveis do projeto, permitindo o avanço para a fase de modelação lógica. O modelo validado constitui assim a base sólida sobre a qual será construída a estrutura da base de dados do sistema Bela Vista.

4. Modelação Lógica

4.1. Construção e Validação do Modelo de Dados Lógico

A conversão do modelo conceptual para o modelo lógico constitui um processo de transformação dos seus elementos em tabelas e relacionamentos concretos.

Cada entidade identificada no modelo conceptual foi representada por uma tabela no modelo lógico, preservando os seus atributos. Os relacionamentos entre as entidades foram implementados por meio de chaves estrangeiras ou tabelas adicionais, conforme a cardinalidade envolvida.

A conversão seguiu uma abordagem estratégica, iniciando pelas entidades com menos dependências e avançando progressivamente para aquelas mais interligadas.

4.2. Apresentação e Explicação do Modelo Lógico Produzido

O processo de conversão teve início com a identificação das entidades fundamentais do sistema, priorizando aquelas com menor número de dependências ou sem dependências (sem chaves estrangeiras) permitindo uma base para a futura criação de tabelas que dependem de outras através das suas chaves estrangeiras.

Passo a passo, o modelo conceptual foi transformado em lógico de maneira estruturada. Esta evolução envolveu definir as relações, normalizar os dados e validar o esquema em cada etapa, o que consolidou uma base de dados que reflete com eficiência a realidade da informação.

4.2.1. Derivação de Relações do Modelo de Dados Local

4.2.1.1. Utilizador

A tabela “Utilizador” foi criada com base na entidade “Cliente” no processo de modelação lógica da base de dados, esta tem como objetivo o registo de indivíduos que publicam as viagens.

Os atributos simples “Id”, “Nome” e “Idade” deram origem a atributos nesta tabela. Para além disso, na conversão do modelo conceptual para o esquema lógico, o atributo composto “Contacto” foi derivado em “Telefone” e “Email”.

O atributo “ID” foi definido como chave primária de forma a garantir a identificação única de cada utilizador.

Tendo em conta que os relacionamentos que a entidade “Utilizador” tem com outras entidades são N:N, nenhum deles leva à criação de uma chave estrangeira nesta tabela.

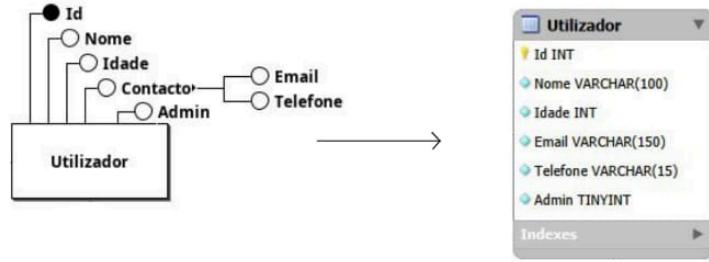


Figura 6: Conversão da Entidade Utilizador - Modelo Lógico

4.2.1.2. Viagem

A tabela “Viagem” foi criada com base na entidade “Viagem” no processo de modelação lógica da base de dados, esta tem como objetivo guardar as informações básicas de cada viagem.

O atributo “Id” foi definido como chave primária de forma a garantir a identificação única de cada viagem. Além disso, foram incluídos os atributos simples “Título” e “Descrição”.

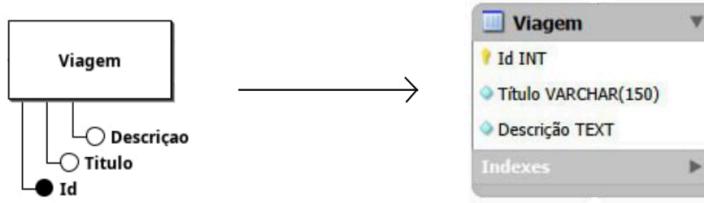


Figura 7: Conversão da Entidade Viagem - Modelo Lógico

4.2.1.3. ViagemRealizadaUtilizador

O relacionamento “Realizada” representa a associação entre as viagens e os utilizadores.

Como este relacionamento é N:M, foi criada a tabela “ViagemRealizadaUtilizador”, que implementa este relacionamento muitos-para-muitos através de associação.

A tabela contém duas chaves estrangeiras para as entidades Viagem e Utilizador, “Id_Viagem” e “Id_Utilizador”, formando uma chave primária composta e inclui os atributos próprios do relacionamento como “Cargo”, “Avaliação Descritiva” e “Avaliação Quantitativa”.

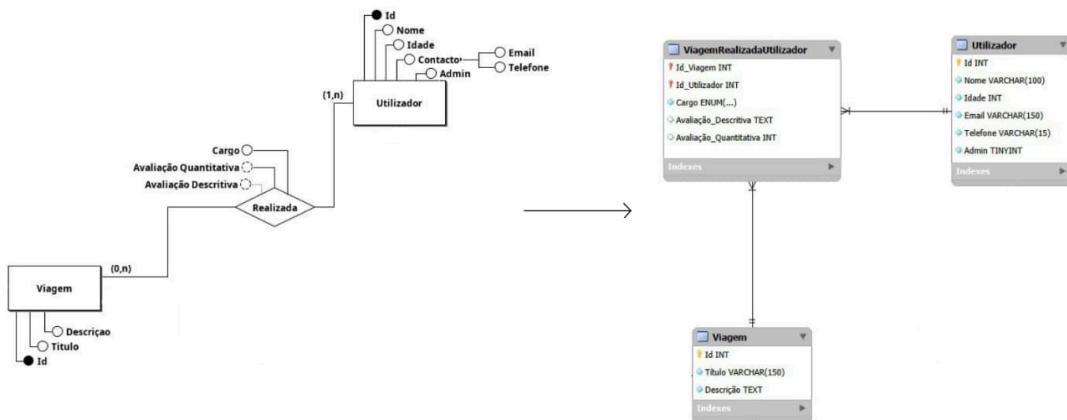


Figura 8: Conversão do Relacionamento Realizada - Modelo Lógico

4.2.1.4. Organização

A tabela “Organização” foi criada com base na entidade “Organização” no processo de modelação lógica da base de dados.

O atributo “Id” foi definido como chave primária de forma a garantir a identificação única de cada organização. Adicionalmente, tem o atributo simples “Nome”.

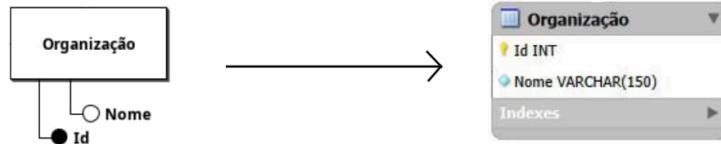


Figura 9: Conversão da Entidade Organização - Modelo Lógico

4.2.1.5. ViagemMencionaOrganização

O relacionamento “Menciona” representa a associação entre as Viagens e as Organizações.

Tendo em conta que este relacionamento é N:M, foi criada a tabela “ViagemMencionaOrganização”, que implementa este relacionamento muitos-para-muitos através de associação.

A tabela contém duas chaves estrangeiras para as entidades Viagem e Organização, “Id_Viagem” e “Id_Organização”, formando uma chave primária composta.

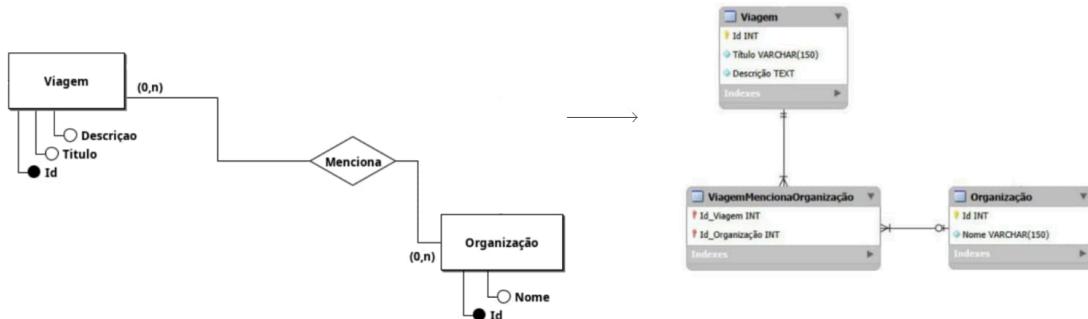


Figura 10: Conversão do Relacionamento Menciona - Modelo Lógico

4.2.1.6. UtilizadorPertenceOrganização

O relacionamento “Pertence” representa a associação entre os Utilizadores e as Organizações.

Foi criada a tabela “UtilizadorPertenceOrganização”, que implementa este relacionamento muitos-para-muitos através de associação. A tabela contém duas chaves estrangeiras para as entidades Viagem e Organização, “Id_Utilizador” e “Id_Organização”, formando uma chave primária composta.

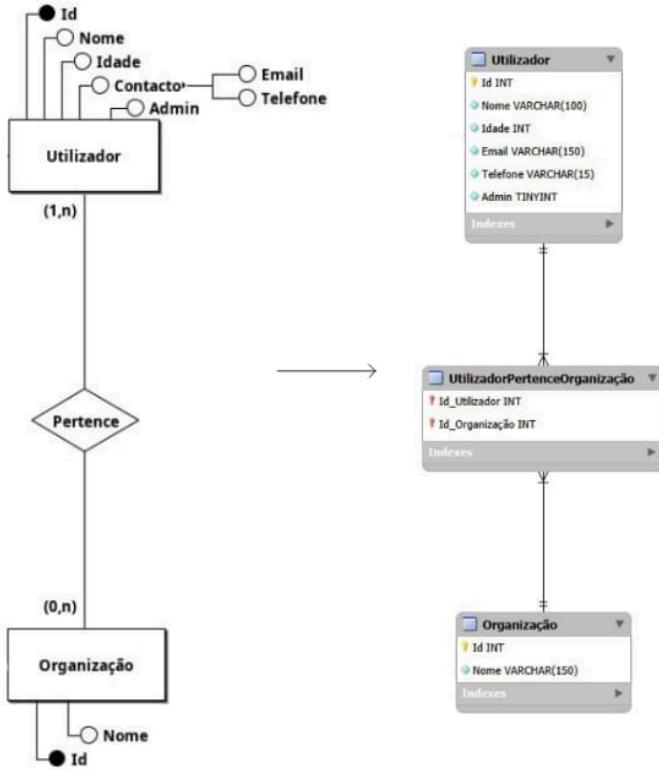


Figura 11: Conversão do Relacionamento Pertence - Modelo Lógico

4.2.1.7. DiadeViagem

A tabela “DiadeViagem” foi criada através da entidade “Dia de Viagem” como parte do processo de modelação lógica da base de dados, destinada a identificar cada dia de uma viagem.

Foram incluídos os atributos “Data”, “Custo”, “Alojamento” e “Transporte”.

Considerando o relacionamento 1:N entre “Viagem” e “Dia de Viagem”, a tabela possui também uma chave estrangeira “Id_Viagem”, que referencia a chave primária “Id” da tabela “Viagem”.

Como chave primária, optou-se por uma chave composta formada por “Data” e “Id_Viagem”. Apenas a “Data” não seria suficiente para identificar de forma única cada registo, uma vez que a mesma data pode ocorrer em viagens diferentes. Assim, a combinação de “Data” com “Id_Viagem” garante a unicidade de cada dia de viagem no sistema.

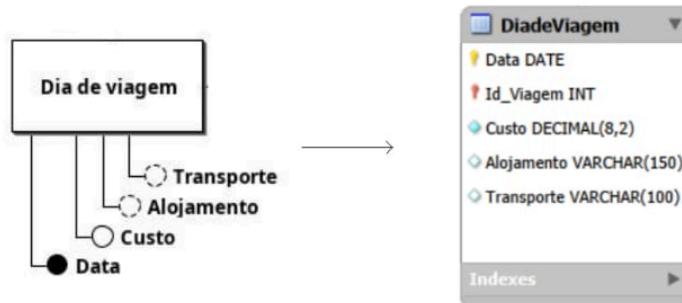


Figura 12: Conversão da Entidade Dia de Viagem - Modelo Lógico

4.2.1.8. Ponto de Interesse

A tabela “PontodeInteresse” foi criada através da entidade “Ponto de Interesse” como parte do processo de modelação lógica da base de dados, destinada a identificar um ponto de interesse da viagem.

O atributo “Id” foi definido como chave primária de forma a garantir a identificação única de local. Além disso, foram incluídos os atributos simples “Nome”, “Localização”, “Descrição” e “Horário”.

Para associar cada ponto de interesse a um dia específico de uma viagem, a tabela inclui também duas chaves estrangeiras: “Data” e “Id_Viagem”. Juntas, formam uma referência à chave primária composta da tabela “DiaDeViagem”, estabelecendo assim o relacionamento que indica em qual dia de qual viagem o ponto foi visitado.

A derivação do atributo multivvalorado “Fotografia” resultou na criação de uma tabela independente denominada “Fotografias”, com a chave primária “Id_Fotografia” e uma chave estrangeira “Id_PontoInteresse”, que referencia a chave primária “Id” da tabela “PontodeInteresse”. A tabela “Fotografias” inclui ainda o atributo “Url_Fotografia”, onde é armazenado o endereço da imagem. Optou-se por não utilizar uma chave primária composta por “Id_PontoInteresse” e “Url_Fotografia”, uma vez que o atributo Url_Fotografia é do tipo TEXT, o que inviabiliza a sua utilização como parte de uma chave primária. Justificando assim a criação da chave primária “Id_Fotografia”.

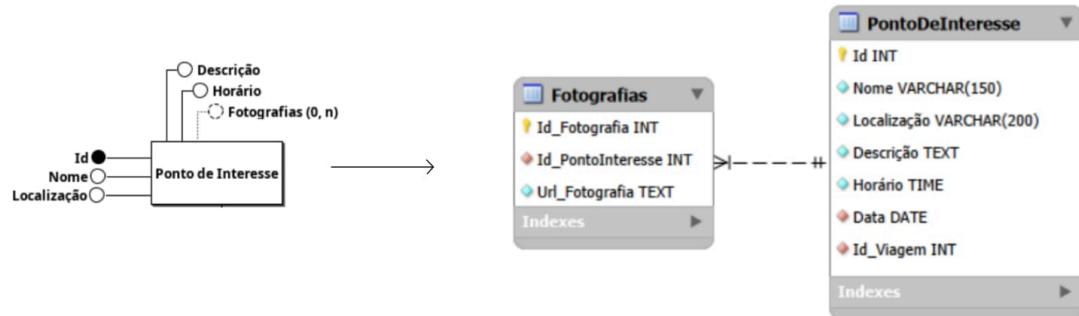


Figura 13: Conversão da Entidade Ponto de Interesse - Modelo Lógico

4.2.2. Esquema Lógico Final

Na imagem seguinte, encontra-se o modelo lógico da base de dados desenvolvido.

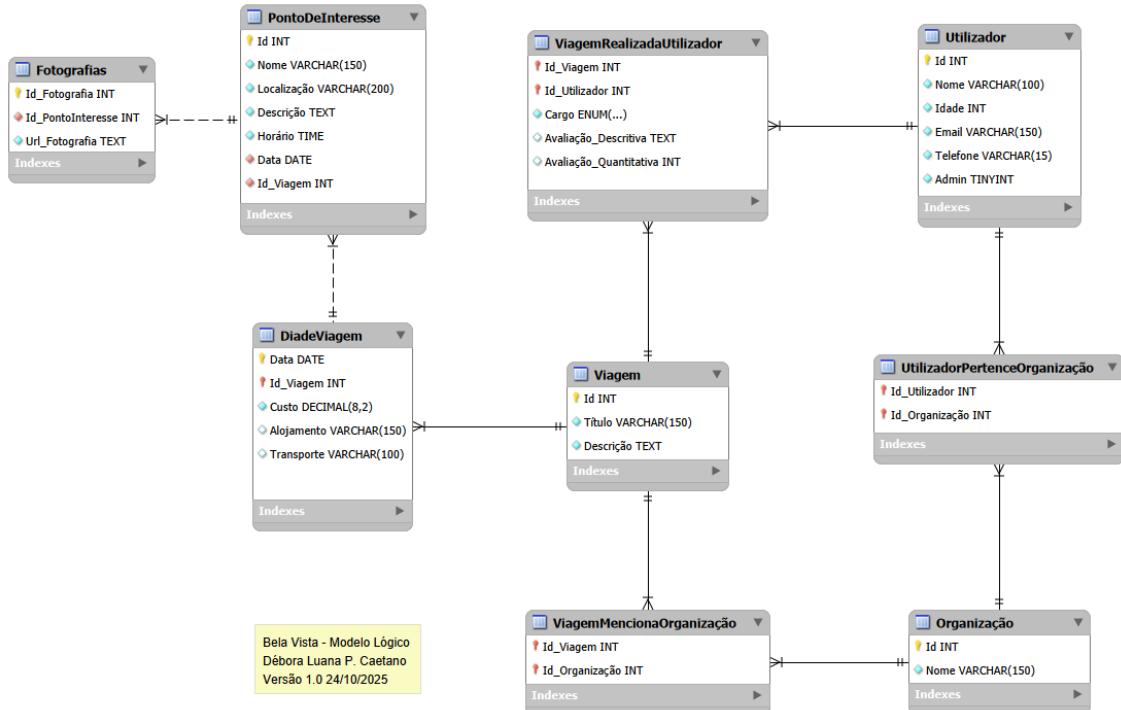


Figura 14: Modelo lógico da base de dados

Para uma melhor leitura do modelo este foi organizado por áreas de trabalho, como é possível verificar na figura abaixo.

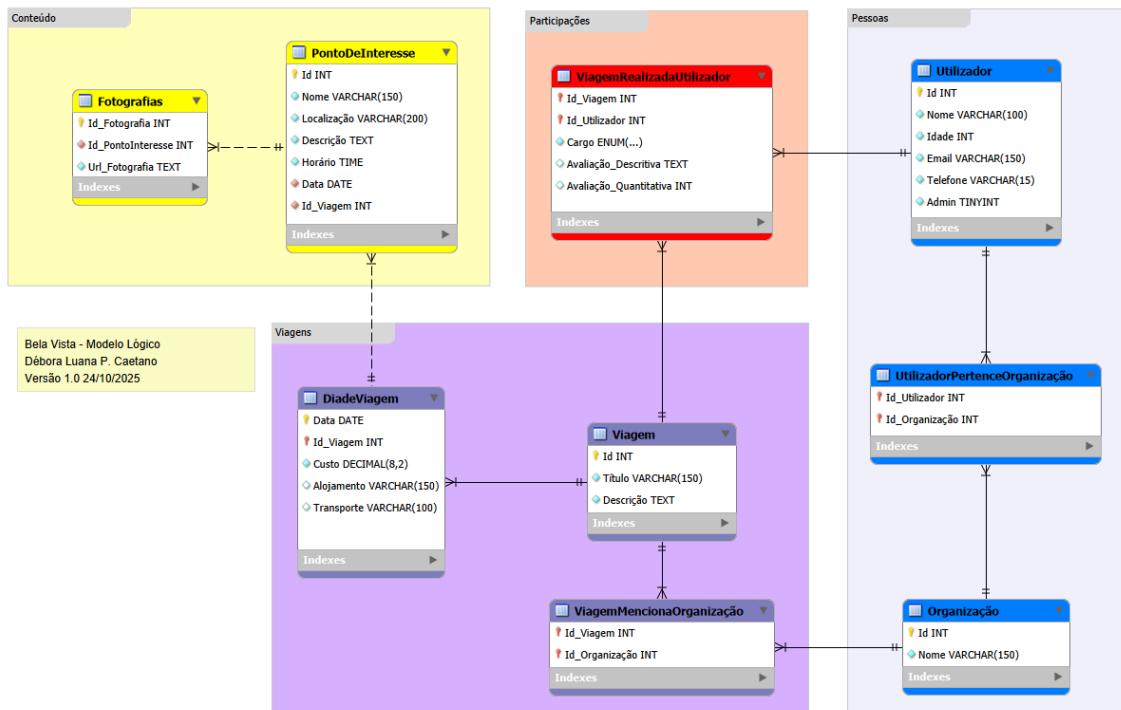


Figura 15: Modelo lógico da base de dados

4.3. Normalização de Dados

Ao criar e desenvolver uma base de dados, é fundamental aplicar o conceito de **normalização de dados**. Este processo assegura que o modelo resultante é um modelo mais próximo da organização que representa, mais consistente, com um baixo grau de redundância e grande estabilidade.

Organiza a informação para garantir dois objetivos principais:

- **Integridade dos Dados** - assegurar que a informação armazenada é precisa e consistente.
- **Redundância controlada** - evitar a duplicação desnecessária de dados, que pode causar inconsistências e desperdício de espaço.

Para alcançar estes objetivos, estão definidas várias regras as quais chamamos **Formas Normais (FN)**.

Nesta secção queremos mostrar que a nossa proposta satisfaz as três primeiras Formas Normais: a 1FN, 2FN e 3FN.

Depois de uma análise cuidada do nosso modelo podemos afirmar que:

- **1FN:** Existe uma chave primária em cada tabela e todos os atributos são atómicos, ou seja, nenhuma das tabelas contém valores repetidos e nem atributos que possuem mais do que um valor.

Para o exemplo do modelo lógico em questão, a entidade “Ponto de Interesse” possui um atributo multi-valorado “Fotografias”. Para este caso, para normalizar, foi construída uma nova tabela com o atributo em questão fazendo uma ligação entre as duas tabelas através de uma chave estrangeira na tabela “Fotografias” que referencia a chave primária da tabela “Ponto de Interesse”.

- **2FN:** Cada atributo de uma tabela depende totalmente da chave primária, assegurando a conformidade com a segunda forma normal (2FN), eliminando dependências parciais.

Para exemplificar esta conformidade, consideremos a tabela “ViagemRealizadaUtilizador”, que estabelece a relação entre utilizadores e viagens através de uma chave primária composta por (Id_Viagem, Id_Utilizador). Esta tabela contém os atributos não-chave Cargo, Avaliação_Descritiva e Avaliação_Quantitativa. A 2FN verifica-se porque cada um destes atributos depende totalmente da combinação completa da chave primária. O atributo “Cargo” não pode ser determinado apenas pelo “Id_Viagem”, pois o mesmo não identifica qual o utilizador em questão, nem apenas pelo “Id_Utilizador”, pois um utilizador pode ter diferentes cargos em diferentes viagens. Apenas a combinação de ambos os identificadores (“Id_Viagem” e “Id_Utilizador”) permite determinar univocamente qual o cargo que determinado utilizador desempenhou numa viagem específica. O mesmo raciocínio aplica-se aos atributos Avaliação_Descritiva e Avaliação_Quantitativa, que representam a avaliação de um utilizador numa participação específica, dependendo portanto da totalidade da chave primária composta.

- **3FN:** Todos os atributos que não pertencem à chave primária devem ser funcionalmente independentes uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela.

Consideremos novamente a tabela “ViagemRealizadaUtilizador”, que possui a chave primária composta (Id_Viagem, Id_Utilizador) e os atributos não-chave Cargo, Avaliação_Descritiva e Avaliação_Quantitativa. A conformidade com a 3FN verifica-se porque todos os atributos não-chave são funcionalmente independentes entre si. O atributo Cargo não determina nem a “Avaliação_Descritiva” nem a “Avaliação_Quantitativa”, pois utilizadores com o mesmo cargo podem receber avaliações diferentes. Da mesma forma, a “Avaliação_Descritiva” não determina a “Avaliação_Quantitativa”, uma vez que dois participantes podem ter descrições diferentes mas pontuações iguais, ou vice-versa. Cada um destes atributos depende exclusivamente da chave primária (Id_Viagem, Id_Utilizador) e não existe qualquer relação de dependência funcional entre eles.

Portanto, com base nesta análise, pode-se afirmar que o modelo lógico está normalizado até à terceira forma normal (3FN).

4.4. Validação do Modelo com Interrogações do Utilizador

Para validar o modelo, a equipa recorreu à álgebra relacional para expressar as interrogações definidas pelos requisitos de manipulação de dados. Para este efeito, foi utilizada a calculadora de álgebra relacional RelaX.

A execução de interrogações na calculadora RelaX requer a definição prévia do esquema das tabelas. Esta definição consiste apenas na listagem dos atributos e respetivos tipos, não sendo necessário especificar chaves primárias ou estrangeiras. O exemplo seguinte ilustra esta especificação:

```
Utilizador = {
    Id:number, Nome:string, Idade:number, Email:string, Telefone:string
    1, 'Maria Oliveira', 28, 'maria.oliveira@example.com', '912345678'
    2, 'João Silva', 35, 'joao.silva@example.com', '913456789'
}
```

Cada tipo de dados do MySQL foi mapeado para os tipos suportados pela calculadora como se pode ver na tabela seguinte.

MySQL	RelaX
DATE	date
DECIMAL(X,Y)	number
INT, INT(X)	number
TEXT	string
VARCHAR(X)	string

Tabela 11: Mapeamento de domínios entre MySQL e RelaX.

Seguem-se então algumas expressões em Álgebra Relacional que foram desenvolvidas para representarem algumas *queries* enunciadas nos requisitos de manipulação estabelecidos anteriormente.

1ª Expressão - RM5

RM5 - Listar todos os utilizadores pertencentes a uma organização:

Uma vez que cada utilizador está associado a uma organização através do relacionamento “Pertence”, esta interrogação torna-se direta. Utiliza-se uma instrução preparada que recebe o identificador da organização e procura na tabela “UtilizadorPertenceOrganização” todos os registo correspondentes. Para se tornar mais legível, utilizamos o operador ρ para atribuir sinónimos às tabelas. Através de uma junção (\bowtie) com a tabela “Utilizador”, obtém-se o nome de cada utilizador que com a operação de seleção (σ), filtram-se os registo pela organização pretendida. Por fim, a operação de projeção (π) gera uma tabela contendo apenas os atributos pedidos: identificador do utilizador (“Id_Utilizador”) e o seu nome (“Nome”). Para a sua execução, substituímos o parâmetro por 2.

Árvore:

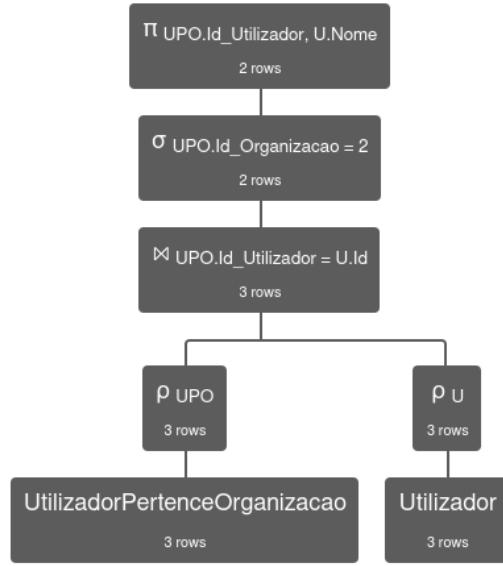


Figura 16: RM5 - Árvore álgebra relacional

Expressão:

$$\Pi_{UPO.Id_Utilizador, U.Nome} \sigma_{UPO.Id_Organizacao = 2} (\rho_{UPO} \\ UtilizadorPertenceOrganizacao \bowtie_{UPO.Id_Utilizador = U.Id} \rho_u Utilizador)$$

Figura 17: RM5 - Expressão álgebra relacional

2ª Expressão - RM11

RM11 - Calcular as datas de ida e regresso das viagens com base nos seus dias:

Para esta query apenas seria necessária a operação sobre a tabela “DiaDeViagem”, contudo, como queremos aceder ao Titulo de cada viagem, é necessário realizar uma junção (\bowtie) da tabela “DiaDeViagem” com “Viagem” através da correspondência entre os identificadores da viagem em cada uma das tabelas. Para cada “v.Id” será calculado a menor e maior data da tabela “DiaDeViagem” denominadas por DataIda e DataRegresso, respetivamente e serão agrupadas através da operação γ . Por fim usamos π para projetar as colunas pretendidas e ρ para atribuirmos nomes mais legíveis.

Árvore:

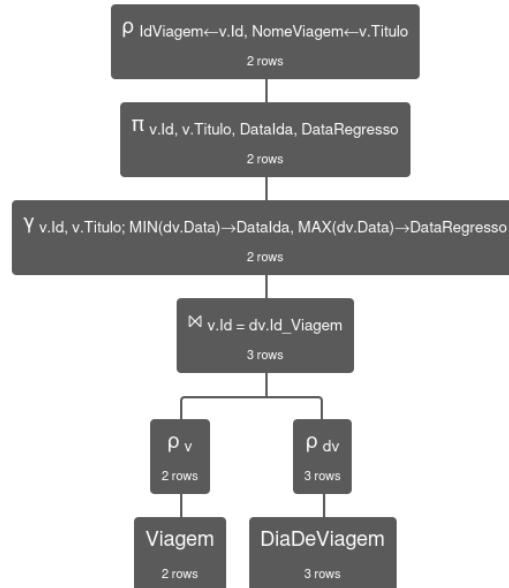


Figura 18: RM11 - Árvore Álgebra Relacional

Expressão:

$$\rho \text{ IdViagem} \leftarrow \text{v.Id}, \text{NomeViagem} \leftarrow \text{v.Titulo} \quad \Pi \text{ v.Id, v.Titulo, DataIda, DataRegresso} \quad \gamma \text{ v.Id, v.Titulo;} \\ \text{MIN(dv.Data)} \rightarrow \text{DataIda}, \text{MAX(dv.Data)} \rightarrow \text{DataRegresso} \quad (\rho \text{ v} \text{ Viagem} \bowtie \text{ v.Id} = \text{dv.Id_Viagem} \rho \text{ dv} \text{ DiaDeViagem})$$

Figura 19: RM11 - Expressão álgebra relacional

3ª Expressão - RM12

RM12 - Calcular o custo total das viagens com base nos custos dos seus dias:

A execução desta query é de certa forma parecida com a anterior. Realizamos uma junção (\bowtie) das tabelas “Viagem” e “DiaDeViagem”, novamente atribuindo-lhes nomes que tornam a leitura mais fácil através do operador ρ . Para cada viagem é calculado o seu custo que será a soma de todos os “DV.Custo” de cada dia de viagem associado aos identificadores da viagem e são agrupados com o operador γ . Por fim, são projetados apenas os atributos necessários e os registo são ordenados de forma crescente pelo seu identificador de viagem através do operador τ .

Árvore:

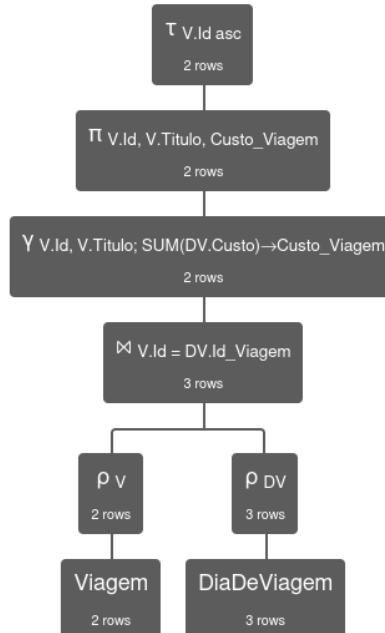


Figura 20: RM12 - Árvore Álgebra Relacional

Expressão:

$$T V.Id asc \Pi V.Id, V.Titulo, Custo_Viagem \gamma V.Id, V.Titulo; SUM(DV.Custo) \rightarrow Custo_Viagem (\rho V \\ Viagem \bowtie V.Id = DV.Id_Viagem \rho DV DiaDeViagem)$$

Figura 21: RM12 - Expressão Álgebra Relacional

4ª Expressão - RM21

RM21 - Listar todos os participantes de uma viagem:

Esta query faz uso de uma instrução preparada que como sendo incompatível com a ferramenta RelaX, é necessário substituir o parâmetro ? utilizado em SQL por um valor específico. Neste caso iremos assumir ? = 1. Para saber os participantes da viagem com Id = 1, é feita uma junção (\bowtie) entre as tabelas “Viagem” e “ViagemRealizadaUtilizador”. Esta junção (\bowtie) já é suficiente para acedermos aos atributos “Id_Utilizador” e “Cargo”. Contudo, como queremos adicionalmente os nomes dos utilizadores, é necessário fazer mais uma junção com a tabela “Utilizador”. Só resta realizar uma seleção (σ) dos registo onde o identificador da viagem (“V.Id”) seja igual a 1 e simultaneamente que o cargo do utilizador (“VRU.Cargo”) seja de participante.

Árvore:

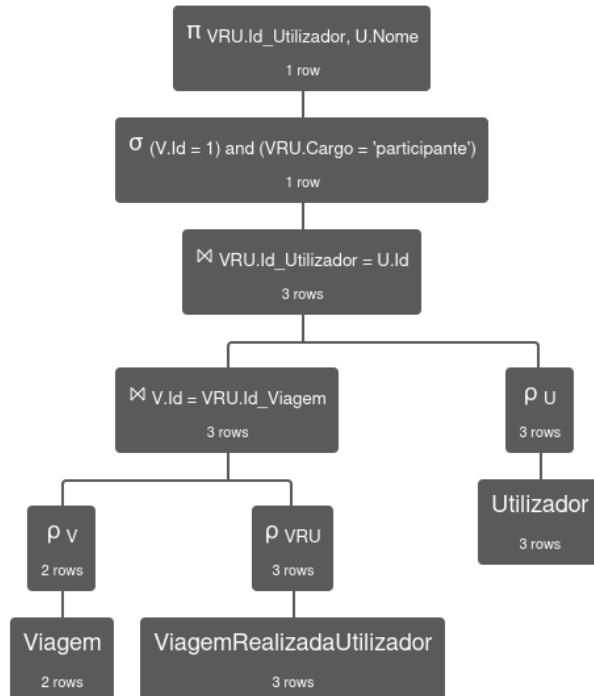


Figura 22: RM21 - Árvore Álgebra Relacional

Expressão:

$$\begin{aligned}
 & \pi_{VRU.Id_Utilizador, U.Nome} \sigma_{(V.Id = 1) \text{ and } (VRU.Cargo = 'participante')} ((\rho_v Viagem \bowtie_{V.Id} \\
 & = VRU.Id_Viagem \rho_{VRU} ViagemRealizadaUtilizador) \bowtie_{VRU.Id_Utilizador = U.Id} \rho_u \\
 & Utilizador)
 \end{aligned}$$

Figura 23: RM21 - Árvore Álgebra Relacional

5ª Expressão - RM28

RM28 - Identificar o utilizador com maior número de viagens realizadas:

Para esta query queremos o identificar (“u.Id”) e o nome do utilizador (“u.Nome”) com o maior número de viagens realizadas. Para isso, já sabemos que terá de ser feita algures na query uma projeção (π) destes atributos. Para tal, precisamos de realizar uma junção (\bowtie) entre as tabelas “Utilizador” e “ViagemRealizadaUtilizador” através do identificador do utilizador. De seguida, para cada “u.Id” é feita a contagem dos respetivos “vru.Id_Viagem”. Os registos são então ordenados (τ) pelo número total de viagens de forma decrescente para termos no topo o maior número de viagens realizadas e é feita a limitação para apenas um registo que nos dará o utilizador correspondente a esse valor.

Árvore:

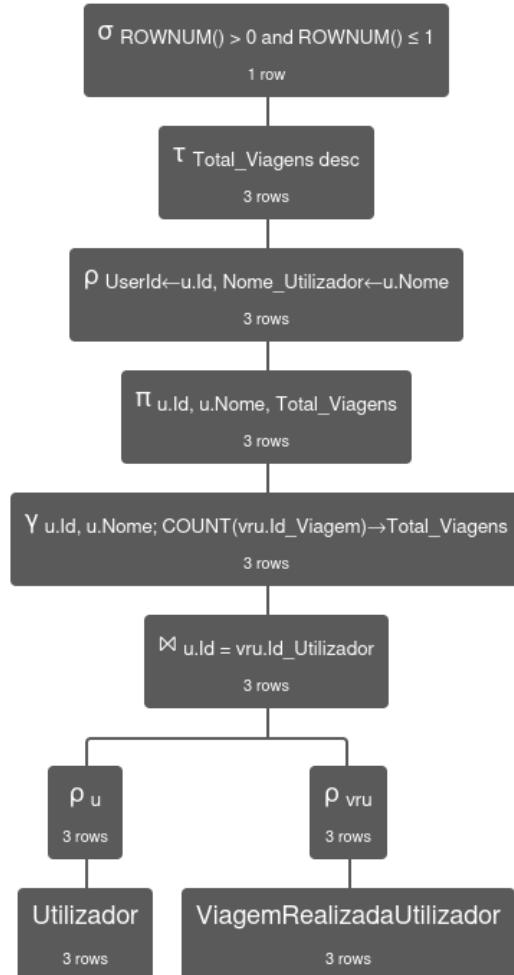


Figura 24: RM28 - Árvore Álgebra Relacional

Expressão:

$$\begin{aligned}
 & \sigma \text{ ROWNUM()} > 0 \text{ and ROWNUM()} \leq 1 \quad \tau \text{ Total_Viagens desc} \quad \rho \text{ UserId} \leftarrow u.Id, \\
 & \text{Nome_Utilizador} \leftarrow u.Nome \quad \Pi \text{ u.Id, u.Nome, Total_Viagens} \quad \gamma \text{ u.Id, u.Nome;} \\
 & \text{COUNT}(vru.Id_Viagem) \rightarrow \text{Total_Viagens} \quad (\rho \text{ u Utilizador} \bowtie \text{u.Id} = vru.Id_Utilizador \rho \text{ vru} \\
 & \text{ViagemRealizadaUtilizador})
 \end{aligned}$$

Figura 25: RM28 - Expressão Álgebra Relacional

6^a Expressão - RM30

RM30 - Listar as avaliações (média de avaliação quantitativa) por viagem:

Para cada viagem é necessário aceder ao campo “Avaliacao_Quantitativa”, para isso é feita uma junção (\bowtie) entre as tabelas “Viagem” e “ViagemRealizadaUtilizador” pois para além do identificador da viagem, queremos saber o título da viagem. Para cada viagem é calculada a média da “Avaliacao_Quantitativa” através da função AVG e é feita uma ordenação (τ) decrescente de acordo com esse valor.

Árvore:

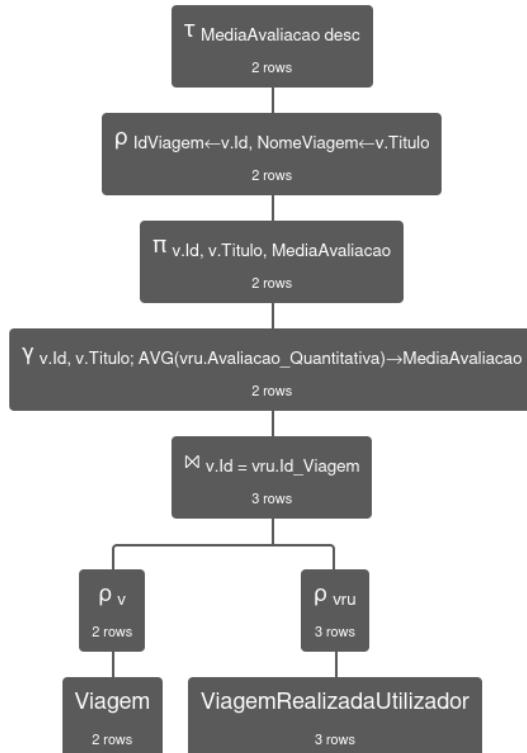


Figura 26: RM30 - Árvore Álgebra Relacional

Expressão:

$$T \text{ MediaAvaliacao desc } \rho \text{ IdViagem} \leftarrow v.\text{Id}, \text{NomeViagem} \leftarrow v.\text{Titulo} \Pi v.\text{Id}, v.\text{Titulo}, \text{MediaAvaliacao} \ Upsilon v.\text{Id}, v.\text{Titulo}; \text{AVG}(vru.\text{Avaliacao}_\text{Quantitativa}) \rightarrow \text{MediaAvaliacao} (\rho v \text{ Viagem} \bowtie v.\text{Id} = vru.\text{Id}_\text{Viagem} \rho vru \text{ ViagemRealizadaUtilizador})$$

Figura 27: RM30 - Expressão Álgebra Relacional

7^a Expressão - RM32

RM32 - Listar quais as organizações com os membros mais ativos (que realizam mais viagens) e melhor avaliam:

Esta query tem como objetivo principal criar um ranking das organizações, ordenando-as primeiro pelas que geram mais atividade (viagens) e, em caso de empate, pelas que têm membros mais satisfeitos.

Árvore:

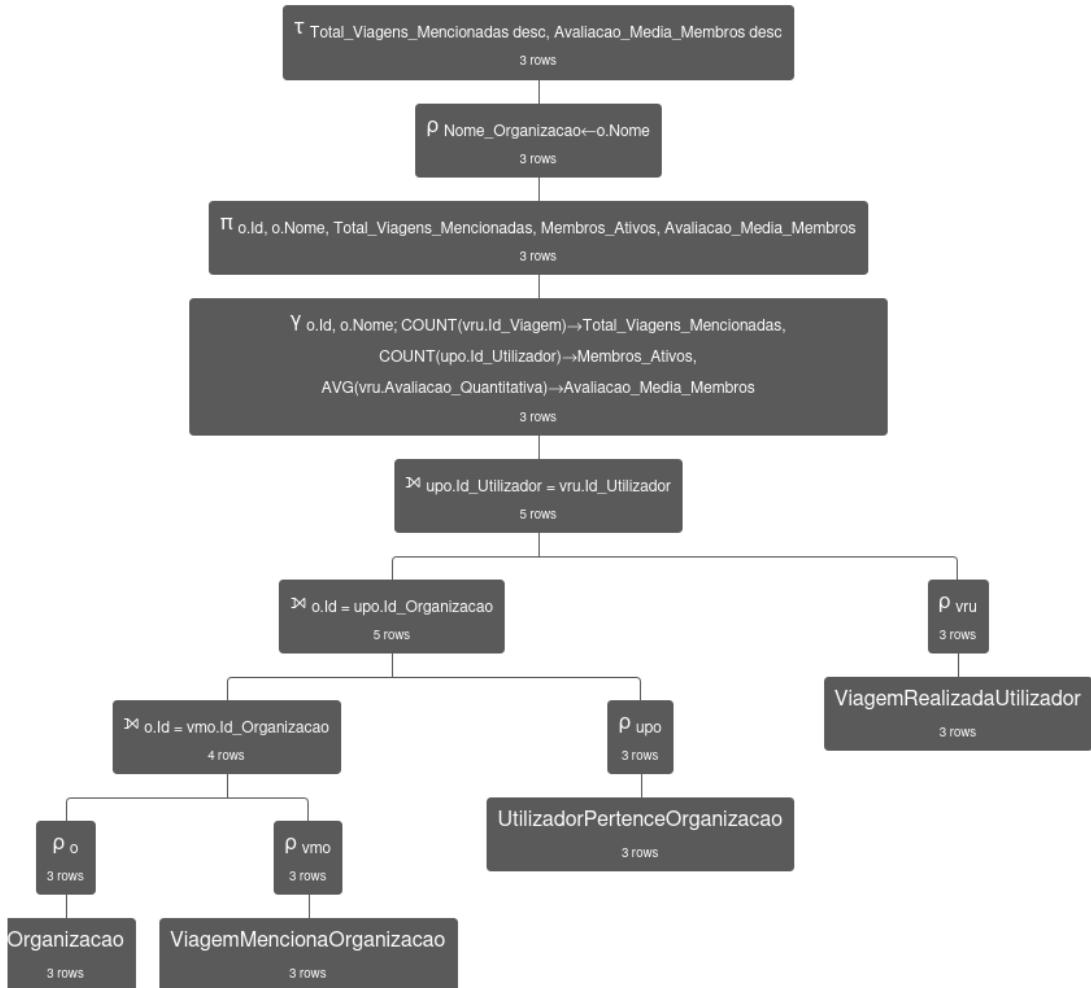


Figura 28: RM32 - Árvore Álgebra Relacional

Expressão:

```

    T Total_Viagens_Mencionadas desc, Avaliacao_Media_Membros desc P Nome_Organizacao<-o.Nome Pi o.Id,
    o.Nome, Total_Viagens_Mencionadas, Membros_Aktiv, Avaliacao_Media_Membros Y o.Id, o.Nome;
    COUNT(vru.Id_Viagem)→Total_Viagens_Mencionadas, COUNT(upo.Id_Utilizador)→Membros_Aktiv,
    AVG(vru.Avaliacao_Quantitativa)→Avaliacao_Media_Membros ( ( ( P o Organizacao &gt; o.Id =
    vmo.Id_Organizacao P vmo ViagemMencionaOrganizacao ) &gt; o.Id = upo.Id_Organizacao P upo
    UtilizadorPertenceOrganizacao ) &gt; upo.Id_Utilizador = vru.Id_Utilizador P vru
    ViagemRealizadaUtilizador )
  
```

Figura 29: RM32 - Expressão Álgebra Relacional

Ressalva-se que a árvore de álgebra relacional gerada, difere do SQL pretendido devido às limitações sintáticas do motor de execução da ferramenta RelaX. Estas restrições impedem a tradução direta de operações complexas de agregação e distinção (como o COUNT DISTINCT), forçando uma simplificação da estrutura lógica para garantir a validade do plano de execução relacional. Tal distinção é explicada no tópico 5.6.

4.5. Validação do Esquema Lógico Final

Após o desenvolvimento do esquema lógico da base de dados, foi realizada a sua validação para assegurar a conformidade com os requisitos e a coerência com o modelo conceptual estabelecido.

Durante este processo, todas as tabelas foram analisadas, verificando-se as dependências funcionais dos atributos e garantindo a ausência de redundância de dados.

Como resultado, constatou-se que o esquema lógico está em conformidade com os princípios de normalização até à terceira forma normal (3FN), demonstrando que o modelo está bem estruturado e apto a garantir a eficiência no armazenamento e gestão dos dados do projeto BelaVista.

5. Implementação Física

5.1. Apresentação e explicação da base de dados implementada

A base de dados Bela Vista foi desenvolvida em *MySQL* com o objetivo de suportar uma aplicação de gestão e partilha de experiências de viagens. A sua conceção baseou-se no modelo lógico previamente definido e apresentado, procurando assegurar coerência entre os requisitos identificados e a implementação física final.

A estrutura adotada organiza a informação em nove tabelas interligadas, refletindo as entidades e relacionamentos do domínio do problema. Esta organização permite representar utilizadores, organizações, viagens, dias de viagem, pontos de interesse e fotografias, garantindo uma separação nítida de responsabilidades e uma gestão eficiente dos dados.

A utilização de chaves primárias e estrangeiras assegura a integridade referencial entre as tabelas, contribuindo para a consistência da informação armazenada. Foram também implementadas restrições de domínio através de *constraints CHECK* e tipos de dados apropriados, prevenindo a inserção de dados inválidos.

5.1.1. Tabela Utilizador

A tabela Utilizador constitui uma das entidades principais do sistema, armazenando os dados dos utilizadores do sistema Bela Vista. Tal como definido no modelo lógico, definem-se os atributos *Id*, definido como chave primária com *AUTO_INCREMENT*, garantido que se trata de um identificador único sequencial para cada utilizador.

Por sua vez o atributo *Idade* utiliza *INT UNSIGNED* para assegurar que são apenas aceites valores positivos, evitando idades negativas que constituiriam um erro. O campo *Admin* é implementado como *BOOLEAN* (equivalente a *TINYINT(1)* em MySQL) com valor *FALSE* por defeito, permitindo assim distinguir utilizadores comuns de administradores do sistema.

Salienta-se ainda a importância de utilizar a propriedade *NOT NULL* nos campos *Nome*, *Email*, *Telefone* e *Idade*, assegurando que estes atributos essenciais são preenchidos durante a inserção de novos utilizadores e adicionalmente a propriedade *UNIQUE* que não permite o mesmo email em mais do que um registo.

```
CREATE TABLE Utilizador (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Idade INT UNSIGNED NOT NULL,
    Email VARCHAR(150) UNIQUE NOT NULL,
    Telefone VARCHAR(15) NOT NULL,
    Admin BOOLEAN NOT NULL DEFAULT FALSE
);
```

5.1.2. Tabela Organização

A tabela Organização representa as organizações (núcleos de estudantes, agências de viagens, empresas, etc.) às quais os utilizadores podem pertencer e que podem estar associadas às viagens.

Neste caso, opta-se por uma estrutura simples, constituída pela chave primária *auto-incremental Id*, semelhante aos casos anteriores, bem como o respetivo Nome, salvaguardado pela propriedade NOT NULL que previne a existência de organizações sem nomenclatura.

```
CREATE TABLE Organizacao (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(150) NOT NULL
);
```

5.1.3. Tabela UtilizadorPertenceOrganização

A seguinte tabela implementa o relacionamento N:N identificado nos modelos do sistema, entre Utilizador e Organização, retratando o facto de um utilizador poder fazer parte de várias organizações e uma organização ser constituída por vários utilizadores.

Ao contrário das tabelas anteriores, neste caso, a chave primária é composta, combinando (Id_Utilizador, Id_Organizacao), o que impede associações duplicadas entre o mesmo utilizador e organização, garantindo a integridade dos dados. Vale ressaltar as restrições FOREIGN KEY com ON DELETE CASCADE que asseguram a integridade referencial, visto que, quando um utilizador ou organização é eliminado do sistema, todas as suas associações são automaticamente removidas, evitando referências a entidades não existentes.

```
CREATE TABLE UtilizadorPertenceOrganizacao (
    Id_Utilizador INT NOT NULL,
    Id_Organizacao INT NOT NULL,
    PRIMARY KEY (Id_Utilizador, Id_Organizacao),
    FOREIGN KEY (Id_Utilizador) REFERENCES Utilizador(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Organizacao) REFERENCES Organizacao(Id) ON DELETE CASCADE
);
```

5.1.4. Tabela Viagem

A tabela Viagem representa a entidade central do sistema, sendo responsável pelo armazenamento das viagens criadas e partilhadas pelos utilizadores. Deste modo, definem-se os atributos da entidade Viagem, Título e Descrição, dando especial importância à propriedade NOT NULL, que reflete a necessidade de todas as viagens terem pelo menos uma identificação básica e uma descrição que permita aos utilizadores compreender o seu propósito.

Além disso, define-se ainda um *Id*, como chave primária *auto-incremental*, o que assegura um identificador único para cada viagem, facilitando as referências apartir de outras tabelas.

```
CREATE TABLE Viagem (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Titulo VARCHAR(150) NOT NULL,
    Descricao TEXT NOT NULL
);
```

5.1.5. Tabela ViagemRealizadaUtilizador

À semelhança da tabela UtilizadorPertenceOrganização, a presente tabela implementa o relacionamento N:N entre as entidades Viagem e Utilizador, representado a participação de utilizadores em viagens. Na sua estrutura, estão presentes atributos adicionais que caracterizam esta participação, nomeadamente o Cargo, a Avaliação_Descritiva e a Avaliação_Quantitativa.

O campo Cargo recorre ao tipo ENUM('participante', 'organizador') de forma a restringir os valores possíveis a apenas estas duas categorias bem definidas, garantindo consistência nos dados, e refletindo dois tipos de associação entre os utilizadores e cada viagem.

Por sua vez, o campo Avaliação_Quantitativa integra a restrição CHECK (Avaliacao_Quantitativa >= 0 AND Avaliacao_Quantitativa <= 5), que assegura que as classificações numéricas respeitam a escala definida (0 a 5 estrelas), evitando valores inválidos. Ambos os campos de avaliação permitem valores NULL, permitindo ao utilizador não avaliar uma viagem na qual tenha participado.

Por fim, a chave identificadora desta tabela é também composta por ("Id_Viagem", "Id_Utilizador"), certificando que cada utilizador pode ter um registo de participação por viagem, impedindo duplicações indevidas.

```
CREATE TABLE ViagemRealizadaUtilizador (
    Id_Viagem INT NOT NULL,
    Id_Utilizador INT NOT NULL,
    Cargo ENUM('participante', 'organizador') NOT NULL,
    Avaliacao_Descritiva TEXT,
    Avaliacao_Quantitativa INT CHECK (Avaliacao_Quantitativa >= 0
                                        AND Avaliacao_Quantitativa <= 5),
    PRIMARY KEY (Id_Viagem, Id_Utilizador),
    FOREIGN KEY (Id_Utilizador) REFERENCES Utilizador(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE
);
```

5.1.6. Tabela ViagemMencionaOrganização

Como nos casos anteriores, esta é uma tabela que implementa a associação N:N entre Viagem e Organização, caracterizando as organizações que estão associadas ou são mencionadas em cada viagem.

Utiliza-se no entanto uma estrutura mais minimalista, que contém apenas as chaves estrangeiras necessárias, formando uma chave primária composta, que impede duplicações da mesma associação. Além disso, a propriedade ON DELETE CASCADE em ambas as chaves estrangeiras, assegura que quando uma viagem ou organização é eliminada do sistema, todas as suas associações são consequentemente removidas, mantendo a integridade referencial.

```
CREATE TABLE ViagemMencionaOrganizacao (
    Id_Viagem INT NOT NULL,
    Id_Organizacao INT NOT NULL,
    PRIMARY KEY (Id_Viagem, Id_Organizacao),
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Organizacao) REFERENCES Organizacao(Id) ON DELETE CASCADE
);
```

5.1.7. Tabela DiaDeViagem

A tabela DiaDeViagem caracteriza cada dia individual que compõe uma viagem, permitindo um planeamento detalhado e estruturado do itinerário. A criação de uma entidade separada para dias de viagem deveu-se à necessidade de armazenar informações específicas para cada dia.

Neste caso, o identificador único trata-se de uma chave primária composta, combinando (Data, Id_Viagem), fruto do relacionamento fraco entre as entidades. Deste modo, a mesma data pode ocorrer em diferentes viagens, contudo, a mesma data não se pode repetir dentro da mesma viagem.

Destaca-se a utilização da restrição CHECK (Custo >= 0), que garante que o custo de um dia não é negativo, assim como a ausência da restrição NOT NULL nos campos Alojamento e Transporte, permitindo flexibilidade para dias onde estes elementos não se aplicam.

Por fim, a chave estrangeira para Viagem é marcada com ON DELETE CASCADE para que, ao eliminar uma viagem, sejam também eliminados os respetivos dias de viagem.

```

CREATE TABLE DiaDeViagem (
    Data DATE NOT NULL,
    Id_Viagem INT NOT NULL,
    Custo DECIMAL(8,2) NOT NULL CHECK (Custo >= 0),
    Alojamento VARCHAR(150),
    Transporte VARCHAR(100),
    PRIMARY KEY(Data, Id_Viagem),
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE
);

```

5.1.8. Tabela PontoDeInteresse

A tabela PontoDeInteresse é responsável por armazenar os pontos de interesse visitados durante os dias de viagem, representando locais específicos do itinerário. É utilizada um chave primária *Id, auto-incremental*, de forma a garantir identificadores únicos independentes das outras características do ponto.

A particularidade mais relevante desta tabela é a chave estrangeira composta *(Data, Id_Viagem)*, que referencia DiaDeViagem. Esta estrutura cria um vínculo entre cada ponto de interesse e um dia específico da viagem em que foi visitado, mantendo a coerência temporal.

```

CREATE TABLE PontoDeInteresse (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(150) NOT NULL,
    Localizacao VARCHAR(200) NOT NULL,
    Descricao TEXT NOT NULL,
    Horario TIME NOT NULL,
    Data DATE NOT NULL,
    Id_Viagem INT NOT NULL,
    FOREIGN KEY (Data, Id_Viagem) REFERENCES DiaDeViagem(Data, Id_Viagem)
        ON DELETE CASCADE
);

```

5.1.9. Tabela Fotografias

Por fim, a tabela Fotografias implementa o atributo multivalorado da entidade Ponto de Interesse, identificado no modelo conceptual. Ao invés de armazenar-mos múltiplos URLs numa única coluna (o que violaria a Primeira Forma Normal), criou-se uma tabela separada onde cada linha representa uma única fotografia.

É importante salientar que nesta tabela, optou-se por uma chave primária artificial, *Id_Fotografia*, em vez de uma chave primária composta, *(Id_PontoInteresse, Url_fotografia)*, visto que o tipo escolhido para URLs, *TEXT*, não é ideal para inclusão em chaves primárias devido a limitações de indexação, e ainda por questões de flexibilidade em operações de junção, e maior facilidade em referências.

Novamente, recorre-se a *ON DELETE CASCADE* na chave estrangeira para assegurar que todas as fotografias associadas a um ponto de interesse, são eliminadas, quando esse ponto é removido.

```

CREATE TABLE Fotografias (
    Id_Fotografia INT AUTO_INCREMENT PRIMARY KEY,
    Id_PontoInteresse INT NOT NULL,
    Url_fotografia TEXT NOT NULL,
    FOREIGN KEY (Id_PontoInteresse) REFERENCES PontoDeInteresse(Id)
        ON DELETE CASCADE
);

```

5.1.10. Ordem de Criação das Tabelas

A ordem de criação das tabelas no *script SQL* segue uma sequência específica, que respeita as dependências entre tabelas devido às chaves estrangeiras.

Esta organização garante que as tabelas referenciadas por chaves estrangeiras existem antes das tabelas que as referenciam, evitando erros durante a execução do *script* de criação.

5.1.11. Políticas de Integridade Referencial

Todas as chaves estrangeiras no sistema foram implementadas com a política `ON DELETE CASCADE`. Esta decisão colocou em causa o bom funcionamento do sistema, pelo que se tiveram em consideração as seguintes implicações:

Embora operações de eliminação passem a afetar múltiplas tabelas, e seja necessário um cuidado redobrado ao eliminar regtos de tabelas no topo da hierarquia (como Viagem e Utilizador), garante-se por outro lado, a consistência de dados evitando-se a existência de referências órfãs, e facilitando possíveis integrações do sistema em aplicações.

5.2. Criação de utilizadores da base de dados

Para gerir a base de dados BelaVista, foram criados três utilizadores com diferentes perfis e privilégios:

- Milena Salomé
- Júlio César
- Utilizador

O ficheiro com as instruções de criação de utilizadores pode ser encontrado no **Anexo XI**.

Os administradores Milena Salomé e Júlio César têm privilégios completos sobre todas as tabelas da base de dados BelaVista, incluindo operações de `INSERT`, `SELECT`, `UPDATE` e `DELETE`. Estes privilégios foram concedidos porque ambos são responsáveis pela gestão global do sistema e pela manutenção dos dados.

Adicionalmente, foram concedidas permissões de `SELECT` sobre diversas *views*, que serão abordadas nos tópicos seguintes permitindo o acesso a informação consolidada e de apoio à tomada de decisão, sem necessidade de consultar diretamente as tabelas base. Isto promove uma utilização mais eficiente e segura dos dados.

A utilização de um *role* em vez de atribuições individuais de privilégios facilita a manutenção, a escalabilidade e a consistência das permissões, uma vez que qualquer alteração futura pode ser feita diretamente no *role* e refletida automaticamente em todos os utilizadores associados.

```
-- Criação de um role
CREATE ROLE IF NOT EXISTS role_administrador;

GRANT INSERT, DELETE, SELECT, UPDATE ON BelaVista.* TO role_administrador;

GRANT SELECT ON BelaVista.vwCustoViagens TO role_administrador;
GRANT SELECT ON BelaVista.vwPontosInteresseViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwPontosdeInteresseFotografias TO role_administrador;
GRANT SELECT ON BelaVista.vwLogisticaDiaDeViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwUtilizadoresViagens TO role_administrador;
GRANT SELECT ON BelaVista.vwOrganizadoresViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwViagensOrganizacoes TO role_administrador;
GRANT SELECT ON BelaVista.vwUtilizadoresOrganizacoes TO role_administrador;
GRANT SELECT ON BelaVista.vwAvaliacoesViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwParticipantesViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwViagemComItinerario TO role_administrador;

-- Criação do administrador Milena Salomé
CREATE USER IF NOT EXISTS 'milena_salome'@'localhost'
IDENTIFIED BY 'msalome123';

-- Criação do administrador Júlio César
CREATE USER IF NOT EXISTS 'julio_cesar'@'localhost'
IDENTIFIED BY 'julioc123';

GRANT role_administrador TO 'milena_salome'@'localhost';
GRANT role_administrador TO 'julio_cesar'@'localhost';
```

```

SET DEFAULT ROLE role_administrador TO
  'milena_salome'@'localhost',
  'julio_cezar'@'localhost';

FLUSH PRIVILEGES;

Para o Utilizador foi criada a vista vw_geral_org que permite expor apenas os dados necessários, sem dar
acesso direto à tabela “Organizacao”.

-- Criação de um utilizador
CREATE USER IF NOT EXISTS 'utilizador'@'localhost'
  IDENTIFIED BY 'password_utilizador';

CREATE VIEW vw_geral_org AS
  SELECT O.Id AS Id_Organização, O.Nome AS Nome_Organização, U.Nome AS
Nome_Pessoa
    FROM Organização AS O INNER JOIN UtilizadorPertenceOrganização AS UPO
    ON O.Id = UPO.Id_Organização
      INNER JOIN Utilizador AS U
      ON UPO.Id_Utilizador = U.Id;

-- Privilégios do Utilizador
GRANT SELECT ON BelaVista.vw_geral_org TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Utilizador TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Viagem TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.PontoDeInteresse TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Fotografias TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.DiaDeViagem TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.UtilizadorPertenceOrganização
TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemRealizadaUtilizador TO
'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemMencionaOrganização TO
'utilizador'@'localhost';

FLUSH PRIVILEGES;

```

5.3. Povoamento da base de dados

O povoamento da base de dados foi realizado através de um script SQL específico que pode ser consultado no **ANEXO VIII**, constituído por um conjunto de instruções INSERT, responsáveis pela inserção de dados em todas as tabelas que compõem o esquema da base de dados.

A ordem de execução destas instruções foi planeada de forma a respeitar as dependências existentes entre as tabelas, resultantes das restrições de integridade referencial impostas pelas chaves estrangeiras. Por exemplo, não seria possível inserir registo na tabela “DiaDeViagem” antes da tabela “Viagem”, uma vez que os registo de “DiaDeViagem” referenciam “Viagem” através de uma chave estrangeira.

Assim, o povoamento teve início na tabela “Utilizador”, seguindo-se as tabelas “Organização”, “UtilizadorPertenceOrganização”, “Viagem”, “ViagemMencionaOrganização”, “DiaDeViagem”, “PontoDeInteresse”, “Fotografias” e, por fim, “ViagemRealizadaUtilizador”.

Este planeamento garantiu que, no momento da inserção de cada registo, todas as chaves estrangeiras já existiam nas respetivas tabelas de referência, assegurando a coerência e consistência dos dados armazenados.

Importa ainda referir que todas as tabelas, com exceção das tabelas intermédias resultantes de relacionamentos N:M, foram definidas com uma chave primária auto-incrementada. Desta forma, não foi necessário especificar manualmente o valor da chave primária aquando da inserção dos registo.

Uma exceção a este caso é a tabela DiaDeViagem, cuja chave primária inclui o atributo Data. Por esse motivo, este atributo teve de ser explicitamente preenchido durante a inserção dos respetivos registo.

5.3.1. Tabela “Utilizador”

O povoamento da tabela “Utilizador” foi efetuado através de várias execuções da instrução `INSERT`. Cada registo corresponde a um utilizador distinto do sistema, contendo informações como o nome, idade, email, número de telefone e indicação de privilégios administrativos.

```
INSERT INTO Utilizador (Nome, Idade, Email, Telefone, Admin) VALUES
    ('Débora Caetano', 19, 'debora.caetano@email.com', '912345678', FALSE),
    ('Gonçalo Fernandes', 78, 'gonçalo.fernandes@email.com', '934567890', TRUE),
    ('Jonas Johanes', 22, 'jonas.johanes@email.com', '914567890', FALSE);
```

5.3.2. Tabela “Organização”

A tabela “Organizacao” foi povoada através da inserção direta de registo representativos das organizações existentes no sistema. Cada registo contém apenas o nome da organização, sendo o identificador atribuído automaticamente.

```
INSERT INTO Organizacao (Nome) VALUES
    ('CeBELUM'),
    ('CeSIUM'),
    ('BestTravel');
```

5.3.3. Tabela “UtilizadorPertenceOrganização”

Esta tabela intermédia representa o relacionamento N:M entre utilizadores e organizações, permitindo associar cada utilizador a uma ou mais organizações.

```
INSERT INTO UtilizadorPertenceOrganizacao (Id_Utilizador, Id_Organizacao) VALUES
    (2, 2),
    (3, 2),
    (2, 3);
```

5.3.4. Tabela “Viagem”

A tabela “Viagem” foi povoada com registo correspondentes às viagens disponíveis no sistema. Para cada viagem foram inseridos o título e uma breve descrição.

```
INSERT INTO Viagem (Titulo, Descricao) VALUES
    ('Tour pelo Douro', 'Viagem de dia inteiro pelo Vale do Douro com visita a quintas'),
    ('Roteiro Histórico Porto', 'Descoberta dos principais monumentos históricos do Porto'),
    ('Escapadinha em Guimarães', 'Fim de semana no berço da nação portuguesa'),
    ('Aventura na Serra da Estrela', 'Experiência de montanha com trilhos e gastronomia local');
```

5.3.5. Tabela “ViagemMencionaOrganização”

Esta tabela estabelece a relação entre viagens e organizações mencionadas ou associadas às mesmas.

```
INSERT INTO ViagemMencionaOrganizacao (Id_Viagem, Id_Organizacao) VALUES
    (1, 1),
    (2, 1),
    (1, 3),
    (3, 2),
    (4, 3);
```

5.3.6. Tabela “DiaDeViagem”

A tabela “DiaDeViagem” representa os diferentes dias associados a cada viagem. Para cada registo foram inseridas informações como a data, custo, alojamento e meio de transporte utilizado.

```
INSERT INTO DiaDeViagem (Data, Id_Viagem, Custo, Alojamento, Transporte) VALUES
    ('2025-03-15', 1, 75.50, 'Hotel Douro Vista', 'Autocarro turístico'),
    ('2025-03-20', 2, 45.00, NULL, 'A pé'),
    ('2025-03-21', 2, 40.00, NULL, 'Metro'),
    ('2025-04-05', 3, 60.00, 'Pousada de Guimarães', 'Carro próprio'),
```

```
('2025-04-06', 3, 55.00, 'Pousada de Guimarães', 'A pé'),
('2025-05-10', 4, 80.00, 'Casa da Montanha', 'Carrinha 4x4'),
('2025-05-11', 4, 70.00, 'Casa da Montanha', 'A pé');
```

5.3.7. Tabela “PontoDeInteresse”

Esta tabela foi povoada com pontos de interesse visitados em cada dia de viagem, contendo informações como o nome, localização, descrição, horário e a respetiva data associada à viagem.

```
INSERT INTO PontoDeInteresse (Nome, Localizacao, Descricao, Horario, Data, Id_Viagem) VALUES
    ('Quinta do Vesúvio', 'Douro', 'Quinta vinícola com provas de vinho', '09:00:00', '2025-03-15',
    1),
    ('Miradouro de São Leonardo', 'Douro', 'Vista panorâmica sobre o rio', '11:30:00', '2025-03-15',
    1),
    ('Restaurante DOC', 'Douro', 'Almoço com vista privilegiada', '13:00:00', '2025-03-15', 1),
    ('Torre dos Clérigos', 'Porto Centro', 'Monumento icónico do Porto', '09:00:00', '2025-03-20',
    2),
    ('Livraria Lello', 'Porto Centro', 'Uma das livrarias mais bonitas do mundo', '10:30:00',
    '2025-03-20', 2),
    ('Ribeira do Porto', 'Porto Centro', 'Passeio junto ao rio com esplanadas', '12:00:00',
    '2025-03-20', 2),
    ('Palácio da Bolsa', 'Porto Centro', 'Salão Árabe e arquitetura impressionante', '14:00:00',
    '2025-03-21', 2),
    ('Caves do Vinho do Porto', 'Vila Nova de Gaia', 'Prova de vinhos do Porto', '16:00:00',
    '2025-03-21', 2),
    ('Castelo de Guimarães', 'Guimarães', 'Castelo medieval, berço de Portugal', '10:00:00',
    '2025-04-05', 3),
    ('Paço dos Duques', 'Guimarães', 'Palácio do século XV', '14:00:00', '2025-04-05', 3),
    ('Centro Histórico', 'Guimarães', 'Passeio pelas ruas medievais', '09:00:00', '2025-04-06', 3),
    ('Torre da Estrela', 'Serra da Estrela', 'Ponto mais alto de Portugal Continental', '10:00:00',
    '2025-05-10', 4),
    ('Lagoa Comprida', 'Serra da Estrela', 'Lagoa glaciar com trilho pedestre', '14:00:00',
    '2025-05-10', 4),
    ('Queijaria Artesanal', 'Serra da Estrela', 'Visita e prova de queijo da serra', '10:00:00',
    '2025-05-11', 4);
```

5.3.8. Tabela “Fotografias”

A tabela “Fotografias” associa imagens aos respetivos pontos de interesse, permitindo armazenar múltiplas fotografias por local.

```
INSERT INTO Fotografias (Id_PontoInteresse, Url_fotografia) VALUES
    (1, 'https://fotos.belavista.com/douro/quinta-vesuvio-1.jpg'),
    (2, 'https://fotos.belavista.com/douro/miradouro-leonardo.jpg'),
    (4, 'https://fotos.belavista.com/porto/torre-clericos-exterior.jpg'),
    (4, 'https://fotos.belavista.com/porto/torre-clericos-vista.jpg'),
    (5, 'https://fotos.belavista.com/porto/lello-escadaria.jpg'),
    (7, 'https://fotos.belavista.com/porto/palacio-bolsa.jpg'),
    (9, 'https://fotos.belavista.com/guimaraes/castelo-exterior.jpg'),
    (10, 'https://fotos.belavista.com/guimaraes/paco-duques.jpg'),
    (12, 'https://fotos.belavista.com/estrela/torre-vista.jpg'),
    (13, 'https://fotos.belavista.com/estrela/lagoa-comprida.jpg');
```

5.3.9. Tabela “ViagemRealizadaUtilizador”

Por fim, a tabela “ViagemRealizadaUtilizador” regista a participação dos utilizadores nas viagens, incluindo o papel desempenhado e a avaliação atribuída.

```
INSERT INTO ViagemRealizadaUtilizador (Id_Viagem, Id_Utilizador, Cargo, Avaliacao_Descritiva,
Avaliacao_Quantitativa) VALUES
    (1, 1, 'organizador', 'Experiência maravilhosa! Tudo muito bem organizado.', 5),
    (1, 2, 'participante', 'Adorei as quintas e as paisagens do Douro.', 5),
    (2, 3, 'organizador', 'Porto é sempre uma boa escolha.', 4),
    (2, 1, 'participante', 'Conheci lugares que não conhecia no Porto!', 5),
    (3, 2, 'organizador', 'Guimarães tem um charme especial.', 5),
    (4, 3, 'organizador', 'A serra estava linda mas o trilho foi cansativo.', 4);
```

Deste modo, o processo de povoamento da base de dados cumpriu todos os requisitos definidos ao nível do modelo lógico e físico, assegurando a integridade referencial e a consistência dos dados. Os registos inseridos são suficientes e adequados para testar e demonstrar o correto funcionamento do sistema.

5.4. Cálculo do espaço da base de dados (inicial e taxa de crescimento anual)

O dimensionamento adequado da base de dados é fundamental para garantir o desempenho e a escalabilidade do sistema BelaVista. Este cálculo permite estimar não apenas o espaço de armazenamento inicial necessário, mas também prever o crescimento futuro da base de dados, facilitando o planeamento de recursos de infraestrutura e a prevenção de problemas relacionados com a capacidade de armazenamento.

Para determinar o espaço ocupado pela base de dados, é necessário conhecer o tamanho em bytes de cada tipo de dado utilizado no modelo relacional. A **Tabela 12** apresenta todos os tipos de dados utilizados para a base de dados BelaVista e o respetivo espaço de armazenamento.

Tipo de Dado	Tamanho(Bytes)	Fonte
INT	4	Oracle, 2024a, p.2269
TINYINT	1	Oracle, 2024a, p.2269
DECIMAL(8,2)	4	Oracle, 2024a, p.2269
DATE	3	Oracle, 2024a, p.2269
VARCHAR(M)	?	Oracle, 2024a, p.2270
ENUM('valor1', 'valor2', ...)	1 ou 2	Oracle, 2024a, p.2270
TEXT	2000	Oracle, 2024a, p.2270
TIME	3	Oracle, 2024a, p.2269

Tabela 12: Espaço ocupado pelos tipos de dados de MySQL“.

Tipos como DECIMAL(x,y) exigem cálculos para determinar o tamanho que ocupam. Com base na documentação do MySQL, (Oracle, 2024a, p.2269), x representa o número total de dígitos e y o número total de dígitos após a vírgula. O MySQL armazena os valores DECIMAL em formato binário compacto, ou seja, agrupa os dígitos em blocos de 9 dígitos cada um ocupando 4 bytes. Os dígitos que sobram, ocupam espaço adicional conforme a tabela seguinte.

Tal como DECIMAL(x,y), VARCHAR(M), também requer cálculos adicionais. Teremos em conta duas formas de calcular o tamanho que ocupa (Oracle, 2024a, p.2270). Assumindo que M é o número máximo de caracteres permitidos e L o comprimento real em bytes do valor armazenado, se M está entre 0 e 255 VARCHAR(M) ocupará L+1 bytes, se M é um valor maior que 255, ocupará L+2 bytes.

Nº dígitos que sobram	Tamanho(Bytes)
0	0
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4

Tabela 13: Espaço ocupado DECIMAL.

Sendo assim, para o exemplo do DECIMAL(8,2), a parte inteira terá 8 - 2 dígitos, 6 dígitos, e a parte decimal 2. Como não dá para formar um grupo completo de 9 dígitos, assume-se que sobram 6 dígitos, que pela

tabela, correspondem a 3 bytes. Segundo a tabela, os 2 dígitos decimais ocupam 1 byte e portanto no total um DECIMAL(8,2) ocupa 4 bytes.

Atributo	Tamanho(Bytes)
Id	4
Nome	100
Idade	4
Email	150
Telefone	15
Admin	1
Total: 274	

Tabela 14: Espaço ocupado pelos atributos de cada registo da tabela “Utilizador”.

Atributo	Tamanho(Bytes)
Id	4
Nome	150
Total: 154	

Tabela 15: Espaço ocupado pelos atributos de cada registo da tabela “Organização”.

Atributo	Tamanho(Bytes)
Id_Utilizador	4
Id_Organização	4
Total: 8	

Tabela 16: Espaço ocupado pelos atributos de cada registo da tabela “UtilizadorPertenceOrganização”.

Atributo	Tamanho(Bytes)
Id	4
Título	150
Descrição	2000
Total : 2154	

Tabela 17: Espaço ocupado pelos atributos de cada registo da tabela “Viagem”.

Atributo	Tamanho(Bytes)
Id_Viagem	4
Id_Utilizador	150
Cargo	1
Avaliação_Descritiva	2000
Avaliação_Quantitativa	4
Total: 2159	

Tabela 18: Espaço ocupado pelos atributos de cada registo da tabela “ViagemRealizadaUtilizador”.

Atributo	Tamanho(Bytes)
Id_Viagem	4
Id_Organização	4
Total: 8	

Tabela 19: Espaço ocupado pelos atributos de cada registo da tabela “ViagemMencionaOrganização”.

Atributo	Tamanho(Bytes)
Data	3
Id_Viagem	4
Custo	4
Alojamento	150
Transporte	100
Total: 261	

Tabela 20: Espaço ocupado pelos atributos de cada registo da tabela “DiaDeViagem”.

Atributo	Tamanho(Bytes)
Id	4
Nome	150
Localização	200
Descrição	2000
Horário	3
Data	3
Id_Viagem	4
Total: 2364	

Tabela 21: Espaço ocupado pelos atributos de cada registo da tabela “PontoDeInteresse”.

Atributo	Tamanho(Bytes)
Id_Fotografia	4
Id_PontoInteresse	150
Url_Fotografia	2000
Total: 2154	

Tabela 22: Espaço ocupado pelos atributos de cada registo da tabela “Fotografias”.

Tabela	Tamanho de um registo	Número de registos	Total(Bytes)
Utilizador	274	3	822
Organização	154	3	462
UtilizadorPertenceOrganização	8	3	24
Viagem	2154	4	8616
ViagemRealizadaUtilizador	2159	6	12954
ViagemMencionaOrganização	8	5	40
DiaDeViagem	261	7	1827
PontoDeInteresse	2364	14	33096
Fotografias	2154	10	24540
			Total: 82381

Tabela 23: Tamanho ocupado por cada tabela na base de dados BelaVista.

Com o total de bytes que a base de dados atual ocupa, 82381 (aproximadamente 80,45 kiloBytes), podemos analisar a taxa de crescimento anual, isto é, como é que o seu tamanho aumentará ao longo do tempo. Iremos então considerar a taxa avaliada para 3 anos e diferentes situações de crescimento esperadas.

Para calcular o tamanho da base de dados em cada ano, será utilizada a seguinte fórmula:

$$\text{Tamanho no ano } N \text{ (kB)} = \text{Tamanho ano 0 (kB)} \times (1 + (c/100))^N$$

onde c representa o crescimento anual em percentagem e N o número do ano.

Crescimento Anual	Ano 0	Ano 1	Ano 2	Ano 3
10%	80,45 kB	88,50 kB	97,34 kB	107,08 kB
15%	80,45 kB	92,52 kB	106,40 kB	122,36 kB
20%	80,45 kB	96,54 kB	115,85 kB	139,02 kB
30%	80,45 kB	104,59 kB	135,96 kB	176,75 kB
50%	80,45 kB	120,68 kB	18,01 kB	271,52 kB

Tabela 24: Evolução do tamanho da base de dados BelaVista.

Para calcular o aumento total para cada cenário de taxa de crescimento anual usaremos a seguinte fórmula:

$$\text{Aumento (\%)} = ((\text{Valor_final} - \text{Valor_inicial}) / \text{Valor_inicial}) \times 100$$

Crescimento atual	Aumento Total
10%	33,1%
15%	52,1%
20%	72,8%
30%	199,7%
50%	237,5%

Tabela 25: Aumento da base de dados BelaVista avaliado em 3 anos.

Sendo assim, podemos observar que mesmo partindo de um volume inicial relativamente reduzido (aproximadamente 80,45 kB), a taxa de crescimento anual tem um impacto significativo no aumento do armazenamento ao longo do tempo.

Observa-se que, em cenários de crescimento moderado, como 10% e 15% ao ano, o aumento total após três anos permanece controlado, variando entre 33,1% e 52,1%, o que indica uma expansão previsível e de fácil gestão.

Por outro lado, taxas de crescimento mais elevadas, como 20% e 30%, já alcançam até 72,8% e 199,7% em três anos, respetivamente. Estes cenários exigem maior atenção no planeamento de recursos, uma vez que o volume de dados pode praticamente duplicar ou triplicar num curto espaço de tempo.

O cenário mais extremo, com um crescimento anual de 50%, mostra um aumento total de 237,5%, o que deixa claro que é necessário ter cuidados adicionais. Nestes casos, é importante garantir que a base de dados consiga crescer sem problemas, que esteja bem organizada e que seja acompanhada regularmente, de forma a evitar falhas ou falta de espaço no futuro.

Assim, conclui-se que a definição realista da taxa de crescimento esperada é fundamental para garantir a sustentabilidade do sistema. Um planeamento antecipado do armazenamento e da infraestrutura associada permitirá evitar limitações futuras, assegurando o desempenho e a disponibilidade da base de dados à medida que o volume de informação aumenta.

5.5. Definição e caracterização de vidas de utilização em SQL

A utilização de vidas em SQL representam uma abordagem eficiente para simplificar o acesso e a gestão da base de dados com estruturas complexas. As vidas consistem em consultas pré-definidas que permitem associar, filtrar e organizar dados de múltiplas tabelas, apresentando a informação de forma mais estruturada e de acordo com as necessidades do cliente e do sistema.

No contexto do sistema de base de dados do projeto **Bela Vista**, foram definidas determinadas vidas com diferentes finalidades, cada uma criada para facilitar operações recorrentes. A definição e caracterização destas vidas encontram-se em detalhe no **Anexo XIII**.

5.5.1. Custo Viagens

A vista vwCustoViagens apresenta o custo total de cada viagem, calculado com base na soma dos custos diárias. São fornecidos o identificador da viagem, o título e o valor total correspondente.

```
CREATE OR REPLACE VIEW vwCustoViagens AS
  SELECT
    v.Id,
    v.Titulo,
    SUM(dv.Custo) AS Custo_Total
  FROM
    Viagem AS v
    JOIN DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
  GROUP BY
    v.Id, v.Titulo
  ORDER BY
    Custo_Total DESC;
```

5.5.2. Pontos de Interesse da Viagem

A vista vwPontosInteresseViagem apresenta os pontos de interesse associados a cada viagem, incluindo a data correspondente. São fornecidos o identificador e o título da viagem, bem como o identificador do ponto de interesse, o nome e a data associada.

```
CREATE OR REPLACE VIEW vwPontosInteresseViagem AS
  SELECT
    v.Id AS Id_Viagem,
    v.Titulo,
    pi.Id AS Id_PontoInteresse,
    pi.Nome,
    dv.Data
  FROM
    Viagem AS v
  JOIN
    DiaDeViagem AS dv
    ON dv.Id_Viagem = v.Id
  JOIN
    PontoDeInteresse AS pi
    ON pi.Id_Viagem = dv.Id_Viagem
    AND pi.Data = dv.Data;
```

5.5.3. Pontos de Interesse e Fotografias

A vista *vwPontosdeInteresseFotografias* apresenta os pontos de interesse de cada viagem, juntamente com o número total de fotografias associadas a cada ponto. Para cada registo são fornecidos o identificador do ponto de interesse, o nome, a localização e a contagem total de fotografias.

```
CREATE OR REPLACE VIEW vwPontosdeInteresseFotografias AS
    SELECT
        pi.Id,
        pi.Nome,
        pi.Localizacao,
        COUNT(f.Id_Fotografia) AS Total_Fotografias
    FROM
        PontodeInteresse pi
        LEFT JOIN
        Fotografias f ON f.Id_PontoInteresse = pi.Id
    GROUP BY pi.Id, pi.Nome, pi.Localizacao;
```

5.5.4. Logística do Dia de Viagem

A vista *vwLogisticaDiaDeViagem* apresenta a logística associada a cada dia de uma viagem. São disponibilizados a data, o identificador da viagem, o custo diário, informações sobre alojamento e transporte, bem como o título da viagem.

```
CREATE OR REPLACE VIEW vwLogisticaDiaDeViagem AS
    SELECT
        d.Data,
        d.Id_Viagem,
        d.Custo,
        d.Alojamento,
        d.Transportes,
        v.Titulo AS Viagem
    FROM
        DiaDeViagem d
    JOIN
        Viagem v ON v.Id = d.Id_Viagem;
```

5.5.5. Utilizadores

A vista *vwUtilizadoresViagens* apresenta os utilizadores e as viagens em que participam, incluindo o cargo desempenhado e a avaliação atribuída. São disponibilizados o nome e email do utilizador, o cargo, o título da viagem e a avaliação quantitativa e descritiva.

```
CREATE OR REPLACE VIEW vwUtilizadoresViagens AS
    SELECT
        u.Nome,
        u.Email,
        vr.u.Cargo,
        v.Titulo AS Viagem,
        vr.u.Avaliacao_Quantitativa,
        vr.u.Avaliacao_Descritiva
    FROM
        Utilizador u
        JOIN
        ViagemRealizadaUtilizador vr ON vr.Id_Utilizador = u.Id
        JOIN
        Viagem v ON v.Id = vr.Id_Viagem;
```

5.5.6. Organizadores

A vista *vwOrganizadoresViagem* apresenta os organizadores associados a cada viagem. Para cada registo, são indicados o título da viagem, o nome do organizador e o respetivo endereço de email.

```
CREATE OR REPLACE VIEW vwOrganizadoresViagem AS
    SELECT
        u.Nome AS Organizador, u.Email, v.Titulo AS Viagem
    FROM
        Viagem v
        JOIN
        ViagemRealizadaUtilizador vr ON vr.Id_Viagem = v.Id
        AND vr.u.Cargo = 'organizador'
        JOIN
        Utilizador u ON u.Id = vr.Id_Utilizador;
```

5.5.7. Organizações das Viagens

A vista *vwViagensOrganizacoes* relaciona cada viagem com as organizações a ela associadas. Para cada registo, são apresentados o título da viagem e o nome da organização correspondente.

```
CREATE OR REPLACE VIEW vwViagensOrganizacoes AS
SELECT
    o.Nome AS Organizacao, v.Titulo AS Viagem
FROM
    Viagem AS v
    JOIN
        ViagemMencionaOrganizacao vmo ON vmo.Id_Viagem = v.Id
        JOIN
            Organizacao AS o ON o.Id = vmo.Id_Organizacao;
```

5.5.8. Utilizadores e suas Organizações

A vista *vwUtilizadoresOrganizacoes* reúne informação sobre os utilizadores e as organizações a que pertencem. Para cada registo, são apresentados o nome da organização, o nome do utilizador e o respetivo endereço de email.

```
CREATE OR REPLACE VIEW vwUtilizadoresOrganizacoes AS
SELECT
    u.Nome AS Utilizador, u.Email, o.Nome AS Organizacao
FROM
    Organizacao AS o
    JOIN
        UtilizadorPertenceOrganizacao AS upo ON upo.Id_Organizacao = o.Id
        JOIN
            Utilizador AS u ON u.Id = upo.Id_Utilizador
ORDER BY u.Nome ASC;
```

5.5.9. Avaliações da Viagem

A vista *vwAvaliacoesViagem* consolida a informação relativa às avaliações efetuadas pelos utilizadores. Para cada registo, são apresentados o título da viagem, o nome do utilizador responsável pela avaliação, bem como os respetivos dados quantitativos e descritivos.

```
CREATE OR REPLACE VIEW vwAvaliacoesViagem AS
SELECT
    v.Titulo,
    vr.u.Avaliacao_Quantitativa,
    vr.u.Avaliacao_Descritiva,
    u.Nome AS Utilizador
FROM
    ViagemRealizadaUtilizador AS vr
    JOIN
        Viagem AS v ON v.Id = vr.Id_Viagem
        JOIN
            Utilizador AS u ON u.Id = vr.Id_Utilizador
ORDER BY v.Titulo;
```

5.5.10. Participantes da Viagem

A vista *vwParticipantesViagem* apresenta uma síntese dos participantes associados a cada viagem. Para cada título de viagem é indicado o número total de participantes registados.

```
CREATE OR REPLACE VIEW vwParticipantesViagem AS
SELECT
    v.Titulo AS Viagem, COUNT(*) AS Total_Participantes
FROM
    Viagem AS v
    JOIN
        ViagemRealizadaUtilizador AS vr ON vr.Id_Viagem = v.Id
WHERE
    vr.Cargo = 'participante'
GROUP BY v.Titulo;
```

5.5.11. Viagem com Itinerário

A vista *vwViagemComItinerario* disponibiliza uma representação consolidada de cada viagem, integrando as respetivas informações descritivas e o itinerário diário. Para cada viagem, são apresentados o identificador,

o título, a descrição e a lista organizada de dias, incluindo data, custo, opções de alojamento e transporte, bem como os pontos de interesse associados.

```

CREATE OR REPLACE VIEW vwViagemComItinerario AS
  SELECT
    v.Id,
    v.Titulo,
    v.Descricao,
    GROUP_CONCAT(DISTINCT CONCAT(DATE_FORMAT(dv.Data, '%d/%m/%Y'),
      ' (' ,
      dv.Custo,
      ' EUR)' ,
      IF(dv.Alojamento IS NOT NULL,
        CONCAT(' - ', dv.Alojamento),
        ''),
      IF(dv.Transportes IS NOT NULL,
        CONCAT(' - ', dv.Transportes),
        '')),
      ' '),
      ' Pontos: ' ,
      (SELECT
        GROUP_CONCAT(CONCAT(TIME_FORMAT(pi2.Horario, '%H:%i'),
          ' - ',
          pi2.Nome,
          ' (' ,
          pi2.Localizacao,
          ')')
        ORDER BY pi2.Horario
        SEPARATOR '')
      )
    FROM
      PontoDeInteresse pi2
    WHERE
      pi2.Data = dv.Data
      AND pi2.Id_Viagem = v.Id))
  ORDER BY dv.Data
  SEPARATOR ''
  ') AS itinerario_completo
FROM
  Viagem AS v
  INNER JOIN
  DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
  LEFT JOIN
  PontoDeInteresse AS pi ON dv.Data = pi.Data
  AND pi.Id_Viagem = v.Id
GROUP BY v.Id , v.Titulo , v.Descricao;

```

5.6. Tradução das interrogações do utilizador para SQL

Nesta secção, serão apresentadas as instruções SQL na mesma ordem que as suas expressões de álgebra relacional correspondentes no tópico “**Validação do Modelo com Interrogações do Utilizador**” Além disso, também as podemos ver no ficheiro **Anexo XIV**.

RM5 - Listar todos os utilizadores pertencentes a uma organização:

```

PREPARE participantes_organização FROM
  'SELECT UPO.Id_Utilizador, U.Nome
  FROM UtilizadorPertenceOrganizacao AS UPO INNER JOIN Utilizador AS U
  ON UPO.Id_Utilizador = U.Id
  WHERE Id_Organizacao = ? ;';

SET @id_org = 2;
EXECUTE participantes_organização USING @id_org;
DEALLOCATE PREPARE participantes_organização;

```

Como já foi explicado no tópico **4.4**, este código serve para obter a lista de pessoas que fazem parte de uma organização específica. Para esta query foi utilizada uma instrução preparada que define uma consulta onde se selecionam duas colunas específicas: o identificador do utilizador (“UPO.Id_Utilizador”) e o seu nome (“U.Nome”). Para obter estes dados, o SQL usa um **INNER JOIN** que liga as duas tabelas relacionadas: a tabela “UtilizadorPertenceOrganizacao”, que indica que utilizadores pertencem a que organizações, e

a tabela “Utilizador”, que contém os dados dos utilizadores. Esta ligação é feita através da condição `UPO.Id_Utilizador = U.Id`, garantindo que apenas são combinados os registos em que o identificador do utilizador coincide nas duas tabelas. De seguida, a cláusula `WHERE Id_Organizacao = ?` é usada para filtrar os resultados por organização, sendo o `?` um parâmetro que será substituído mais tarde por um valor concreto, permitindo reutilizar a mesma instrução para diferentes organizações sem reescrever a query.

Para ser executada basta definir qual a organização sobre a qual queremos listar os seus utilizadores (`SET @id_org = 2`) e executar usando esse valor (“`EXECUTE participantes_organização USING @id_org`”). No final, a instrução preparada é apagada da memória. O resultado final é uma lista com os identificadores e nomes das pessoas que pertencem à organização número 2.

RM11 - Calcular as datas de ida e regresso das viagens com base nos seus dias:

```
SELECT
    v.Id AS IdViagem,
    v.Titulo AS NomeViagem,
    MIN(dv.Data) AS DataIda,
    MAX(dv.Data) AS DataRegresso
FROM
    Viagem AS v
    JOIN
        DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
GROUP BY v.Id , v.Titulo;
```

Para esta query é utilizada uma instrução **SELECT** que devolve quatro informações: o identificador da viagem (“`v.Id`”), o nome da viagem (“`v.Titulo`”), a data de ida (“`DataIda`”) e a data de regresso (“`DataRegresso`”). As datas de ida e regresso são obtidas através das funções `MIN(dv.Data)` e `MAX(dv.Data)`, que permitem identificar, respetivamente, o primeiro e o último dia associados a cada viagem.

Para obter estes valores, o SQL recorre a um **JOIN** entre a tabela “Viagem”, que contém a informação geral das viagens, e a tabela “DiaDeViagem”, onde estão registadas as datas correspondentes a cada dia de viagem. Esta ligação é feita através da condição `v.Id = dv.Id_Viagem`, garantindo que apenas são considerados os dias pertencentes a cada viagem.

De seguida, a cláusula `GROUP BY v.Id, v.Titulo` é utilizada para agrupar todos os dias associados à mesma viagem, permitindo que as funções `MIN` e `MAX` sejam aplicadas corretamente a cada grupo. O resultado final é uma lista onde cada linha representa uma viagem, indicando o seu identificador, o nome, a data de início e a data de fim da viagem.

RM12 - Calcular o custo total das viagens com base nos custos dos seus dias:

```
SELECT V.Id, V.Titulo, SUM(DV.Custo) AS Custo_Viagem
  FROM Viagem AS V INNER JOIN DiaDeViagem AS DV
    ON V.Id = DV.Id_Viagem
   GROUP BY V.Id
  ORDER BY V.Id ASC;
```

Este código serve para listar as viagens e o total de custos associados a cada uma, permitindo perceber quanto foi gasto em cada viagem. Para esta query é utilizada uma instrução **SELECT** que devolve três informações: o identificador da viagem (“V.Id”), o título da viagem (“V.Titulo”) e a soma dos custos associados a essa viagem, calculada através da função `SUM(DV.Custo)` e apresentada com o nome “Custo_Viagem”. Para obter estes valores, o SQL recorre a um **INNER JOIN** que liga a tabela “Viagem”, onde estão registadas as viagens, à tabela “DiaDeViagem”, que contém os custos diários de cada viagem. Esta ligação é feita pela condição `V.Id = DV.Id_Viagem`, garantindo que apenas são considerados os dias que pertencem efetivamente a cada viagem.

De seguida, a cláusula `GROUP BY V.Id` é utilizada para agrupar todos os dias pertencentes à mesma viagem, permitindo que a soma dos custos seja calculada corretamente por viagem. Por fim, a cláusula `ORDER BY V.Id ASC` organiza os resultados por ordem crescente do identificador da viagem. O resultado final é uma lista onde cada linha representa uma viagem, o seu título e o valor total gasto nessa viagem, ordenada pelo seu identificador.

RM21 - Listar todos os participantes de uma viagem:

```
PREPARE participantes_viajem FROM
  'SELECT VRU.Id_Utilizador, U.Nome
   FROM Viagem AS V INNER JOIN ViagemRealizadaUtilizador AS VRU
     ON V.Id = VRU.Id_Viagem INNER JOIN Utilizador AS U
       ON VRU.Id_Utilizador = U.Id
      WHERE (V.Id = ?) AND (VRU.Cargo = 'participante'); '

SET @id_viajem = 1;
EXECUTE participantes_viajem USING @id_viajem;
```

Como já foi explicado anteriormente, este código tem como objetivo listar os participantes de uma viagem específica. Para isso, é utilizada uma instrução preparada que define uma consulta onde se selecionam o identificador do utilizador (“VRU.Id_Utilizador”) e o seu nome (“U.Nome”). A query começa por fazer um **INNER JOIN** entre a tabela “Viagem” e a tabela “ViagemRealizadaUtilizador”, que regista as participações dos utilizadores nas viagens, ligando-as através da condição `V.Id = VRU.Id_Viagem`. Esta ligação permite obter apenas as participações associadas a uma viagem concreta. Em seguida, é feito um segundo **INNER JOIN** com a tabela “Utilizador”, usando a condição “`VRU.Id_Utilizador = U.Id`”, o que permite associar cada participação aos dados do respetivo utilizador e obter o seu nome. A cláusula `WHERE` é usada para filtrar os resultados, restringindo a query à viagem indicada através de um parâmetro (“`V.Id = ?`”) e selecionando apenas os utilizadores cujo cargo seja “participante”. Para executar a instrução, basta definir o identificador da viagem pretendida e executar a query usando esse valor. No final, o resultado é uma lista com os identificadores e nomes dos utilizadores que participaram na viagem indicada.

RM28 - Identificar o utilizador com maior número de viagens realizadas:

```
SELECT
  u.Id AS UserId,
  u.Nome AS Nome_Utilizador,
  COUNT(vru.Id_Viagem) AS Total_Viagens
FROM
  Utilizador AS u
  JOIN
  ViagemRealizadaUtilizador AS vru
    ON u.Id = vru.Id_Utilizador
GROUP BY u.Id, u.Nome
ORDER BY Total_Viagens DESC
LIMIT 1;
```

Para esta query é utilizada uma instrução **SELECT** que devolve três informações: o identificador do utilizador (“`u.Id`”), o nome do utilizador (“`u.Nome`”) e o número total de viagens realizadas por esse utilizador, calculado através da função `COUNT(vru.Id_Viagem)` e apresentado com o nome `Total_Viagens`.

Para obter estes valores, o SQL recorre a um **JOIN** entre a tabela “Utilizador”, que contém os dados dos utilizadores, e a tabela “ViagemRealizadaUtilizador”, onde estão registadas as associações entre utilizadores e viagens realizadas. Esta ligação é feita pela condição `u.Id = vru.Id_Utilizador`, garantindo que apenas são consideradas as viagens efetivamente realizadas por cada utilizador.

De seguida, a cláusula `GROUP BY u.Id, u.Nome` é utilizada para agrupar todas as viagens pertencentes ao mesmo utilizador, permitindo calcular corretamente o número total de viagens por utilizador. Posteriormente, a cláusula `ORDER BY Total_Viagens DESC` ordena os resultados por ordem decrescente do número de viagens, colocando o utilizador com mais viagens no topo da lista. Por fim, a cláusula `LIMIT 1` restringe o resultado a apenas um registo, identificando assim o utilizador com o maior número de viagens realizadas.

RM30 - Listar as avaliações (média de avaliação quantitativa) por viagem:

```
SELECT
    v.Id AS IdViagem,
    v.Titulo AS NomeViagem,
    AVG(vru.Avaliacao_Quantitativa) AS MediaAvaliacao
FROM
    Viagem AS v
    JOIN
        ViagemRealizadaUtilizador AS vru ON v.Id = vru.Id_Viagem
GROUP BY v.Id, v.Titulo
ORDER BY MediaAvaliacao DESC;
```

Esta query devolve três informações: o identificador da viagem (“`v.Id`”), o nome da viagem (“`v.Titulo`”) e a média da avaliação quantitativa associada a essa viagem, apresentada com o nome `MediaAvaliacao`. Este valor é calculado através da função `AVG(v.Id)`, que recebe o identificador da viagem e devolve a respetiva média de avaliação.

Para obter estes resultados, o SQL recorre a um **JOIN** entre a tabela “Viagem”, que contém a informação geral das viagens, e a tabela “ViagemRealizadaUtilizador”, onde estão registadas as viagens realizadas pelos utilizadores. Esta ligação é feita através da condição `v.Id = vru.Id_Viagem`, garantindo que apenas são consideradas viagens que tenham sido efetivamente realizadas e avaliadas.

De seguida, a cláusula `GROUP BY v.Id, v.Titulo` é utilizada para agrupar os registo pertencentes à mesma viagem, assegurando que a média de avaliação é calculada corretamente para cada uma. Por fim, a cláusula `ORDER BY MediaAvaliacao DESC` organiza os resultados por ordem decrescente da média de avaliação, permitindo identificar no topo da lista as viagens melhor classificadas pelos utilizadores.

RM34 - Listar quais as organizações com os membros mais ativos (que realizam mais viagens) e melhor avaliam:

```
SELECT
    o.Id,
    o.Nome AS Nome_Organizacao,
    COUNT(DISTINCT vru.Id_Viagem) AS Total_Viagens_Feitas,
    COUNT(DISTINCT vru.Id_Utilizador) AS Membros_Ativos,
    ROUND(AVG(vru.Avaliacao_Quantitativa), 2) AS Avaliacao_Media_Membros
FROM Organizacao o
LEFT JOIN UtilizadorPertenceOrganizacao upo ON o.Id = upo.Id_Organizacao
LEFT JOIN ViagemRealizadaUtilizador vru ON upo.Id_Utilizador = vru.Id_Utilizador
GROUP BY o.Id, o.Nome
ORDER BY Total_Viagens_Feitas DESC, Avaliacao_Media_Membros DESC;
```

A consulta parte da tabela principal “Organizacao” e, através de uma série de operações **LEFT JOIN**, vai buscar informação às tabelas que relacionam organizações com utilizadores “UtilizadorPertenceOrganizacao” e, por sua vez, utilizadores com as viagens que realizaram e avaliaram “ViagemRealizadaUtilizador”.

A utilização do **LEFT JOIN** permite que todas as organizações apareçam, sem exceção, mesmo as que estão completamente inativas.

A ideia reside em agrupar os resultados por cada organização (`GROUP BY o.Id, o.Nome`) para conseguirmos resumir a atividade de todos os seus membros em valores únicos. É aqui que é importante o uso funções de agregação, especialmente com o uso crítico de `DISTINCT`.

O cálculo para `Total_Viagens_Feitas` utiliza `COUNT(DISTINCT vru.Id_Viagem)`. Isto é fundamental porque, sem o `DISTINCT`, a contagem seria feita por linhas. Imaginando que dois membros da mesma organização

participaram na mesma viagem. Isso geraria duas linhas na junção de tabelas. Um `COUNT` simples contaria 2 viagens, o que é incorreto, pois foi apenas uma viagem única com dois participantes. O `DISTINCT` assegura que o identificador de cada viagem é contado uma única vez, independentemente de quantos membros da organização nela tenham estado, dando-nos assim o número real e não inflacionado de viagens distintas em que a organização esteve envolvida.

A mesma lógica aplica-se a `Membros_Ativos`, com `COUNT(DISTINCT vru.Id_Utilizador)`. Um membro muito ativo que participou em dez viagens apareceria em dez linhas. Contar simplesmente as linhas diria que há dez “ocorrências de membro”, mas na realidade é uma única pessoa. O `DISTINCT` garante que cada utilizador é contado uma só vez, resultando no número real de indivíduos ativos da organização.

Para a `Avaliacao_Media_Membros`, a query calcula `ROUND(AVG(vru.Avaliacao_Quantitativa), 2)`. A função `AVG` soma todas as avaliações quantitativas dadas por membros da organização e divide pelo número total de avaliações, obtendo a nota média. No entanto, esse resultado pode ser um número com muitas casas decimais (como 3.1416). O `ROUND` com o parâmetro 2 serve precisamente para tornar este resultado apresentável, arredondando-o a duas casas decimais. Assim, em vez de um valor extenso, obtemos uma média clara, como 3.14.

Com todos estes valores calculados para cada organização, a query ordena o resultado. A ordenação é feita pelo `Total_Viagens_Feitas`, de forma decrescente, colocando assim as organizações mais ativas no topo da lista. Para organizações com o mesmo número de viagens, aplica-se um critério de desempate ordenando-as também de forma decrescente pela `Avaliacao_Media_Membros`.

Visto que a ferramenta RelaX muitas vezes não suporta múltiplos `DISTINCT` dentro de agregados ou tem dificuldades com a ordem de execução dos `JOINS` quando misturados com funções de grupo, foi necessário o uso de uma versão mais simplificada para a execução desta query.

```
SELECT
    o.Id,
    o.Nome AS Nome_Organizacao,
    COUNT(vru.Id_Viagem) AS Total_Viagens_Mencionadas,
    COUNT(upo.Id_Utilizador) AS Membros_Ativos,
    AVG(vru.Avaliacao_Quantitativa) AS Avaliacao_Media_Membros
FROM Organizacao AS o
LEFT JOIN ViagemMencionaOrganizacao AS vmo ON o.Id = vmo.Id_Organizacao
LEFT JOIN UtilizadorPertenceOrganizacao AS upo ON o.Id = upo.Id_Organizacao
LEFT JOIN ViagemRealizadaUtilizador AS vru ON upo.Id_Utilizador = vru.Id_Utilizador
GROUP BY o.Id, o.Nome
ORDER BY Total_Viagens_Mencionadas DESC, Avaliacao_Media_Membros DESC;
```

5.7. Indexação do Sistema de Dados

A implementação de um plano de indexação no Projeto Bela Vista visa otimizar a execução de consultas e operações sobre as tabelas mais acessadas e com maior número de relações. A análise das queries utilizadas pelos utilizadores revelou que certas tabelas são consultadas com frequência, especialmente em junções, agregações e filtros.

O objetivo é melhorar o desempenho do sistema, permitindo localizar rapidamente registos por chaves primárias, chaves estrangeiras ou outros atributos muito consultados, sem necessidade de pesquisas completas na tabela. Entre os atributos mais relevantes estão: IDs, datas, cargos e avaliações quantitativas.

A seguir, exemplos de índices implementados para aumentar a performance das consultas:

5.7.1. Tabela “Viagem”

```
CREATE INDEX idx_Viagem_Id ON Viagem(Id);
CREATE INDEX idx_Viagem_Titulo ON Viagem(Titulo);
```

Os índices `idx_Viagem_Id` e `idx_Viagem_Titulo` são criados na tabela `Viagem` para melhorar o desempenho das consultas que envolvem a identificação da viagem ou pesquisa por título. Estes índices são particularmente úteis nas queries que listam viagens, calculam datas de ida e regresso, agregam custos ou médias de avaliação.

5.7.2. Tabela “DiaDeViagem”

```
CREATE INDEX idx_DiaDeViagem_IdViagem ON DiaDeViagem(Id_Viagem);
CREATE INDEX idx_DiaDeViagem_Data ON DiaDeViagem(Data);
```

O índice `idx_DiaDeViagem_IdViagem` permite associar rapidamente cada dia à viagem correspondente, enquanto `idx_DiaDeViagem_Data` acelera consultas ordenadas por data, como listagens cronológicas ou cálculo de datas de ida/regresso.

5.7.3. Tabela “ViagemRealizadaUtilizador”

```
CREATE INDEX idx_VRU_IdViagem ON ViagemRealizadaUtilizador(Id_Viagem);
CREATE INDEX idx_VRU_IdUtilizador ON ViagemRealizadaUtilizador(Id_Utilizador);
CREATE INDEX idx_VRU_Cargo ON ViagemRealizadaUtilizador(Cargo);
```

Na tabela `ViagemRealizadaUtilizador`, os índices `idx_VRU_IdViagem` e `idx_VRU_IdUtilizador` agilizam junções frequentes com `Viagem` e `Utilizador`, e `idx_VRU_Cargo` permite filtrar rapidamente por participantes ou organizadores.

5.7.4. Tabela “Utilizador”

```
CREATE INDEX idx_Utilizador_Id ON Utilizador(Id);
CREATE INDEX idx_Utilizador_Nome ON Utilizador(Nome);
```

Para a tabela `Utilizador`, os índices `idx_Utilizador_Id` e `idx_Utilizador_Nome` aceleram pesquisas por `utilizador` ou junções com tabelas relacionadas.

5.7.5. Tabela “UtilizadorPertenceOrganizacao”

```
CREATE INDEX idx_UPO_IdOrganizacao ON UtilizadorPertenceOrganizacao(Id_Organizacao);
CREATE INDEX idx_UPO_IdUtilizador ON UtilizadorPertenceOrganizacao(Id_Utilizador);
```

Em `UtilizadorPertenceOrganizacao`, os índices `idx_UPO_IdOrganizacao` e `idx_UPO_IdUtilizador` permitem identificar rapidamente membros de uma organização e realizar junções com `Utilizador` de forma eficiente.

5.8. Implementação de procedimentos, funções e gatilhos

A incorporação de procedimentos, funções e gatilhos num sistema de base de dados é crucial para automatizar processos, preservar a integridade das informações e otimizar o desempenho das operações. Os procedimentos armazenados permitem a execução de tarefas complexas utilizando comandos SQL pré-definidos, o que eleva a eficiência e reforça a segurança do sistema. As funções são ferramentas eficazes para realizar cálculos e manipulações de dados, fornecendo resultados específicos que podem ser aplicados em consultas. A seguir, são apresentados vários exemplos de procedimentos e funções criados para o sistema, também disponíveis no **Anexo X**, acompanhados de detalhes sobre os seus objetivos.

5.8.1. Function - calculateTripRating (RM29)

```
DELIMITER $$

CREATE FUNCTION calculateTripRating(p_Id INT)
RETURNS DECIMAL(3,2) DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT AVG(Avaliacao_Quantitativa)
        FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id
    );
END $$

DELIMITER ;
```

A função `calculateTripRating` tem como objetivo calcular e retornar a avaliação média de uma viagem com base no seu identificador (`Id`). Sempre que a função é chamada, realizam-se as seguintes etapas:

1. Recebe como parâmetro o *Id* da viagem.;
2. Consulta a tabela “ViagemRealizadaUtilizador” e procura ocorrências do *Id* fornecido.;
3. Calcula e retorna a média da avaliação, combinando as avaliações quantitativas de cada utilizador que tenha realizado e avaliado a viagem, usando `AVG()`.

Desta forma, esta função simplifica o acesso à avaliação global de viagens, abrindo um leque de possibilidades em interrogações e procedimentos.

5.8.2. Function - isAdmin

```
DELIMITER $$

CREATE FUNCTION isAdmin(user_id INT)
RETURNS BOOLEAN DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT Admin FROM Utilizador
        WHERE Id = user_id
    );
END $$

DELIMITER ;
```

A função `isAdmin` foca-se na verificação do tipo de permissões de um utilizador, retornando um `BOOLEAN` que indica se um utilizador é ou não *Admin*. Sempre que a função é chamada, realizam-se as seguintes etapas:

1. Recebe como parâmetro o *Id* de um utilizador;
2. Consulta a tabela “Utilizador” e procura a ocorrência com o *Id* pretendido;
3. Retorna o valor do atributo “Admin” do utilizador selecionado.

Deste modo, a função simplifica o acesso às permissões do utilizador, facilitando esta verificação durante procedimentos.

5.8.3. Function - calculateTripCost (RM12)

```
DELIMITER $$

CREATE FUNCTION calculateTripCost(trip_id INT)
RETURNS DECIMAL(8,2) DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT SUM(Custo) FROM DiaDeViagem
        WHERE Id_Viagem = trip_id
    );
END $$

DELIMITER ;
```

Tal como o nome indica, a função `calculateTripCost` realiza o cálculo do custo total de uma viagem, somando os custos diários da viagem. Na sua execução, realizam-se os seguintes passos:

1. Recebe como parâmetro o *Id* de uma viagem;
2. Consulta a tabela “DiaDeViagem” e procura ocorrências que pertençam ao dia pretendido;
3. Cálcula e retorna o valor do custo total, usando `SUM()` para somar os custos diários.

Deste modo, facilita-se o acesso a esta informação, que pode ser pertinente, evitando o seu armazenamento desnecessário visto ser um derivado que pode ser apenas calculado quando necessário.

5.8.4. Procedure - getTripByLoc (RM34)

```
DELIMITER $$

CREATE PROCEDURE getTripByLoc(IN current_user_id INT, IN p_loc VARCHAR(200))
BEGIN
    DECLARE isAdmin BOOLEAN;
    SELECT isAdmin(current_user_id) INTO isAdmin;
```

```

    IF isAdmin = 1 THEN
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN PontoDeInteresse pi ON pi.Id_Viagem = vci.Id
        WHERE pi.Localizacao REGEXP p_loc;
    ELSE
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador vu ON vci.Id = vu.Id_Viagem
        INNER JOIN PontoDeInteresse pi ON pi.Id_Viagem = vci.Id
        WHERE pi.Localizacao REGEXP p_loc AND vu.Id_Utilizador = current_user_id;
    END IF;

END $$

DELIMITER ;

```

O Procedimento `getTripByLoc` faz a busca de viagens que tenham passado por um determinado local (Ponto de Interesse). Neste procedimento recorre-se ainda à função `isAdmin` de modo a determinar o tipo de acessos do utilizador em questão. A execução genérica deste procedimento envolve os seguintes passos:

1. Recebe como parâmetros `current_user_id` e `p_loc`, isto é, o *Id* do utilizador requisitou a informação, e o respetivo input de procura para a localização;
2. Consulta a vista `ViagemComItinerario` e procura viagens com pontos de interesse relevantes na pesquisa (a procura é feita utilizando `REGEXP`);
3. Seleciona todos os dados pretendidos com base nas ocorrências encontradas.

Apesar destas etapas serem genéricas, o tipo de acesso do utilizador terá influência direta nos resultados da busca uma vez que, um Admin, executa uma busca em todas as viagens, isto é, incluindo aquelas nas quais não participou. Por outro lado, um utilizador comum, tem apenas acesso às viagens na qual tenha participado.

Este procedimento facilita a pesquisa filtrada de viagens, facilitando a consulta de informação por parte dos utilizadores.

5.8.5. Procedure - getUserTrips (RM24)

```

DELIMITER $$

CREATE PROCEDURE getUserTrips(IN current_user_id INT, IN p_Id INT)
BEGIN
    DECLARE isAdmin BOOLEAN;
    SELECT isAdmin(current_user_id) INTO isAdmin;

    IF isAdmin = 1 THEN
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador as vru ON vci.Id = vru.Id_Viagem
        WHERE Id_Utilizador = p_Id;
    ELSE
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador ON vci.Id = Id_Viagem
        WHERE Id_Utilizador = p_Id AND Id_Utilizador = current_user_id;
    END IF;
END $$

DELIMITER ;

```

O Procedimento `getUserTrips` lista todas as viagens de um determinado utilizador. Tal como no procedimento anterior, recorre-se à função `isAdmin` para determinar o tipo de acessos do utilizador em questão. Por conseguinte, um Admin é portanto capaz de listar as viagens de qualquer utilizador, pelo seu *Id*, já um utilizador comum tem apenas acesso às suas próprias viagens. A execução genérica deste procedimento envolve os seguintes passos:

1. Recebe como parâmetros `current_user_id` e `p_Id`, isto é, o *Id* do utilizador que faz a chamada do procedimento, e o respetivo *Id* para a procura. No caso de uma chamada feita por um utilizador comum, este segundo argumento é o seu próprio *Id*;
2. Consulta a vista `ViagemComItinerario` e procura as viagens nas quais tenha participado o utilizador em questão;
3. Seleciona todos os dados pretendidos com base nas ocorrências encontradas.

Este procedimento facilita uma função básica do sistema, tornando a experiência de consultas mais simples.

5.8.6. Procedure - getTripsByOrgId (RM31)

```
DELIMITER $$

CREATE PROCEDURE getTripsByOrgId(IN p_Org_Id INT)
BEGIN
    SELECT DISTINCT vci.* FROM ViagemComItinerario vci
    INNER JOIN ViagemMencionaOrganizacao AS vmo ON vci.Id = vmo.Id_Viagem
    WHERE Id_Organizacao = p_Org_Id;
END $$

DELIMITER ;
```

O procedimento `getTripsByOrgId` tem como objetivo listar todas as viagens associadas a uma organização específica. Este procedimento é útil para consultar o histórico de viagens relacionadas com entidades organizacionais registadas no sistema. A execução deste procedimento envolve os seguintes passos:

1. Recebe como parâmetro `p_Org_Id`, correspondente ao `Id` da organização pretendida;
2. Consulta a vista `ViagemComItinerario` através de uma junção com a tabela “`ViagemMencionaOrganizacao`”, estabelecendo a relação entre viagens e organizações, e filtra as ocorrências pelo identificador da organização fornecido;
3. Seleciona e retorna todos os dados das viagens que mencionam ou estão associadas à organização em questão.

Este procedimento simplifica a consulta de viagens por organização, sendo particularmente útil para análises de parcerias, patrocínios ou colaborações registadas.

5.8.7. Procedure - getTripsByCost (RM22)

```
DELIMITER $$

CREATE PROCEDURE getTripsByCost(IN p_bottom INT, IN p_top INT)
BEGIN
    SELECT DISTINCT vci.* FROM ViagemComItinerario vci
    WHERE calculateTripCost(Id) BETWEEN p_bottom AND p_top;
END $$

DELIMITER ;
```

O procedimento `getTripsByCost` permite filtrar viagens com base num intervalo de custos. Este procedimento recorre à função `calculateTripCost` para obter o custo total de cada viagem. A sua execução envolve as seguintes etapas:

1. Recebe como parâmetros `p_bottom` e `p_top`, que definem o limite inferior e superior do intervalo de custos pretendido;
2. Consulta a vista `ViagemComItinerario` e, para cada viagem, invoca a função `calculateTripCost` para calcular o seu custo total;
3. Filtra as viagens cujo custo se encontra dentro do intervalo especificado, utilizando o operador `BETWEEN`;
4. Seleciona e retorna todos os dados das viagens que satisfazem o critério de custo.

5.8.8. Procedure - getTripsByDates (RM23)

```
DELIMITER $$

CREATE PROCEDURE getTripsByDates(IN p_bottom DATE, IN p_top DATE)
BEGIN
    SELECT DISTINCT vci.*
    FROM ViagemComItinerario vci
    WHERE vci.Id IN (
        SELECT dv.Id_Viagem
        FROM DiaDeViagem dv
        GROUP BY dv.Id_Viagem
        HAVING MIN(dv.Data) >= p_bottom AND MAX(dv.Data) <= p_top
    );
END $$

DELIMITER ;
```

O procedimento `getTripsByDates` possibilita a filtragem de viagens com base num intervalo de datas, retornando apenas as viagens que ocorreram completamente dentro do período especificado. Este procedimento é fundamental para consultas temporais e análises históricas. A sua execução envolve os seguintes passos:

1. Recebe como parâmetros `p_bottom` e `p_top`, que correspondem às datas limite inferior e superior do intervalo pretendido;
2. Executa uma subconsulta na tabela “DiaDeViagem”, agrupando os registos por `Id` de viagem;
3. Utiliza as funções agregadas `MIN()` e `MAX()` para determinar, respetivamente, a data de início e a data de fim de cada viagem;
4. Utiliza `HAVING` para filtrar apenas as viagens cuja data de início é igual ou posterior a `p_bottom` e cuja data de fim é igual ou anterior a `p_top`, garantindo que toda a viagem ocorreu dentro do intervalo;
5. Consulta a vista `ViagemComItinerario` e seleciona apenas as viagens cujo `Id` consta na lista retornada pela subconsulta;
6. Retorna todos os dados das viagens que satisfazem o intervalo pretendido.

Este procedimento é essencial para funcionalidades de pesquisa avançada, permitindo aos utilizadores encontrar viagens realizadas em períodos específicos, como determinadas estações do ano ou intervalos de férias.

5.9. Implementação de transações

A implementação de transações é fundamental para garantir a integridade, consistência e segurança dos dados em operações complexas que envolvem várias tabelas. No contexto do sistema Bela Vista, as transações asseguram que operações críticas, como a criação de viagens, edição de itinerários ou gestão de avaliações, sejam executadas de forma atómica, ou seja, ou todas as operações são concluídas com sucesso, ou nenhuma alteração é aplicada à base de dados.

Este mecanismo foi pensado pela equipa visto mostrar ser relevante ao levantar a possibilidade de futuras implementações do sistema em aplicações web ou móveis, onde múltiplos utilizadores podem aceder e modificar dados simultaneamente. As transações implementadas incluem verificações de permissões, validações de integridade referencial e tratamento de erros.

A seguir, apresentam-se os procedimentos transacionais desenvolvidos para o sistema:

5.9.1. Procedure - `createTrip` (RM35)

```
DELIMITER $$

CREATE PROCEDURE createTrip(
    IN p_Id_Utilizador INT,
    IN p_Titulo VARCHAR(150),
    IN p_Descricao TEXT
)
BEGIN
    DECLARE v_Id_Viagem INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao criar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    INSERT INTO Viagem (Titulo, Descricao)
        VALUES (p_Titulo, p_Descricao);

    SET v_Id_Viagem = LAST_INSERT_ID();

    INSERT INTO ViagemRealizadaUtilizador (Id_Viagem, Id_Utilizador, Cargo)
        VALUES (v_Id_Viagem, p_Id_Utilizador, 'organizador');
```

```

    COMMIT;
    SELECT v_Id_Viagem AS Id_Viagem, 'Viagem criada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

O procedimento `createTrip` permite que qualquer utilizador crie uma nova viagem, sendo automaticamente registado como organizador da mesma. Esta transação é composta por dois passos:

1. Criação da viagem: Insere um novo registo na tabela “Viagem” com o título e descrição fornecidos;
2. Associação do organizador: Regista o utilizador como organizador na tabela “ViagemRealizadaUtilizador”.

A utilização de uma transação garante que ambas as operações são executadas de forma única. Caso ocorra algum erro, o `handler EXIT HANDLER FOR SQLEXCEPTION` captura a exceção, executa um `ROLLBACK` para reverter todas as alterações e sinaliza o erro ao utilizador. A função `LAST_INSERT_ID()` é utilizada para capturar o identificador da viagem recém-criada, assegurando a correta associação com o organizador.

5.9.2. Procedure - editTrip (RM38)

```

DELIMITER $$

CREATE PROCEDURE editTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Titulo VARCHAR(150),
    IN p_Descricao TEXT
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao editar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'O Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem editar viagens.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    UPDATE Viagem
    SET Titulo = p_Titulo,
        Descricao = p_Descricao
    WHERE Id = p_Id_Viagem;

    COMMIT;
    SELECT 'Viagem editada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

O procedimento `editTrip` implementa a edição de viagens com controlo rigoroso de permissões. A transação executa os seguintes passos:

1. Verificação de privilégios: utiliza a função `isAdmin()` para determinar se o utilizador tem permissões elevadas;

2. Validação de participação e cargo: para utilizadores não administradores, verifica se participam na viagem e se possuem o cargo de organizador;
3. Atualização dos dados: após as validações, procede à atualização do título e descrição da viagem.

Este modelo de é relevante numa aplicação onde diferentes níveis de acesso coexistem já que previne modificações não autorizadas.

5.9.3. Procedure - deleteTrip (RM38)

```

DELIMITER $$

CREATE PROCEDURE deleteTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'O Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem eliminar viagens.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    DELETE FROM Viagem WHERE Id = p_Id_Viagem;

    COMMIT;
    SELECT 'Viagem eliminada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

O procedimento `deleteTrip` implementa a eliminação de viagens com as mesmas validações de segurança do procedimento de edição. A operação de `DELETE` beneficia das políticas `ON DELETE CASCADE` definidas nas chaves estrangeiras

A transação assegura que, em caso de falha durante o processo de eliminação em “cascata”, nenhuma alteração parcial permanece na base de dados.

5.9.4. Procedure - createTripDay (RM39)

```

DELIMITER $$

CREATE PROCEDURE createTripDay(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE,
    IN p_Custo DECIMAL(8,2),
    IN p_Alojamento VARCHAR(150),
    IN p_Transporte VARCHAR(100)
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);

```

```

DECLARE v_IsAdmin BOOLEAN;
DECLARE v_Erro VARCHAR(255);

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    SET v_Erro = 'Erro ao criar dia de viagem. Transação revertida.';
    ROLLBACK;
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END;

START TRANSACTION;

SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

IF v_IsAdmin = 0 THEN
    SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Cargo IS NULL THEN
        SET v_Erro = 'O Utilizador não participa nesta viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF v_Cargo != 'organizador' THEN
        SET v_Erro = 'Apenas organizadores podem criar dias de viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;
END IF;

IF p_Custo < 0 THEN
    SET v_Erro = 'O custo não pode ser negativo.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

INSERT INTO DiaDeViagem (Data, Id_Viagem, Custo, Alojamento, Transporte)
VALUES (p_Data, p_Id_Viagem, p_Custo, p_Alojamento, p_Transporte);

COMMIT;
SELECT 'Dia de viagem criado com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

De forma semelhante às transações referentes a viagens, criaram-se também as transações `editTripDay` e `deleteTripDay`.

O procedimento `createTripDay` adiciona um novo dia ao itinerário de uma viagem existente. Para além das validações de permissões já descritas, este procedimento inclui uma validação de domínio adicional:

1. Validação de custo: verifica se o custo fornecido é não negativo, complementando a restrição `CHECK` definida na tabela.

5.9.5. Procedure - `createInterestPoint` (RM40)

```

DELIMITER $$

CREATE PROCEDURE createInterestPoint(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE,
    IN p_Nome VARCHAR(150),
    IN p_Localizacao VARCHAR(200),
    IN p_Descricao TEXT,
    IN p_Horario TIME
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Dia_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao criar ponto de interesse. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

```

```

START TRANSACTION;

SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

IF v_IsAdmin = 0 THEN
    SELECT Cargo INTO v_Cargo
    FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Cargo IS NULL THEN
        SET v_Erro = 'Utilizador não participa nesta viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF v_Cargo != 'organizador' THEN
        SET v_Erro = 'Apenas organizadores podem criar pontos de interesse.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;
END IF;

SELECT COUNT(*) INTO v_Dia_Existe
FROM DiaDeViagem
WHERE Data = p_Data AND Id_Viagem = p_Id_Viagem;

IF v_Dia_Existe = 0 THEN
    SET v_Erro = 'Dia de viagem não existe. Crie o dia primeiro.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

INSERT INTO PontoDeInteresse (Nome, Localizacao, Descricao, Horario, Data, Id_Viagem)
VALUES (p_Nome, p_Localizacao, p_Descricao, p_Horario, p_Data, p_Id_Viagem);

COMMIT;
SELECT LAST_INSERT_ID() AS Id_PontoInteresse, 'Ponto de interesse criado com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

De forma semelhante às transações referentes a viagens, criaram-se também as transações editInterestPoint e deleteInterestPoint.

O procedimento createInterestPoint adiciona pontos de interesse a dias específicos de uma viagem. A transação inclui uma validação crítica de integridade referencial:

1. Verificação de existência do dia: antes de criar o ponto de interesse, confirma primeiramente se o dia de viagem especificado existe na base de dados.

5.9.6. Procedure - reviewTrip (RM41)

```

DELIMITER $$

CREATE PROCEDURE reviewTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Avaliacao_Descriptiva TEXT,
    IN p_Avaliacao_Quantitativa INT
)
BEGIN
    DECLARE v_Participa INT;
    DECLARE v_Avaliacao_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao avaliar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Participa = 0 THEN
        SET v_Erro = 'Apenas participantes da viagem podem avaliá-la.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

```

```

END IF;

IF p_Avaliacao_Quantitativa < 0 OR p_Avaliacao_Quantitativa > 5 THEN
    SET v_Erro = 'A avaliação quantitativa deve estar entre 0 e 5.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador
AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

IF v_Avaliacao_Existe > 0 THEN
    SET v_Erro = 'Já existe uma avaliação para esta viagem.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

UPDATE ViagemRealizadaUtilizador
SET Avaliacao_Descritiva = p_Avaliacao_Descritiva,
    Avaliacao_Quantitativa = p_Avaliacao_Quantitativa
WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

COMMIT;
SELECT 'Avaliação criada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

O procedimento `reviewTrip` permite que participantes de uma viagem a avaliem. A transação implementa três validações sequenciais:

1. Validação de participação: confirma que o utilizador efetivamente participou na viagem;
2. Validação de domínio: verifica se a avaliação quantitativa está dentro do intervalo válido (0-5);
3. Prevenção de duplicação: garante que o utilizador ainda não avaliou esta viagem.

5.9.7. Procedure - editTripReview (RM42)

```

DELIMITER $$

CREATE PROCEDURE editTripReview(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Avaliacao_Descritiva TEXT,
    IN p_Avaliacao_Quantitativa INT
)
BEGIN
    DECLARE v_Participa INT;
    DECLARE v_Avaliacao_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao editar avaliação. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Participa = 0 THEN
        SET v_Erro = 'A avaliação não existe ou não pertence ao utilizador.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem
        AND Id_Utilizador = p_Id_Utilizador
        AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

    IF v_Avaliacao_Existe = 0 THEN
        SET v_Erro = 'A avaliação não existe';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF p_Avaliacao_Quantitativa < 0 OR p_Avaliacao_Quantitativa > 5 THEN

```

```

        SET v_Erro = 'A avaliação quantitativa deve estar entre 0 e 5.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    UPDATE ViagemRealizadaUtilizador
        SET Avaliacao_Descritiva = p_Avaliacao_Descritiva,
            Avaliacao_Quantitativa = p_Avaliacao_Quantitativa
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    COMMIT;
    SELECT 'Avaliação editada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

```

O procedimento `editTripReview` permite que utilizadores modifiquem avaliações previamente submetidas. A transação implementa validações específicas para garantir que apenas avaliações existentes podem ser editadas:

1. Validação de participação: confirma se existe um registo de participação do utilizador na viagem especificada;
2. Verificação de existência da avaliação: garante que o utilizador já submeteu uma avaliação (pelo menos um dos campos de avaliação não é nulo);
3. Validação de domínio: verifica se a nova avaliação quantitativa respeita o intervalo válido (0-5);
4. Atualização dos campos: sobrescreve os valores anteriores com os novos dados fornecidos.

5.9.8. Procedure - deleteTripReview (RM42)

```

DELIMITER $$

CREATE PROCEDURE deleteTripReview(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT
)
BEGIN
    DECLARE v_Participa INT;
    DECLARE v_Avaliacao_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar avaliação. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Participa = 0 THEN
        SET v_Erro = 'A avaliação não existe ou não pertence ao utilizador.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem
        AND Id_Utilizador = p_Id_Utilizador
        AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

    IF v_Avaliacao_Existe = 0 THEN
        SET v_Erro = 'A avaliação não existe.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    UPDATE ViagemRealizadaUtilizador
        SET Avaliacao_Descritiva = NULL,
            Avaliacao_Quantitativa = NULL
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    COMMIT;
    SELECT 'Avaliação eliminada com sucesso!' AS Mensagem;
END$$

```

```
DELIMITER ;
```

O procedimento `deleteTripReview` permite a remoção de avaliações existentes, implementando uma abordagem de “eliminação lógica” em vez de eliminação física do registo, isto é, os campos são passados a `NULL`. A transação executa as seguintes validações:

1. Validação de propriedade: confirma se o utilizador possui uma participação registada na viagem;
2. Verificação de existência: garante que existe efetivamente uma avaliação a eliminar;
3. Anulação dos campos: Define ambos os campos de avaliação como `NULL`, preservando o registo de participação.

5.10. Exportação de informação

Este código cria um procedimento em MySQL chamada `gerar_resumo_viajem` que gera um arquivo de texto com o resumo de uma viagem.

Contudo, existem limitações no caminho de escrita:

- O MySQL só permite escrever arquivos no diretório especificado pela variável `secure_file_priv`. O código usa um caminho fixo que pode não coincidir com essa configuração.
- O usuário do MySQL precisa ter permissão FILE para usar INTO OUTFILE.
- O caminho contém referência específica ao Windows (`C:/ProgramData/...`), tornando o código não portável para outros sistemas operacionais.
- O arquivo de destino não pode já existir, caso contrário ocorrerá erro.
- O diretório deve ter permissões de escrita para o usuário do MySQL.
- O caminho usa uma versão específica do MySQL Server (8.0), que pode não corresponder à versão instalada.

Para funcionar corretamente, é necessário verificar o valor de `secure_file_priv` com `SHOW VARIABLES LIKE 'secure_file_priv'` e ajustar o caminho conforme o diretório permitido pelo servidor MySQL.

```
DELIMITER $$
```

```
CREATE PROCEDURE gerar_resumo_viajem(IN p_id_viajem INT)
BEGIN
    DECLARE v_titulo VARCHAR(150);
    DECLARE v_custo_total DECIMAL(10,2);
    DECLARE v_data_dia DATE;
    DECLARE v_custo_dia DECIMAL(8,2);
    DECLARE done INTEGER DEFAULT 0;
    DECLARE v_resumo TEXT DEFAULT '';
    DECLARE v_linha TEXT;

    -- Cursor para percorrer os dias da viagem
    DECLARE cursor_dias CURSOR FOR
        SELECT Data, Custo
        FROM DiadeViagem
        WHERE Id_Viagem = p_id_viajem
        ORDER BY Data;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Obter título da viagem
    SELECT Titulo INTO v_titulo
    FROM Viagem
    WHERE Id = p_id_viajem;

    -- Verificar se a viagem existe
    IF v_titulo IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Viagem não encontrada';
    END IF;

    -- Calcular custo total
    SELECT COALESCE(SUM(Custo), 0) INTO v_custo_total
```

```

    FROM DiadeViagem
    WHERE Id_Viagem = p_id_viagem;

    -- Construir cabeçalho da fatura
    SET v_resumo = CONCAT(
        '=====\\n',
        '      Resumo de uma viagem\\n',
        '=====\\n',
        'Viagem: ', v_titulo, '\\n',
        '=====\\n\\n',
        'DETALHES DOS DIAS:\\n',
        '-----\\n'
    );
    OPEN cursor_dias;

    loop_dias: LOOP
        FETCH cursor_dias INTO v_data_dia, v_custo_dia;

        IF done = 1 THEN
            LEAVE loop_dias;
        END IF;

        SET v_linha = CONCAT(
            DATE_FORMAT(v_data_dia, '%d/%m/%Y'), REPEAT(' ', 5), v_custo_dia, ' €\\n'
        );
        SET v_resumo = CONCAT(v_resumo, v_linha);
    END LOOP;

    CLOSE cursor_dias;

    -- Adicionar rodapé com total
    SET v_resumo = CONCAT(
        v_resumo,
        '-----\\n',
        'TOTAL:', REPEAT(' ', 21),
        v_custo_total, ' €\\n',
        '=====\\n'
    );
    SET @caminho = CONCAT('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/resumo_viagem_',
    p_id_viagem, '.txt');
    SET @sql = CONCAT('SELECT ? INTO OUTFILE ''', @caminho, '''');
    PREPARE stmt FROM @sql;
    SET @conteudo = v_resumo;
    EXECUTE stmt USING @conteudo;
    DEALLOCATE PREPARE stmt;

END$$

DELIMITER ;

CALL gerar_resumo_viagem(7);

-- DROP PROCEDURE gerar_resumo_viagem;
-- SHOW VARIABLES LIKE 'secure_file_priv';

```

6. Conclusão

Fica então concluído o projeto no âmbito da unidade curricular Bases de Dados, integrada no 2º ano da Licenciatura em Engenharia Informática da Universidade do Minho, durante o ano letivo de 2025/2026.

A seguir faremos uma análise do que foi feito e do que poderá vir a ser feito no futuro, e uma reflexão final sobre o percurso do projeto.

6.1. Análise do Diagrama de Gantt Planeado x Real

A comparação entre o planeamento inicial e a execução real do projeto revelou alguns desvios esperados em trabalhos desta natureza.

Na fase de conceção e modelação, o tempo estimado revelou-se insuficiente, tendo a elaboração do modelo Entidade-Relacionamento exigido mais tempo do que o inicialmente previsto. A necessidade de compreender profundamente os requisitos do sistema, identificar todas as entidades e relações relevantes, e assegurar que o modelo conceptual refletia adequadamente a realidade a representar, conduziu a várias iterações e ajustes que prolongaram esta fase.

Por outro lado, a fase de implementação do esquema SQL decorreu de forma mais célere do que planeado, beneficiando da solidez do modelo conceptual previamente desenvolvido. A tradução do modelo Entidade-Relacionamento para o esquema relacional e a subsequente implementação em SQL revelaram-se processos mais diretos pois as decisões estruturais fundamentais já haviam sido tomadas na fase anterior.

A criação de vidas, funções e procedimentos armazenados manteve-se dentro dos prazos estabelecidos, aproveitando a clareza proporcionada pelo esquema bem definido.

O desenvolvimento da documentação acompanhou regularmente o progresso do projeto, evitando a acumulação de trabalho no final. Esta abordagem iterativa permitiu manter o relatório atualizado e facilitou a redação das secções finais.

De um modo geral, embora tenha havido pequenas flutuações nos prazos de algumas tarefas, o projeto foi concluído dentro do calendário académico estabelecido, demonstrando uma gestão eficaz do tempo e dos recursos disponíveis.

6.2. Trabalho futuro

De um modo geral, os objetivos estabelecidos foram alcançados com sucesso, resultando numa base de dados funcional e devidamente estruturada. Embora a solução desenvolvida seja sólida e esteja alinhada com os objetivos propostos, é possível identificar oportunidades de melhoria e evolução, tanto a nível técnico como conceptual.

A implementação de funcionalidades analíticas mais avançadas, como relatórios estatísticos e dashboards interativos, acrescentaria valor ao sistema permitindo extrair insights mais profundos da informação armazenada.

Adicionalmente, o desenvolvimento de uma interface de utilizador intuitiva tornaria o sistema mais acessível a utilizadores sem conhecimentos técnicos de SQL, facilitando a adoção e utilização quotidiana.

6.3. Reflexão Final

O trabalho realizado permitiu consolidar conhecimentos práticos relacionados com a definição de estruturas de dados e com a utilização de mecanismos como vistas, funções e procedimentos armazenados, os quais contribuíram para uma maior modularidade e organização do sistema.

Um dos aspectos mais valiosos foi trabalhar com um cenário próximo de situações reais, com requisitos por vezes ambíguos que exigiram tomadas de decisão fundamentadas. A necessidade de equilibrar a complexidade do modelo com a sua usabilidade ensinou-nos a importância de soluções pragmáticas.

Para além dos aspectos técnicos, o desenvolvimento do projeto revelou-se igualmente importante, promovendo a cooperação entre os elementos da equipa, a comunicação eficaz e uma adequada distribuição de tarefas, competências fundamentais para o futuro percurso académico e profissional.

Este projeto constituiu uma experiência formativa completa que ultrapassou o âmbito puramente técnico de Bases de Dados, reforçando a nossa capacidade de transformar requisitos abstratos em soluções concretas e funcionais.

7. Bibliografia

1. Chen, P. (1976, março). The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), 9–36. <https://doi.org/10.1145/320434.320440>
2. Chen, P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned [PDF]. Em Software Pioneers (pp. 296–310). Springer-Verlag. ISBN 978-3-540-43081-0.
3. Fuchs, M. (2021, 3 de março). Entity-Relationship Diagram (ERD). Obtido de <https://michael-fuchs-sql.netlify.app/2021/03/03/entity-relationship-diagram-erd/>
4. MySQL. (s.d.). MySQL Workbench. Obtido de <https://www.mysql.com/products/workbench/>
5. MySQL (s.d.). Documentação SQL. Obtido de <https://dev.mysql.com/doc/>
6. Oracle. (2024a, May 4) Documentação MySQL. Obtido a partir de <https://downloads.mysql.com/docs/refman-8.0-en.a4.pdf>
7. Sis4. (s.d.). Modelo BR. Obtido de <http://www.sis4.com/brModelo/>
8. Kessler, J. (2022, setembro). Relax. Obtido de <https://dbis-uibk.github.io/relax/landing>

Lista de Siglas e Acrónimos

RD Requisitos de Descrição

RM Requisitos de Manipulação

RC Requisitos de Controlo

ID *Identifier*

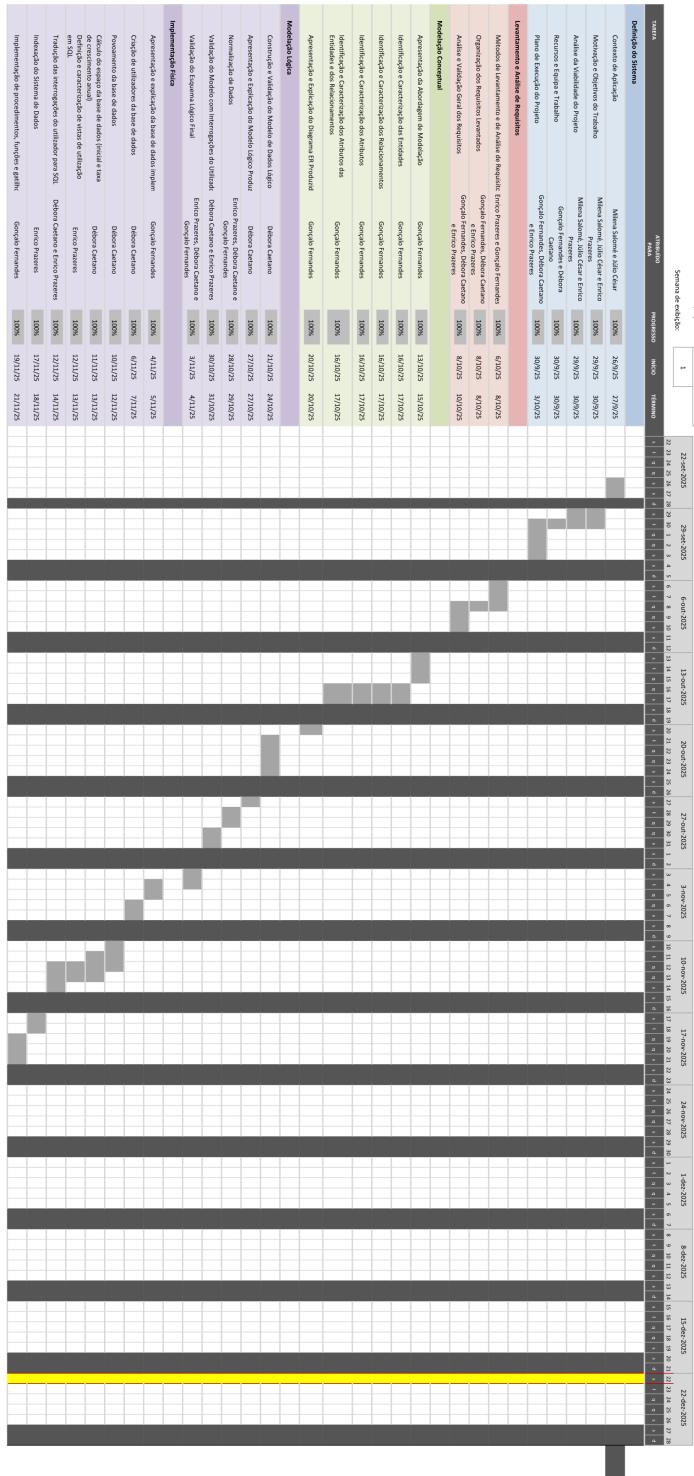
CeBELUM Centro de Estudantes de Biotecnologia e Engenharia de Laboratório da Universidade do Minho

Anexos

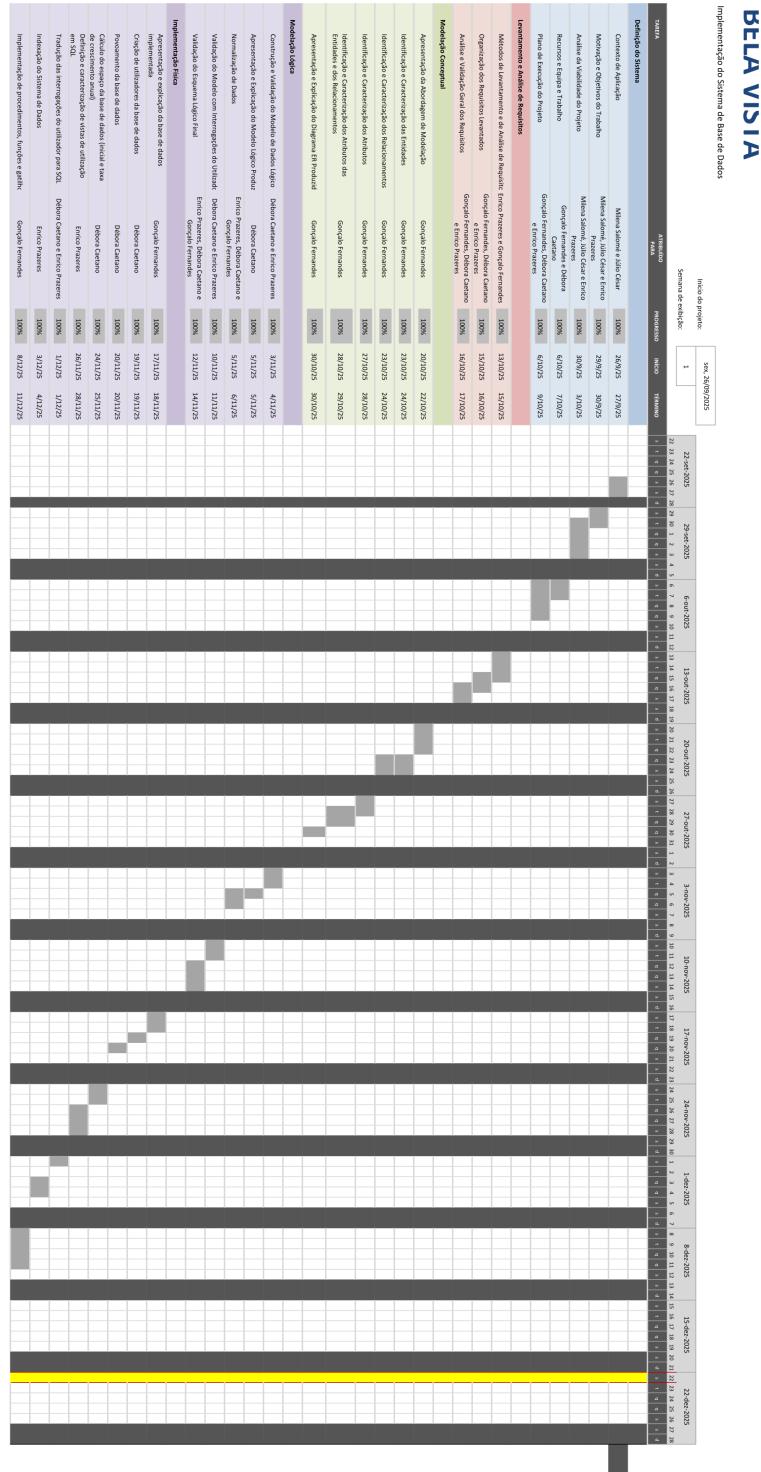
I. Diagrama de Gantt - Planeamento de Tarefas

BELA VISION

Implementação do Sistema de Base de Dados



II. Diagrama de Gantt - Tarefas Realizadas



III. Requisitos Levantados

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos

Documento de Requisitos

Nr	Data e Hora	Descrição	Área	Fonte	Analista
1	06/10/2025 08:15	Um utilizador é caracterizado por um identificador único, nome, idade, contacto (telefone e email) e se é admin ou não.	Pessoas	Milena Salomé	Enrico
2	06/10/2025 08:30	Uma organização é caracterizada por um identificador único e nome.	Pessoas	Júlio Cesar	Débora
3	06/10/2025 08:45	Um utilizador pode pertencer a zero ou mais organizações.	Pessoas	Milena Salomé	Gonçalo
4	06/10/2025 08:53	Uma organização deve ter pelo menos um utilizador associado.	Pessoas	Milena Salomé	Enrico
5	06/10/2025 09:00	Uma viagem é caracterizada por um identificador único, título e descrição.	Viagens	Júlio Cesar	Enrico
6	06/10/2025 09:10	A data de ida de uma viagem corresponde à data mais antiga dos seus dias de viagem.	Viagens	Júlio Cesar	Débora
7	06/10/2025 09:10	A data de regresso de uma viagem corresponde à data mais recente dos seus dias de viagem.	Viagens	Júlio Cesar	Gonçalo
8	06/10/2025 09:13	O custo total de uma viagem corresponde à soma dos custos dos seus dias de viagem.	Viagens	Milena Salomé	Débora
9	06/10/2025 09:17	Uma viagem é constituída por um ou mais dias de viagem.	Viagens	Júlio Cesar	Débora
10	06/10/2025 09:30	Um dia de viagem é caracterizado por uma data, custo, e opcionalmente por alojamento e transporte.	Viagens	Milena Salomé	Enrico
11	06/10/2025 09:50	Um ponto de interesse é caracterizado por um identificador único, nome, localização, descrição e horário.	Conteúdo	Júlio Cesar	Gonçalo
12	06/10/2025 10:00	Um ponto de interesse pertence a um e apenas um dia de viagem.	Conteúdo, Viagens	Júlio Cesar	Gonçalo

13	06/10/2025 10:30	Um dia de viagem pode conter zero ou mais pontos de interesse.	Conteúdo, Viagens	Milena Salomé	Enrico
14	06/10/2025 10:33	Um ponto de interesse pode ter zero ou mais fotografias associadas.	Conteúdo	Milena Salomé	Enrico
15	06/10/2025 10:37	Uma fotografia é caracterizada por um caminho/URL único.	Conteúdo	Júlio Cesar	Débora
16	06/10/2025 10:40	Uma viagem é realizada por um ou mais utilizadores.	Participações	Milena Salomé	Débora
17	06/10/2025 10:50	Na realização de uma viagem, cada utilizador tem um cargo associado (organizador ou participante).	Participações	Júlio Cesar	Débora
18	06/10/2025 10:52	Cada viagem deve ter pelo menos um utilizador com o cargo de organizador.	Participações	Júlio Cesar	Gonçalo
19	06/10/2025 10:55	Um utilizador pode realizar zero ou mais viagens.	Pessoas, Participações	Júlio Cesar	Enrico
20	06/10/2025 11:00	Uma avaliação é caracterizada por uma classificação quantitativa (escala de 0 a 5) e/ou um comentário descriptivo.	Participações	Júlio Cesar	Gonçalo
21	06/10/2025 11:05	Uma organização pode estar mencionada em zero ou mais viagens.	Pessoas, Viagens	Milena Salomé	Gonçalo
22	06/10/2025 11:07	Uma viagem pode mencionar zero ou mais organizações.	Pessoas, Viagens	Júlio Cesar	Débora
23	06/10/2025 11:15	Um utilizador pode fazer no máximo uma avaliação por viagem realizada.	Pessoas, Participações	Milena Salomé	Gonçalo
24	06/10/2025 11:15	A avaliação média de uma viagem corresponde à média aritmética das avaliações quantitativas dos utilizadores que a realizaram.	Participações	Milena Salomé	Gonçalo
25	06/10/2025 11:17	Um dia de viagem constitui uma e apenas uma viagem.	Viagens	Júlio Cesar	Enrico
26	06/10/2025 11:30	Não podem existir dois dias de viagem com a mesma data dentro da mesma viagem.	Viagens	Milena Salomé	Débora
27	06/10/2025 11:35	A avaliação quantitativa deve estar entre 0 e 5 (inclusive).	Participações	Milena Salomé	Enrico

28	07/10/2025 09:30	O sistema deve permitir registar um novo utilizador.	Pessoas	Júlio Cesar	Débora
29	07/10/2025 09:33	O sistema deve permitir atualizar os dados de um utilizador.	Pessoas	Júlio Cesar	Enrico
30	07/10/2025 09:36	O sistema deve permitir eliminar um utilizador.	Pessoas	Júlio Cesar	Enrico
31	07/10/2025 09:45	O sistema deve permitir criar uma nova organização.	Pessoas	Milena Salomé	Débora
32	07/10/2025 09:52	O sistema deve listar todos os utilizadores pertencentes a uma organização.	Pessoas	Júlio Cesar	Gonçalo
33	07/10/2025 09:58	O sistema deve permitir associar utilizadores a organizações.	Pessoas	Milena Salomé	Enrico
34	07/10/2025 10:30	O sistema deve permitir remover utilizadores de organizações.	Pessoas	Milena Salomé	Gonçalo
35	07/10/2025 10:35	O sistema deve permitir criar uma nova viagem.	Viagens	Júlio Cesar	Enrico
36	07/10/2025 10:38	O sistema deve permitir editar os dados de uma viagem existente.	Viagens	Júlio Cesar	Débora
37	07/10/2025 10:40	O sistema deve permitir eliminar uma viagem.	Viagens	Milena Salomé	Enrico
38	07/10/2025 11:00	O sistema deve calcular as datas de ida e regresso das viagens com base nos seus dias.	Viagens	Júlio Cesar	Gonçalo
39	07/10/2025 11:30	O sistema deve calcular o custo total das viagens com base nos custos dos seus dias.	Viagens	Milena Salomé	Débora
40	07/10/2025 11:33	O sistema deve permitir criar um dia de viagem associado a uma viagem.	Viagens	Milena Salomé	Gonçalo
41	07/10/2025 11:34	O sistema deve permitir editar um dia de viagem.	Viagens	Milena Salomé	Enrico
42	07/10/2025 11:34	O sistema deve permitir eliminar um dia de viagem.	Viagens	Júlio Cesar	Enrico

43	07/10/2025 14:13	O sistema deve permitir criar um ponto de interesse associado a um dia de viagem.	Conteúdo, Viagens	Júlio Cesar	Débora
44	07/10/2025 14:15	O sistema deve permitir editar um ponto de interesse.	Conteúdo	Júlio Cesar	Gonçalo
45	07/10/2025 14:20	O sistema deve permitir eliminar um ponto de interesse.	Conteúdo	Júlio Cesar	Gonçalo
46	07/10/2025 14:24	O sistema deve permitir associar fotografias a um ponto de interesse.	Conteúdo	Júlio Cesar	Enrico
47	07/10/2025 14:27	O sistema deve permitir remover fotografias de um ponto de interesse.	Conteúdo	Júlio Cesar	Enrico
48	07/10/2025 14:50	O sistema deve permitir listar todos os participantes de uma viagem	Viagens	Milena Salomé	Gonçalo
49	07/10/2025 15:03	O sistema deve permitir pesquisar viagens por intervalo de custo.	Viagens	Milena Salomé	Gonçalo
50	07/10/2025 15:10	O sistema deve permitir pesquisar viagens por intervalo de data.	Viagens	Milena Salomé	Gonçalo
51	07/10/2025 15:22	O sistema deve permitir consultar o histórico de viagens de um utilizador.	Viagens	Júlio Cesar	Débora
52	07/10/2025 15:30	O sistema deve permitir que um utilizador avalie uma viagem que realizou.	Participações	Júlio Cesar	Enrico
53	07/10/2025 15:34	O sistema deve permitir que um utilizador edite a sua avaliação de uma viagem.	Participações	Júlio Cesar	Débora
54	07/10/2025 15:39	O sistema deve permitir que um utilizador remova a sua avaliação de uma viagem.	Participações	Júlio Cesar	Gonçalo
55	07/10/2025 15:50	O sistema deve identificar o utilizador com maior número de viagens realizadas.	Participações	Milena Salomé	Débora
56	07/10/2025 15:56	O sistema deve calcular a avaliação média de uma viagem.	Participações	Júlio Cesar	Débora
57	07/10/2025 15:59	O sistema deve listar as avaliações (média de avaliação quantitativa) por viagem.	Participações	Milena Salomé	Enrico

58	07/10/2025 16:10	O sistema deve listar todas as viagens associadas a uma organização.	Viagens	Júlio Cesar	Gonçalo
59	07/10/2025 16:15	O sistema deve permitir listar quais as organizações com os membros mais ativos (que realizam mais viagens) e melhor avaliam	Pessoas, Participações	Milena Salomé	Débora
60	07/10/2025 16:20	O sistema deve permitir visualizar todas as fotografias de um ponto de interesse.	Conteúdo	Milena Salomé	Gonçalo
61	07/10/2025 16:23	O sistema deve permitir pesquisar viagens por locais visitados (localizações dos pontos de interesse).	Conteúdo	Júlio Cesar	Débora
62	07/10/2025 16:30	O sistema deve permitir que um utilizador normal crie uma nova viagem, assumindo automaticamente o cargo de organizador.	Viagens, Participações	Júlio Cesar	Enrico
63	07/10/2025 16:42	O sistema deve permitir que um utilizador normal edite apenas os seus próprios dados.	Pessoas, Participações	Júlio Cesar	Débora
64	07/10/2025 16:45	O sistema deve permitir que utilizadores normais visualizem apenas viagens nas quais tenham participado (como organizador ou participante).	Pessoas, Viagens	Júlio Cesar	Gonçalo
65	07/10/2025 16:47	O sistema deve permitir que utilizadores com cargo de organizador numa viagem, editem ou eliminem essa viagem.	Participações	Milena Salomé	Débora
66	07/10/2025 16:50	O sistema deve permitir que utilizadores com cargo de organizador numa viagem, criem, editem ou eliminem dias de viagem dessa viagem.	Participações, Viagens	Milena Salomé	Débora
67	08/10/2025 09:25	O sistema deve permitir que Utilizadores com cargo de organizador numa viagem, criem, editem ou eliminem pontos de interesse dessa viagem.	Participações, Conteúdo	Júlio Cesar	Débora
68	08/10/2025 09:27	O sistema deve permitir a um utilizador avaliar apenas viagens em que tenha participado (organizador ou participante).	Participações	Milena Salomé	Enrico
69	08/10/2025 09:32	O sistema deve permitir utilizador editar ou eliminar apenas as suas próprias avaliações.	Participações	Júlio Cesar	Gonçalo
70	08/10/2025 09:36	O sistema deve permitir a um utilizador fazer no máximo uma avaliação por viagem.	Participações	Júlio Cesar	Gonçalo
71	08/10/2025 14:38	Administradores podem criar, editar ou eliminar organizações.	Pessoas	Milena Salomé	Enrico
72	08/10/2025 14:42	Administradores podem eliminar utilizadores.	Pessoas	Milena Salomé	Enrico

73	08/10/2025 14:43	Administradores podem editar ou eliminar qualquer viagem, dia de viagem ou ponto de interesse.	Viagens	Júlio Cesar	Débora
74	08/10/2025 14:34	O sistema deve implementar dois tipos de perfis: Gestor e Utilizador Normal.	-	Milena Salomé	Enrico
75	08/10/2025 14:57	Administradores têm permissões de leitura, escrita e em todas as tabelas.	-	Milena Salomé	Gonçalo
76	08/10/2025 15:20	Administradores podem consultar estatísticas globais do sistema.	-	Júlio Cesar	Enrico
77	08/10/2025 15:24	Utilizadores normais têm permissões de leitura, escrita e em todas as tabelas exceto a tabela Organização.	-	Júlio Cesar	Gonçalo

IV. Requisitos de Descrição

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos
Documento de Requisitos de Descrição

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RD1	06/10/2025 08:15	Um utilizador é caracterizado por um identificador único, nome, idade, contacto (telefone e email) e se é admin ou não.	Pessoas	Milena Salomé	Enrico
RD2	06/10/2025 08:30	Uma organização é caracterizada por um identificador único e nome.	Pessoas	Júlio Cesar	Débora
RD3	06/10/2025 08:45	Um utilizador pode pertencer a zero ou mais organizações.	Pessoas	Milena Salomé	Gonçalo
RD4	06/10/2025 08:53	Uma organização deve ter pelo menos um utilizador associado.	Pessoas	Milena Salomé	Enrico
RD5	06/10/2025 09:00	Uma viagem é caracterizada por um identificador único, título e descrição.	Viagens	Júlio Cesar	Enrico
RD6	06/10/2025 09:10	A data de ida de uma viagem corresponde à data mais antiga dos seus dias de viagem.	Viagens	Júlio Cesar	Débora
RD7	06/10/2025 09:10	A data de regresso de uma viagem corresponde à data mais recente dos seus dias de viagem.	Viagens	Júlio Cesar	Gonçalo
RD8	06/10/2025 09:13	O custo total de uma viagem corresponde à soma dos custos dos seus dias de viagem.	Viagens	Milena Salomé	Débora
RD9	06/10/2025 09:17	Uma viagem é constituída por um ou mais dias de viagem.	Viagens	Júlio Cesar	Débora
RD10	06/10/2025 09:30	Um dia de viagem é caracterizado por uma data, custo, e opcionalmente por alojamento e transporte.	Viagens	Milena Salomé	Enrico
RD11	06/10/2025 09:50	Um ponto de interesse é caracterizado por um identificador único, nome, localização, descrição e horário.	Conteúdo	Júlio Cesar	Gonçalo

RD12	06/10/2025 10:00	Um ponto de interesse pertence a um e apenas um dia de viagem.	Conteúdo, Viagens	Júlio Cesar	Gonçalo
RD13	06/10/2025 10:30	Um dia de viagem pode conter zero ou mais pontos de interesse.	Conteúdo, Viagens	Milena Salomé	Enrico
RD14	06/10/2025 10:33	Um ponto de interesse pode ter zero ou mais fotografias associadas.	Conteúdo	Milena Salomé	Enrico
RD15	06/10/2025 10:37	Uma fotografia é caracterizada por um caminho/URL único.	Conteúdo	Júlio Cesar	Débora
RD16	06/10/2025 10:40	Uma viagem é realizada por um ou mais utilizadores.	Participações	Milena Salomé	Débora
RD17	06/10/2025 10:50	Na realização de uma viagem, cada utilizador tem um cargo associado (organizador ou participante).	Participações	Júlio Cesar	Débora
RD18	06/10/2025 10:52	Cada viagem deve ter pelo menos um utilizador com o cargo de organizador.	Participações	Júlio Cesar	Gonçalo
RD19	06/10/2025 10:55	Um utilizador pode realizar zero ou mais viagens.	Pessoas, Participações	Júlio Cesar	Enrico
RD20	06/10/2025 11:00	Uma avaliação é caracterizada por uma classificação quantitativa (escala de 0 a 5) e/ou um comentário descriptivo.	Participações	Júlio Cesar	Gonçalo
RD21	06/10/2025 11:05	Uma organização pode estar mencionada em zero ou mais viagens.	Pessoas, Viagens	Milena Salomé	Gonçalo
RD22	06/10/2025 11:07	Uma viagem pode mencionar zero ou mais organizações.	Pessoas, Viagens	Júlio Cesar	Débora
RD23	06/10/2025 11:15	Um utilizador pode fazer no máximo uma avaliação por viagem realizada.	Pessoas, Participações	Milena Salomé	Gonçalo
RD24	06/10/2025 11:15	A avaliação média de uma viagem corresponde à média aritmética das avaliações quantitativas dos utilizadores que a realizaram.	Participações	Milena Salomé	Gonçalo
RD25	06/10/2025 11:17	Um dia de viagem constitui uma e apenas uma viagem.	Viagens	Júlio Cesar	Enrico
RD26	06/10/2025 11:30	Não podem existir dois dias de viagem com a mesma data dentro da mesma viagem.	Viagens	Milena Salomé	Débora
RD27	06/10/2025 11:35	A avaliação quantitativa deve estar entre 0 e 5 (inclusive).	Participações	Milena Salomé	Enrico

V. Requisitos de Manipulação

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos

Documento de Requisitos de Manipulação

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RM1	07/10/2025 09:30	O sistema deve permitir registar um novo utilizador.	Pessoas	Júlio Cesar	Débora
RM2	07/10/2025 09:33	O sistema deve permitir atualizar os dados de um utilizador.	Pessoas	Júlio Cesar	Enrico
RM3	07/10/2025 09:36	O sistema deve permitir eliminar um utilizador.	Pessoas	Júlio Cesar	Enrico
RM4	07/10/2025 09:45	O sistema deve permitir criar uma nova organização.	Pessoas	Milena Salomé	Débora
RM5	07/10/2025 09:52	O sistema deve listar todos os utilizadores pertencentes a uma organização.	Pessoas	Júlio Cesar	Gonçalo
RM6	07/10/2025 09:58	O sistema deve permitir associar utilizadores a organizações.	Pessoas	Milena Salomé	Enrico
RM7	07/10/2025 10:30	O sistema deve permitir remover utilizadores de organizações.	Pessoas	Milena Salomé	Gonçalo
RM8	07/10/2025 10:35	O sistema deve permitir criar uma nova viagem.	Viagens	Júlio Cesar	Enrico
RM9	07/10/2025 10:38	O sistema deve permitir editar os dados de uma viagem existente.	Viagens	Júlio Cesar	Débora
RM10	07/10/2025 10:40	O sistema deve permitir eliminar uma viagem.	Viagens	Milena Salomé	Enrico
RM11	07/10/2025 11:00	O sistema deve calcular as datas de ida e regresso das viagens com base nos seus dias.	Viagens	Júlio Cesar	Gonçalo
RM12	07/10/2025 11:30	O sistema deve calcular o custo total das viagens com base nos custos dos seus dias.	Viagens	Milena Salomé	Débora
RM13	07/10/2025 11:33	O sistema deve permitir criar um dia de viagem associado a uma viagem.	Viagens	Milena Salomé	Gonçalo

RM14	07/10/2025 11:34	O sistema deve permitir editar um dia de viagem.	Viagens	Milena Salomé	Enrico
RM15	07/10/2025 11:34	O sistema deve permitir eliminar um dia de viagem.	Viagens	Júlio Cesar	Enrico
RM16	07/10/2025 14:13	O sistema deve permitir criar um ponto de interesse associado a um dia de viagem.	Conteúdo, Viagens	Júlio Cesar	Débora
RM17	07/10/2025 14:15	O sistema deve permitir editar um ponto de interesse.	Conteúdo	Júlio Cesar	Gonçalo
RM18	07/10/2025 14:20	O sistema deve permitir eliminar um ponto de interesse.	Conteúdo	Júlio Cesar	Gonçalo
RM19	07/10/2025 14:24	O sistema deve permitir associar fotografias a um ponto de interesse.	Conteúdo	Júlio Cesar	Enrico
RM20	07/10/2025 14:27	O sistema deve permitir remover fotografias de um ponto de interesse.	Conteúdo	Júlio Cesar	Enrico
RM21	07/10/2025 14:50	O sistema deve permitir listar todos os participantes de uma viagem	Viagens	Milena Salomé	Gonçalo
RM22	07/10/2025 15:03	O sistema deve permitir pesquisar viagens por intervalo de custo.	Viagens	Milena Salomé	Gonçalo
RM23	07/10/2025 15:10	O sistema deve permitir pesquisar viagens por intervalo de data.	Viagens	Milena Salomé	Gonçalo
RM24	07/10/2025 15:22	O sistema deve permitir consultar o histórico de viagens de um utilizador.	Viagens	Júlio Cesar	Débora
RM25	07/10/2025 15:30	O sistema deve permitir que um utilizador avalie uma viagem que realizou.	Participações	Júlio Cesar	Enrico
RM26	07/10/2025 15:34	O sistema deve permitir que um utilizador edite a sua avaliação de uma viagem.	Participações	Júlio Cesar	Débora
RM27	07/10/2025 15:39	O sistema deve permitir que um utilizador remova a sua avaliação de uma viagem.	Participações	Júlio Cesar	Gonçalo
RM28	07/10/2025 15:50	O sistema deve identificar o utilizador com maior número de viagens realizadas.	Participações	Milena Salomé	Débora
RM29	07/10/2025 15:56	O sistema deve calcular a avaliação média de uma viagem.	Participações	Júlio Cesar	Débora

RM30	07/10/2025 15:59	O sistema deve listar as avaliações (média de avaliação quantitativa) por viagem.	Participações	Milena Salomé	Enrico
RM31	07/10/2025 16:10	O sistema deve listar todas as viagens associadas a uma organização.	Viagens	Júlio Cesar	Gonçalo
RM32	07/10/2025 16:15	O sistema deve permitir listar quais as organizações com os membros mais ativos (que realizam mais viagens) e melhor avaliam	Pessoas, Participações	Milena Salomé	Débora
RM33	07/10/2025 16:20	O sistema deve permitir visualizar todas as fotografias de um ponto de interesse.	Conteúdo	Milena Salomé	Gonçalo
RM34	07/10/2025 16:23	O sistema deve permitir pesquisar viagens por locais visitados (localizações dos pontos de interesse).	Conteúdo	Júlio Cesar	Débora
RM35	07/10/2025 16:30	O sistema deve permitir que um utilizador normal crie uma nova viagem, assumindo automaticamente o cargo de organizador.	Viagens, Participações	Júlio Cesar	Enrico
RM36	07/10/2025 16:42	O sistema deve permitir que um utilizador normal edite apenas os seus próprios dados.	Pessoas, Participações	Júlio Cesar	Débora
RM37	07/10/2025 16:45	O sistema deve permitir que utilizadores normais visualizem apenas viagens nas quais tenham participado (como organizador ou participante).	Pessoas, Viagens	Júlio Cesar	Gonçalo
RM38	07/10/2025 16:47	O sistema deve permitir que utilizadores com cargo de organizador numa viagem, editem ou eliminem essa viagem.	Participações	Milena Salomé	Débora
RM39	07/10/2025 16:50	O sistema deve permitir que utilizadores com cargo de organizador numa viagem, criem, editem ou eliminem dias de viagem dessa viagem.	Participações, Viagens	Milena Salomé	Débora
RM40	08/10/2025 09:25	O sistema deve permitir que Utilizadores com cargo de organizador numa viagem, criem, editem ou eliminem pontos de interesse dessa viagem.	Participações, Conteúdo	Júlio Cesar	Débora
RM41	08/10/2025 09:27	O sistema deve permitir a um utilizador avaliar apenas viagens em que tenha participado (organizador ou participante).	Participações	Milena Salomé	Enrico
RM42	08/10/2025 09:32	O sistema deve permitir utilizador editar ou eliminar apenas as suas próprias avaliações.	Participações	Júlio Cesar	Gonçalo
RM43	08/10/2025 09:36	O sistema deve permitir a um utilizador fazer no máximo uma avaliação por viagem.	Participações	Júlio Cesar	Gonçalo
RM44	08/10/2025 14:38	Administradores podem criar, editar ou eliminar organizações.	Pessoas	Milena Salomé	Enrico
RM45	08/10/2025 14:42	Administradores podem eliminar utilizadores.	Pessoas	Milena Salomé	Enrico
RM46	08/10/2025 14:43	Administradores podem editar ou eliminar qualquer viagem, dia de viagem ou ponto de interesse.	Viagens	Júlio Cesar	Débora

VI. Requisitos de Controlo

Bela Vista

Processo de Desenvolvimento do Sistema de Base de Dados

Levantamento de Requisitos

Documento de Requisitos de Controlo

Nr	Data e Hora	Descrição	Área	Fonte	Revisor
RC1	08/10/2025 14:34	O sistema deve implementar dois tipos de perfis: Gestor e Utilizador Normal.	-	Milena Salomé	Enrico
RC2	08/10/2025 14:57	Administradores têm permissões de leitura, escrita e em todas as tabelas.	-	Milena Salomé	Gonçalo
RC3	08/10/2025 15:20	Administradores podem consultar estatísticas globais do sistema.	-	Júlio Cesar	Enrico
RC4	08/10/2025 15:24	Utilizadores normais têm permissões de leitura, escrita e em todas as tabelas exceto a tabela Organização.	-	Júlio Cesar	Gonçalo

VII. Modelo Físico

```

-- == ====== CREATION ====== ==
-- Criação da Base de Dados BelaVista.
-- == ====== ==
CREATE DATABASE IF NOT EXISTS BelaVista;

USE BelaVista;

CREATE TABLE Utilizador (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Idade INT UNSIGNED NOT NULL,
    Email VARCHAR(150) UNIQUE NOT NULL,
    Telefone VARCHAR(15) NOT NULL,
    Admin BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE Organizacao (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(150) NOT NULL
);

CREATE TABLE UtilizadorPertenceOrganizacao (
    Id_Utilizador INT NOT NULL,
    Id_Organizacao INT NOT NULL,
    PRIMARY KEY (Id_Utilizador, Id_Organizacao),
    FOREIGN KEY (Id_Utilizador) REFERENCES Utilizador(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Organizacao) REFERENCES Organizacao(Id) ON DELETE CASCADE
);

CREATE TABLE Viagem (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Titulo VARCHAR(150) NOT NULL,
    Descricao TEXT NOT NULL
);

CREATE TABLE ViagemRealizadaUtilizador (
    Id_Viagem INT NOT NULL,
    Id_Utilizador INT NOT NULL,
    Cargo ENUM('participante', 'organizador') NOT NULL,
    Avaliacao_Descritiva TEXT,
    Avaliacao_Quantitativa INT CHECK (Avaliacao_Quantitativa >= 0 AND Avaliacao_Quantitativa <=5),
    PRIMARY KEY (Id_Viagem, Id_Utilizador),
    FOREIGN KEY (Id_Utilizador) REFERENCES Utilizador(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE
);

CREATE TABLE ViagemMencionaOrganizacao (
    Id_Viagem INT NOT NULL,
    Id_Organizacao INT NOT NULL,
    PRIMARY KEY (Id_Viagem, Id_Organizacao),
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE,
    FOREIGN KEY (Id_Organizacao) REFERENCES Organizacao(Id) ON DELETE CASCADE
);

CREATE TABLE DiaDeViagem (
    Data DATE NOT NULL,
    Id_Viagem INT NOT NULL,
    Custo DECIMAL(8,2) NOT NULL CHECK (Custo >= 0),
    Alojamento VARCHAR(150),
    Transporte VARCHAR(100),
    PRIMARY KEY(Data, Id_Viagem),
    FOREIGN KEY (Id_Viagem) REFERENCES Viagem(Id) ON DELETE CASCADE
);

CREATE TABLE PontoDeInteresse (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(150) NOT NULL,
    Localizacao VARCHAR(200) NOT NULL,

```

```

Descricao TEXT NOT NULL,
Horario TIME NOT NULL,
Data DATE NOT NULL,
Id_Viagem INT NOT NULL,
FOREIGN KEY (Data,Id_Viagem) REFERENCES DiaDeViagem(Data, Id_Viagem) ON DELETE CASCADE
);

CREATE TABLE Fotografias (
Id_Fotografia INT AUTO_INCREMENT PRIMARY KEY,
Id_PontoInteresse INT NOT NULL,
Url_fotografia TEXT NOT NULL,
FOREIGN KEY (Id_PontoInteresse) REFERENCES PontoDeInteresse(Id) ON DELETE CASCADE
);

```

VIII. Povoamento da base de dados

```

-- == ===== POPULATION ===== --
-- Povoamento da base de dados Bela Vista.
-- == ===== --

USE BelaVista;

INSERT INTO Utilizador (Nome, Idade, Email, Telefone, Admin) VALUES
('Débora Caetano', 19, 'debora.caetano@email.com', '912345678', FALSE),
('Gonçalo Fernandes', 78, 'gonçalo.fernandes@email.com', '934567890', TRUE),
('Jonas Johanes', 22, 'jonas.johanes@email.com', '914567890', FALSE);

INSERT INTO Organizacao (Nome) VALUES
('CeBELUM'),
('CeSIUM'),
('BestTravel');

INSERT INTO UtilizadorPertenceOrganizacao (Id_Utilizador, Id_Organizacao) VALUES
(2, 2),
(3, 2),
(2, 3);

INSERT INTO Viagem (Titulo, Descricao) VALUES
('Tour pelo Douro', 'Viagem de dia inteiro pelo Vale do Douro com visita a quintas'),
('Roteiro Histórico Porto', 'Descoberta dos principais monumentos históricos do Porto'),
('Escapadinha em Guimarães', 'Fim de semana no berço da nação portuguesa'),
('Aventura na Serra da Estrela', 'Experiência de montanha com trilhos e gastronomia local');

INSERT INTO ViagemMencionaOrganizacao (Id_Viagem, Id_Organizacao) VALUES
(1, 1),
(2, 1),
(1, 3),
(3, 2),
(4, 3);

INSERT INTO DiaDeViagem (Data, Id_Viagem, Custo, Alojamento, Transporte) VALUES
('2025-03-15', 1, 75.50, 'Hotel Douro Vista', 'Autocarro turístico'),
('2025-03-20', 2, 45.00, NULL, 'A pé'),
('2025-03-21', 2, 40.00, NULL, 'Metro'),
('2025-04-05', 3, 60.00, 'Pousada de Guimarães', 'Carro próprio'),
('2025-04-06', 3, 55.00, 'Pousada de Guimarães', 'A pé'),
('2025-05-10', 4, 80.00, 'Casa da Montanha', 'Carrinha 4x4'),
('2025-05-11', 4, 70.00, 'Casa da Montanha', 'A pé');

INSERT INTO PontoDeInteresse (Nome, Localizacao, Descricao, Horario, Data, Id_Viagem) VALUES
('Quinta do Vesúvio', 'Douro', 'Quinta vinícola com provas de vinho', '09:00:00', '2025-03-15',
1),
('Miradouro de São Leonardo', 'Douro', 'Vista panorâmica sobre o rio', '11:30:00', '2025-03-15',
1),
('Restaurante DOC', 'Douro', 'Almoço com vista privilegiada', '13:00:00', '2025-03-15', 1),
('Torre dos Clérigos', 'Porto Centro', 'Monumento icónico do Porto', '09:00:00', '2025-03-20',
2),
('Livraria Lello', 'Porto Centro', 'Uma das livrarias mais bonitas do mundo', '10:30:00',
'2025-03-20', 2),
('Ribeira do Porto', 'Porto Centro', 'Passeio junto ao rio com esplanadas', '12:00:00',
'2025-03-20', 2),
('Palácio da Bolsa', 'Porto Centro', 'Salão Árabe e arquitetura impressionante', '14:00:00',
'2025-03-21', 2),
('Caves do Vinho do Porto', 'Vila Nova de Gaia', 'Prova de vinhos do Porto', '16:00:00',

```

```

'2025-03-21', 2),
    ('Castelo de Guimarães', 'Guimarães', 'Castelo medieval, berço de Portugal', '10:00:00',
'2025-04-05', 3),
    ('Paço dos Duques', 'Guimarães', 'Palácio do século XV', '14:00:00', '2025-04-05', 3),
    ('Centro Histórico', 'Guimarães', 'Passeio pelas ruas medievais', '09:00:00', '2025-04-06', 3),
    ('Torre da Estrela', 'Serra da Estrela', 'Ponto mais alto de Portugal Continental', '10:00:00',
'2025-05-10', 4),
    ('Lagoa Comprida', 'Serra da Estrela', 'Lagoa glaciar com trilho pedestre', '14:00:00',
'2025-05-10', 4),
    ('Queijaria Artesanal', 'Serra da Estrela', 'Visita e prova de queijo da serra', '10:00:00',
'2025-05-11', 4);

INSERT INTO Fotografias (Id_PontoInteresse, Url_fotografia) VALUES
(1, 'https://fotos.belavista.com/douro/quinta-vesuvio-1.jpg'),
(2, 'https://fotos.belavista.com/douro/miradouro-leonardo.jpg'),
(4, 'https://fotos.belavista.com/porto/torre-clericos-exterior.jpg'),
(4, 'https://fotos.belavista.com/porto/torre-clericos-vista.jpg'),
(5, 'https://fotos.belavista.com/porto/lello-escadaria.jpg'),
(7, 'https://fotos.belavista.com/porto/palacio-bolsa.jpg'),
(9, 'https://fotos.belavista.com/guimaraes/castelo-exterior.jpg'),
(10, 'https://fotos.belavista.com/guimaraes/paco-duques.jpg'),
(12, 'https://fotos.belavista.com/estrela/torre-vista.jpg'),
(13, 'https://fotos.belavista.com/estrela/lagoa-comprida.jpg');

INSERT INTO ViagemRealizadaUtilizador (Id_Viagem, Id_Utilizador, Cargo, Avaliacao_Descritiva,
Avaliacao_Quantitativa) VALUES
(1, 1, 'organizador', 'Experiência maravilhosa! Tudo muito bem organizado.', 5),
(1, 2, 'participante', 'Adorei as quintas e as paisagens do Douro.', 5),
(2, 3, 'organizador', 'Porto é sempre uma boa escolha!', 4),
(2, 1, 'participante', 'Conheci lugares que não conhecia no Porto!', 5),
(3, 2, 'organizador', 'Guimarães tem um charme especial.', 5),
(4, 3, 'organizador', 'A serra estava linda mas o trilho foi cansativo.', 4);

```

IX. Índices criados

```

-- ===== INDEXES =====
-- Indexação do Sistema de Dados.
-- =====

USE BelaVista;

-- Viagem
CREATE INDEX idx_Viagem_Id ON Viagem(Id);
CREATE INDEX idx_Viagem_Titulo ON Viagem(Titulo);

-- DiaDeViagem
CREATE INDEX idx_DiaDeViagem_IdViagem ON DiaDeViagem(Id_Viagem);
CREATE INDEX idx_DiaDeViagem_Data ON DiaDeViagem(Data);

-- ViagemRealizadaUtilizador
CREATE INDEX idx_VRU_IdViagem ON ViagemRealizadaUtilizador(Id_Viagem);
CREATE INDEX idx_VRU_IdUtilizador ON ViagemRealizadaUtilizador(Id_Utilizador);
CREATE INDEX idx_VRU_Cargo ON ViagemRealizadaUtilizador(Cargo);

-- Utilizador
CREATE INDEX idx_Utilizador_Id ON Utilizador(Id);
CREATE INDEX idx_Utilizador_Nome ON Utilizador(Nome);

-- UtilizadorPertenceOrganizacao
CREATE INDEX idx_UPO_IdOrganizacao ON UtilizadorPertenceOrganizacao(Id_Organizacao);
CREATE INDEX idx_UPO_IdUtilizador ON UtilizadorPertenceOrganizacao(Id_Utilizador);

```

X. Procedimentos, funções e gatilhos

```

-- ===== PROCEDURES/FUNCTIONS =====
-- Operações de consulta e cálculo sobre viagens.
-- =====

```

```

USE BelaVista;

-- STORED PROCEDURES

-- Criação do procedimento "getTripByLoc".
-- Retorna viagens que passam por um determinado local (ponto de interesse).
DROP PROCEDURE IF EXISTS getTripByLoc;
DELIMITER $$

CREATE PROCEDURE getTripByLoc(IN current_user_id INT, IN p_loc VARCHAR(200))
BEGIN
    DECLARE isAdmin BOOLEAN;
    SELECT isAdmin(current_user_id) INTO isAdmin;

    IF isAdmin = 1 THEN
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN PontoDeInteresse pi ON pi.Id_Viagem = vci.Id
        WHERE pi.Localizacao REGEXP p_loc;
    ELSE
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador vu ON vci.Id = vu.Id_Viagem
        INNER JOIN PontoDeInteresse pi ON pi.Id_Viagem = vci.Id
        WHERE pi.Localizacao REGEXP p_loc AND vu.Id_Utilizador = current_user_id;
    END IF;

END $$

DELIMITER ;

-- Criação do procedimento "getUserTrips".
-- Retorna o histórico de viagens realizadas por um utilizador.
DROP PROCEDURE IF EXISTS getUserTrips;
DELIMITER $$

CREATE PROCEDURE getUserTrips(IN current_user_id INT, IN p_Id INT)
BEGIN
    DECLARE isAdmin BOOLEAN;
    SELECT isAdmin(current_user_id) INTO isAdmin;

    IF isAdmin = 1 THEN
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador as vru ON vci.Id = vru.Id_Viagem
        WHERE Id_Utilizador = p_Id;
    ELSE
        SELECT DISTINCT vci.* FROM ViagemComItinerario vci
        INNER JOIN ViagemRealizadaUtilizador ON vci.Id = Id_Viagem
        WHERE Id_Utilizador = p_Id AND Id_Utilizador = current_user_id;
    END IF;

END $$

DELIMITER ;

-- Criação do procedimento "getTripsByOrgId".
-- Retorna todas as viagens associadas a uma organização.
DROP PROCEDURE IF EXISTS getTripsByOrgId;
DELIMITER $$

CREATE PROCEDURE getTripsByOrgId(IN p_Org_Id INT)
BEGIN
    SELECT DISTINCT vci.* FROM ViagemComItinerario vci
    INNER JOIN ViagemMencionaOrganizacao as vmo ON vci.Id = vmo.Id_Viagem
    WHERE Id_Organizacao = p_Org_Id;
END $$

DELIMITER ;

-- Criação do procedimento "getTripsByCost".
-- Retorna viagens num dado intervalo de custo.
DROP PROCEDURE IF EXISTS getTripsByCost;
DELIMITER $$

CREATE PROCEDURE getTripsByCost(IN p_bottom INT, IN p_top INT)
BEGIN
    SELECT DISTINCT vci.* FROM ViagemComItinerario vci
    WHERE calculateTripCost(Id) BETWEEN p_bottom AND p_top;
END $$

DELIMITER ;

-- Criação do procedimento "getTripsByDates".
-- Retorna viagens num dado intervalo de datas.
DROP PROCEDURE IF EXISTS getTripsByDates;
DELIMITER $$

CREATE PROCEDURE getTripsByDates(IN p_bottom DATE, IN p_top DATE)
BEGIN
    SELECT DISTINCT vci.*
    FROM ViagemComItinerario vci
    WHERE vci.Id IN (

```

```

        SELECT dv.Id_Viagem
        FROM DiaDeViagem dv
        GROUP BY dv.Id_Viagem
        HAVING MIN(dv.Data) >= p_bottom AND MAX(dv.Data) <= p_top
    );
END $$;
DELIMITER ;

-- F U N C T I O N S

-- Criação da função "calculateTripRating".
-- Calcula a avaliação média associada a uma viagem.
DROP FUNCTION IF EXISTS calculateTripRating;
DELIMITER $$;
CREATE FUNCTION calculateTripRating(p_Id INT)
RETURNS DECIMAL(3,2) DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT AVG(Avaliacao_Quantitativa)
        FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id
    );
END $$;
DELIMITER ;

-- Criação da função "isAdmin".
-- Verifica se um utilizador é administrador.
DROP FUNCTION IF EXISTS isAdmin;
DELIMITER $$;
CREATE FUNCTION isAdmin(user_id INT)
RETURNS BOOLEAN DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT Admin FROM Utilizador
        WHERE Id = user_id
    );
END $$;
DELIMITER ;

-- Criação da função "calculateTripCost".
-- Calcula o custo total de uma viagem.
DROP FUNCTION IF EXISTS calculateTripCost;
DELIMITER $$;
CREATE FUNCTION calculateTripCost(trip_id INT)
RETURNS DECIMAL(8,2) DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (
        SELECT SUM(Custo) FROM DiaDeViagem
        WHERE Id_Viagem = trip_id
    );
END $$;
DELIMITER ;

```

XI. Script de criação de utilizadores

```

-- == =====
-- == ===== USERS =====
-- == Criação de utilizadores da Base de Dados Bela Vista.
-- == =====

USE BelaVista;

CREATE ROLE IF NOT EXISTS role_administrador;

GRANT INSERT, DELETE, SELECT, UPDATE ON BelaVista.* TO role_administrador;

GRANT SELECT ON BelaVista.vwCustoViagens TO role_administrador;
GRANT SELECT ON BelaVista.vwPontosInteresseViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwPontosdeInteresseFotografias TO role_administrador;
GRANT SELECT ON BelaVista.vwLogisticaDiaDeViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwUtilizadoresViagens TO role_administrador;
GRANT SELECT ON BelaVista.vwOrganizadoresViagem TO role_administrador;

```

```

GRANT SELECT ON BelaVista.vwViagensOrganizacoes TO role_administrador;
GRANT SELECT ON BelaVista.vwUtilizadoresOrganizacoes TO role_administrador;
GRANT SELECT ON BelaVista.vwAvaliacoesViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwParticipantesViagem TO role_administrador;
GRANT SELECT ON BelaVista.vwViagemComItinerario TO role_administrador;

CREATE USER IF NOT EXISTS 'milena_salome'@'localhost'
IDENTIFIED BY 'msalome123';

CREATE USER IF NOT EXISTS 'julio_cesar'@'localhost'
IDENTIFIED BY 'julioc1234';

GRANT role_administrador TO 'milena_salome'@'localhost';
GRANT role_administrador TO 'julio_cesar'@'localhost';

SET DEFAULT ROLE role_administrador TO
    'milena_salome'@'localhost',
    'julio_cesar'@'localhost';

FLUSH PRIVILEGES;

-- 
-- REVOKE role_administrador FROM 'milena_salome'@'localhost';
-- REVOKE role_administrador FROM 'julio_cesar'@'localhost';
-- DROP ROLE IF EXISTS role_administrador;
-- 

-- Criação de um utilizador
CREATE USER IF NOT EXISTS 'utilizador'@'localhost'
    IDENTIFIED BY 'password_utilizador';

CREATE VIEW vw_geral_org AS
SELECT O.Id AS Id_Organizacao, O.Nome AS Nome_Organização, U.Nome AS Nome_Pessoa
    FROM Organizacao AS O INNER JOIN UtilizadorPertenceOrganizacao AS UPO
        ON O.Id = UPO.Id_Organizacao
            INNER JOIN Utilizador AS U
                ON UPO.Id_Utilizador = U.Id;

GRANT SELECT ON BelaVista.vw_geral_org TO 'utilizador'@'localhost';
GRANT SELECT ON BelaVista.vw_geral_org TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Utilizador TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Viagem TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.PontoDeInteresse TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.Fotografias TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.DiaDeViagem TO 'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.UtilizadorPertenceOrganizacao TO
    'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemRealizadaUtilizador TO
    'utilizador'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemMcionaOrganizacao TO
    'utilizador'@'localhost';

FLUSH PRIVILEGES;

-- 
-- DROP USER IF EXISTS 'utilizador'@'localhost';
-- REVOKE SELECT ON BelaVista.vw_geral_org FROM 'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.Utilizador FROM 'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.Viagem FROM 'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.PontoDeInteresse FROM
    'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.Fotografias FROM 'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.DiaDeViagem FROM 'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.UtilizadorPertenceOrganizacao FROM
    'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemRealizadaUtilizador FROM
    'utilizador'@'localhost';
-- REVOKE SELECT, INSERT, DELETE, UPDATE ON BelaVista.ViagemMcionaOrganizacao FROM
    'utilizador'@'localhost';
-- FLUSH PRIVILEGES;
-- 

```

XII. Script calculadora RelaX

```
group: BelaVista

-- Tabela Utilizador
Utilizador = {
  Id:number, Nome:string, Idade:number, Email:string, Telefone:string, Admin:number
  1, 'Débora Caetano', 19, 'debora.caetano@email.com', '912345678', 0
  2, 'Gonçalo Fernandes', 78, 'gonçalo.fernandes@email.com', '934567890', 1
  3, 'Jonas Johanes', 22, 'jonas.johanes@email.com', '914567890', 0
}

-- Tabela Organizacao
Organizacao = {
  Id:number, Nome:string
  1, 'CeBELUM'
  2, 'CeSIUM'
  3, 'BestTravel'
}

-- Tabela UtilizadorPertenceOrganizacao
UtilizadorPertenceOrganizacao = {
  Id_Utilizador:number, Id_Organizacao:number
  2, 2
  3, 2
  2, 3
}

-- Tabela Viagem
Viagem = {
  Id:number, Titulo:string, Descricao:string
  1, 'Tour pelo Douro', 'Viagem de dia inteiro pelo Vale do Douro com visita a quintas'
  2, 'Roteiro Histórico Porto', 'Descoberta dos principais monumentos históricos do Porto'
}

-- Tabela ViagemMencionaOrganizacao
ViagemMencionaOrganizacao = {
  Id_Viagem:number, Id_Organizacao:number
  1, 1
  2, 1
  1, 3
}

-- Tabela DiaDeViagem
DiaDeViagem = {
  Data:string, Id_Viagem:number, Custo:number, Alojamento:string, Transporte:string
  '2025-03-15', 1, 75.50, 'Hotel Douro Vista', 'Autocarro turístico'
  '2025-03-20', 2, 45.00, null, 'A pé'
  '2025-03-21', 2, 40.00, null, 'Metro'
}

-- Tabela PontoDeInteresse
PontoDeInteresse = {
  Id:number, Nome:string, Localizacao:string, Descricao:string, Horario:string, Data:string,
  Id_Viagem:number
  1, 'Quinta do Vesúvio', 'Douro', 'Quinta vinícola com provas de vinho', '09:00:00',
  '2025-03-15', 1
  2, 'Torre dos Clérigos', 'Porto Centro', 'Monumento icónico do Porto', '09:00:00', '2025-03-20',
  2
  3, 'Livraria Lello', 'Porto Centro', 'Uma das livrarias mais bonitas do mundo', '10:00:00',
  '2025-03-20', 2
  4, 'Palácio da Bolsa', 'Porto Centro', 'Um centro muito bonito', '14:00:00', '2025-03-21', 2
}

-- Tabela Fotografias
Fotografias = {
  Id_Fotografia:number, Id_PontoInteresse:number, Url_fotografia:string
  1, 1, 'https://fotos.belavista.com/douro/quinta-vesuvio-1.jpg'
  2, 2, 'https://fotos.belavista.com/porto/torre-clericos-exterior.jpg'
  3, 2, 'https://fotos.belavista.com/porto/torre-clericos-vista.jpg'
  4, 3, 'https://fotos.belavista.com/porto/lello-escadaria.jpg'
  5, 4, 'https://fotos.belavista.com/porto/palacio-bolsa.jpg'
}

-- Tabela ViagemRealizadaUtilizador
ViagemRealizadaUtilizador = {
  Id_Viagem:number, Id_Utilizador:number, Cargo:string, Avaliacao_Descritiva:string,
  Avaliacao_Quantitativa:number
  1, 1, 'organizador', 'Experiência maravilhosa! Tudo muito bem organizado.', 5
}
```

```
1, 2, 'participante', 'Adorei as quintas e as paisagens do Douro.', 5
2, 3, 'organizador', 'Porto é sempre uma boa escolha.', 4
}
```

XIII. Criação de vistas

```

-- == ===== VIEWS ===== --
-- Vistas de utilização do sistema de bases de dados.
-- == ===== --

USE BelaVista;

-- Custo das Viagens (ordem decrescente)
CREATE OR REPLACE VIEW vwCustoViagens AS
    SELECT
        v.Id,
        v.Titulo,
        SUM(dv.Custo) AS Custo_Total
    FROM
        Viagem AS v
        JOIN DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
    GROUP BY
        v.Id, v.Titulo
    ORDER BY
        Custo_Total DESC;

-- 
-- DROP VIEW vwCustoViagens;
-- 

-- Pontos de Interesse das Viagens
CREATE OR REPLACE VIEW vwPontosInteresseViagem AS
    SELECT
        v.Id AS Id_Viagem,
        v.Titulo,
        pi.Id AS Id_PontoInteresse,
        pi.Nome,
        dv.Data
    FROM
        Viagem AS v
    JOIN
        DiaDeViagem AS dv
        ON dv.Id_Viagem = v.Id
    JOIN
        PontoDeInteresse AS pi
        ON pi.Id_Viagem = dv.Id_Viagem
        AND pi.Data = dv.Data;

-- 
-- DROP VIEW vwPontosInteresseViagem;
-- 

-- Pontos de Interesse e a sua quantidade de Fotografias
CREATE OR REPLACE VIEW vwPontosdeInteresseFotografias AS
    SELECT
        pi.Id,
        pi.Nome,
        pi.Localizacao,
        COUNT(f.Id_Fotografia) AS Total_Fotografias
    FROM
        PontodeInteresse pi
        LEFT JOIN
        Fotografias AS f ON f.Id_PontoInteresse = pi.Id
    GROUP BY pi.Id, pi.Nome, pi.Localizacao;

-- 
-- DROP VIEW vwPontosdeInteresseFotografias;
-- 

-- Custos e Logistica por dia de viagem
CREATE OR REPLACE VIEW vwLogisticaDiaDeViagem AS
    SELECT
        d.Data,
        d.Id_Viagem,
        d.Custo,
        d.Alojamento,
        d.Transportes,
        v.Titulo AS Viagem
    FROM
        DiaDeViagem AS d
    JOIN

```

```

Viagem AS v ON v.Id = d.Id_Viagem;

-- 
-- DROP VIEW vwLogisticaDiaDeViagem;
-- 

-- Utilizadores e as suas viagens
CREATE OR REPLACE VIEW vwUtilizadoresViagens AS
SELECT
    u.Nome,
    u.Email,
    vru.Cargo,
    v.Titulo AS Viagem,
    vru.Avaliacao_Quantitativa,
    vru.Avaliacao_Descritiva
FROM
    Utilizador AS u
    JOIN
    ViagemRealizadaUtilizador AS vru ON vru.Id_Utilizador = u.Id
    JOIN
    Viagem v ON v.Id = vru.Id_Viagem;

-- 
-- DROP VIEW vwUtilizadoresViagens;
-- 

-- Organizadores e as suas viagens
CREATE OR REPLACE VIEW vwOrganizadoresViagem AS
SELECT
    u.Nome AS Organizador, u.Email, v.Titulo AS Viagem
FROM
    Viagem v
    JOIN
    ViagemRealizadaUtilizador vru ON vru.Id_Viagem = v.Id
        AND vru.Cargo = 'organizador'
    JOIN
    Utilizador u ON u.Id = vru.Id_Utilizador;

-- 
-- DROP VIEW vwOrganizadoresViagem;
-- 

-- Organizações associadas às Viagens
CREATE OR REPLACE VIEW vwViagensOrganizacoes AS
SELECT
    o.Nome AS Organizacao, v.Titulo AS Viagem
FROM
    Viagem AS v
    JOIN
    ViagemMencionaOrganizacao vmo ON vmo.Id_Viagem = v.Id
    JOIN
    Organizacao AS o ON o.Id = vmo.Id_Organizacao;

-- 
-- DROP VIEW vwViagensOrganizacoes;
-- 

-- Utilizadores e suas Organizações
CREATE OR REPLACE VIEW vwUtilizadoresOrganizacoes AS
SELECT
    u.Nome AS Utilizador, u.Email, o.Nome AS Organizacao
FROM
    Organizacao AS o
    JOIN
    UtilizadorPertenceOrganizacao AS upo ON upo.Id_Organizacao = o.Id
    JOIN
    Utilizador AS u ON u.Id = upo.Id_Utilizador
ORDER BY u.Nome ASC;

-- 
-- DROP VIEW vwUtilizadoresOrganizacoes;
-- 

-- Viagens e as suas Avaliações
CREATE OR REPLACE VIEW vwAvaliacoesViagem AS

```

```

SELECT
    v.Titulo,
    vru.Avaliacao_Quantitativa,
    vru.Avaliacao_Descritiva,
    u.Nome AS Utilizador
FROM
    ViagemRealizadaUtilizador AS vru
        JOIN
    Viagem AS v ON v.Id = vru.Id_Viagem
        JOIN
    Utilizador AS u ON u.Id = vru.Id_Utilizador
ORDER BY v.Titulo;

-- 
-- DROP VIEW vwAvaliacoesViagem;
-- 

-- Participantes por Viagem
CREATE OR REPLACE VIEW vwParticipantesViagem AS
SELECT
    v.Titulo AS Viagem, COUNT(*) AS Total_Participantes
FROM
    Viagem AS v
        JOIN
    ViagemRealizadaUtilizador AS vru ON vru.Id_Viagem = v.Id
WHERE
    vru.Cargo = 'participante'
GROUP BY v.Titulo;

-- 
-- DROP VIEW vwParticipantesViagem;
-- 

CREATE OR REPLACE VIEW vwViagemComItinerario AS
SELECT
    v.Id,
    v.Titulo,
    v.Descricao,
    GROUP_CONCAT(DISTINCT CONCAT(DATE_FORMAT(dv.Data, '%d/%m/%Y'),
        ',',
        dv.Custo,
        ' EUR'),
        IF(dv.Alojamento IS NOT NULL,
            CONCAT(' - ', dv.Alojamento),
            ''),
        IF(dv.Transportador IS NOT NULL,
            CONCAT(' - ', dv.Transportador),
            '')),
        ',
        'Pontos: ',
        (SELECT
            GROUP_CONCAT(CONCAT(TIME_FORMAT(pi2.Horario, '%H:%i'),
                ' - ',
                pi2.Nome,
                ',',
                pi2.Localizacao,
                ''))
            ORDER BY pi2.Horario
            SEPARATOR ''
        )
    )
        FROM
            PontoDeInteresse pi2
        WHERE
            pi2.Data = dv.Data
            AND pi2.Id_Viagem = v.Id)
    ORDER BY dv.Data
    SEPARATOR ''
    ') AS itinerario_completo
FROM
    Viagem AS v
        INNER JOIN
    DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
        LEFT JOIN
    PontoDeInteresse AS pi ON dv.Data = pi.Data
        AND pi.Id_Viagem = v.Id
GROUP BY v.Id , v.Titulo , v.Descricao;

```

```
--  
-- DROP VIEW vwViagemComItinerario;  
--
```

XIV. Interrogações do Utilizador

```
-- == ===== QUERIES ===== --
-- Interrogações utilizadas pelos utilizadores do sistema de base de dados.
-- == =====

USE BelaVista;

-- Calcular as datas de ida e regresso das viagens com base nos seus dias
SELECT
    v.Id AS IdViagem,
    v.Titulo AS NomeViagem,
    MIN(dv.Data) AS DataIda,
    MAX(dv.Data) AS DataRegresso
FROM
    Viagem AS v
    JOIN
        DiaDeViagem AS dv ON v.Id = dv.Id_Viagem
GROUP BY v.Id , v.Titulo;

-- Identificar o utilizador com maior número de viagens realizadas
SELECT
    u.Id AS UserId,
    u.Nome AS Nome_Utilizador,
    COUNT(vru.Id_Viagem) AS Total_Viagens
FROM
    Utilizador AS u
    JOIN
        ViagemRealizadaUtilizador AS vru ON u.Id = vru.Id_Utilizador
GROUP BY u.Id
ORDER BY Total_Viagens DESC
LIMIT 1;

-- Listar as avaliações (média de avaliação quantitativa) por viagem
SELECT
    v.Id AS IdViagem,
    v.Titulo AS NomeViagem,
    AVG(vru.Avaliacao_Quantitativa) AS MediaAvaliacao
FROM
    Viagem AS v
    JOIN
        ViagemRealizadaUtilizador AS vru ON v.Id = vru.Id_Viagem
GROUP BY v.Id , v.Titulo
ORDER BY MediaAvaliacao DESC;

-- Listar o custo total das viagens com base nos seus dias
SELECT
    V.Id, V.Titulo, SUM(DV.Custo) AS Custo_Viagem
FROM
    Viagem AS V
    INNER JOIN
        DiaDeViagem AS DV ON V.Id = DV.Id_Viagem
GROUP BY V.Id
ORDER BY V.Id;

-- Listar todos os utilizadores que pertencem a uma organização
PREPARE participantes_organização FROM
    'SELECT UPO.Id_Utilizador, U.Nome
     FROM UtilizadorPertenceOrganizacao AS UPO INNER JOIN Utilizador AS U
     ON UPO.Id_Utilizador = U.Id
     WHERE Id_Organizacao = ? ;';

SET @id_org = 2;
EXECUTE participantes_organização USING @id_org;
DEALLOCATE PREPARE participantes_organização;

-- Listar todos os participantes de uma viagem
PREPARE participantes_viagem FROM
    'SELECT VRU.Id_Utilizador, U.Nome
     FROM Viagem AS V INNER JOIN ViagemRealizadaUtilizador AS VRU
     ON V.Id = VRU.Id_Viagem INNER JOIN Utilizador AS U
     ON VRU.Id_Utilizador = U.Id
     WHERE (V.Id = ? ) AND (VRU.Cargo = ''participante''); ';
```

```

SET @id_viagem = 1;
EXECUTE participantes_viagem USING @id_viagem;
DEALLOCATE PREPARE participantes_viagem;

-- Listar quais as organizações com os membros mais ativos (que realizam mais viagens) e melhor
avaliam
SELECT
    o.Id,
    o.Nome AS Nome_Organizacao,
    COUNT(DISTINCT vru.Id_Viagem) AS Total_Viagens_Feitas,
    COUNT(DISTINCT vru.Id_Utilizador) AS Membros_Ativos,
    ROUND(AVG(vru.Avaliacao_Quantitativa), 2) AS Avaliacao_Media_Membros
FROM Organizacao o
LEFT JOIN UtilizadorPertenceOrganizacao upo ON o.Id = upo.Id_Organizacao
LEFT JOIN ViagemRealizadaUtilizador vru ON upo.Id_Utilizador = vru.Id_Utilizador
GROUP BY o.Id, o.Nome
ORDER BY Total_Viagens_Feitas DESC, Avaliacao_Media_Membros DESC;

```

XV. Transações

```

-- == ====== TRANSACTIONS ====== --
-- Procedures com transactions para manipulação segura de dados.
-- == ====== --

USE BelaVista;

DELIMITER $$

-- O procedimento createTrip permite que qualquer utilizador crie uma viagem, ficando
automaticamente como organizador
DROP PROCEDURE IF EXISTS createTrip$$
CREATE PROCEDURE createTrip(
    IN p_Id_Utilizador INT,
    IN p_Titulo VARCHAR(150),
    IN p_Descricao TEXT
)
BEGIN
    DECLARE v_Id_Viagem INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao criar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    INSERT INTO Viagem (Titulo, Descricao)
        VALUES (p_Titulo, p_Descricao);

    SET v_Id_Viagem = LAST_INSERT_ID();

    INSERT INTO ViagemRealizadaUtilizador (Id_Viagem, Id_Utilizador, Cargo)
        VALUES (v_Id_Viagem, p_Id_Utilizador, 'organizador');

    COMMIT;
    SELECT v_Id_Viagem AS Id_Viagem, 'Viagem criada com sucesso!' AS Mensagem;
END$$

-- O prodecimento editTrip permite que um organizador ou admin edite uma viagem
DROP PROCEDURE IF EXISTS editTrip$$
CREATE PROCEDURE editTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Titulo VARCHAR(150),
    IN p_Descricao TEXT
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

```

```

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    SET v_Erro = 'Erro ao editar viagem. Transação revertida.';
    ROLLBACK;
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END;

START TRANSACTION;

SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

IF v_IsAdmin = 0 THEN
    SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Cargo IS NULL THEN
        SET v_Erro = 'O Utilizador não participa nesta viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF v_Cargo != 'organizador' THEN
        SET v_Erro = 'Apenas organizadores podem editar viagens.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;
END IF;

UPDATE Viagem
SET Titulo = p_Titulo,
    Descricao = p_Descricao
WHERE Id = p_Id_Viagem;

COMMIT;
SELECT 'Viagem editada com sucesso!' AS Mensagem;
END$$

-- O procedimento deleteTrip permite que um organizador ou admin elimine uma viagem
DROP PROCEDURE IF EXISTS deleteTrip$$
CREATE PROCEDURE deleteTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'O Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem eliminar viagens.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    DELETE FROM Viagem WHERE Id = p_Id_Viagem;

    COMMIT;
    SELECT 'Viagem eliminada com sucesso!' AS Mensagem;
END$$

```

```

-- O prodecimento createTripDay permite que um utilizador organizador ou admin crie um dia de viagem
DROP PROCEDURE IF EXISTS createTripDay$$
CREATE PROCEDURE createTripDay(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE,
    IN p_Custo DECIMAL(8,2),
    IN p_Alojamento VARCHAR(150),
    IN p_Transporte VARCHAR(100)
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao criar dia de viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'O Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem criar dias de viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    IF p_Custo < 0 THEN
        SET v_Erro = 'O custo não pode ser negativo.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    INSERT INTO DiaDeViagem (Data, Id_Viagem, Custo, Alojamento, Transporte)
    VALUES (p_Data, p_Id_Viagem, p_Custo, p_Alojamento, p_Transporte);

    COMMIT;
    SELECT 'Dia de viagem criado com sucesso!' AS Mensagem;
END$$

-- O prodecimento editTripDay permite que um utilizador organizador ou admin edite um dia de viagem
DROP PROCEDURE IF EXISTS editTripDay$$
CREATE PROCEDURE editTripDay(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE,
    IN p_Custo DECIMAL(8,2),
    IN p_Alojamento VARCHAR(150),
    IN p_Transporte VARCHAR(100)
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao editar dia de viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

```

```

SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

IF v_IsAdmin = 0 THEN
    SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Cargo IS NULL THEN
        SET v_Erro = 'O Utilizador não participa nesta viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF v_Cargo != 'organizador' THEN
        SET v_Erro = 'Apenas organizadores podem editar dias de viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;
END IF;

IF p_Custo < 0 THEN
    SET v_Erro = 'O custo não pode ser negativo.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

UPDATE DiaDeViagem
SET Custo = p_Custo,
    Alojamento = p_Alojamento,
    Transporte = p_Transporte
WHERE Data = p_Data AND Id_Viagem = p_Id_Viagem;

COMMIT;
SELECT 'Dia de viagem editado com sucesso!' AS Mensagem;
END$$

-- O procedimento deleteTripDay permite que um utilizador organizador ou admin elimine um dia de viagem
DROP PROCEDURE IF EXISTS deleteTripDay$$
CREATE PROCEDURE deleteTripDay(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar dia de viagem. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'O Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem eliminar dias de viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    DELETE FROM DiaDeViagem
    WHERE Data = p_Data AND Id_Viagem = p_Id_Viagem;

    COMMIT;
    SELECT 'Dia de viagem eliminado com sucesso!' AS Mensagem;
END$$

```

```

-- O prodecimento createInterestPoint permite que um utilizador organizador ou admin crie um ponto de interesse.
DROP PROCEDURE IF EXISTS createInterestPoint$$
CREATE PROCEDURE createInterestPoint(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Data DATE,
    IN p_Nome VARCHAR(150),
    IN p_Localizacao VARCHAR(200),
    IN p_Descricao TEXT,
    IN p_Horario TIME
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Dia_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao criar ponto de interesse. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN
        SELECT Cargo INTO v_Cargo
        FROM ViagemRealizadaUtilizador
        WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

        IF v_Cargo IS NULL THEN
            SET v_Erro = 'Utilizador não participa nesta viagem.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;

        IF v_Cargo != 'organizador' THEN
            SET v_Erro = 'Apenas organizadores podem criar pontos de interesse.';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
        END IF;
    END IF;

    SELECT COUNT(*) INTO v_Dia_Existe
    FROM DiaDeViagem
    WHERE Data = p_Data AND Id_Viagem = p_Id_Viagem;

    IF v_Dia_Existe = 0 THEN
        SET v_Erro = 'Dia de viagem não existe. Crie o dia primeiro.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    INSERT INTO PontoDeInteresse (Nome, Localizacao, Descricao, Horario, Data, Id_Viagem)
    VALUES (p_Nome, p_Localizacao, p_Descricao, p_Horario, p_Data, p_Id_Viagem);

    COMMIT;
    SELECT LAST_INSERT_ID() AS Id_PontoInteresse, 'Ponto de interesse criado com sucesso!' AS Mensagem;
END$$

-- O prodecimento editInterestPoint permite que um utilizador organizador ou admin edite um ponto de interesse.
DROP PROCEDURE IF EXISTS editInterestPoint$$
CREATE PROCEDURE editInterestPoint(
    IN p_Id_Utilizador INT,
    IN p_Id INT,
    IN p_Nome VARCHAR(150),
    IN p_Localizacao VARCHAR(200),
    IN p_Descricao TEXT,
    IN p_Horario TIME
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Id_Viagem INT;
    DECLARE v_Erro VARCHAR(255);

```

```

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    SET v_Erro = 'Erro ao editar ponto de interesse. Transação revertida.';
    ROLLBACK;
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END;

START TRANSACTION;

SELECT Id_Viagem INTO v_Id_Viagem FROM PontoDeInteresse
WHERE Id = p_Id;

IF v_Id_Viagem IS NULL THEN
    SET v_Erro = 'Ponto de interesse não encontrado.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

IF v_IsAdmin = 0 THEN
    SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = v_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Cargo IS NULL THEN
        SET v_Erro = 'O Utilizador não participa nesta viagem.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF v_Cargo != 'organizador' THEN
        SET v_Erro = 'Apenas organizadores podem editar pontos de interesse.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;
END IF;

UPDATE PontoDeInteresse
SET Nome = p_Nome,
    Localizacao = p_Localizacao,
    Descricao = p_Descricao,
    Horario = p_Horario
WHERE Id = p_Id;

COMMIT;
SELECT 'Ponto de interesse editado com sucesso!' AS Mensagem;
END$$

-- O procedimento deleteInterestPoint permite que um utilizador organizador ou admin elimine um
ponto de interesse.
DROP PROCEDURE IF EXISTS deleteInterestPoint$$
CREATE PROCEDURE deleteInterestPoint(
    IN p_Id_Utilizador INT,
    IN p_Id INT
)
BEGIN
    DECLARE v_Cargo VARCHAR(20);
    DECLARE v_IsAdmin BOOLEAN;
    DECLARE v_Id_Viagem INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar ponto de interesse. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT Id_Viagem INTO v_Id_Viagem FROM PontoDeInteresse
    WHERE Id = p_Id;

    IF v_Id_Viagem IS NULL THEN
        SET v_Erro = 'Ponto de interesse não encontrado.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    SELECT isAdmin(p_Id_Utilizador) INTO v_IsAdmin;

    IF v_IsAdmin = 0 THEN

```

```

SELECT Cargo INTO v_Cargo FROM ViagemRealizadaUtilizador
WHERE Id_Viagem = v_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

IF v_Cargo IS NULL THEN
    SET v_Erro = 'Utilizador não participa nesta viagem.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

IF v_Cargo != 'organizador' THEN
    SET v_Erro = 'Apenas organizadores podem eliminar pontos de interesse.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;
END IF;

DELETE FROM PontoDeInteresse WHERE Id = p_Id;

COMMIT;
SELECT 'Ponto de interesse eliminado com sucesso!' AS Mensagem;
END$$

-- O procedimento reviewTrip permite que um utilizador avalie uma viagem em que tenha participado.
DROP PROCEDURE IF EXISTS reviewTrip $$
CREATE PROCEDURE reviewTrip(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT,
    IN p_Avaliacao_Descritiva TEXT,
    IN p_Avaliacao_Quantitativa INT
)
BEGIN
DECLARE v_Participa INT;
DECLARE v_Avaliacao_Existe INT;
DECLARE v_Erro VARCHAR(255);

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    SET v_Erro = 'Erro ao avaliar viagem. Transação revertida.';
    ROLLBACK;
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END;

START TRANSACTION;

SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

IF v_Participa = 0 THEN
    SET v_Erro = 'Apenas participantes da viagem podem avaliá-la.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

IF p_Avaliacao_Quantitativa < 0 OR p_Avaliacao_Quantitativa > 5 THEN
    SET v_Erro = 'A avaliação quantitativa deve estar entre 0 e 5.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador
AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

IF v_Avaliacao_Existe > 0 THEN
    SET v_Erro = 'Já existe uma avaliação para esta viagem.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

UPDATE ViagemRealizadaUtilizador
SET Avaliacao_Descritiva = p_Avaliacao_Descritiva,
    Avaliacao_Quantitativa = p_Avaliacao_Quantitativa
WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

COMMIT;
SELECT 'Avaliação criada com sucesso!' AS Mensagem;
END$$

-- O procedimento editTripReview permite que um utilizador edite uma avaliação sua numa dada
viagem em que tenha participado.
DROP PROCEDURE IF EXISTS editTripReview $$
CREATE PROCEDURE editTripReview(
    IN p_Id_Utilizador INT,

```

```

        IN p_Id_Viagem INT,
        IN p_Avaliacao_Descritiva TEXT,
        IN p_Avaliacao_Quantitativa INT
    )
BEGIN
    DECLARE v_Participa INT;
    DECLARE v_Avaliacao_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao editar avaliação. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Participa = 0 THEN
        SET v_Erro = 'A avaliação não existe ou não pertence ao utilizador.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem
    AND Id_Utilizador = p_Id_Utilizador
    AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

    IF v_Avaliacao_Existe = 0 THEN
        SET v_Erro = 'A avaliação não existe';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    IF p_Avaliacao_Quantitativa < 0 OR p_Avaliacao_Quantitativa > 5 THEN
        SET v_Erro = 'A avaliação quantitativa deve estar entre 0 e 5.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

    UPDATE ViagemRealizadaUtilizador
    SET Avaliacao_Descritiva = p_Avaliacao_Descritiva,
        Avaliacao_Quantitativa = p_Avaliacao_Quantitativa
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    COMMIT;
    SELECT 'Avaliação editada com sucesso!' AS Mensagem;
END$$

-- O procedimento deleteTripReview permite que um utilizador elimine uma avaliação sua, numa dada viagem em que tenha participado.
DROP PROCEDURE IF EXISTS deleteTripReview $$
CREATE PROCEDURE deleteTripReview(
    IN p_Id_Utilizador INT,
    IN p_Id_Viagem INT
)
BEGIN
    DECLARE v_Participa INT;
    DECLARE v_Avaliacao_Existe INT;
    DECLARE v_Erro VARCHAR(255);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET v_Erro = 'Erro ao eliminar avaliação. Transação revertida.';
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END;

    START TRANSACTION;

    SELECT COUNT(*) INTO v_Participa FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

    IF v_Participa = 0 THEN
        SET v_Erro = 'A avaliação não existe ou não pertence ao utilizador.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
    END IF;

```

```

SELECT COUNT(*) INTO v_Avaliacao_Existe FROM ViagemRealizadaUtilizador
    WHERE Id_Viagem = p_Id_Viagem
      AND Id_Utilizador = p_Id_Utilizador
      AND (Avaliacao_Descritiva IS NOT NULL OR Avaliacao_Quantitativa IS NOT NULL);

IF v_Avaliacao_Existe = 0 THEN
    SET v_Erro = 'A avaliação não existe.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_Erro;
END IF;

UPDATE ViagemRealizadaUtilizador
    SET Avaliacao_Descritiva = NULL,
        Avaliacao_Quantitativa = NULL
    WHERE Id_Viagem = p_Id_Viagem AND Id_Utilizador = p_Id_Utilizador;

COMMIT;
SELECT 'Avaliação eliminada com sucesso!' AS Mensagem;
END$$

DELIMITER ;

-- EXEMPLOS:

-- CALL createTrip(3, 'Viagem a Paris', 'Uma viagem incrível pela cidade do amor');

-- CALL editTrip(2, 1, 'Novo Título', 'Nova Descrição');

-- CALL deleteTrip(2, 1);

-- CALL createTripDay(2, 1, '2024-06-15', 150.50, 'Hotel XYZ', 'Autocarro');

-- CALL editTripDay(2, 1, '2024-06-15', 180.00, 'Hotel ABC', 'Comboio');

-- CALL deleteTripDay(2, 1, '2024-06-15');

-- CALL createInterestPoint(2, 1, 'Torre Eiffel', 'Paris, França', 'Monumento histórico', '14:30:00');

-- CALL editInterestPoint(2, 1, 'Torre Eiffel', 'Paris, França', 'Descrição atualizada', '15:00:00');

-- CALL deleteInterestPoint(2, 1);

-- CALL reviewTrip(3, 1, 'Viagem excelente!', 5);

-- CALL editTripReview(3, 1, 'Viagem muito boa.', 4);

-- CALL deleteTripReview(3, 1);

```