Universidade do Minho
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2025/2026

BeLIUM Viagens

Bruno Magalhães, Diogo Azevedo, Simão Santos,

Vera Almeida

Dezembro, 2025

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

BeLIUM Viagens

Bruno Magalhães, Diogo Azevedo, Simão Santos,

Vera Almeida

Dezembro, 2025

Resumo

O presente trabalho descreve o processo de conceção, modelação e implementação do sistema de base de dados intitulado "BeLIUM Viagens", desenvolvido sob solicitação do CeSIUM (Centro de Estudantes de Engenharia Informática da Universidade do Minho). A investigação e desenvolvimento aqui detalhados visam mitigar a fragmentação e dispersão da informação histórica e operacional do núcleo, atualmente distribuída por canais informais e documentos isolados, fator que compromete a preservação da memória institucional e a eficácia no planeamento de atividades pedagógicas.

A metodologia adotada seguiu rigorosamente o ciclo de vida do desenvolvimento de um sistema de base de dados, iniciando-se com o levantamento de requisitos de descrição, manipulação e controlo. A arquitetura do sistema fundamentou-se numa modelação conceptual baseada em diagramas Entidade-Relacionamento (ER), seguida de uma modelação lógica normalizada até à Terceira Forma Normal (3FN), garantindo a integridade referencial e a eliminação de redundâncias. A implementação física foi concretizada no Sistema de Gestão de Bases de Dados MySQL, onde foram criados diversos utilizadores com diferentes privilégios de acesso. A base de dados foi povoada com dados representativos, e foram desenvolvidas as interrogações necessárias ao seu funcionamento, complementadas por procedimentos, funções e gatilhos.

A equipa de desenvolvimento mantém agora um compromisso de prestação de serviços contínuos ao CeSIUM, assegurando a operacionalidade da base de dados e propondo melhorias contínuas com base no feedback dos utilizadores que a utilizam diariamente.

Área de Aplicação: Desenho e Arquitetura de Bases de Dados

Palavras-Chave: Bases de Dados, Diagramas ER, Modelo Relacional, Normalização de Bases de Dados Relacionais, Álgebra Relacional, SQL, MySQL, InnoDB

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

1.1. Contexto de Aplicação

O CeSIUM (Centro de Estudantes de Engenharia Informática da Universidade do Minho) é uma organização composta por alunos voluntários, cujo principal objetivo é representar, promover e apoiar o curso de Engenharia Informática e a sua comunidade académica. Para isso, dinamiza diversas atividades de carácter pedagógico e social, como workshops, sessões de estudo e eventos de convívio, que complementam a formação dos estudantes e fortalecem o espírito de equipa.

Ao longo dos anos, o CeSIUM tem realizado várias viagens e iniciativas que marcaram diferentes gerações de estudantes. Contudo, estas informações encontram-se atualmente dispersas por conversas informais, redes sociais e documentos isolados, o que dificulta a preservação estruturada da memória histórica do núcleo.

Face a esta limitação, surgiu a necessidade de criar um sistema centralizado que permita registar e consultar as viagens associadas ao CeSIUM. Com o apoio dos alumni do núcleo, foi então proposta a implementação de um Sistema de Base de Dados (SBD) que permita organizar e preservar estes registos de forma consistente e acessível.

1.2. Motivação e Objetivos do Trabalho

O constante e exponencial crescimento do interesse e impacto da área da informática nas últimas décadas, assim como a sua natural internacionalização, torna cada vez mais pertinente expandir o alcance e a presença do CeSIUM para além do ambiente académico local.

Tendo isto em conta, o CeSIUM pretende organizar, ocasionalmente, viagens tanto com fins pedagógicos como recreativos para a sua comunidade. No entanto, o facto de a informação se encontrar distribuída por várias fontes dificulta imensamente este processo se for levado à escala pretendida. Neste contexto, o desenvolvimento do projeto BeLIUM Viagens, que reúne as informações das viagens e experiências das diversas gerações da organização, é perfeitamente justificável.

Para executar um projeto desta dimensão, e considerando o seu provável crescimento futuro, uma estruturação eficiente e consistente dos dados relativos às viagens dos membros, através de um sistema de bases de dados, é crucial. Com esta ferramenta de planeamento e gestão ativa, o CeSIUM pretende:

- Criar um portfólio público que consolide, num só local, os momentos partilhados na comunidade do CeSIUM, preservando a sua memória histórica e cultural.
- Simplificar as decisões relativas a futuros eventos fora do ambiente académico, melhorando a gestão do investimento financeiro associado.
- Garantir que as viagens organizadas alcancem os objetivos pretendidos, facilitando a escolha dos destinos mais adequados através de uma organização eficiente dos dados.
- Assegurar a segurança e integridade dos dados, evitando dependência de plataformas externas e garantindo que o CeSIUM mantém controlo sobre a informação partilhada.
- Alargar a rede de *networking* e promover a partilha de informações entre as diferentes gerações de estudantes do CeSIUM.

1.3. Análise da Viabilidade do processo

A viabilidade do projeto será analisada em três componentes principais – técnica, económica e operacional – de modo a garantir que o desenvolvimento do sistema BeLIUM seja exequível, sustentável e capaz de atingir os objetivos estabelecidos.

- A nível técnico, o projeto é viável, uma vez que o CeSIUM dispõe de recursos humanos qualificados, nomeadamente estudantes e *alumni* de Engenharia Informática, capazes de desenvolver e manter o sistema utilizando tecnologias modernas.
- A nível económico, o investimento necessário é reduzido, pois o desenvolvimento será realizado internamente por uma equipa do Centro de Apoio ao Open Source (CAOS), um departamento do CeSIUM, com custos limitados à manutenção periódica do sistema. Além disso, o projeto já conta com apoio financeiro dos *alumni*, e a expansão da presença do CeSIUM aumentará o interesse de patrocinadores, consolidando a sustentabilidade financeira.
- A nível operacional, a adesão da comunidade é expectável, dado o interesse natural dos membros em partilhar as suas experiências e registar as suas viagens.

Para além destes fatores, com a centralização dos dados e a implementação de um espaço digital colaborativo, o CeSIUM estima que, ao adotar um sistema mais eficiente de armazenamento e gestão de dados, conseguirá:

- Reduzir em 40% a perda de informações sobre viagens realizadas, através da centralização dos dados num único repositório digital, dado que, atualmente, estes estão completamente dispersos.
- Aumentar em 15% o número de partilhas e interações entre membros e *alumni*, mediante a disponibilização de um espaço digital colaborativo.
- Aumentar em 30% o retorno financeiro derivado de patrocínios, a principal fonte de fundos do CeSIUM, considerando a nova possibilidade de patrocinadores internacionais.

Assim, o CeSIUM conclui que a implementação do BeLIUM Viagens é viável e rentável, desde que os custos de desenvolvimento sejam controlados.

1.4. Recursos e Equipa de Trabalho

Os recursos humanos necessários para o projeto são constituídos inteiramente por membros e ex-membros do CeSIUM, divididos em dois grupos: supervisores e desenvolvedores. Os supervisores possuem experiência na liderança e organização de projetos do núcleo, enquanto os desenvolvedores são estudantes proficientes em programação e gestão de bases de dados. Estes dois grupos constituem o corpo colaborativo do projeto (CCP).

No que diz respeito aos supervisores, estes são responsáveis pela coordenação do projeto, definição de objetivos, validação das etapas do desenvolvimento do sistema e gestão das tarefas atribuídas ao grupo de desenvolvedores, assegurando a orientação e organização do projeto de acordo com os objetivos definidos. O grupo é constituído por:

1. Júlio Pinto: atual presidente do CeSIUM e líder do projeto.
2. Gustavo Pereira: ex-presidente do CeSIUM e atual representante da comunidade *alumni*, responsável pela distribuição das tarefas e pela validação de etapas.
3. Rui Afonso: *alumni* fundador do CeSIUM, encarregado da manutenção geral da organização do projeto.

Quanto aos desenvolvedores, este grupo é constituído por Bruno Magalhães, Diogo Azevedo, Simão Santos e Vera Almeida, membros do CAOS. São responsáveis pelo desenvolvimento efetivo do sistema, incluindo a implementação da base de dados, programação das funcionalidades, testes e manutenção do software. Cada membro contribui com competências específicas, garantindo que o projeto seja desenvolvido de forma eficiente, organizada e alinhada com os objetivos definidos pelos supervisores.

A tabela seguinte resume os recursos humanos do projeto:

Nome	*Júlio Pinto*	*Gustavo Pereira*	*Rui Afonso*	*Bruno Magalhães*	*Diogo Azevedo*	*Simão Santos*	*Vera Almeida*
Função	Supervisor	Supervisor	Supervisor	Desenv.	Desenv.	Desenv.	Desenv.
Relação com o CeSIUM	Atual Presidente	Ex-Presidente	Fundador	Atual sócio	Atual sócio	Atual sócio	Atual sócio

Tabela 1 - Corpo colaborativo do projeto BeLIUM Viagens.

Relativamente aos recursos materiais, estes dividem-se em *software* e *hardware*:

- *Software*: serão utilizadas ferramentas adequadas ao desenvolvimento do sistema, incluindo uma ferramenta de modelação ER, um sistema de gestão de bases de dados (MySQL) e plataformas de colaboração online para comunicação e acompanhamento das tarefas.
- *Hardware*: cada membro do grupo de desenvolvedores deve dispor de um computador pessoal, essencial para a implementação e manutenção do sistema de base de dados (SBD).

1.5. Plano de Execução do Projeto

O corpo colaborativo do projeto reuniu-se com o representante dos *alumni* e com o atual presidente do núcleo para planear o desenvolvimento do sistema. Dessa reunião resultou um plano de trabalhos detalhado num diagrama de Gantt no qual foram definidos o período de execução, as fases do ciclo de vida do sistema, as tarefas a realizar e os intervenientes em cada uma delas.

O diagrama distribui o tempo do projeto de forma faseada e lógica. As fases iniciais têm menor duração por serem mais conceptuais, enquanto as fases de modelação e implementação carecem de mais tempo por serem críticas e exigirem maior detalhe técnico. O cronograma foi ainda elaborado com alguma margem de manobra, de modo a acomodar eventuais imprevistos ou períodos de menor disponibilidade da equipa.

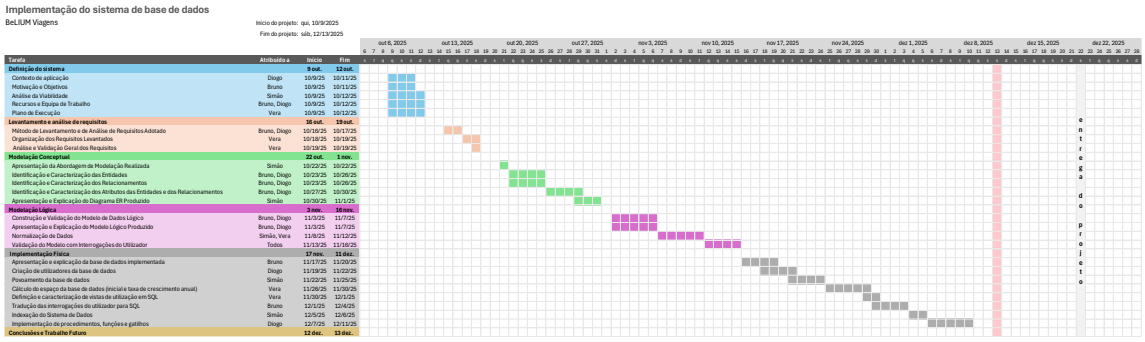
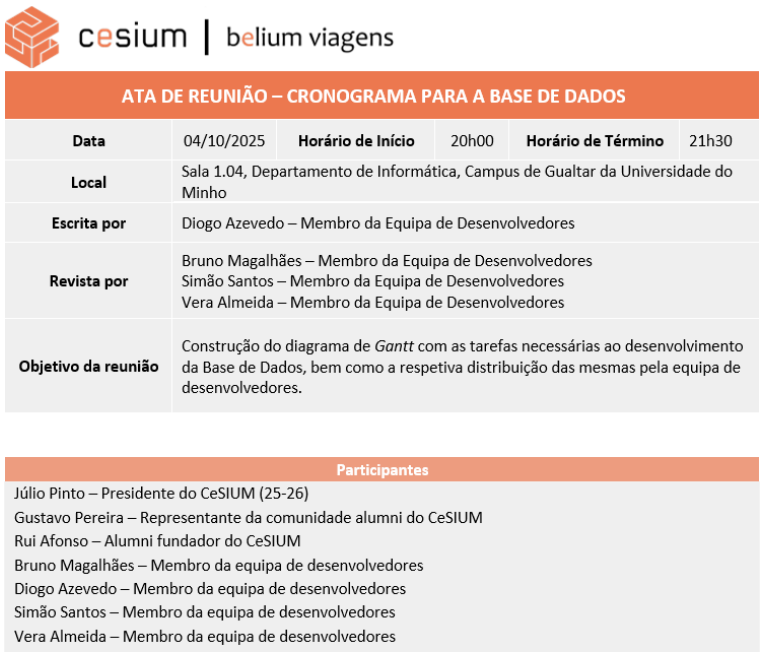


Figura 1 - Diagrama de Gantt relativo ao planeamento de tarefas.



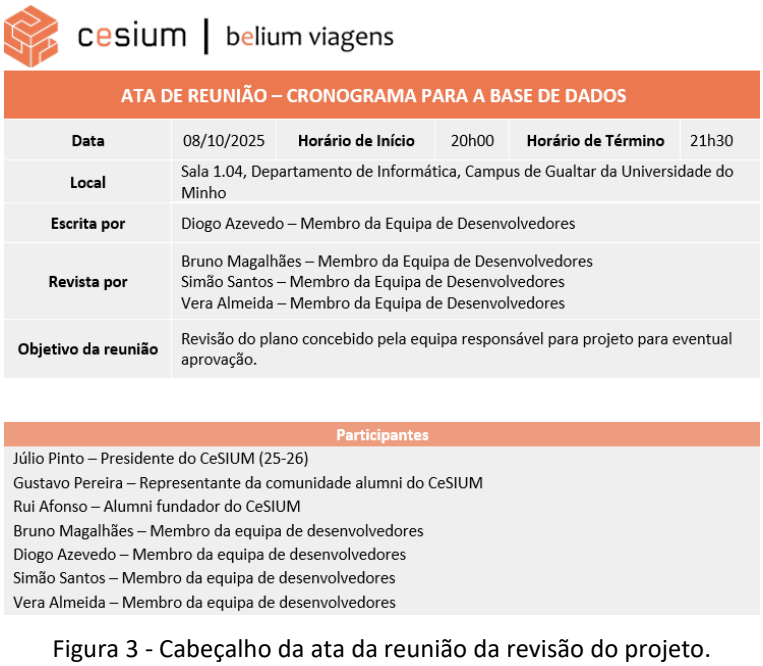
cesium | belium viagens

ATA DE REUNIÃO – CRONOGRAMA PARA A BASE DE DADOS					
Data	04/10/2025	**Horário de Início**	20h00	**Horário de Término**	21h30
Local	Sala 1.04, Departamento de Informática, Campus de Gualtar da Universidade do Minho				
Escrita por	Diogo Azevedo – Membro da Equipa de Desenvolvedores				
Revista por	Bruno Magalhães – Membro da Equipa de Desenvolvedores Simão Santos – Membro da Equipa de Desenvolvedores Vera Almeida – Membro da Equipa de Desenvolvedores				
Objetivo da reunião	Construção do diagrama de *Gantt* com as tarefas necessárias ao desenvolvimento da Base de Dados, bem como a respetiva distribuição das mesmas pela equipa de desenvolvedores.				

Participantes
Júlio Pinto – Presidente do CeSIUM (25-26)
Gustavo Pereira – Representante da comunidade alumni do CeSIUM
Rui Afonso – Alumni fundador do CeSIUM
Bruno Magalhães – Membro da equipa de desenvolvedores
Diogo Azevedo – Membro da equipa de desenvolvedores
Simão Santos – Membro da equipa de desenvolvedores
Vera Almeida – Membro da equipa de desenvolvedores

Figura 2 - Cabeçalho da ata da reunião de construção do cronograma para o desenvolvimento da BD.

Após uma análise a fundo ao plano de execução, a equipa responsável pelo projeto agendou novamente uma reunião com os representantes dos *alumni* onde foi novamente reiterado o plano do projeto a desenvolver. Ambos os representantes confirmaram o que foi definido, fechando assim o contrato com o CeSIUM num valor de 500€, por parte dos *alumni*, para desenvolver o sistema de bases de dados.



cesium | belium viagens

ATA DE REUNIÃO – CRONOGRAMA PARA A BASE DE DADOS					
Data	08/10/2025	**Horário de Início**	20h00	**Horário de Término**	21h30
Local	Sala 1.04, Departamento de Informática, Campus de Gualtar da Universidade do Minho				
Escrita por	Diogo Azevedo – Membro da Equipa de Desenvolvedores				
Revista por	Bruno Magalhães – Membro da Equipa de Desenvolvedores Simão Santos – Membro da Equipa de Desenvolvedores Vera Almeida – Membro da Equipa de Desenvolvedores				
Objetivo da reunião	Revisão do plano concebido pela equipa responsável para projeto para eventual aprovação.				

Participantes
Júlio Pinto – Presidente do CeSIUM (25-26)
Gustavo Pereira – Representante da comunidade alumni do CeSIUM
Rui Afonso – Alumni fundador do CeSIUM
Bruno Magalhães – Membro da equipa de desenvolvedores
Diogo Azevedo – Membro da equipa de desenvolvedores
Simão Santos – Membro da equipa de desenvolvedores
Vera Almeida – Membro da equipa de desenvolvedores

Figura 3 - Cabeçalho da ata da reunião da revisão do projeto.

2. Levantamento e Análise de Requisitos

2.1. Método de Levantamento e Análise de Requisitos Adotado

Para o levantamento de requisitos, com o objetivo de compreender as necessidades e expectativas dos *alumni* e do CeSIUM, o corpo colaborativo do projeto recorreu a duas metodologias:

- Reuniões entre todos os elementos do corpo colaborativo do projeto, nas quais foram discutidas, em particular, as necessidades internas, tais como os custos associados às viagens.
- Inquéritos à comunidade académica local, cujo principal objetivo era determinar os interesses dos membros da mesma no que diz respeito ao acesso a informação de viagens de terceiros.

Reuniões entre o corpo colaborativo do projeto

Nas reuniões, os representantes dos ex-alunos apresentaram e clarificaram a sua visão inicial, explicitando as funcionalidades pretendidas para o sistema de base de dados a desenvolver. Estas conversas permitiram identificar quais os dados que deveriam ser registados e posteriormente consultados, assim como os grupos de pessoas que teriam permissão para o fazer. Paralelamente, o grupo de desenvolvedores orientou as discussões de modo a assegurar que toda a informação essencial fosse recolhida com rigor, clareza e coerência através das seguintes práticas:

- Para clarificar o comportamento esperado do sistema, foram elaborados casos de uso e cenários exemplificativos que descrevem como cada tipo de utilizador interage com as principais funcionalidades. A partir dessas situações foi possível identificar quais os utilizadores com permissão para executar determinadas ações como o registo de viagens ou a consulta de experiências anteriores.
- Foram levantadas, frequentemente, questões complementares relacionadas com os limites e intervalos de determinados atributos (por exemplo, se era possível a mesma paragem possuir diversas fotos associadas), com o intuito de definir valores coerentes e contribuir para a precisão e consistência dos dados modelados.

- Antes de terminar as reuniões, era feita uma apresentação da informação acumulada ao grupo de supervisores, de forma a confirmar se os requisitos definidos correspondiam às suas necessidades e expectativas.

Inquéritos à comunidade

Para além da expansão do alcance do CeSIUM, o projeto BeLIUM Viagens visa criar um local virtual público para auxiliar os estudantes da comunidade de informática. Por esse motivo, o corpo colaborativo do projeto decidiu que a realização de inquéritos à comunidade era vital na determinação de formas de tornar a plataforma realmente útil e de interesse para os estudantes.

Os inquéritos foram enviados na plataforma *Outlook*, uma vez que todos os estudantes da Universidade do Minho possuem uma conta na mesma. Tendo isto em conta, os inquéritos realizados apresentaram a seguinte estrutura:

cesium | belium viagens

INQUÉRITO À COMUNIDADE ESTUDANTIL DE ENGENHARIA INFORMÁTICA DA UM

Data de Envio	16/10/2025	**Data de Recolha**	31/10/2025
Escrita por	Diogo Azevedo e Bruno Magalhães – Membro da equipa de desenvolvedores		
Revista por	Simão Santos – Membro da equipa de desenvolvedores Vera Almeida – Membro da equipa de desenvolvedores		

Inquérito

Contexto

Com o objetivo de formar um portfólio para as experiências da comunidade CeSIUM, o núcleo pretende desenvolver um sistema capaz de armazenar viagens, tanto do foro pessoal como profissional. Embora apenas sócios do CeSIUM possam publicar as suas experiências na plataforma, esta será aberta para todos.

Tendo em conta a crescente importância das experiências internacionais na vida dos estudantes, em particular na área da informática, esta plataforma permitirá, tanto a nós como a ti, facilitar as decisões futuras nesta vertente.

Para determinar os requisitos mais importantes para esta plataforma, pedimos-te que respondesses à seguinte pergunta:

Se tivesses acesso à base de dados em questão, que dados e informações gostarias de ver disponibilizados?

- Despesas
- Localização
- Fotos
- Descrição da Experiência
- Outros (escreve no espaço em branco)

Figura 4 - Estrutura do inquérito enviado à comunidade académica.

Toda a informação recolhida foi devidamente registada e documentada, constituindo a base estruturante para o seu desenvolvimento conceptual, lógico e físico. Este processo de recolha e validação sistemática de requisitos garantiu que o sistema projetado respondesse, efetivamente, às necessidades expressas pela comunidade de antigos alunos e às exigências funcionais estabelecidas no âmbito do projeto.

2.2. Organização dos Requisitos Levantados

De modo a organizar os diferentes requisitos para a base de dados, a equipa decidiu categorizá-los em requisitos de descrição (RD), de manipulação (RM) e de controlo (RC), consoante o tipo de operações a que estão associados: criação da base de dados, manipulação da mesma e controlo do acesso aos dados por diferentes utilizadores, respetivamente.

Os requisitos de descrição (Requisitos de Descrição) foram definidos a partir da identificação das principais entidades, bem como dos relacionamentos existentes entre elas. Foram descritos os atributos essenciais de cada entidade, distinguindo os obrigatórios dos opcionais, e estabelecidas as respetivas identificações únicas. Adicionalmente, foram especificadas as associações entre utilizadores, sócios, viagens, paragens e patrocinadores, incluindo as suas cardinalidades e restrições.

Número	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista
RD1	20/10/2025	10:16	Uma viagem é identificada pelo seu ID. Cada viagem contém um objetivo, uma data inicial e uma data final, um custo associado, o número de participantes, o número de paragens e um ou mais tipos de deslocamento.	Utilizadores	Reunião do CCP	Desenvolvedores
RD2	20/10/2025	10:18	Um sócio pode publicar viagens. Cada viagem pode ou não ser publicada por um ou mais sócios.	Sócios	Reunião do CCP	Desenvolvedores
RD3	20/10/2025	10:20	Cada sócio é caracterizado pelo seu número de sócio e, opcionalmente, pelo estatuto dentro do núcleo e por uma foto de perfil.	Sócios	Reunião do CCP	Desenvolvedores
RD4	20/10/2025	10:32	Uma viagem possui pelo menos uma paragem. Cada paragem pertence obrigatoriamente a uma única viagem. A viagem deve, também, possuir o número de paragens associadas.	Utilizadores	Reunião do CCP	Desenvolvedores

Tabela 2 - Primeiros requisitos de descrição.

Os requisitos de manipulação (Requisitos de Manipulação) foram elaborados para suportar as consultas essenciais do sistema, estruturando-se em três categorias principais: operações CRUD (Create, Read, Update, Delete) para gestão das entidades, consultas parametrizadas com ordenação e filtros (datas, custos, localização, etc.), e agregações para cálculo de totais, médias e rankings.

Número	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista
RM1	27/10/2025	14:10	Registar novos sócios e alterar os dados dos existentes.	Administração	Reunião do CCP	Desenvolvedores
RM2	27/10/2025	14:12	Registar novos utilizadores e alterar os dados dos existentes.	Administração	Reunião do CCP	Desenvolvedores
RM3	27/10/2025	14:16	Listar, por ordem, todas as paragens de uma dada viagem.	Utilizadores/ Administração	Reunião do CCP	Desenvolvedores
RM4	27/10/2025	14:20	Listar todos os sócios (número de sócio, nome e estatuto) que realizaram uma determinada viagem (por ordem alfabética).	Utilizadores	Reunião do CCP	Desenvolvedores

Tabela 3 - Primeiros requisitos de manipulação.

Os requisitos de controlo (Requisitos de Controlo) foram definidos para garantir a segurança e integridade dos dados, estruturados em três níveis de acesso com permissões hierárquicas (utilizador, sócio, administrador). A autenticação é obrigatória e as credenciais iniciais são distribuídas de forma segura pelos administradores. Operações críticas, como a modificação ou eliminação de conteúdo, são restritas aos criadores ou a administradores. A segurança e controlo são assegurados por um plano de backups automatizados (diários e mensais).

Número	Data	Hora	Descrição	Vista de utilização	Fonte	Analista
RC1	28/10/2025	14:39	Para se registar, um utilizador deve pedir diretamente a um administrador ou enviar um pedido, por email. Se o pedido for aceite, um administrador deve registá-lo no sistema e comunicá-lo do seu ID e da sua password.	Administração	Reunião do CCP	Desenvolvedores
RC2	28/10/2025	14:42	Um utilizador pode consultar todos os dados dos sócios, das viagens e das paragens no sistema.	Utilizadores	Reunião do CCP	Desenvolvedores
RC3	28/10/2025	14:46	Um utilizador pode atualizar a sua password, se souber a atual.	Utilizadores	Reunião do CCP	Desenvolvedores
RC4	28/10/2025	14:54	Não devem existir múltiplos registos de utilizadores com o mesmo email.	Utilizadores	Reunião do CCP	Desenvolvedores

Tabela 4 - Primeiros requisitos de controlo.

Com base na análise dos requisitos e da estrutura da base de dados BeLIUM_Viagens, identificam-se três vistas de utilização principais, de acordo com as suas necessidades específicas dentro do sistema, denominadas "Utilizadores", "Sócios" e "Administração".

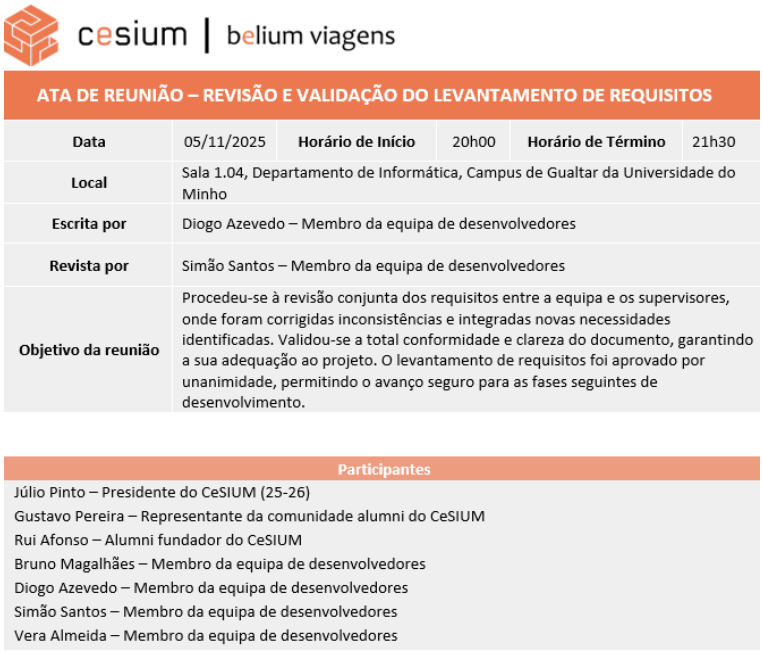
A vista "Utilizadores" engloba todas as funcionalidades destinadas aos utilizadores gerais da plataforma, incluindo membros da comunidade académica e visitantes. Estes utilizadores têm acesso limitado aos dados, focando-se principalmente na consulta e visualização de informações públicas sobre as viagens.

A vista "Sócios" é destinada aos sócios do CeSIUM, incluindo membros da direção e *alumni*. Estes utilizadores têm privilégios adicionais que lhes permitem contribuir ativamente para o conteúdo da plataforma e aceder a informações mais detalhadas sobre as atividades do núcleo.

Finalmente, a vista de utilização "Administração" é reservada aos administradores do sistema (membros do CCP), que têm responsabilidades de gestão global do sistema, incluindo aspetos financeiros, de segurança e de manutenção da integridade dos dados.

2.3. Análise e Validação Geral dos Requisitos

Após a conclusão do levantamento de requisitos, a equipa de desenvolvedores, em conjunto com os supervisores, realizaram uma revisão detalhada de todas as vistas, vertentes e requisitos. Durante esta reunião, a documentação foi analisada de forma sistemática, permitindo a deteção e correção de pequenas inconsistências, bem como a integração de requisitos adicionais resultantes da discussão entre as partes. O processo de validação assegurou conformidade, completude e clareza do documento de requisitos, garantindo a sua adequação às necessidades identificadas. No final desta revisão exaustiva, todos os intervenientes aprovaram de forma unânime os requisitos levantados, estabelecendo uma base sólida para as fases subsequentes do desenvolvimento do projeto.



cesium | belium viagens

ATA DE REUNIÃO – REVISÃO E VALIDAÇÃO DO LEVANTAMENTO DE REQUISITOS					
Data	05/11/2025	**Horário de Início**	20h00	**Horário de Término**	21h30
Local	Sala 1.04, Departamento de Informática, Campus de Gualtar da Universidade do Minho				
Escrita por	Diogo Azevedo – Membro da equipa de desenvolvedores				
Revista por	Simão Santos – Membro da equipa de desenvolvedores				
Objetivo da reunião	Procedeu-se à revisão conjunta dos requisitos entre a equipa e os supervisores, onde foram corrigidas inconsistências e integradas novas necessidades identificadas. Validou-se a total conformidade e clareza do documento, garantindo a sua adequação ao projeto. O levantamento de requisitos foi aprovado por unanimidade, permitindo o avanço seguro para as fases seguintes de desenvolvimento.				

Participantes
Júlio Pinto – Presidente do CeSIUM (25-26)
Gustavo Pereira – Representante da comunidade alumni do CeSIUM
Rui Afonso – Alumni fundador do CeSIUM
Bruno Magalhães – Membro da equipa de desenvolvedores
Diogo Azevedo – Membro da equipa de desenvolvedores
Simão Santos – Membro da equipa de desenvolvedores
Vera Almeida – Membro da equipa de desenvolvedores

Figura 5 - Cabeçalho da ata da reunião de revisão e validação do levantamento de requisitos.

3. Modelação Conceptual

3.1. Apresentação da Abordagem de Modelação Realizada

À medida que se realizava o levantamento e análise de requisitos, a equipa de desenvolvedores iniciou a estruturação da base de dados do sistema BeLIUM Viagens. O processo começou com a construção de um diagrama Entidade-Relacionamento (ER), que permite representar de forma visual as entidades principais do sistema e os relacionamentos entre elas. Para a criação do diagrama foi utilizada a ferramenta *brModelo*, escolhida pela sua simplicidade e pela experiência prévia dos elementos da equipa na sua utilização. Esta ferramenta adota uma notação de Peter Chen Min-Max, muito reconhecida na modelação de bases de dados.

A equipa começou por identificar as entidades e, de seguida, os relacionamentos entre elas. Após esta identificação, foram definidos os atributos de cada entidade e de cada relacionamento, assim como as respetivas cardinalidades, de modo a garantir que o modelo refletisse com rigor os requisitos recolhidos. Foi adotada uma abordagem monovista, isto é, a criação de um modelo único que integra todas as perspetivas de utilização do sistema. Esta opção foi considerada a mais adequada, tendo em conta o número reduzido de requisitos e a necessidade de manter uma estrutura coerente e simples de interpretar.

O resultado desta fase foi um modelo conceptual que serviu de base sólida para as etapas seguintes de desenvolvimento, estando assim garantido que o sistema BeLIUM Viagens representa de forma clara e estruturada toda a informação relativa às viagens e experiências da comunidade CeSIUM.

3.2. Identificação e Caracterização das Entidades

O desenvolvimento de uma base de dados exige uma identificação rigorosa das entidades que a compõem. Esta etapa é fundamental, pois constitui o alicerce para, numa fase posterior, se proceder à determinação dos respetivos relacionamentos e atributos.

O processo de identificação das entidades baseou-se na análise e interpretação dos requisitos, acompanhadas de uma reflexão crítica sobre a possibilidade de cada elemento identificado reunir as condições necessárias para ser considerado uma entidade.

Identificou-se, primeiramente, com base no requisito **RD7**, a entidade 'Utilizador', que representa a superclasse de todos os indivíduos que interagem com o sistema.

Desta entidade genérica deriva o 'Sócio', uma entidade que herda as características do 'Utilizador', mas distingue-se pelos seus atributos exclusivos, tal como o número de sócio, como enuncia o requisito **RD3**: "Um sócio é identificado pelo seu número de sócio do CeSIUM…".

Posteriormente, é identificada, em **RD1**, a entidade designada 'Viagem', elemento central para o funcionamento da base de dados, dado que representa a principal atividade publicada pelos sócios e visualizada pelos utilizadores.

De seguida, o requisito **RD5** levou à identificação de uma nova entidade, 'Paragem', essencial para a representação detalhada do itinerário das viagens. A 'Paragem' assume a natureza de entidade fraca, o que significa que não possui existência autónoma, dependendo da entidade 'Viagem' para ser identificada.

Por último, a entidade designada 'Patrocinador', expressa no requisito **RD9**, é responsável pela representação das organizações ou instituições que apoiam financeiramente determinadas viagens, sendo, deste modo, uma entidade de existência independente.

Segue uma tabela com o resumo das informações apresentadas acima:

Entidade	Definição	Requisito
Utilizador	Entidade genérica que representa todos os indivíduos que interagem com o sistema.	RD7
Sócio	Entidade derivada de 'Utilizador'. Corresponde a um sócio da organização.	RD3
Viagem	Atividade principal do sistema, correspondente às viagens publicadas pelos sócios e visualizadas pelos utilizadores.	RD1
Paragem	Ponto incluído no itinerário de uma viagem. Depende de uma viagem para existir e ser identificado.	RD5
Patrocinador	Organização ou instituição que apoia financeiramente determinadas viagens, possuindo existência independente.	RD9

Tabela 5 - Entidades identificadas pela equipa de desenvolvimento.

3.3. Identificação e Caracterização dos Relacionamentos

Após a identificação das entidades da base de dados, procedeu-se à determinação dos relacionamentos que estabelecem as interligações lógicas entre estas, de modo a representar corretamente o comportamento e as dependências existentes no domínio do problema. Este processo de análise baseou-se na leitura detalhada dos requisitos funcionais e descritivos, dos quais foi possível extrair, de forma direta ou indireta, as associações entre entidades, em relação às respetivas cardinalidades e participações.

Em primeiro lugar, o relacionamento 'Possui' foi identificado com base no requisito **RD4**, onde se enuncia que "Uma viagem possui pelo menos uma paragem. Cada paragem pertence obrigatoriamente a uma única viagem". Este requisito evidencia a associação entre as entidades 'Viagem' e 'Paragem', uma vez que uma viagem é constituída por várias paragens, cada uma representando um ponto de passagem. Adicionalmente, entende-se que uma 'Paragem' está associada a uma única 'Viagem', enquanto cada 'Viagem' pode estar associada a um conjunto de paragens, o que define uma cardinalidade um para n.

O relacionamento 'Patrocinada' resulta do requisito **RD8**, o qual estabelece que "Um patrocinador patrocina uma ou mais viagens. Cada viagem pode ou não ser patrocinada por um ou mais patrocinadores". Deste modo, torna-se evidente a existência de uma associação entre 'Viagem' e 'Patrocinador'. O relacionamento é, portanto, de natureza muitos para muitos, visto que um patrocinador pode patrocinar várias viagens e, simultaneamente, uma viagem pode receber o apoio de diversos patrocinadores. Além disso, nem todas as viagens são patrocinadas, assim como nem todos os patrocinadores são obrigados a apoiar.

Seguidamente, o requisito **RD2**, que refere que "Um sócio pode publicar viagens. Cada viagem pode ou não ser publicada por um ou mais sócios", levou à identificação do relacionamento 'Publica' entre as entidades 'Sócio' e 'Viagem'. Este relacionamento reflete o ato de um sócio tornar pública uma determinada viagem na plataforma, associando-lhe informações descritivas. Cada viagem pode ser publicada por vários sócios, ao passo que um sócio pode publicar várias viagens, o que determina uma cardinalidade de muitos para muitos. Uma vez que uma viagem publicada deve ter, no mínimo, um sócio associado, conclui-se que a participação da entidade 'Viagem' é obrigatória, enquanto a participação do 'Sócio' é facultativa, considerando que podem existir sócios registados que ainda não publicaram qualquer viagem.

O relacionamento 'Visualiza' foi identificado a partir do requisito **RD12**, o qual estabelece que "Um utilizador pode visualizar zero ou mais viagens. Cada viagem pode ser visualizada por zero ou mais utilizadores…". Este requisito permite relacionar as entidades 'Utilizador' e 'Viagem', representando a interação dos utilizadores com os conteúdos partilhados. Assim, verifica-se que uma 'Viagem' pode ser

visualizada por múltiplos 'Utilizador' e que um 'Utilizador' pode visualizar diversas 'Viagem', o que define uma cardinalidade de muitos para muitos. No que diz respeito à participação, ambos os lados apresentam participação parcial, uma vez que podem existir utilizadores que ainda não visualizaram nenhuma viagem, bem como viagens que ainda não tenham sido visualizadas por ninguém.

Por último, o requisito **RD6** indica que "Cada sócio corresponde a exatamente um utilizador. Um utilizador pode ou não ser sócio". Este requisito fundamenta a existência do relacionamento entre 'Sócio' e 'Utilizador', o qual estabelece a equivalência entre membro físico do CeSIUM e o seu registo digital na base de dados. Assim, cada 'Sócio' está associado obrigatoriamente a um único 'Utilizador', enquanto um 'Utilizador' pode ou não estar associado a um 'Sócio'.

Segue uma tabela com o resumo das informações apresentadas acima:

Entidade	Relacionamento	Cardinalidade	Participação	Entidade	Requisito
Viagem	Possui	1 : N	T : T	Paragem	RD4
Viagem	Patrocinada	N : M	P : P	Patrocinador	RD8
Sócio	Publica	N : M	P : T	Viagem	RD2
Utilizador	Visualiza	N : M	P : P	Viagem	RD12
Sócio	-	1 : 1	T : P	Utilizador	RD6

Tabela 6 - Relacionamentos identificados pela equipa de desenvolvimento.

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

Seguindo a identificação das entidades e dos respetivos relacionamentos, torna-se necessário avançar para a definição e caracterização dos atributos. Esta etapa visa detalhar a informação que cada entidade e relacionamento deve armazenar, assegurando que todos os dados relevantes ao domínio do problema são corretamente representados e estruturados.

Primeiramente, com base no requisito **RD3**, obtêm-se diretamente os atributos da entidade 'Sócio', nomeadamente 'Nr_Sócio', 'Estatuto' e 'Foto_Perfil'. O atributo 'Nr_Sócio' é obrigatório e de natureza simples. O atributo 'Estatuto' utiliza um tipo ENUM, descrevendo a categoria a que o sócio pertence. Por fim, o atributo 'Foto_Perfil' corresponde ao caminho para o ficheiro de imagem associado ao sócio, sendo opcional.

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
Nr_Sócio	Simples	INT	Não	Número de sócio no CeSIUM	2468
Estatuto	Simples	ENUM (...)	Sim	Define o estatuto do sócio no CeSIUM	Alumni
Foto_Perfil	Simples	VARCHAR (100)	Sim	Caminho para a fotografia de perfil do sócio	/imagens/FP1854.png

Tabela 7 - Atributos da entidade 'Sócio'.

De seguida, o requisito **RD7** apresenta os atributos do 'Utilizador'. O atributo 'ID' funciona como chave identificadora única, sendo obrigatório e de natureza simples. Os atributos 'Nome', 'Telemóvel', 'Email' e 'Pass_Word' são igualmente simples e obrigatórios, contendo dados de identificação e comunicação essenciais ao funcionamento do sistema.

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
ID	Simples	INT	Não	Identificador único do utilizador	2
Nome	Simples	VARCHAR (100)	Não	Nome e apelido do utilizador	Rui Almeida
Telemóvel	Simples	VARCHAR (13)	Não	Número de telemóvel do utilizador	+351911111111
Email	Simples	VARCHAR (100)	Não	Email do utilizador	rui@email.com
Pass_Word	Simples	VARCHAR (100)	Não	Password do utilizador	rui123

Tabela 8 - Atributos da entidade 'Utilizador'.

A entidade 'Patrocinador' reúne os dados necessários para registar as entidades externas que financiam as deslocações. O atributo 'ID' atua como identificador único, sendo um valor inteiro e obrigatório. Já o atributo 'Nome' é um campo de texto simples e igualmente obrigatório, destinado a guardar o nome do patrocinador (**RD9**).

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
ID	Simples	INT	Não	Identificador único do patrocinador	1
Nome	Simples	VARCHAR (40)	Não	Nome do patrocinador	Yari Labs

Tabela 9 - Atributos da entidade 'Patrocinador'.

Os atributos definidos para a entidade 'Viagem', conforme o requisito **RD1**, descrevem detalhadamente a informação necessária para representar cada deslocação. O atributo 'ID' garante a distinção inequívoca entre viagens. Os atributos 'Data_Início' e 'Data_Término' registam, respetivamente, a data de início e fim da viagem. O atributo 'Objetivo' descreve a finalidade da viagem e o 'Custo' representa o valor total, armazenado em formato decimal. Os atributos 'Nr_Participantes' e 'Nr_Paragens' indicam o número de participantes e de paragens, sendo 'Nr_Paragens' derivado das paragens associadas à viagem. Por fim, o 'Tipo_Deslocamento' é um atributo multivalor que identifica os diversos meios de transporte utilizados.

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
ID	Simples	INT	Não	Identificador único da viagem	10
Data_Início	Simples	DATETIME	Não	Data em que a viagem teve início	2023-10-22 19:15:00
Data_Término	Simples	DATETIME	Não	Data em que a viagem terminou	2023-10-25 19:17:00
Objetivo	Simples	ENUM (...)	Não	Objetivo da viagem	Pedagógico
Custo	Simples	DECIMAL (8,2)	Não	Custo da viagem	23.60
Nr_Participantes	Simples	INT	Não	Número de participantes da viagem	6
Nr_Paragens	Simples	INT	Não	Número de paragens da viagem	3
Tipo_Deslocamento	Multivalor	ENUM (...)	Não	Tipos de deslocamento	Carro, Avião

Tabela 10 - Atributos da entidade 'Viagem'.

A entidade 'Paragem', conforme o requisito **RD5**, inclui os atributos 'Nr' (identificador), 'Localização' (cidade e país) e 'Foto', um atributo multivalor com os caminhos das imagens associadas. Inclui também 'Data_Chegada' e 'Data_Partida', correspondentes à chegada e partida dos sócios.

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
Nr	Simples	INT	Não	Identificador da paragem	45
Localização	Composto	-	-	Local onde se situa a paragem	-
País	Simples	VARCHAR (50)	Não	País onde se situa a paragem	Áustria
Cidade	Simples	VARCHAR (50)	Não	Cidade onde se situa a paragem	Viena
Foto	Multivalor	VARCHAR (100)	Sim	Caminho para as fotos associadas à paragem	/Fotos/Viagem1_Vigo_01.png
Data_Chegada	Simples	DATETIME	Não	Data de chegada à paragem	2023-10-23 19:19:00
Data_Partida	Simples	DATETIME	Não	Data de partida da paragem	2023-10-24 19:21:00

Tabela 11 - Atributos da entidade 'Paragem'.

Relativamente ao relacionamento 'Patrocinada', entre 'Viagem' e 'Patrocinador', destacam-se os atributos do grupo 'Detalhes'. O atributo 'Valor' indica o montante financeiro do apoio, enquanto 'Motivo', sendo multivalor, permite registar uma ou várias razões para cada patrocínio, garantindo que toda a justificação do apoio fica documentada (**RD10**).

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
Detalhes	Composto	-	-	Detalhes do patrocínio	-
Valor	Simples	DECIMAL (8,2)	Não	Investimento do patrocinador na viagem	1200.05
Motivo	Multivalor	VARCHAR (200)	Não	Motivos para o patrocínio da viagem	Apoiar um projeto de investigação

Tabela 12 - Atributos do relacionamento 'Patrocinada'.

O relacionamento 'Publica', do requisito **RD11**, associa um 'Sócio' a uma 'Viagem' e permite registar o feedback da experiência através do atributo composto 'Descrição', composto por 'Avaliação' e 'Comentário' (opcional).

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
Descrição	Composto	-	-	Descrição da viagem	-
Avaliação	Simples	ENUM (...)	Não	Avaliação do sócio à viagem	Muito positiva
Comentário	Simples	VARCHAR (150)	Sim	Comentário do sócio à viagem	Expandiu os meus horizontes

Tabela 13 - Atributos do relacionamento 'Publica'.

Finalmente, o relacionamento 'Visualiza', que associa 'Utilizador' a 'Viagem', aloja o atributo simples 'Reação' (**RD12**). Esta modelação justifica-se pelo contexto da interação: um utilizador pode reagir de forma diferente a várias viagens, e uma mesma viagem pode gerar reações distintas em diferentes utilizadores.

Atributo	Tipo	Domínio	Nulo	Descrição	Exemplo
Reação	Simples	ENUM (...)	Não	Opinião do utilizador em relação à viagem	Negativa

Tabela 14 - Atributos do relacionamento 'Visualiza'.

3.5. Apresentação e Explicação do Diagrama ER Produzido

O processo de construção do diagrama foi dividido em cinco etapas:

1. A equipa de desenvolvimento iniciou o trabalho com a identificação e inserção de todas as entidades no programa utilizado para a criação do diagrama.
2. Foram definidos e incluídos todos os relacionamentos entre as entidades, bem como as respetivas cardinalidades e participações.
3. Definiu-se os atributos de cada entidade, estabelecendo os respetivos domínios, identificando os atributos identificadores e distinguindo os que são opcionais.
4. Procedeu-se à releitura de todos os requisitos, garantindo que o modelo permitia o cumprimento de cada um deles.
5. Ajustou-se a disposição dos elementos gráficos do diagrama para tornar a representação mais clara e legível.

O diagrama ER final, elaborado após uma revisão solicitada pela equipa de supervisores, é apresentado de seguida.

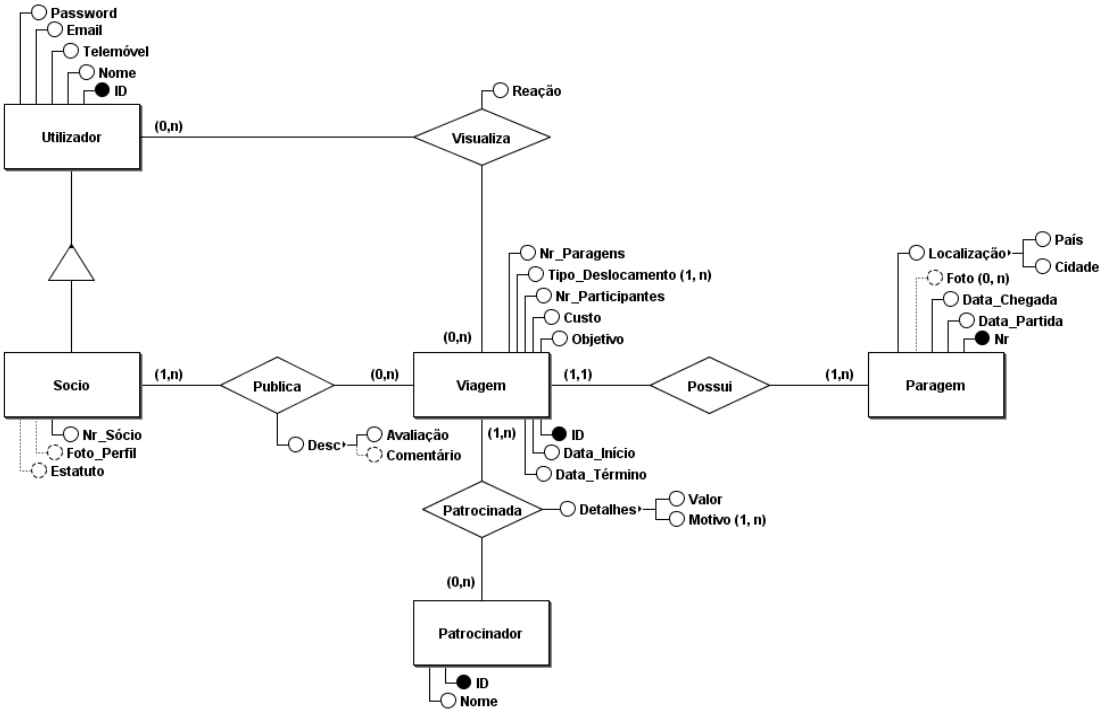


Figura 6 - Versão final do diagrama ER produzido pela equipa de desenvolvedores.

4. Modelação Lógica

4.1. Construção e Validação do Modelo de Dados Lógico

Após a definição do modelo conceptual, iniciou-se a construção do modelo lógico da base de dados do sistema BeLIUM Viagens, com o objetivo de preparar a estrutura de informação para a futura implementação no Sistema de Gestão de Base de Dados, MySQL. No contexto da modelação lógica, manteve-se a abordagem monovista, ou seja, o modelo foi desenvolvido na sua totalidade de uma só vez, utilizando o *MySQL Workbench*.

4.2. Apresentação e Explicação do Modelo Lógico Produzido

Para iniciar a construção do esquema lógico, a equipa procedeu à criação de uma tabela correspondente a cada entidade identificada no modelo conceptual. Na definição da estrutura de cada tabela, foram considerados não apenas os atributos simples das entidades, mas também os atributos compostos, já devidamente decompostos em atributos elementares.

Num segundo momento, foram tratados os atributos multivalor. Cada atributo deste tipo foi convertido numa tabela autónoma, garantindo uma representação adequada no modelo relacional. Estas tabelas incluem uma chave estrangeira que referencia a entidade à qual o atributo multivalor estava originalmente associado, assegurando a ligação entre ambas as estruturas. Considerando que estes atributos são totalmente dependentes da tabela de origem, a chave estrangeira faz parte da chave primária da tabela adicional.

De seguida, foi efetuado o mapeamento dos relacionamentos, os quais, neste caso, eram exclusivamente binários. Dependendo das cardinalidades associadas a cada relacionamento, foram adotados diferentes procedimentos de representação. Nos relacionamentos do tipo muitos-para-muitos, foi criada uma tabela intermédia contendo duas chaves estrangeiras, cada uma referenciando as entidades participantes. Em conjunto, estas chaves estrangeiras constituem a chave primária da tabela intermediária, considerando, novamente, que esta é totalmente dependente das duas entidades envolvidas. Já no relacionamento do tipo um-para-muitos, o mapeamento traduziu-se na introdução de uma chave estrangeira na tabela correspondente à entidade do lado múltiplo, assegurando a ligação ao

registo da entidade do lado um. Adicionalmente, o relacionamento de especialização, configura-se num relacionamento do tipo um-para-um entre a entidade genérica e a sua entidade especializada. Neste caso, optou-se por manter tabelas distintas, em que a chave primária da entidade especializada coincide com a chave primária da entidade genérica, funcionando simultaneamente como chave estrangeira, garantindo assim a integridade e a correspondência direta entre ambas.

Destes procedimentos resultou o esquema seguinte:

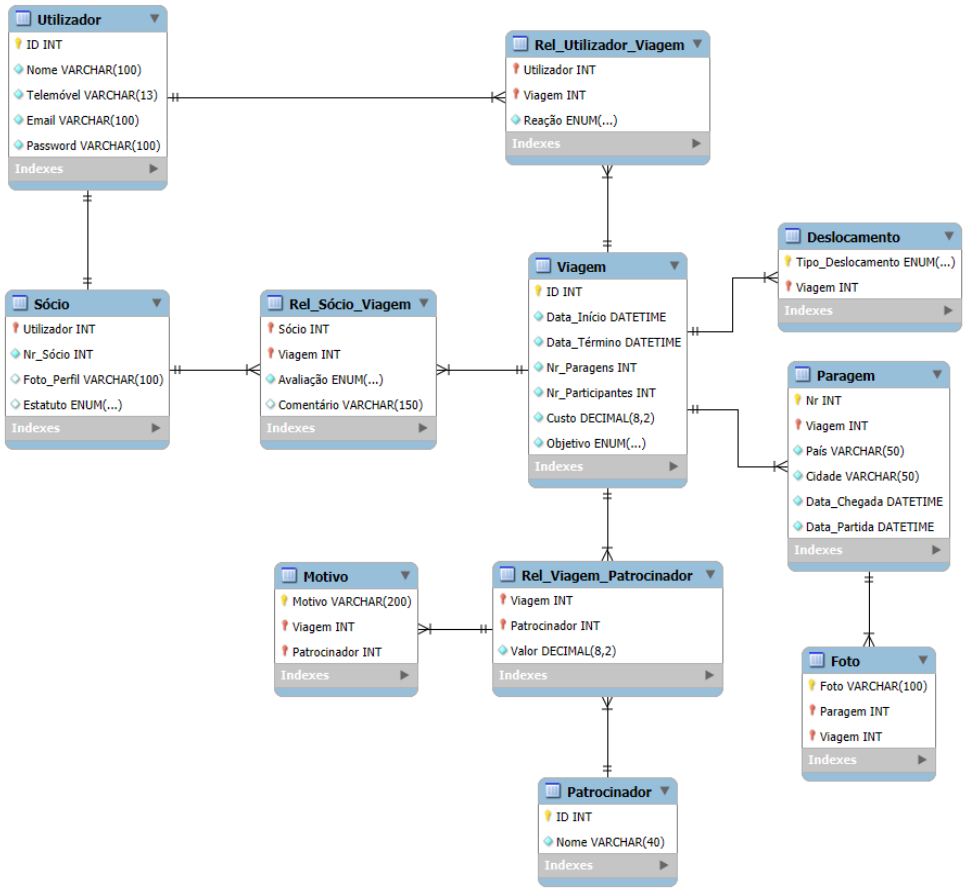


Figura 7 - Versão final do esquema lógico realizado no MySQL.

A seguir apresenta-se o dicionário de dados, acompanhado pela descrição do processo adotado para a definição de cada tabela.

A tabela 'Utilizador' foi criada a partir da entidade com o mesmo nome. Esta tabela é composta pelas colunas 'ID' (definido como chave primária), 'Nome', 'Telemóvel' e 'Email', as quais foram diretamente transpostas do modelo conceptual. Todas foram definidas como não nulas, garantindo que cada registo possua informação completa relativamente à identificação e contacto do utilizador.

Atributo	Domínio	Nulo	Chave Estrangeira
ID	INT	Não	Não
Nome	VARCHAR (100)	Não	Não
Telemóvel	VARCHAR (13)	Não	Não
Email	VARCHAR (100)	Não	Não
Password	VARCHAR (100)	Não	Não

Tabela 15 - Dicionário de dados da relação 'Utilizador'.

A tabela 'Sócio' resulta da especialização da entidade 'Utilizador'. Por essa razão, utiliza a chave estrangeira 'Utilizador' (que referencia o utilizador correspondente) como chave primária. Também inclui as colunas 'Nr_Sócio' e 'Estatuto', implementando este último como um campo do tipo ENUM com as opções permitidas: 'Direção' e 'Alumni'. Para além destes, integra ainda a coluna opcional 'Foto_Perfil', representada no modelo como um campo que pode assumir o valor NULL.

Atributo	Domínio	Nulo	Chave Estrangeira
Utilizador	INT	Não	Sim
Nr_Sócio	INT	Não	Não
Foto_Perfil	VARCHAR (100)	Sim	Não
Estatuto	ENUM (…)	Sim	Não

Tabela 16 - Dicionário de dados da relação 'Sócio'.

A tabela 'Patrocinador' foi derivada da entidade 'Patrocinador' e contém apenas duas colunas: 'ID', que identifica unicamente cada patrocinador, e 'Nome', que armazena a designação do mesmo.

Atributo	Domínio	Nulo	Chave Estrangeira
ID	INT	Não	Não
Nome	VARCHAR (40)	Não	Não

Tabela 17 - Dicionário de dados da relação 'Patrocinador'.

A tabela 'Viagem' foi obtida a partir da entidade 'Viagem'. É constituída pelas colunas 'ID' (chave primária), 'Data_Início', 'Data_Término', 'Objetivo', 'Custo', 'Nr_Participantes' e 'Nr_Paragens'. A coluna 'Objetivo' é do tipo ENUM, podendo possuir um dos seguintes valores: 'Pedagógico', 'Recreativo' e 'Ambos'.

Atributo	Domínio	Nulo	Chave Estrangeira
ID	INT	Não	Não
Data_Início	DATETIME	Não	Não
Data_Término	DATETIME	Não	Não
Nr_Paragens	INT	Não	Não
Nr_Participantes	INT	Não	Não
Objetivo	ENUM (…)	Não	Não
Custo	DECIMAL (8,2)	Não	Não

Tabela 18 - Dicionário de dados da relação 'Viagem'.

A tabela 'Paragem' foi derivada da entidade com o mesmo nome, que possui os atributos simples 'Nr', 'País', 'Cidade', 'Data_Partida' e 'Data_Chegada', todos convertidos da mesma forma em colunas. Além destes, inclui-se a coluna 'Viagem', proveniente do relacionamento 'Possui', de cardinalidade um-para-muitos, que estabelece a associação entre cada paragem e a viagem a que pertence. Por tratar-se de uma chave estrangeira, o atributo 'Viagem' deve ter o mesmo domínio que a chave primária da tabela 'Viagem', ou seja, INT. Adicionalmente, uma vez que cada paragem está inteiramente dependente da viagem associada, sendo, por isso, uma entidade fraca, o atributo 'Viagem', juntamente com o 'Nr', definem uma chave primária composta.

Atributo	Domínio	Nulo	Chave Estrangeira
Nr	INT	Não	Não
Viagem	INT	Não	Sim
País	VARCHAR (50)	Não	Não
Cidade	VARCHAR (50)	Não	Não
Data_Partida	DATETIME	Não	Não
Data_Chegada	DATETIME	Não	Não

Tabela 19 - Dicionário de dados da relação 'Paragem'.

Para representar o relacionamento 'Visualiza' criou-se a tabela 'Rel_Utilizador_Viagem', que contém duas chaves estrangeiras, 'Utilizador' e 'Viagem', que formam conjuntamente a chave primária da tabela. Esta última é o que permite ao utilizador visualizar várias viagens, e vice-versa. Além disso, a tabela inclui a coluna 'Reação', do tipo ENUM, que pode assumir o valor 'Positiva' ou 'Negativa'.

Atributo	Domínio	Nulo	Chave Estrangeira
Utilizador	INT	Não	Sim
Viagem	INT	Não	Sim
Reação	ENUM (…)	Não	Não

Tabela 20 - Dicionário de dados da relação 'Rel_Utilizador_Viagem'.

De forma semelhante, o relacionamento 'Publica' deu origem à tabela 'Rel_Sócio_Viagem', que também possui duas chaves estrangeiras, 'Sócio' e 'Viagem', derivadas das tabelas 'Sócio' e 'Viagem', respetivamente, e que constituem a chave primária composta da tabela. Esta apresenta ainda colunas adicionais provenientes do atributo composto 'Descrição': 'Avaliação', do tipo ENUM (de valor 'Muito Negativa', 'Negativa', 'Mediana', 'Positiva' ou 'Muito Positiva'), e 'Comentário', que é opcional.

Atributo	Domínio	Nulo	Chave Estrangeira
Sócio	INT	Não	Sim
Viagem	INT	Não	Sim
Avaliação	ENUM (…)	Não	Não
Comentário	VARCHAR (150)	Sim	Não

Tabela 21 - Dicionário de dados da relação 'Rel_Sócio_Viagem'.

Seguidamente, a tabela proveniente do relacionamento 'Patrocinada', 'Rel_Viagem_Patrocinador', é formada por duas chaves estrangeiras: 'Patrocinador' (da tabela 'Patrocinador') e 'Viagem' (da tabela 'Viagem'), que constituem conjuntamente a chave primária da relação. Além destas, a tabela inclui a coluna 'Valor'.

Atributo	Domínio	Nulo	Chave Estrangeira
Patrocinador	INT	Não	Sim
Viagem	INT	Não	Sim
Valor	DECIMAL (8,2)	Não	Não

Tabela 22 – Dicionário de dados da relação 'Rel_Viagem_Patrocinador'.

A tabela 'Deslocamento' deriva do atributo multivalor 'Tipo_Deslocamento' associado à entidade 'Viagem', permitindo registar os diferentes meios de transporte utilizados em cada deslocação. É constituída pelas colunas 'Tipo_Deslocamento', que identifica o meio de transporte (pode assumir os valores: 'Carro', 'Autocarro', 'Comboio', 'Avião', 'Barco', 'Navio', 'Bicicleta', 'Mota', 'A pé' e 'Outro'), e 'Viagem', definido como chave estrangeira para a tabela 'Viagem'. A chave primária da tabela é composta por ambas as colunas, garantindo que o mesmo tipo de deslocamento não é registado duas vezes para a mesma viagem.

Atributo	Domínio	Nulo	Chave Estrangeira
Tipo_Deslocamento	ENUM (…)	Não	Não
Viagem	INT	Não	Sim

Tabela 23 - Dicionário de dados da relação 'Deslocamento'.

O atributo multivalor 'Foto' da entidade 'Paragem' foi transformado na tabela 'Foto'. Esta inclui duas colunas: 'Foto' e as chaves estrangeiras obrigatórias 'Paragem' e 'Viagem', que referenciam a tabela 'Paragem'. A chave primária é composta por ambas as três.

Atributo	Domínio	Nulo	Chave Estrangeira
Foto	VARCHAR (100)	Não	Não
Paragem	INT	Não	Sim
Viagem	INT	Não	Sim

Tabela 24 - Dicionário de dados da relação 'Foto'.

O atributo multivalor 'Motivo', presente no atributo composto 'Detalhes' do relacionamento 'Patrocinada', deu origem à tabela 'Motivo'. Esta tabela contém a chave estrangeira composta ('Patrocinador', 'Viagem'), que referencia a tabela 'Patrocinada', e a coluna 'Motivo'. Em conjunto, formam a chave primária da tabela. Cada registo representa um motivo específico associado a um determinado patrocinador e viagem.

Atributo	Domínio	Nulo	Chave Estrangeira
Motivo	VARCHAR (200)	Não	Não
Patrocinador	INT	Não	Sim
Viagem	INT	Não	Sim

Tabela 25 - Dicionário de dados da relação 'Motivo'.

4.3. Normalização de Dados

A normalização é uma técnica orientada para a organização dos dados numa base de dados relacional, visando minimizar a redundância e eliminar eventuais anomalias de inserção, atualização e eliminação. No âmbito do projeto BeLIUM Viagens, procedeu-se à verificação das dependências funcionais e da atomicidade das colunas das tabelas resultantes do modelo lógico para garantir o cumprimento das três primeiras Formas Normais (FN).

Todas as relações cumprem a Primeira Forma Normal (1FN), apresentando chave primária definida, valores atómicos em todos os atributos e ausência de grupos de dados repetidos. Durante a passagem do modelo conceptual para o lógico, todos os atributos compostos foram decompostos nos seus componentes simples e os atributos multivalorados identificados foram devidamente normalizados. Um exemplo é o atributo 'Tipo_Deslocamento' da entidade 'Viagem', que deu origem à criação da tabela 'Deslocamento'. Na estrutura resultante, cada registo na tabela 'Deslocamento' representa um único meio de transporte associado a uma viagem específica.

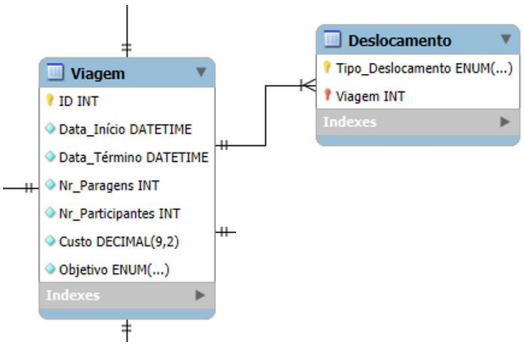


Figura 8 - Relação 'Deslocamento'.

O mesmo princípio foi aplicado às fotos de cada paragem e aos motivos de cada patrocínio, resultando na criação das tabelas 'Foto' e 'Motivo', respetivamente.

Uma tabela encontra-se na Segunda Forma Normal (2FN) quando já cumpre a 1FN e todos os atributos não-primos (isto é, que não pertencem à chave) dependem totalmente da chave primária, não existindo dependências parciais. As tabelas dos atributos multivalorados explicadas acima, por conterem apenas atributos primos e já se encontrarem na 1FN, satisfazem automaticamente esta condição. Nas tabelas com chave primária simples, esta condição verifica-se igualmente de forma automática.

Procurou-se garantir a normalização das restantes tabelas, começando por 'Rel_Sócio_Viagem': o diagrama de dependências funcionais gerado apresenta apenas as seguintes dependências: {Sócio, Viagem} → Avaliação, Comentário. Isto significa que a avaliação e o comentário dependem simultaneamente do sócio e da viagem, já que cada registo da tabela corresponde à descrição que um determinado sócio deu a uma determinada viagem. Assim: 'Sócio' isoladamente não identifica uma descrição, porque um mesmo sócio pode descrever várias viagens. 'Viagem' isoladamente também não identifica uma descrição, porque vários sócios podem descrever a mesma viagem. Apenas a combinação {Sócio, Viagem} identifica univocamente uma descrição. Deste modo, a 'Avaliação' e o 'Comentário' são totalmente dependentes da chave composta, e não existe qualquer dependência parcial (ou seja, não há atributos que dependam apenas de uma parte da chave). Consequentemente, a relação encontra-se em conformidade com a 2FN.

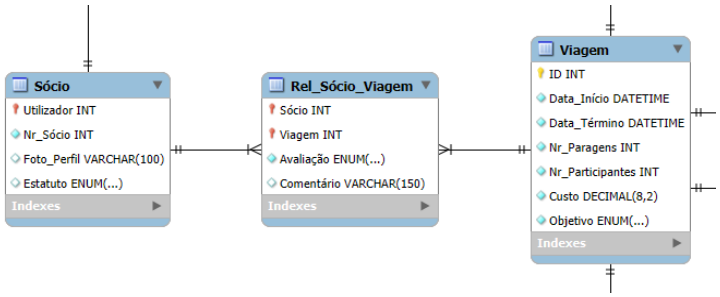


Figura 9 - Relação 'Rel_Sócio_Viagem'.

Aplicou-se o mesmo raciocínio às restantes tabelas resultantes de relacionamentos muitos-para-muitos, que apresentam igualmente dependências funcionais totalmente determinadas pelas respetivas chaves compostas.

Há ainda um caso particular: a tabela 'Paragem'. Como a entidade que lhe deu origem é uma entidade fraca, a identificação de cada paragem não é autónoma, isto é, necessita da chave estrangeira 'Viagem' para que o identificador fique completo. Consequentemente, os atributos de 'Paragem' não dependem apenas da própria paragem, mas da combinação Paragem + Viagem. Uma vez que todos os atributos dependem da chave composta na sua totalidade, e não existe qualquer dependência parcial, a relação encontra-se em conformidade com a 2FN.

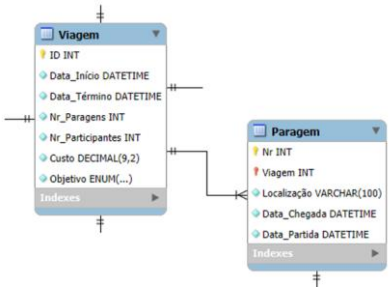


Figura 10 - Relação 'Paragem'.

Uma tabela está na Terceira Forma Normal (3FN) se, para além de estar na 2FN, todos os atributos não-chave forem mutuamente independentes, não existindo dependências funcionais transitivas. Após a análise de todas as tabelas do modelo, concluiu-se que esta condição é satisfeita em todo o esquema relacional, como é possível visualizar através da seguinte tabela:

Relação	Dependências funcionais
Utilizador	{ID} → Nome, Telemóvel, Email, Password
Sócio	{Utilizador} → Nr_Sócio, Foto_Perfil, Estatuto
Viagem	{ID} → Data_Início, Data_Término, Nr_Paragens, Nr_Participantes, Custo, Objetivo
Patrocinador	{ID} → Nome
Paragem	{Nr, Viagem} → Cidade, País, Data_Partida, Data_Chegada
Deslocamento	{Tipo_Deslocamento, Viagem}
Foto	{Foto, Paragem, Viagem}
Motivo	{Motivo, Viagem, Patrocinador}
Rel_Sócio_Viagem	{Sócio, Viagem} → Avaliação, Comentário
Rel_Utilizador_Viagem	{Utilizador, Viagem} → Reação
Rel_Viagem_Patrocinador	{Viagem, Patrocinador} → Valor

Tabela 26 - Dependências funcionais entre os diversos elementos de dados de cada relação.

Importa esclarecer que não existe uma dependência transitiva na relação 'Paragem', pois tanto a 'Cidade' como o 'País' dependem diretamente da chave primária, não havendo uma relação de dependência entre os dois atributos de localização. Embora, no domínio do problema, uma cidade pertença sempre a um país específico, essa dependência não foi considerada para efeitos de normalização. Esta decisão visa melhorar a eficiência na execução de *queries*, em particular porque filtrar viagens pela sua localização será uma operação frequente.

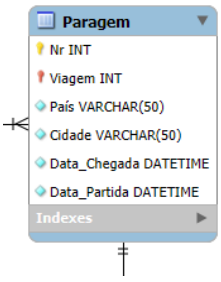


Figura 11 - Relação 'Paragem'.

Deste modo, todas as tabelas do modelo encontram-se normalizadas até à 3FN, garantindo consistência, ausência de redundâncias não controladas e minimizando a ocorrência de anomalias de inserção, atualização ou eliminação.

4.4. Validação do Modelo com Interrogações do Utilizador

A validação do modelo lógico é uma etapa crítica do desenvolvimento de bases de dados, confrontando o esquema relacional com os requisitos operacionais. Esta fase garante robustez, integridade e eficiência do sistema, assegurando que os dados são armazenados de forma normalizada e que consultas complexas são suportadas. Para tal, foram selecionados os Requisitos de Manipulação (RM) identificados durante as reuniões com o Corpo Colaborativo do Projeto (CCP) e através dos inquéritos à comunidade, e procedeu-se à formalização das operações em Álgebra Relacional.

A estratégia de validação seguiu uma abordagem dual: primeiramente, a formalização da lógica de acesso aos dados através da Álgebra Relacional e, seguidamente, a sua verificação prática utilizando a calculadora de álgebra relacional RelaX (Relational Algebra Calculator). Como o RelaX não aceita certos caracteres (como @, acentos e símbolos especiais), os registos tiveram de ser adaptados para os testes. Mesmo com esta limitação, a ferramenta permitiu verificar a correção das operações e a eficácia das estruturas normalizadas, executando as expressões sobre uma instância de dados de teste preparada para o efeito (ver Relações RelaX).

No âmbito da verificação das operações de consulta, constatou-se que algumas interrogações requerem a passagem explícita de argumentos. Por exemplo, para obter a lista de paragens associadas a uma determinada viagem, é necessário identificar a viagem através do seu identificador ('Paragem.Viagem'). Assim, a abordagem adotada pelo grupo consistiu em representar, nas expressões das interrogações, os elementos parametrizáveis através de um ponto de interrogação, sendo que a consulta final é gerada substituindo esse marcador pelo valor concreto fornecido no momento da execução.

O requisito **RM3** ("Listar, por ordem, todas as paragens de uma dada viagem") recorre à tabela 'Paragem', onde cada registo inclui uma referência à viagem correspondente. Para satisfazer este requisito, selecionam-se todas as linhas cuja coluna 'Viagem' coincide com o identificador fornecido como argumento. Após a filtragem, os registos são ordenados pela data de chegada, garantindo que a lista reflete a sequência real das paragens na viagem especificada.

$$\tau_{\texttt{Data\_Chegada ASC}} \left(\pi_{\texttt{Nr, País, Cidade, Data\_Chegada, Data\_Partida}} \left(\sigma_{\texttt{Viagem = ?}} \texttt{Paragem}\right)\right)$$

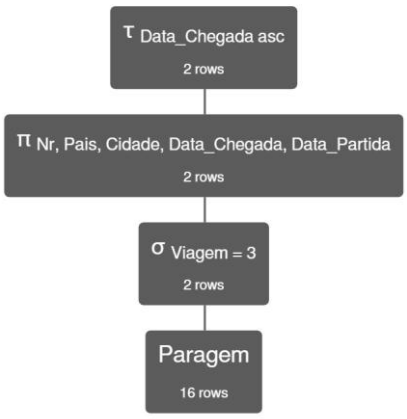


Figura 12 - Árvore da interrogação do RM3 para a 'Paragem' com Viagem = 3

O requisito **RM4** consiste em "Listar todos os sócios (número de sócio, nome e estatuto) que realizaram uma determinada viagem (por ordem alfabética)". Para isso, a informação necessária encontra-se distribuída por três tabelas: 'Utilizador', que contém os nomes necessários para a interrogação; 'Sócio', que liga cada utilizador ao respetivo número de sócio; e 'Rel_Sócio_Viagem', que regista a participação de cada sócio nas viagens. A equipa começou por realizar a junção entre as tabelas 'Sócio' e 'Rel_Sócio_Viagem', estabelecendo a relação através do número de sócio. Seguidamente, efetuou-se uma segunda junção com a tabela 'Utilizador', utilizando o identificador do utilizador como chave de ligação. Após estas junções, aplicou-se o filtro para selecionar apenas os registos correspondentes à viagem especificada. Finalmente, procedeu-se à projeção dos atributos e à ordenação alfabética dos resultados pelo nome.

$$\tau_{\texttt{Nome ASC}}\left(\pi_{\texttt{ID, Nome}}\left(\sigma_{\texttt{Viagem = ?}}\left(\left(\texttt{Sócio} \bowtie_{\texttt{Rel\_Sócio\_Viagem.Sócio = Sócio.Utilizador}} \texttt{Rel\_Sócio\_Viagem}\right) \bowtie_{\texttt{Utilizador.ID = Sócio.Utilizdor}} \texttt{Utilizador}\right)\right)\right)$$

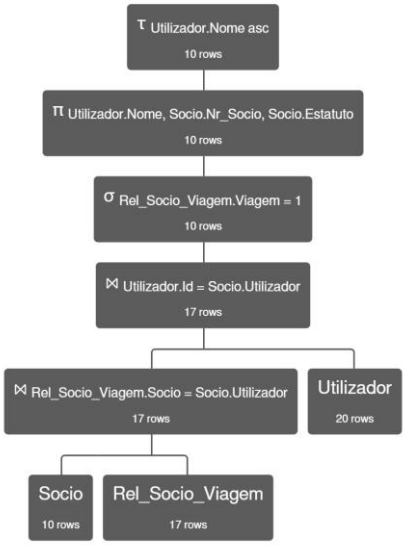


Figura 13 - Árvore da interrogação do RM4 para a 'Rel_Sócio_Viagem' com Viagem = 1.

De seguida, foi implementado o requisito **RM6**, que consiste em "Consultar os patrocínios (nome do patrocinador e valor) de uma dada viagem, ordenados pelo valor e pelo nome". Neste caso, a informação necessária encontra-se distribuída por duas tabelas distintas: a relação intermédia 'Rel_Viagem_Patrocinador', que associa cada viagem aos respetivos patrocinadores, e a tabela 'Patrocinador', onde estão armazenados os detalhes de cada entidade patrocinadora. Assim, a estratégia definida foi realizar uma equi-junção entre estas duas tabelas, utilizando como condição a correspondência entre o identificador do patrocinador na relação intermédia e o atributo 'ID' da tabela 'Patrocinador'. Depois da junção, aplica-se uma seleção sobre a viagem pretendida, em seguida, uma projeção sobre os atributos necessários – ID, Nome e Valor – e por fim, ordena-se pelo Valor e Nome.

$$\tau_{\texttt{Valor, Nome}} \left(\pi_{\texttt{ID, Nome, Valor}} \left(\sigma_{\texttt{Viagem = ?}} \left(\texttt{Patrocinador} \bowtie_{\texttt{ID = Patrocinador}} \texttt{Rel\_Viagem\_Patrocinador}\right)\right)\right)$$

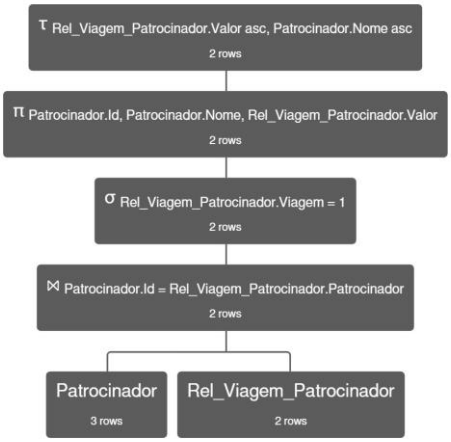


Figura 14 - Árvore da interrogação do RM6 para a 'Rel_Viagem_Patrocinador' com Viagem = 1.

O requisito **RM8** consiste em "Filtrar viagens pela localização das paragens". A informação necessária encontra-se distribuída entre as relações 'Viagem' e 'Paragem'. Para identificar as viagens que possuem paragens na cidade e no país pretendido, realiza-se uma equi-junção entre estas duas relações, usando como condição a igualdade do identificador da viagem. Após a junção, aplica-se uma seleção sobre os atributos de localização ('Cidade' e 'País') para obter apenas as tuplas que satisfazem o critério definido.

$$\pi_{\texttt{ID, Data\_Início, Data\_Término, Nr\_Participantes, Custo, Objetivo}} \left(\sigma_{\texttt{País = ? } \wedge \texttt{ Cidade = ?}} \left(\texttt{Viagem} \bowtie_{\texttt{ID = Viagem}} \texttt{Paragem}\right)\right)$$

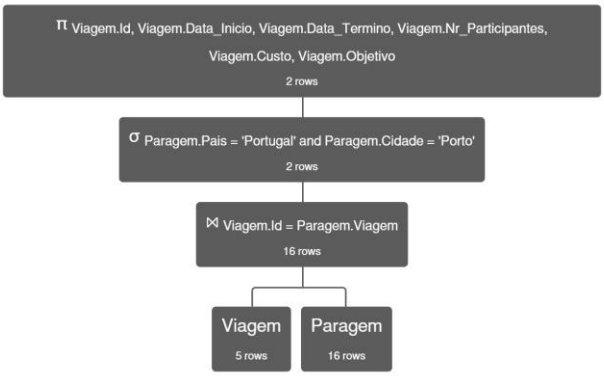


Figura 15 - Árvore da interrogação do RM8 para as 'Paragem' com País = 'Portugal' e Cidade = 'Porto'.

O requisito **RM9** consiste em "Filtrar viagens pelas datas, pelo custo e pelo objetivo". Pode ser satisfeito recorrendo exclusivamente à relação 'Viagem', dado que todos os atributos relevantes para este filtro se encontram presentes nesta tabela. A operação consiste apenas na aplicação de seleções sobre os atributos da 'Viagem'.

$$\sigma_{\texttt{Data\_Início >= ? } \wedge \texttt{ Data\_Término <= ? } \wedge \texttt{ Custo <= ? } \wedge \texttt{ Custo >= ? } \wedge \texttt{ Objetivo = ?}} \texttt{Viagem}$$

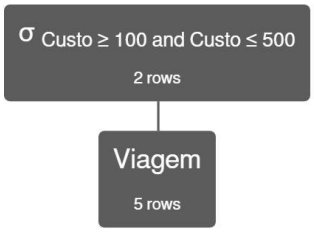


Figura 16 - Árvore de interrogação do RM9 para as 'Viagem' com custo entre 100 e 500.

Com o requisito **RM11**, "Calcular o investimento total de cada patrocinador", pretende-se calcular quanto é que cada patrocinador já investiu no total. Esta operação implica o agrupamento de dados por patrocinador e a aplicação de funções de agregação (SUM) sobre o atributo 'Valor' presente na tabela de relacionamento 'Rel_Viagem_Patrocinador'. É necessário realizar uma junção externa à esquerda com a tabela 'Patrocinador' para apresentar o nome da entidade e não apenas o seu código numérico.

$$\pi_{\texttt{ID, Nome, Investimento\_Total}} (\gamma_{\texttt{ID, Nome, SUM (Valor)} \rightarrow \texttt{ Investimento\_Total}} (\texttt{Patrocinador} \bowtie_{\texttt{ID = Patrocinador}} \texttt{Rel\_Viagem\_Patrocinador}))$$

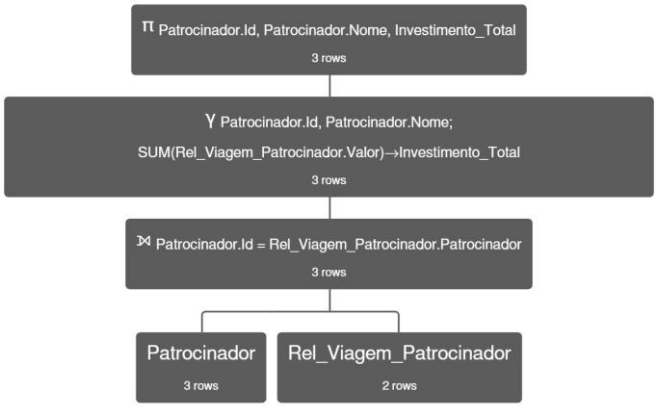


Figura 17 - Árvore de interrogação do RM11.

O requisito **RM12** consiste em "Calcular a média de investimento de um patrocinador por viagem", ou seja, pretende-se determinar quanto, em média, um determinado patrocinador investe numa viagem. Tal como no **RM11**, esta operação envolve os dados da tabela de relacionamento 'Rel_Viagem_Patrocinador', onde se encontra o atributo 'Valor'. Filtra-se o patrocinador através do seu identificador e calcula-se a média (AVG) dos valores de investimento.

$$\pi_{\texttt{Investimento\_Médio}}\left(\gamma_{\texttt{AVG (Valor)}\rightarrow\texttt{Investimento\_Médio}}\left(\sigma_{\texttt{Patrocinador = ?}}\texttt{Rel\_Viagem\_Patrocinador}\right)\right)$$

Figura 18 – Árvore de interrogação do RM12 para a 'Rel_Viagem_Patrocinador' com Patrocinador = 2.

O requisito **RM15** consiste em "Listar as viagens realizadas por um sócio (através do número de sócio)". A informação necessária encontra-se distribuída por três tabelas: 'Viagem', 'Sócio' e 'Rel_Sócio_Viagem'. A estratégia definida foi realizar primeiro uma junção entre 'Viagem' e 'Rel_Sócio_Viagem', utilizando como condição a correspondência entre 'Viagem.Id' e 'Rel_Sócio_Viagem.Viagem'. Seguiu-se uma segunda junção com a tabela 'Sócio', ligando 'Sócio.Utilizador' a 'Rel_Sócio_Viagem.Sócio'.

Após as junções, aplica-se uma seleção para filtrar apenas os registos correspondentes ao sócio identificado pelo número '?'. Finalmente, procede-se à projeção dos atributos necessários e à ordenação dos resultados por 'Viagem.Data_Início' por ordem decrescente.

$$\tau_{\texttt{Data\_Início DESC}} (\pi_{\texttt{ID, Data\_Início, Data\_Término, Nr\_Participantes, Custo, Objetivo}} (\sigma_{\texttt{Nr\_Sócio = ?}} ((\texttt{Viagem} \bowtie_{\texttt{ID = Viagem}} \texttt{Rel\_Sócio\_Viagem}) \bowtie_{\texttt{Utilizador = Sócio}} \texttt{Sócio})$$

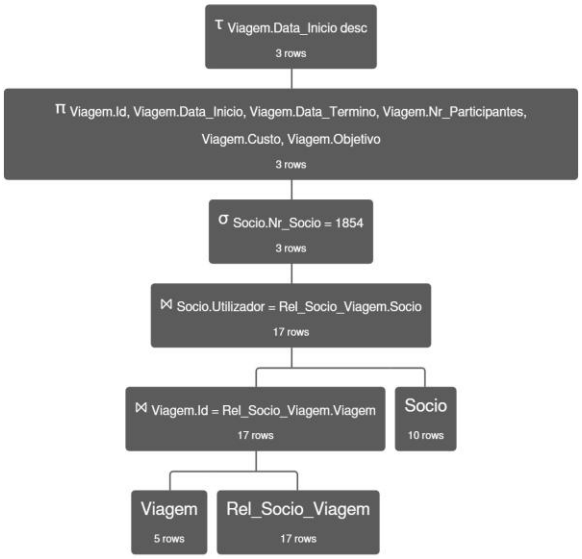


Figura 19 - Árvore de interrogação do RM15 para o 'Sócio' com Nr_Sócio = 1854.

O requisito **RM16** permite "Consultar os motivos de um patrocínio". Cada patrocinador contém uma ou mais razões para apoiar determinada viagem, registada na tabela 'Motivo'. Para satisfazer este requisito, seleciona-se da tabela 'Motivo' apenas os registos que correspondem ao identificador da viagem e ao identificador do patrocinador pretendido. Desta forma, é possível obter diretamente os motivos que justificam o patrocínio, facilitando a análise e a gestão das relações entre patrocinadores e viagens.

$$\pi_{\texttt{Motivo}} (\sigma_{\texttt{Patrocinador = ? } \wedge \texttt{ Viagem = ?}} \texttt{Motivo})$$

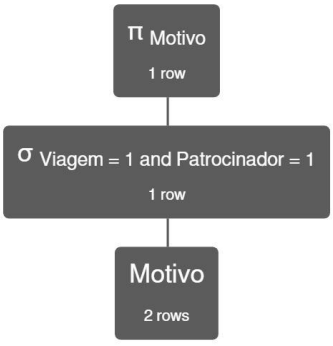


Figura 20 - Árvore de interrogação do RM16 para o 'Motivo' com Viagem = 1 e Patrocinador = 1.

O requisito **RM17** permite "Consultar o número de viagens publicadas por cada sócio (ordenado por número de publicações)", tendo como objetivo quantificar a participação dos membros na criação de conteúdo para a plataforma. A informação necessária encontra-se distribuída entre três tabelas: 'Sócio', da qual se utiliza apenas o 'Utilizador' para conectar o sócio às outras tabelas; 'Rel_Sócio_Viagem', que relaciona os sócios às viagens publicadas; e 'Utilizador', onde se encontra o nome do sócio. Primeiro, é realizada uma junção interna entre 'Utilizador' e 'Sócio' para associar cada sócio aos seus dados de identificação. Em seguida, aplica-se uma junção externa à esquerda com 'Rel_Sócio_Viagem', garantindo que todos os sócios ficam representados no resultado, mesmo que não tenham publicado qualquer viagem. Por fim, aplica-se um agrupamento pelo número do sócio e conta-se o número de viagens associadas a cada um. O resultado é ordenado de forma decrescente relativamente ao número de viagens publicadas.

$$\tau_{\texttt{Nr\_Viagens\_Publicadas DESC}} (\pi_{\texttt{Nr\_Sócio, Nome, Nr\_Viagens\_Publicadas}} (\gamma_{\texttt{Nr\_Sócio, Nome, COUNT (Viagem)} \rightarrow \texttt{Nr\_Viagens\_Publicadas}} ((\texttt{Utilizador} \bowtie_{\texttt{Utilizador = ID}} \texttt{Sócio}) \bowtie_{\texttt{Utilizador = Sócio}} \texttt{Rel\_Sócio\_Viagem}))))$$

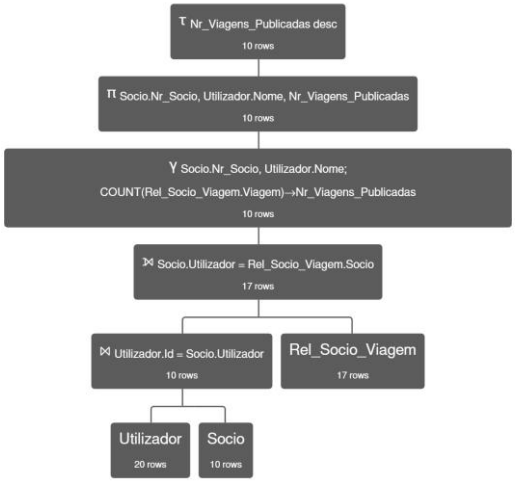


Figura 21 - Árvore de interrogação do RM17.

O requisito **RM18** – "Filtrar viagens por percentagem de reações positivas (ordenado por percentagem)" é essencial para simplificar a decisão de futuros participantes, um dos objetivos centrais do projeto. Calcula-se, através da tabela 'Rel_Utilizador_Viagem' a percentagem de reações positivas para cada viagem: soma-se o número de reações positivas e divide-se pelo total de reações existentes (ignorando valores nulos). Depois, realiza-se uma junção interna com a tabela 'Viagem' para obter todos os atributos das viagens correspondentes. Aplica-se uma seleção para filtrar apenas as viagens cuja percentagem de reações positivas seja igual ou superior ao valor definido, e finalmente ordena-se o resultado pela percentagem de reações positivas por ordem crescente.

$$\tau_{\texttt{Percentagem DESC}} (\pi_{\texttt{v.*, P.Percentagem}} (\sigma_{\texttt{Percentagem >= ?}} (\texttt{Viagem V} \bowtie_{\texttt{V.ID = P.Viagem}} (\texttt{P} \leftarrow \rho_{\texttt{Percentagem} \leftarrow \texttt{Conta}} (\gamma_{\texttt{Viagem, Conta}} \texttt{Rel\_Utilizador\_Viagem})))))$$

'Conta' representa o cálculo da percentagem positiva descrito acima, que não se consegue detalhar em álgebra relacional.

Devido à complexidade envolvida no cálculo para a execução da interrogação, não foi possível utilizar a calculadora RelaX. Em alternativa, a representação da árvore foi elaborada através de um editor online de diagramas, *Mermaid*, sendo a visualização ilustrada apenas pelo nome das operações e relações pertinentes.

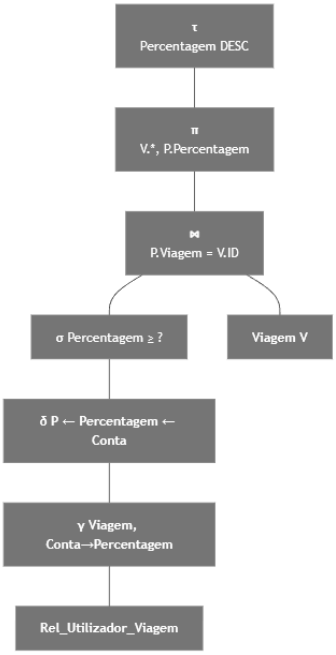


Figura 22 - Árvore de interrogação do RM18.

O requisito **RM19** "Consultar o top 10 viagens mais populares, isto é, com mais visualizações" foca-se na identificação das tendências de visualização. O conceito de popularidade de cada viagem é determinado pelo número de interações (visualizações) registadas pelos utilizadores. A informação necessária reside na tabela 'Rel_Utilizador_Viagem', que regista cada interação entre um utilizador e uma viagem. Para apresentar todos os detalhes de cada viagem, realiza-se uma junção externa à esquerda com a tabela 'Viagem', de modo a associar cada 'ID' da viagem às suas informações completas. A estratégia adotada consiste em contar o número de ocorrências de cada viagem e ordenar os resultados de forma decrescente, destacando as viagens com maior número de visualizações, refletindo as preferências dos utilizadores. Apesar de o requisito referir o top 10, essa limitação não foi considerada na álgebra relacional.

$$\tau_{\texttt{Nr\_Visualizações DESC}} (\pi_{\texttt{V.*, Nr\_Visualizações}} (\gamma_{\texttt{Viagem, COUNT (Utilizador)} \rightarrow \texttt{Nr\_Visualizações}} (\texttt{Viagem} \rtimes_{\texttt{ID = Viagem}} \texttt{Rel\_Utilizador\_Viagem})))$$

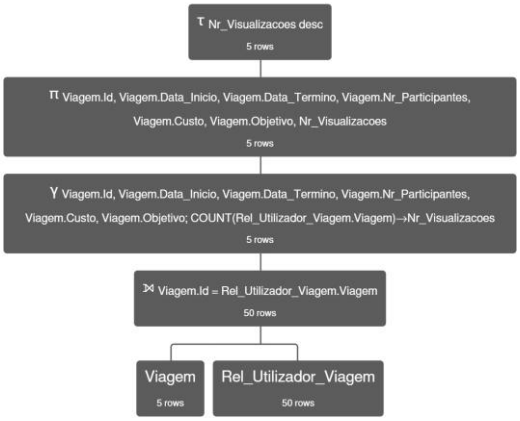


Figura 23 - Árvore de interrogação do RM19.

O requisito **RM20** consiste em "Consultar as opiniões dos sócios em relação a uma dada viagem". A informação necessária encontra-se nas tabelas 'Rel_Sócio_Viagem' e 'Sócio'. A relação 'Rel_Sócio_Viagem' armazena as avaliações e comentários efetuados pelos sócios relativamente às viagens, enquanto a tabela 'Sócio' permite identificar cada sócio através do respetivo número. A estratégia adotada consiste numa junção entre 'Rel_Sócio_Viagem' e 'Sócio', ligando o atributo 'Sócio.Utilizador' ao atributo 'Rel_Sócio_Viagem.Sócio'. Após a junção, aplica-se uma seleção para filtrar a viagem pretendida e, por fim, uma projeção sobre o número de sócio, a avaliação e o comentário associados.

$$\pi_{\texttt{Nr\_Sócio, Avaliação, Comentário}} (\sigma_{\texttt{Viagem = ?}} (\texttt{Rel\_Sócio\_Viagem} \bowtie_{\texttt{Utilizador = Sócio}} \texttt{Sócio}))$$

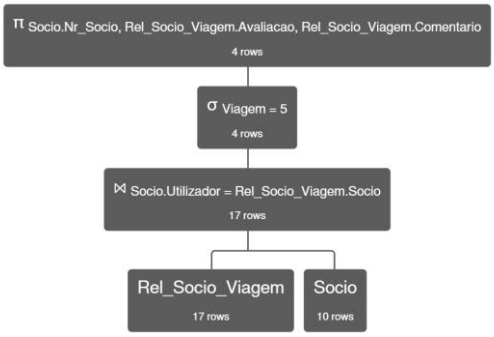


Figura 24 - Árvore de interrogação do RM20 para a 'Rel_Sócio_Viagem' com Viagem = 5.

5. Implementação Física

5.1. Apresentação e Explicação da Base de Dados Implementada

Após a construção do modelo lógico, procedeu-se à implementação do esquema físico da base de dados. Esta implementação consiste na tradução do modelo lógico para um conjunto de instruções SQL, que são interpretadas pelo SGBD MySQL para criar a base de dados e todas as tabelas correspondentes.

O primeiro passo consistiu na criação da base de dados, efetuada através da execução dos seguintes comandos:

```
DROP DATABASE IF EXISTS BeLIUM_Viagens;
CREATE DATABASE IF NOT EXISTS BeLIUM_Viagens;
USE BeLIUM_Viagens;
```

A primeira instrução `DROP DATABASE IF EXISTS BeLIUM_Viagens`, elimina a BD "BeLIUM_Viagens" caso ela já exista, permitindo que o script seja executado novamente sem gerar erros. O uso de `IF EXISTS` garante que não ocorrerá uma tentativa de remoção de uma base de dados inexistente. Em seguida, a instrução `CREATE DATABASE IF NOT EXISTS BeLIUM_Viagens`, cria a base de dados de nome "BeLIUM_Viagens". Por último, a diretiva `USE BeLIUM_Viagens` permite informar o MySQL que todas as instruções subsequentes deverão ser aplicadas a esta base de dados recém-criada.

Seguiu-se a etapa de criação das tabelas. No entanto, antes disso, tornou-se necessário definir as dependências entre elas, uma vez que algumas tabelas contêm chaves estrangeiras que fazem referência a outras – por exemplo, 'Sócio' depende de 'Utilizador'. Ou seja, para que uma chave estrangeira seja validada corretamente durante a criação de uma tabela, a tabela de referência já deve existir. Embora seja possível usar instruções `ALTER TABLE` para adicionar chaves estrangeiras posteriormente, esta abordagem é menos eficiente. Por esta razão, optou-se por determinar antecipadamente a sequência de criação das tabelas. Para auxiliar neste processo, foi gerado um grafo de dependências, em que cada nodo representa uma tabela e existe uma aresta A $\rightarrow$ B sempre que a tabela B possui uma chave estrangeira que referencia atributos da tabela A.

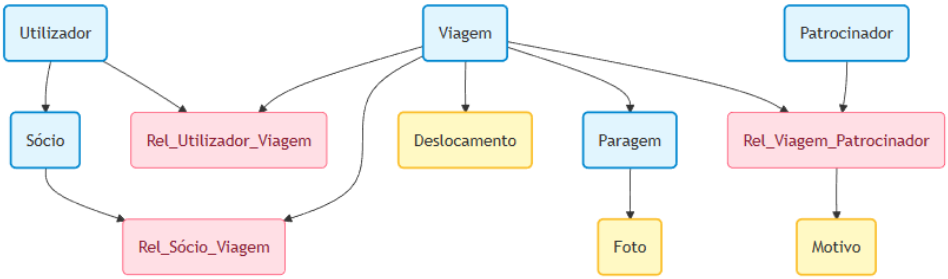


Figura 25 - Grafo do diagrama de dependências da criação da BD.

Com base no grafo, é possível identificar claramente as dependências entre as tabelas e, a partir disso, definir uma sequência lógica para a sua criação. A tabela 'Utilizador' é base para 'Sócio' e para o relacionamento 'Rel_Utilizador_Viagem', pois estas tabelas possuem chaves estrangeiras que a referenciam. A tabela 'Sócio', por sua vez, é necessária para o relacionamento 'Rel_Sócio_Viagem'. A tabela 'Viagem' serve de referência para várias tabelas: 'Deslocamento', 'Rel_Utilizador_Viagem', 'Rel_Sócio_Viagem', 'Rel_Viagem_Patrocinador' e 'Paragem'. 'Patrocinador' é referenciado apenas por 'Rel_Viagem_Patrocinador', que, por sua vez, é base para a tabela 'Motivo'. Finalmente, 'Paragem' é referenciada pela tabela 'Foto'.

Assim, primeiramente foram criadas as entidades principais: 'Utilizador', 'Sócio', 'Viagem', 'Paragem' e 'Patrocinador'. Em seguida, foram definidas as tabelas de relacionamentos: 'Rel_Utilizador_Viagem', 'Rel_Sócio_Viagem' e 'Rel_Viagem_Patrocinador'. Por último, criaram-se as tabelas dos atributos multivalor: 'Deslocamento', 'Foto' e 'Motivo'.

As tabelas apresentadas nesta secção, seguem o mesmo padrão de criação já exemplificado na implementação lógica. Por esse motivo, aspetos já discutidos, como a definição básica de atributos, não serão detalhados novamente.

Seguindo a ordem topológica apresentada acima, começamos pela criação da tabela 'Utilizador', tal como mostra a diretiva:

```
CREATE TABLE IF NOT EXISTS Utilizador (

    ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Telemóvel VARCHAR(13) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Pass_Word VARCHAR(100) NOT NULL,

    PRIMARY KEY (ID),

    CHECK (Telemóvel LIKE '+%'),
```

```
    CHECK (Email LIKE '%@%.%')
);
```

O código SQL apresentado define os campos essenciais da tabela: ID, Nome, Telemóvel, Email e Pass_Word, e estabelece o ID como chave primária. Cada um destes campos é obrigatório e o ID é incrementado automaticamente a cada novo registo, permitindo a criação de novos utilizadores sem necessidade de atribuição manual de identificadores. A integridade dos dados é reforçada por definições específicas e restrições: o uso de `INT UNSIGNED` garante que o identificador seja sempre positivo. A restrição `UNIQUE` no Email impede duplicados (conforme **RC4**) e as cláusulas `CHECK` validam formatos básicos, assegurando que o Telemóvel comece pelo indicativo internacional ("+") e que o Email contenha os caracteres essenciais ("@" e um ponto).

A próxima relação criada foi 'Sócio', através da seguinte instrução SQL:

```
CREATE TABLE IF NOT EXISTS Sócio (

    Utilizador INT UNSIGNED NOT NULL,
    Nr_Sócio INT UNSIGNED NOT NULL UNIQUE,
    Foto_Perfil VARCHAR(100) NULL,
    Estatuto ENUM('Direção','Alumni') NULL,

    PRIMARY KEY (Utilizador),

    CONSTRAINT FK_Sócio_To_Utilizador FOREIGN KEY
    (Utilizador) REFERENCES Utilizador(ID),

    CHECK (Foto_Perfil IS NULL OR Foto_Perfil LIKE
    'Imagens/%.png')
);
```

A tabela 'Sócio' contém os campos Utilizador, Nr_Sócio, Foto_Perfil e Estatuto. O Utilizador é chave primária e também chave estrangeira que referencia 'Utilizador'. O campo Nr_Sócio possui uma restrição de unicidade (`UNIQUE`) e o tipo `UNSIGNED` impede valores negativos. A integridade dos restantes dados é assegurada de duas formas distintas: a restrição `CHECK` valida se a Foto_Perfil (caso exista) segue o diretório e formato 'Imagens/%.png', enquanto o tipo de dados `ENUM` restringe as opções do Estatuto exclusivamente a 'Direção' ou 'Alumni'.

Já a criação da tabela 'Viagem' foi obtida através da instrução abaixo:

```
CREATE TABLE IF NOT EXISTS Viagem (
```

```
    ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Data_Início DATETIME NOT NULL,
    Data_Término DATETIME NOT NULL,
    Nr_Paragens INT NOT NULL DEFAULT 0,
    Nr_Participantes INT NOT NULL,
    Custo DECIMAL(8,2) NOT NULL,
    Objetivo ENUM('Pedagógico','Recreativo','Ambos')
    NOT NULL,

    PRIMARY KEY (ID),

    CHECK (Data_Início < Data_Término),
    CHECK (Nr_Paragens >= 0),
    CHECK (Nr_Participantes >= 0),
    CHECK (Custo > 0)
);
```

O campo ID é definido como chave primária auto-incrementada, sendo a sua natureza não negativa garantida pelo tipo `INT UNSIGNED`. As restrições `CHECK` são aplicadas para validar a lógica de negócio: a Data_Início deve ser cronologicamente anterior à Data_Término, o Custo deve ser estritamente positivo e tanto o Nr_Participantes como o Nr_Paragens (que assume o valor 0 por defeito caso não especificado) não podem ser negativos. Por fim, o campo Objetivo é restrito aos valores 'Pedagógico', 'Recreativo' ou 'Ambos' através da definição do tipo `ENUM`.

A tabela 'Paragem' representa uma entidade fraca, cuja existência depende da Viagem:

```
CREATE TABLE IF NOT EXISTS Paragem (

    Nr INT UNSIGNED NOT NULL,
    Viagem INT UNSIGNED NOT NULL,
    Cidade VARCHAR(50) NOT NULL,
    País VARCHAR(50) NOT NULL,
    Data_Chegada DATETIME NOT NULL,
    Data_Partida DATETIME NOT NULL,

    PRIMARY KEY (Nr,Viagem),

    CONSTRAINT FK_Paragem_To_Viagem FOREIGN KEY (Viagem)
    REFERENCES Viagem(ID),

    CHECK (Data_Chegada < Data_Partida)
);
```

A tabela possui uma chave primária composta pelo par (Nr, Viagem), refletindo a sua dependência em relação à entidade forte. A integridade referencial é assegurada pela chave estrangeira Viagem, que liga cada paragem ao identificador único da tabela 'Viagem'. Em termos de validação de dados, o tipo `INT UNSIGNED` garante que o número sequencial da paragem (Nr) nunca seja negativo. Por sua vez, a restrição `CHECK` é utilizada para garantir a consistência temporal, assegurando que a Data_Chegada é cronologicamente anterior à Data_Partida.

A vertente financeira é suportada pela tabela Patrocinador, uma entidade simples que armazena os dados das organizações financiadoras:

```
CREATE TABLE IF NOT EXISTS Patrocinador (

    ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(40) NOT NULL,

    PRIMARY KEY (ID)
);
```

O campo ID atua como chave primária auto-incrementada e positiva (`UNSIGNED`), e o campo Nome armazena a designação da entidade, sendo de preenchimento obrigatório.

A tabela 'Rel_Utilizador_Viagem' regista a interação básica de qualquer utilizador:

```
CREATE TABLE IF NOT EXISTS Rel_Utilizador_Viagem (

    Utilizador INT UNSIGNED NOT NULL,
    Viagem INT UNSIGNED NOT NULL,
    Reação ENUM('Negativa','Positiva') NOT NULL,

    PRIMARY KEY (Utilizador,Viagem),

    CONSTRAINT FK_Rel_Utilizador_Viagem_To_Utilizador
    FOREIGN KEY (Utilizador) REFERENCES Utilizador(ID),
    CONSTRAINT FK_Rel_Utilizador_Viagem_To_Viagem FOREIGN
    KEY (Viagem) REFERENCES Viagem(ID)
);
```

Esta tabela materializa o relacionamento entre as entidades 'Utilizador' e 'Viagem', contendo um campo Reação que é obrigatoriamente preenchido com 'Negativa' ou 'Positiva' (validado pelo tipo `ENUM`). A integridade dos dados é assegurada de duas formas: as chaves estrangeiras garantem a integridade referencial (o utilizador e a viagem têm de existir) e a chave primária composta (Utilizador, Viagem) que é o mecanismo que impede a duplicação.

Para os membros do núcleo, a tabela 'Rel_Sócio_Viagem' permite um feedback mais detalhado:

```
CREATE TABLE IF NOT EXISTS Rel_Sócio_Viagem (

    Sócio INT UNSIGNED NOT NULL,
    Viagem INT UNSIGNED NOT NULL,
    Avaliação ENUM('Muito Negativa', 'Negativa', 'Mediana',
    'Positiva', 'Muito Positiva') NOT NULL,
    Comentário VARCHAR(150) NULL,

    PRIMARY KEY (Sócio,Viagem),

    CONSTRAINT FK_Rel_Sócio_Viagem_To_Sócio FOREIGN KEY
    (Sócio) REFERENCES Sócio(Utilizador),
    CONSTRAINT FK_Rel_Sócio_Viagem_To_Viagem FOREIGN KEY
    (Viagem) REFERENCES Viagem(ID)
);
```

Esta relação utiliza uma chave primária composta por (Sócio, Viagem). A estrutura de dados impõe o preenchimento obrigatório do campo Avaliação, cujos valores são estritamente limitados pelo tipo `ENUM` a uma escala de cinco níveis (de 'Muito Negativa' a 'Muito Positiva'). O campo Comentário, definido como opcional (`NULL`) e limitado a 150 caracteres, permite observações textuais complementares.

A relação de muitos-para-muitos (N:M) entre viagens e patrocinadores é materializada na tabela 'Rel_Viagem_Patrocinador':

```
CREATE TABLE IF NOT EXISTS Rel_Viagem_Patrocinador (

    Viagem INT UNSIGNED NOT NULL,
    Patrocinador INT UNSIGNED NOT NULL,
    Valor DECIMAL(8,2) NOT NULL,

    PRIMARY KEY (Viagem,Patrocinador),

    CONSTRAINT FK_Rel_Viagem_Patrocinador_To_Viagem FOREIGN
    KEY (Viagem) REFERENCES Viagem(ID),
    CONSTRAINT FK_Rel_Viagem_Patrocinador_To_Patrocinador
    FOREIGN KEY (Patrocinador) REFERENCES Patrocinador(ID),

    CHECK (Valor > 0)
);
```

A tabela 'Rel_Viagem_Patrocinador' contém uma chave primária composta por (Viagem, Patrocinador). O atributo Valor, definido com precisão decimal para suportar montantes monetários, é protegido pela restrição `CHECK` (Valor > 0), assegurando que apenas quantias estritamente positivas (maiores que zero) sejam registadas no sistema. As chaves estrangeiras garantem a integridade referencial, ligando 'Viagem' à tabela 'Viagem' e 'Patrocinador' à tabela 'Patrocinador'.

Como uma viagem pode utilizar diferentes meios de transporte, a tabela 'Deslocamento' foi criada para normalizar este atributo multivalorado.

```
CREATE TABLE IF NOT EXISTS Deslocamento (

    Tipo_Deslocamento ENUM('Carro', 'Autocarro', 'Comboio',
    'Avião', 'Barco', 'Navio', 'Bicicleta', 'Mota', 'A pé',
    'Outro') NOT NULL,
    Viagem INT UNSIGNED NOT NULL,

    PRIMARY KEY (Tipo_Deslocamento,Viagem),

    CONSTRAINT FK_Deslocamento_To_Viagem FOREIGN KEY
    (Viagem) REFERENCES Viagem(ID)
);
```

A tabela possui uma chave primária composta por (Tipo_Deslocamento, Viagem) e uma chave estrangeira 'Viagem' que referencia a tabela 'Viagem'. O campo Tipo_Deslocamento assegura a consistência dos dados através de um `ENUM`, restringindo as entradas a uma lista fechada de 10 opções de mobilidade, que vão desde veículos motorizados até à opção 'A pé'.

Para registos multimédia, criou-se a tabela 'Foto':

```
CREATE TABLE IF NOT EXISTS Foto (

    Foto VARCHAR(100) NOT NULL,
    Paragem INT UNSIGNED NOT NULL,
    Viagem INT UNSIGNED NOT NULL,

    PRIMARY KEY (Foto,Paragem,Viagem),

    CONSTRAINT FK_Foto_To_Paragem FOREIGN KEY
    (Paragem,Viagem) REFERENCES Paragem(Nr,Viagem),

    CHECK (Foto LIKE 'Fotos/%.png')
);
```

A tabela 'Foto' armazena imagens associadas a cada paragem de uma viagem. Possui chave primária composta por (Foto, Paragem, Viagem) e uma chave estrangeira que referencia a tabela 'Paragem'. A restrição `CHECK` garante que o nome do ficheiro da foto segue o formato 'Fotos/%.png'.

Por último, para detalhar as razões do financiamento, foi criada a tabela Motivo:

```
CREATE TABLE IF NOT EXISTS Motivo (

    Motivo VARCHAR(200) NOT NULL,
    Patrocinador INT UNSIGNED NOT NULL,
    Viagem INT UNSIGNED NOT NULL,

    PRIMARY KEY (Motivo,Patrocinador,Viagem),

    CONSTRAINT FK_Motivo_To_Rel_Viagem_Patrocinador FOREIGN
    KEY (Viagem,Patrocinador) REFERENCES
    Rel_Viagem_Patrocinador(Viagem,Patrocinador)
);
```

A tabela 'Motivo' depende da chave composta (Viagem, Patrocinador) da relação 'Rel_Viagem_Patrocinador'. Esta estrutura permite que um mesmo patrocínio tenha várias justificações associadas, preservando o detalhe da informação e mantendo a base de dados normalizada.

5.2. Criação de Utilizadores na Base de Dados

O passo seguinte na implementação do sistema BeLIUM Viagens é a criação dos utilizadores, conforme os requisitos de controlo. Esta etapa é fundamental para garantir o correto funcionamento do sistema ao longo do tempo, estabelecendo as permissões aplicáveis a cada tipo de utilizador.

Primeiramente, é necessário identificar os diferentes tipos de utilizador a serem criados. Com base nos requisitos, identifica-se uma clara distinção entre três categorias:

- **Utilizadores**: Possuem permissões muito restritas, limitadas à consulta dos dados centrais da base de dados (**RC2**), atualização da própria password (**RC3**) – desde que saibam a password atual – e gestão de visualizações das viagens (**RC5**), isto é, registos na tabela 'Rel_Utilizador_Viagem', realizadas pelo próprio.

- **Sócios**: Possuem todas as permissões dos utilizadores (**RC6**). Adicionalmente, podem inserir novos registos de viagens e paragens, assim como os respetivos tipos de deslocamento e fotos associadas (**RC7**). Podem consultar todos os dados do sistema, exceto os dados pessoais dos

utilizadores (**RC8**), o que, na prática, significa que podem consultar todas as tabelas exceto a tabela 'Utilizador'.

- **Administradores:** Têm controlo total sobre o sistema (**RC10**), podendo realizar qualquer operação, incluindo modificações estruturais. São responsáveis pela gestão de tabelas que possuem controlo restrito: 'Utilizador', 'Sócio' e todas as associadas aos patrocínios – 'Rel_Viagem_Patrocinador', 'Patrocinador' e 'Motivo'. São ainda responsáveis pelo registo de novos utilizadores (**RC1**) e pela gestão de permissões na base de dados (**RC11**). Este grupo, composto, inicialmente, apenas pelos membros do CCP, é fundamental para a resolução de problemas ocasionais na base de dados.

Após a distinção das categorias de utilizadores, procede-se à sua implementação física no sistema através das seguintes instruções SQL:

```
CREATE USER IF NOT EXISTS 'UTILIZADOR'@'localhost'
  IDENTIFIED BY 'BeLIUM_Viagens_UTILIZADOR_1025';

CREATE USER IF NOT EXISTS 'SOCIO'@'localhost'
  IDENTIFIED BY 'BeLIUM_Viagens_SOCIO_2047';

CREATE USER IF NOT EXISTS 'ADMIN'@'localhost'
    IDENTIFIED BY 'BeLIUM_Viagens_VDBS';
```

A criação dos utilizadores é realizada através do comando `CREATE USER`, que estabelece as identidades de autenticação para os três papéis definidos. A cláusula `IF NOT EXISTS` previne erros caso o comando seja executado quando os utilizadores já existiam. A sintaxe `'TIPO_UTILIZADOR'@'localhost'` define o nome do utilizador e permite que a base de dados seja acessada a partir do *host* local, impedindo acessos de conexão remota, com a respetiva senha na cláusula `IDENTIFIED BY`. Por motivos de segurança, as senhas reais foram substituídas por *placeholders*.

Para efeitos de reversão ou manutenção, foram também definidas as instruções para eliminar estes utilizadores:

```
DROP USER IF EXISTS 'UTILIZADOR'@'localhost';
  DROP USER IF EXISTS 'SOCIO'@'localhost';
  DROP USER IF EXISTS 'ADMIN'@'localhost';
```

A cláusula `IF EXISTS` serve, de forma análoga, para evitar erros ao tentar eliminar utilizadores que não existam. O comando `DROP USER` remove o utilizador do sistema, sendo a operação inversa de `CREATE USER`. Dada a correspondência das designações, estas instruções removem, de forma segura, os três tipos de utilizador definidos.

Finalmente, é necessário atribuir as permissões a cada um dos utilizadores. Primeiramente, foram definidas as permissões dos utilizadores comuns, através das seguintes instruções *SQL*:

```
GRANT SELECT ON BeLIUM_Viagens.Sócio TO 'UTILIZADOR'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Viagem TO 'UTILIZADOR'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Paragem TO 'UTILIZADOR'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Deslocamento TO 'UTILIZADOR'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Foto TO 'UTILIZADOR'@'localhost';

GRANT EXECUTE ON PROCEDURE MudarPassword TO 'UTILIZADOR'@'localhost';
GRANT EXECUTE ON PROCEDURE InserirVisualização TO 'UTILIZADOR'@'localhost';
GRANT EXECUTE ON PROCEDURE AtualizarVisualização TO 'UTILIZADOR'@'localhost';
GRANT EXECUTE ON PROCEDURE RemoverVisualização TO 'UTILIZADOR'@'localhost';
```

O comando `GRANT` atribui permissões a utilizadores. Visto que os utilizadores devem ter acesso às tabelas 'Sócio', 'Viagem', 'Paragem', 'Deslocamento' e 'Foto', foram concedidas estas permissões através da instrução `GRANT SELECT`. Posteriormente, o comando `GRANT EXECUTE ON PROCEDURE` permite ao utilizador executar quatro procedimentos, cujo funcionamento detalhado será explicado na secção 5.8. O procedimento 'MudarPassword' permite ao utilizador alterar a sua password, conforme **RC3**, e os restantes procedimentos permitem inserir, atualizar e remover registos de visualizações (tabela 'Rel_Utilizador_Viagem'), conforme **RC5**. A opção pela utilização de procedimentos é fundamental, pois garante que um utilizador apenas efetua alterações caso conheça a password do utilizador envolvido. No entanto, uma vez que a sua implementação ainda não foi realizada, os comandos de concessão de permissões sobre os procedimentos não foram executados até essa fase. É importante salientar que, para que um utilizador se registe no sistema, deverá contactar um administrador, conforme **RC1**. Após obter um registo na base de dados, o utilizador poderá, autenticando-se com a respetiva password, usufruir das permissões atribuídas. Deste modo, um utilizador não registado pode consultar os dados a que tem acesso, embora não possa interagir diretamente com o sistema ao nível de alterações.

Segue-se o estabelecimento das permissões dos sócios. De acordo com o **RC6**, os sócios possuem todas as permissões dos utilizadores. Por esse motivo, foram executadas as mesmas instruções para o utilizador dos sócios:

```
GRANT SELECT ON BeLIUM_Viagens.Sócio TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Viagem TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Paragem TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Deslocamento TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Foto TO 'SOCIO'@'localhost';

GRANT EXECUTE ON PROCEDURE MudarPassword TO 'SOCIO'@'localhost';
GRANT EXECUTE ON PROCEDURE InserirVisualização TO 'SOCIO'@'localhost';
GRANT EXECUTE ON PROCEDURE AtualizarVisualização TO 'SOCIO'@'localhost';
```

```
GRANT EXECUTE ON PROCEDURE RemoverVisualização TO 'SOCIO'@'localhost';
```

Para completar a definição das permissões dos sócios, foram executadas as seguintes instruções SQL:

```
GRANT INSERT, UPDATE, DELETE ON BeLIUM_Viagens.Viagem TO 'SOCIO'@'localhost';
GRANT INSERT, UPDATE, DELETE ON BeLIUM_Viagens.Paragem TO 'SOCIO'@'localhost';
GRANT INSERT, UPDATE, DELETE ON BeLIUM_Viagens.Deslocamento TO 'SOCIO'@'localhost';
GRANT INSERT, UPDATE, DELETE ON BeLIUM_Viagens.Foto TO 'SOCIO'@'localhost';
GRANT INSERT, UPDATE, DELETE ON BeLIUM_Viagens.Rel_Sócio_Viagem TO 'SOCIO'@'localhost';

GRANT SELECT ON BeLIUM_Viagens.Rel_Utilizador_Viagem TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Rel_Sócio_Viagem TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Rel_Viagem_Patrocinador TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Patrocinador TO 'SOCIO'@'localhost';
GRANT SELECT ON BeLIUM_Viagens.Motivo TO 'SOCIO'@'localhost';
```

As permissões de `INSERT`, `UPDATE` e `DELETE` sobre as tabelas 'Viagem', 'Paragem', 'Deslocamento' e 'Foto' permitem aos sócios gerir integralmente o conteúdo das viagens, conforme estabelecido em **RC7**. Esta autorização abrange a criação de novas viagens, a edição das existentes e a sua remoção quando necessário. Essas mesmas permissões sobre a tabela 'Rel_Sócio_Viagem' permitem aos sócios publicar, editar e eliminar avaliações e comentários de viagens, conforme o requisito **RC9**. As restantes permissões de `SELECT` garantem que os sócios podem consultar todos os dados do sistema, exceto os dados pessoais dos utilizadores (na tabela 'Utilizador'), conforme especificado em **RC8**. Esta abordagem assegura que os sócios têm uma visão completa das operações do sistema, incluindo as reações dos utilizadores, as avaliações dos sócios e os patrocínios às viagens.

Resta, para finalizar esta etapa, estabelecer as permissões dos administradores. Estas são definidas através da seguinte instrução:

```
GRANT ALL PRIVILEGES ON BeLIUM_Viagens.*
TO 'ADMIN'@'localhost' WITH GRANT OPTION;
```

Este comando concede aos administradores controlo total sobre o sistema. A expressão `ALL PRIVILEGES` atribui todas as permissões possíveis no sistema BeLIUM_Viagens, incluindo as operações `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, `DROP`, `INDEX`, e todas as demais. O uso do asterisco assegura que estas permissões se aplicam a todas as tabelas, views, procedimentos e outros objetos, tanto os atuais como os que venham a ser criados no futuro.

A cláusula `WITH GRANT OPTION` é particularmente significativa, pois concede aos administradores a capacidade de atribuir as suas próprias permissões a outros utilizadores. Isto implementa diretamente o **RC11**, permitindo que administradores existentes possam promover sócios de confiança ao estatuto de administrador, concedendo-lhes os mesmos privilégios. Esta definição cumpre integralmente o

requisito **RC10**, que estabelece que os administradores podem realizar qualquer tipo de operação no sistema, incluindo alterações estruturais. Os administradores ficam assim responsáveis por todas as operações fora do alcance dos sócios, tais como: a gestão direta da tabela Utilizador (incluindo registo de novos utilizadores - **RC1**), a modificação da estrutura da base de dados, a gestão de patrocínios e a resolução de problemas que exijam intervenção profunda no sistema.

De seguida, determinou-se quais os tipos de utilizador que devem ter acesso às diferentes vistas criadas (secção 5.5.). A tabela seguinte resume as relações envolvidas em cada vista e os respetivos perfis de utilizador autorizados:

Vista	Relações	Utilizadores
VisualizarInvestimentoTotal	Patrocinador	Administradores, Sócios
VisualizarNrPublicações	Utilizador, Sócio	Todos os utilizadores
VisualizarViagensPopulares	Viagem	Todos os utilizadores

Tabela 27 - Explicação das vistas criadas.

Analisando a tabela, verifica-se que administradores e sócios devem ter acesso às três vistas definidas. No entanto, considerando que os dados relativos a patrocínios são de natureza sensível e não devem ser acessíveis a utilizadores comuns, a vista 'VisualizarInvestimentoTotal' foi restringida apenas aos perfis de administrador e sócio.

Para implementar este modelo de permissões, foram executadas as seguintes instruções *SQL* (utilizando o comando `GRANT`), conforme mencionado anteriormente:

```
GRANT SELECT ON BeLIUM_Viagens.VisualizarInvestimentoTotal TO
'SOCIO'@'localhost', 'ADMIN'@'localhost';

GRANT SELECT ON BeLIUM_Viagens.VisualizarNrPublicações TO
'UTILIZADOR'@'localhost', 'SOCIO'@'localhost', 'ADMIN'@'localhost';

GRANT SELECT ON BeLIUM_Viagens.VisualizarViagensPopulares TO
'UTILIZADOR'@'localhost', 'SOCIO'@'localhost', 'ADMIN'@'localhost';
```

Adicionalmente, por efeitos de completude, foram concedidas permissões de acesso a funções (`FUNCTIONS`, secção 5.8.). As permissões foram configuradas através das seguintes instruções *SQL*:

```
GRANT EXECUTE ON FUNCTION CalcularInvestimentoTotal
TO 'SOCIO'@'localhost', 'ADMIN'@'localhost';

GRANT EXECUTE ON FUNCTION CalcularInvestimentoMédio
TO 'SOCIO'@'localhost', 'ADMIN'@'localhost';

GRANT EXECUTE ON FUNCTION CalcularPercentagemPositiva
TO 'UTILIZADOR'@'localhost', 'SOCIO'@'localhost', 'ADMIN'@'localhost';
```

Em suma, a arquitetura resultante organiza-se em três níveis hierárquicos bem diferenciados: os **Utilizadores** com permissões básicas de consulta e gestão das suas próprias interações; os **Sócios**, que acrescentam a capacidade de gerir conteúdo e aceder à quase totalidade dos dados; e os **Administradores**, com controlo absoluto e responsabilidade pela manutenção global do sistema. Esta estrutura assegura o equilíbrio entre funcionalidade, autonomia dos diferentes atores e segurança dos dados, estabelecendo uma base sólida para a operação do sistema.

5.3. Povoamento da Base de Dados

O povoamento inicial da base de dados visa testar o cumprimento de todos os requisitos funcionais do sistema. Para este efeito, a equipa de desenvolvedores optou pela utilização de procedimentos armazenados (*stored procedures*), através da instrução `INSERT` da DML (*Data Manipulation Language)* do SQL.

Um dos aspetos críticos no povoamento é a ordem de inserção dos registos, que deve respeitar as dependências de integridade referencial. Por exemplo, não é possível inserir registos em 'Sócio' antes dos respetivos 'Utilizador', pois a chave estrangeira referenciaria um registo inexistente. Para garantir esta coerência, seguiu-se a ordem topológica das dependências entre tabelas, que pode ser representada pela seguinte sequência:

1. 'Utilizador' (entidade independente)
2. 'Sócio' (depende de 'Utilizador')
3. 'Viagem' (entidade independente)
4. 'Deslocamento' (depende de 'Viagem')
5. 'Paragem' (depende de 'Viagem')
6. 'Foto' (depende de 'Paragem')
7. 'Rel_Sócio_Viagem' (depende de 'Sócio' e de 'Viagem')
8. 'Patrocinador' (entidade independente)
9. 'Rel_Viagem_Patrocinador' (depende de 'Viagem' e de 'Patrocinador')
10. 'Motivo' (depende de 'Rel_Viagem_Patrocinador')
11. 'Rel_Utilizador_Viagem' (depende de 'Utilizador' e 'Viagem')

O método escolhido consiste na criação e execução de procedimentos armazenados. Esta abordagem apresenta várias vantagens:

- **Persistência –** Uma vez criados, os procedimentos mantêm-se armazenados no servidor de base de dados e permanecem acessíveis a todos os utilizadores com permissões de execução.

- **Reutilização –** Elimina a necessidade de desenvolver instruções SQL específicas para cada operação de inserção. Para adicionar novos registos, basta executar o procedimento correspondente através do comando `CALL`.
- **Manutenção Centralizada –** As regras de inserção e validação são definidas uma única vez, facilitando atualizações futuras.
- **Segurança –** Permite controlar o acesso aos dados através de permissões granulares sobre os procedimentos, conforme os requisitos de controlo analisados na etapa anterior.

Cada procedimento recebe como parâmetros de entrada os valores a registar e, após realizar as validações necessárias, executa a operação `INSERT` correspondente. O funcionamento detalhado de cada procedimento será apresentado na secção 5.8.

A tabela seguinte resume os registos realizados nas entidades independentes do sistema:

Relação	Número de registos	Método de obtenção dos dados
Utilizador	20	Selecionaram-se 10 sócios da organização (incluindo os membros do CCP) e 10 voluntários não sócios.
Viagem	5	Selecionou-se uma viagem realizada por cada desenvolvedor e uma viagem coletiva da organização.
Patrocinador	3	Selecionou-se o patrocinador da viagem coletiva e dois patrocinadores históricos do CeSIUM.

Tabela 28 - Metodologia de povoamento da base de dados.

Para garantir uma cobertura completa dos testes, foram também inseridos:

- Os 10 sócios selecionados (tabela 'Sócio')
- Relacionamentos entre entidades (tabelas 'Rel_X_Y')
- Tipos de deslocamento associados a cada viagem (tabela 'Deslocamento')
- Fotografias das paragens (tabela 'Foto')
- Motivos dos patrocínios (tabela 'Motivo')

Os registos do povoamento encontram-se no anexo V.

5.4. Cálculo do Espaço da Base de Dados (Inicial e Taxa de Crescimento Anual)

O planeamento da capacidade de armazenamento assume um papel preponderante na fase de implementação física de qualquer sistema de gestão de bases de dados. Esta etapa ultrapassa a mera formalidade técnica, constituindo um requisito essencial para garantir a sustentabilidade operacional da plataforma a longo prazo.

Dado que o sistema será alojado na infraestrutura do CAOS, onde os recursos computacionais são partilhados e finitos, a equipa de desenvolvimento adotou uma postura de rigor extremo na estimativa dos requisitos de espaço em disco.

A metodologia de cálculo apresentada neste capítulo não se limita a uma aritmética simples dos dados brutos. Pelo contrário, baseia-se numa análise da arquitetura interna do motor de armazenamento InnoDB, o *storage engine* selecionado para o MySQL 8.0, devido à sua conformidade com as propriedades ACID e ao suporte robusto para integridade referencial.

Para fundamentar os cálculos subsequentes, foi necessário estabelecer um mapeamento preciso entre os tipos de dados do modelo lógico e a sua representação física no MySQL/InnoDB. A escolha do charset utf8mb4 (UTF-8) para todos os campos textuais é uma decisão estratégica para suportar a internacionalização e o uso de emojis nos comentários das viagens, mas acarreta implicações significativas no armazenamento, uma vez que cada carácter pode ocupar até 4 bytes.

A tabela 28 sistematiza as regras de consumo de espaço adotadas para este estudo, servindo de referência transversal para todas as tabelas de cálculo.

Tipo de Dados (Lógico)	Tipo Físico (MySQL)	Espaço Fixo (Bytes)	Espaço Variável (Fórmula)	Observações Técnicas e Impacto no Modelo
Identificador / Chave	INT (Signed/Unsigned)	4	0	Utilizado para a identificação de registos e estabelecimento de ligações entre tabelas (Chaves Primárias e Estrangeiras), este formato representa o padrão e oferece a maior eficiência de processamento.
Texto Curto / Longo	VARCHAR(N)	0	L x C + H	L = comprimento da string; C = média de bytes por carácter (estimado em 1,5 para PT/EN); H = cabeçalho do comprimento (1 byte se $N < 255$, 2 bytes se $N > 255$).
Enumeração	ENUM	1	0	Em vez de guardar a palavra toda, guarda apenas um número. Usado em estados e categorias fixas.
Data e Hora	DATETIME	5	0	Guarda a data e hora exata num formato comprimido. Não guardamos milissegundos para poupar espaço.
Monetário	DECIMAL(8,2)	4	0	Específico para dinheiro. Garante que não há erros de arredondamento e ocupa sempre 4 bytes fixos.
Cabeçalho de Registo	Row Header	5	0	Informação administrativa que o sistema adiciona automaticamente para gerir a linha (ex: saber se foi apagada).
ID de Transação	TRX_ID	6	0	Registar o utilizador responsável pela última modificação de um dado é crucial, especialmente em ambientes com múltiplos utilizadores a aceder simultaneamente.
Ponteiro de Rollback	ROLL_PTR	7	0	Funciona como um botão de "Desfazer". Permite ao sistema recuperar o valor antigo se uma operação der erro.
Bitmap de Nulidade	Null Bitmap	N_{null} / 8	0	Um mapa de bits compacto que identifica os campos com valores nulos (NULL), otimizando assim o tempo de processamento do sistema ao evitar a leitura desnecessária desses campos.

Tabela 29 - Regras de consumo de espaço adotadas.

A seguir, procede-se à decomposição da estrutura de cada uma das tabelas definidas na secção 5.1. , com o objetivo de calcular o espaço ocupado por registo. Esta análise exaustiva é fundamental para a identificação das tabelas que exercem maior influência no crescimento da base de dados.

A tabela 'Utilizador' representa a entidade fundamental para a gestão de identidades. Como superclasse das entidades 'Sócio', antecipa-se um volume de dados moderado, porém com crescimento sustentado.

Atributo	Tipo de dados	Tamanho Teórico (Bytes)
Id	INT	4
Nome	VARCHAR(100)	31
Telemóvel	VARCHAR(12)	13
Email	VARCHAR(45)	26
Pass_word	VARCHAR (100)	60
Total por Registo	-	152

Tabela 30 - Espaço ocupado pelos atributos de cada registo da relação 'Utilizador'.

A tabela 'Sócio' estabelece uma relação de um para um com a tabela 'Utilizador'. Embora a implementação física em tabelas distintas implique um ligeiro consumo adicional de recursos, resultante da duplicação de chaves e cabeçalhos, esta abordagem é justificada pela normalização e clareza que proporciona ao modelo de dados.

Atributo	Tipo de dados	Tamanho Teórico (Bytes)
Nr_Sócio	INT	4
Utilizador	INT	4
Estatuto	ENUM	1
Foto_Perfil	VARCHAR(100)	41
Null Bitmap	System	1
Total por Registo	-	69 Bytes

Tabela 31 - Espaço ocupado pelos atributos de cada registo da relação 'Sócio'.

A entidade 'Viagem' desempenha um papel central no sistema, consolidando a maioria das relações. O seu tamanho é relativamente reduzido, dado que a complexidade inerente, incluindo paragens e participantes, foi externalizada para outras tabelas.

Atributo	Tipo de dados	Tamanho Teórico (Bytes)
ID	INT	4
Datas (Início/Fim)	DATETIME	10
Objetivo	ENUM	1
Custo	DECIMAL(8,2)	4
Contadores	INT	8
Total por Registo	-	45 Bytes

Tabela 32 - Espaço ocupado pelos atributos de cada registo da relação 'Viagem'.

A Paragem, enquanto entidade fraca, detém uma chave composta que incorpora a chave da Viagem.

Atributo	Tipo de dados	Tamanho Teórico (Bytes)
PK Composta	INT x2	8
Localização	VARCHAR x2	32
Datas	DATETIME x2	10
Total por Registo	-	68 Bytes

Tabela 33 - Espaço ocupado pelos atributos de cada registo da relação 'Paragem'.

A normalização para a Primeira Forma Normal (1FN) determinou a criação de tabelas distintas para a gestão de atributos multivalorados. Esta abordagem resulta, frequentemente, numa proporção desfavorável entre a estrutura de controlo e os dados úteis, uma vez que a base de dados impõe um custo fixo de gestão de 18 bytes a cada registo, independentemente da sua dimensão.

Este impacto na eficiência de armazenamento é particularmente evidente na relação 'Deslocamento', onde os dados úteis (a referência da viagem e o tipo de deslocamento) ocupam apenas 5 bytes, mas o registo final consome 23 bytes devido ao peso da estrutura interna. O padrão repete-se na tabela de Fotos das paragens: considerando as chaves e um caminho de ficheiro médio (cerca de 58 bytes de dados mais um byte de controlo), o tamanho total atinge os 77 bytes. Mesmo na tabela relativa aos Motivos de Patrocínio, que armazena descrições mais extensas totalizando cerca de 88 bytes de informação efetiva, o espaço final ocupado em disco ascende aos 106 bytes por força dos requisitos administrativos do sistema.

Relação	Tamanho Médio (Bytes)
Deslocamento	23
Foto	77
Motivo	106

Tabela 34 - Resumo do espaço ocupado pelas tabelas dos atributos multivalor.

Estas tabelas constituem o "tecido social" da aplicação, apresentando um padrão de crescimento que não é meramente linear face ao número de viagens, mas sim polinomial ou exponencial, impulsionado pela intensidade da interação dos utilizadores.

A tabela de publicações 'Rel_Sócio_Viagem' combina as chaves de identificação com a avaliação e um campo de comentários opcional. Para este último, assume-se uma média ponderada de 33 bytes (considerando uma taxa de preenchimento de 40% com comentários de 80 caracteres). Somando o byte de gestão de valores nulos e o custo fixo de estrutura, o tamanho estimado por registo é de 61 bytes.

Por sua vez, a tabela de visualizações 'Rel_Utilizador_Viagem', que se prevê ser a mais volumosa em quantidade de registos, é estruturalmente mais leve. Ao armazenar as chaves de ligação, a reação do utilizador e o indicador de nulos, cada interação ocupa 28 bytes, já incluído o consumo de recursos de gestão. Finalmente, a relação de patrocínios 'Rel_Viagem_Patrocinador', que regista a ligação à viagem e o respetivo valor monetário, totaliza 30 bytes por registo, somando os dados úteis à estrutura de controlo.

Relação	Tamanho (Bytes)
Rel_Sócio_Viagem	61
Rel_Utilizador_Viagem	28
Rel_Viagem_Patrocinador	30

Tabela 35 - Resumo do espaço pelas tabelas dos relacionamentos N:M.

Os valores anteriormente calculados referem-se exclusivamente ao volume dos dados "em bruto", correspondente à sua organização primária. Contudo, uma análise de capacidade rigorosa deve considerar os custos operacionais, os quais, na prática, incrementam substancialmente o espaço necessário em disco. Face a esta questão, a equipa identificou dois fatores críticos que funcionam como multiplicadores do armazenamento.

O primeiro fator consiste nos Índices Secundários, que operam como "atalhos" de pesquisa. Dado o modelo de dados apresentar múltiplas ligações entre tabelas (Chaves Estrangeiras), o sistema gera automaticamente estruturas auxiliares para gerir essas relações. Estes índices requerem espaço adicional, uma vez que necessitam de duplicar certas referências para o seu funcionamento, estimando-se que incrementem entre 30% a 50% o volume inicial das tabelas.

A fragmentação é outro fator a considerar. A base de dados armazena a informação em blocos de memória que raramente atingem a capacidade máxima. Dada a natureza aleatória das interações dos utilizadores, estima-se que cerca de 25% do espaço reservado em disco permaneça vazio (espaço entre registos). Para compensar estes dois fenómenos, a equipa aplicou um fator de segurança de 1,5 sobre os cálculos teóricos, tendo em conta tanto o impacto dos índices como o desperdício devido à fragmentação. Esta abordagem garante uma estimativa de hardware mais realista.

No âmbito do lançamento da plataforma, o CeSIUM pretende proceder à importação de um conjunto de dados históricos, de forma a assegurar que o sistema não inicie as suas operações em estado de vazio. Esta medida visa incentivar a adesão imediata dos utilizadores. As estimativas de volume para este povoamento são as seguintes:

Categoria	Dados
Utilizadores	1.000 (importação de base de dados de antigos alunos + estudantes atuais).
Sócios	400 (membros ativos e antigos alunos-sócios).
Viagens	50 (histórico de atividades da última década).
Paragens	Média de 4 por viagem (Total: 200).
Multimédia	500 referências de fotos e 100 registos de deslocação.
Interações	2.000 registos de participação (sócios) e visualização (utilizadores), refletindo o interesse inicial.

Tabela 36 - Estimativa do volume de dados iniciais para a plataforma.

A tabela 36 projeta o espaço ocupado no momento do lançamento da plataforma.

Relação	Tamanho do registo (Bytes)	Registos	Tamanho dos dados (KB)	Espaço em Disco
Utilizador	152	1.000	148.4 KB	224 KB (14 págs)
Sócio	69	400	27 KB	48 KB (3 págs)
Viagem	45	50	2.2 KB	16 KB (1 pág)
Paragem	68	200	13.3 KB	32 KB (2 págs)
Patrocinador	38	20	0.7 KB	16 KB (1 pág)

Deslocamento	23	100	2.2 KB	16 KB (1 pág)
Foto	77	500	37.6 KB	64 KB (4 págs)
Motivo	106	40	4.1 KB	16 KB (1 pág)
Rel_Sócio_Viagem	61	2.000	119 KB	192 KB (12 págs)
Rel_Utilizador_Viagem	28	2.000	54.7 KB	96 KB (6 págs)
Rel_Viagem_Patrocinador	30	30	0.9 KB	16 KB (1 pág)
TOTAL	-	6.340	351 KB	656 KB

Tabela 37 - Estimativa do tamanho ocupado por cada relação (ano 0).

A análise preliminar revela que o volume inicial da base de dados é extremamente reduzido, ocupando aproximadamente 0,6 MB em disco. Este dado valida inequivocamente a opção arquitetural de armazenar apenas as referências das imagens, delegando os ficheiros binários (BLOBs) para um sistema de armazenamento externo. Caso contrário, a inclusão direta das imagens na base de dados elevaria o consumo de espaço numa magnitude de pelo menos mil vezes, transformando uma estrutura de Kilobytes numa de Gigabytes.

No entanto, o planeamento de capacidade exige uma perspetiva de longo prazo. Reconhecendo que a evolução de uma rede social académica não segue uma trajetória linear, a equipa modelou as projeções futuras com base em taxas de crescimento compostas. O modelo contempla um crescimento orgânico de utilizadores na ordem dos 20% ao ano, impulsionado por novas matrículas, a par de um incremento constante na atividade de 10 novas viagens anuais. O vetor mais significativo reside no crescimento viral das interações, estimado em 40% ao ano: à medida que a base de utilizadores e o arquivo histórico se expandem, o volume de visualizações e reações tende a crescer exponencialmente, dado que os novos utilizadores interagem também com o conjunto de conteúdos passados.

Ano	Fator de Crescimento (Utilizadores)	Fator de Crescimento (Interações)	Tamanho Estimado (Dados + Índices)	Espaço em Disco (Alocação Recomendada
Ano 0	Base (1.000)	Base (2.000)	0.53 MB	5 MB
Ano 1	+20%	+40%	0.85 MB	5 MB
Ano 2	+20%	+40%	1.4 MB	10 MB
Ano 3	+20%	+40%	2.3 MB	10 MB
Ano 4	+20%	+40%	4.1 MB	20 MB
Ano 5	+20%	+40%	7.5 MB	20 MB

Tabela 38 - Evolução do tamanho da BD BeLIUM_Viagens (Projeção a 5 anos).

Mesmo considerando um cenário de crescimento acentuado ao atingir o quinto ano de operação, projeta-se que o volume total da base de dados relacional permaneça inferior a 10 MB. Este indicador demonstra a elevada eficiência e escalabilidade do modelo de dados, sugerindo que a infraestrutura atual do CAOS, apesar das suas limitações de hardware, possuirá capacidade para suportar o componente SQL do sistema durante a próxima década sem necessidade de expansão de armazenamento.

É imperativo, contudo, analisar a discrepância natural entre os cálculos teóricos e a ocupação real observada em ambiente de produção (frequentemente reportada por comandos como `SHOW TABLE STATUS`).

5.5. Definição e Caracterização de Vistas de Utilização em SQL

O uso de vistas (*Views*) em *SQL* constitui um mecanismo fundamental para a gestão de acesso aos dados e simplificação de interrogações. Tecnicamente, uma vista comporta-se como uma tabela virtual, cujos dados não são armazenados fisicamente de forma duplicada, mas sim derivados dinamicamente de tabelas base ou de outras vistas através de uma consulta *SQL* pré-definida.

A diretiva `CREATE VIEW` permite a definição destas estruturas, que oferecem duas grandes vantagens num sistema de informação:

1. **Segurança e Controlo de Acesso –** Permitem restringir o acesso a colunas específicas (projeções) ou a registos que cumpram determinada condição (seleções), sem necessidade de conceder permissões diretas sobre as tabelas subjacentes.
2. **Abstração e Simplificação –** Permitem encapsular a complexidade como junções (*joins*), agregações e subconsultas, oferecendo aos utilizadores finais ou à camada de aplicação uma interface de dados mais intuitiva e com menor complexidade sintática.

Contudo, as vistas apresentam algumas limitações, sendo a principal a incapacidade de receber parâmetros diretamente. Esta restrição torna-as inadequadas para requisitos que necessitem de filtragem dinâmica baseada em valores fornecidos em tempo de execução.

Apesar desta limitação, três requisitos específicos do sistema – **RM11**, **RM17** e **RM19** – distinguem-se por não necessitarem de parâmetros de entrada, sendo, portanto, passíveis de implementação através de vistas, tal como se detalha de seguida.

```
CREATE VIEW VisualizarInvestimentoTotal AS
     SELECT P.ID, P.Nome, SUM(RVP.Valor) AS Investimento_Total
     FROM Patrocinador AS P
     LEFT OUTER JOIN Rel_Viagem_Patrocinador AS RVP
          ON P.ID = RVP.Patrocinador
     GROUP BY P.ID, P.Nome;

CREATE VIEW VisualizarNrPublicações AS
     SELECT S.Nr_Sócio, U.Nome, COUNT(RSV.Viagem) AS
     Nr_Viagens_Publicadas FROM Utilizador AS U
     INNER JOIN Sócio AS S ON U.ID = S.Utilizador
```

```
        LEFT OUTER JOIN Rel_Sócio_Viagem AS RSV
            ON S.Utilizador = RSV.Sócio
        GROUP BY S.Nr_Sócio, U.Nome
        ORDER BY Nr_Viagens_Publicadas DESC;

    CREATE VIEW VisualizarViagensPopulares AS
        SELECT V.ID, V.Data_Início, V.Data_Término, V.Nr_Participantes,
        V.Custo, V.Objetivo, COUNT(RUV.Viagem) AS Nr_Visualizações FROM
        Viagem AS V
        LEFT OUTER JOIN Rel_Utilizador_Viagem AS RUV
            ON V.ID = RUV.Viagem
        GROUP BY V.ID, V.Data_Início, V.Data_Término, V.Nr_Participantes,
        V.Custo, V.Objetivo
        ORDER BY Nr_Visualizações DESC
        LIMIT 10;
```

A instrução `CREATE VIEW` cria cada vista, sendo a sua lógica definida pela consulta `SELECT` que segue o operador `AS`. A partir do momento da criação, cada vista pode ser consultada como uma tabela virtual, fornecendo sempre resultados atualizados relativamente aos dados subjacentes.

A explicação detalhada da construção de cada uma das consultas encontra-se na secção 5.6.

A implementação de vistas, complementada com um sistema rigoroso de controlo de acessos, permitiu satisfazer os requisitos **RM11**, **RM17** e **RM19** de forma segura e eficiente.

Esta abordagem oferece:

- Abstração da complexidade – as consultas complexas ficam encapsuladas nas vistas
- Segurança dos dados – restrições de acesso conforme a sensibilidade da informação
- Desempenho otimizado – reutilização de consultas pré-compiladas

O sistema assegura assim que cada tipo de utilizador acede apenas à informação necessária para as suas funções, mantendo a confidencialidade dos dados financeiros enquanto disponibiliza estatísticas úteis à comunidade. Esta arquitetura estabelece uma base sólida para futuras expansões do sistema, permitindo a fácil adição de novas vistas e permissões conforme as necessidades evolutivas da organização.

5.6. Tradução das Interrogações do Utilizador para SQL

Uma vez definidas as consultas em álgebra relacional para satisfazer os requisitos de manipulação da base de dados, a transição para SQL é, na maioria dos casos, quase imediata. Trata-se essencialmente de

uma adaptação sintática, que preserva intacto o significado das operações originais. Contudo, enquanto a calculadora RelaX – utilizada para testar expressões de álgebra relacional – não suporta a parametrização dessas mesmas expressões, o SQL oferece uma flexibilidade adicional. Esta linguagem permite criar instruções parametrizadas, que podem ser pré-preparadas e ter os valores dos parâmetros substituídos apenas no momento da execução. Assim, nesta secção serão apresentadas as instruções SQL correspondentes aos requisitos de manipulação identificados.

Primeiramente, foram identificados os melhores métodos para a satisfação de cada interrogação. Os critérios utilizados foram:

1. Interrogações simples e com número reduzidos de parâmetros foram implementadas através de instruções preparadas (*prepared statements*).
2. Interrogações que não necessitam de parâmetros foram implementadas com vistas.
3. Interrogações complexas foram implementadas com procedimentos.

Adicionalmente, importa salientar que, para algumas interrogações, foram utilizadas funções de apoio aos cálculos. A explicação detalhada dos procedimentos e das funções encontra-se na secção 5.8. 5.8.

A interrogação SQL correspondente ao requisito **RM3** realiza a listagem de todas as paragens de uma determinada viagem. O identificador da viagem na cláusula `WHERE` é representado por um ponto de interrogação, que será substituído por um valor específico no momento da execução. A interrogação SQL projeta o número da paragem, o país, a cidade e as datas, ordenando o resultado cronologicamente pela data de chegada.

```
PREPARE RM3 FROM
'SELECT Nr, País, Cidade, Data_Chegada, Data_Partida
    FROM Paragem
    WHERE Viagem = ?
    ORDER BY Data_Chegada ASC;';

 SET @ViagemRM3 = 1;
 EXECUTE RM3 USING @ViagemRM3;
 DEALLOCATE PREPARE RM3;
```

É importante salientar que as instruções `SET`, `EXECUTE` e `DEALLOCATE` foram definidas apenas para exemplificar como é que as instruções preparadas são utilizadas e, quando não são necessárias, libertadas (`DEALLOCATE`).

Em seguida, o requisito **RM4** consiste em listar todos os sócios que realizaram uma determinada viagem. A interrogação recorre a junções (`INNER JOIN`) entre as relações 'Sócio', 'Rel_Sócio_Viagem' e 'Utilizador' para reunir o nome, número de sócio e estatuto. A condição `WHERE RSV.Viagem = ?` aplica o filtro dinâmico, e os resultados são apresentados por ordem alfabética do nome do utilizador.

```
PREPARE RM4 FROM
'SELECT U.Nome, S.Nr_Sócio, S.Estatuto FROM Sócio AS S
     INNER JOIN Rel_Sócio_Viagem AS RSV ON RSV.Sócio = S.Utilizador
     INNER JOIN Utilizador AS U ON U.ID = S.Utilizador
          WHERE RSV.Viagem = ?
          ORDER BY U.Nome ASC;';

SET @ViagemRM4 = 1;
EXECUTE RM4 USING @ViagemRM4;
DEALLOCATE PREPARE RM4;
```

Relativamente ao requisito **RM6**, cujo objetivo é obter a lista de patrocinadores de uma viagem e o respetivo valor investido, a interrogação projeta o identificador, o nome e o valor. A junção com a tabela 'Rel_Viagem_Patrocinador' permite associar o investimento ao patrocinador, filtrando pela viagem desejada. A ordenação é feita primeiramente pelo valor (crescente) e, em caso de empate, pelo nome do patrocinador.

```
PREPARE RM6 FROM
'SELECT P.ID, P.Nome, RVP.Valor FROM Patrocinador AS P
     INNER JOIN Rel_Viagem_Patrocinador AS RVP
     ON P.ID = RVP.Patrocinador
           WHERE RVP.Viagem = ?
           ORDER BY RVP.Valor ASC, P.Nome ASC;';

SET @ViagemRM6 = 1;
EXECUTE RM6 USING @ViagemRM6;
DEALLOCATE PREPARE RM6;
```

O requisito **RM8** introduz uma lógica de filtragem mais complexa e opcional. O objetivo é filtrar viagens pela localização ('País' e/ou 'Cidade'), mas permitindo que qualquer um dos parâmetros seja nulo. A cláusula `WHERE (? IS NULL OR P.País = ?)` assegura que, se o parâmetro for `NULL`, a condição é ignorada, devolvendo todas as ocorrências; caso contrário, filtra pelo valor especificado. A palavra-chave `DISTINCT` é utilizada para evitar duplicação de viagens nos resultados.

```
SELECT DISTINCT V.* FROM Viagem AS V
     INNER JOIN Paragem AS P ON V.ID = P.Viagem
          WHERE (? IS NULL OR P.País = ?) AND
                (? IS NULL OR P.Cidade = ?);
```

De forma análoga, o requisito **RM9** aplica a lógica de filtros opcionais a múltiplos critérios: datas, custo e objetivo. A consulta verifica intervalos de datas e custos, bem como a igualdade do objetivo. Cada

condição segue o padrão (`? is NULL OR Coluna Operador ?`), permitindo ao utilizador preencher apenas os critérios que deseja restringir, mantendo a consulta flexível e robusta.

```
SELECT * FROM Viagem
     WHERE (? IS NULL OR Data_Início >= ?) AND
           (? IS NULL OR Data_Término <= ?) AND
           (? IS NULL OR Custo >= ?) AND
           (? IS NULL OR Custo <= ?) AND
           (? IS NULL OR Objetivo = ?);
```

O requisito **RM11** solicita o cálculo do investimento total de cada patrocinador (com o auxílio da função 'CalcularInvestimentoTotal' – ver secção 5.8.). Utiliza-se uma `LEFT OUTER JOIN` para garantir que mesmo patrocinadores sem investimentos registados (caso existam) sejam listados (com valor nulo ou zero). A função de agregação `SUM(RVP.Valor)` realiza o somatório, agrupado pelo identificador do patrocinador.

```
SELECT P.ID, P.Nome, CalcularInvestimentoTotal(P.ID)
AS Investimento_Total
FROM Patrocinador AS P
LEFT OUTER JOIN Rel_Viagem_Patrocinador AS RVP
     ON P.ID = RVP.Patrocinador
GROUP BY P.ID, P.Nome;
```

Para o requisito **RM12**, calcula-se a média de investimento de um patrocinador específico (com o auxílio da função 'CalcularInvestimentoMédio' – ver secção 5.8.). A função `AVG(Valor)` é aplicada sobre a relação 'Rel_Viagem_Patrocinador', filtrando-se pelo identificador através da cláusula `WHERE ID = ?`, e o resultado é arredondado a duas casas decimais com `ROUND`.

```
PREPARE RM12 FROM
'SELECT CalcularInvestimentoMédio(P.ID) AS Investimento_Médio
FROM Rel_Viagem_Patrocinador
          WHERE Patrocinador = ?;';



SET @IdRM12 = 1;
EXECUTE RM12 USING @IdRM12;
DEALLOCATE PREPARE RM12;
```

O requisito **RM15** lista as viagens realizadas por um determinado sócio. A consulta efetua junções entre 'Viagem', 'Rel_Sócio_Viagem' e 'Sócio' para conectar as entidades, filtrando pelo 'Nr_Sócio' fornecido. A ordenação decrescente por 'Data_Início' permite visualizar primeiro as viagens mais recentes.

```
PREPARE RM15 FROM
'SELECT V.* FROM Viagem AS V
      INNER JOIN Rel_Sócio_Viagem AS RSV ON V.ID = RSV.Viagem
      INNER JOIN Sócio AS S ON S.Utilizador = RSV.Sócio
           WHERE S.Nr_Sócio = ?
           ORDER BY V.Data_Início DESC;';

SET @SocioRM15 = 123;
EXECUTE RM15 USING @SocioRM15;
DEALLOCATE PREPARE RM15;
```

No requisito **RM16**, consulta-se o motivo de um patrocínio específico. A cláusula `WHERE` exige dois parâmetros: o identificador da viagem e o identificador do patrocinador, garantindo que se obtém o motivo exato dessa relação específica.

```
PREPARE RM16 FROM
'SELECT Motivo FROM Motivo
          WHERE Viagem = ? AND Patrocinador = ?;';

SET @ViagemRM16 = 1;
SET @PatrocinadorRM16 = 5;
EXECUTE RM16 USING @ViagemRM16, @PatrocinadorRM16;
DEALLOCATE PREPARE RM16;
```

O requisito **RM17** contabiliza o número de viagens publicadas por cada sócio. Recorre-se a uma `LEFT OUTER JOIN` entre 'Sócio' e 'Rel_Sócio_Viagem' para assegurar que sócios sem publicações também aparecem na listagem (com contagem zero). O agrupamento é feito pelos dados do sócio e a ordenação é decrescente pelo número de publicações.

```
SELECT S.Nr_Sócio, U.Nome, COUNT(RSV.Viagem) AS
Nr_Viagens_Publicadas FROM Utilizador AS U
     INNER JOIN Sócio AS S ON U.ID = S.Utilizador
     LEFT OUTER JOIN Rel_Sócio_Viagem AS RSV
     ON S.Utilizador = RSV.Sócio
          GROUP BY S.Nr_Sócio, U.Nome
          ORDER BY Nr_Viagens_Publicadas DESC;
```

O requisito **RM18** apresenta uma complexidade superior, pois requer a filtragem de viagens baseada na percentagem de reações positivas. Para tal, utiliza-se uma subconsulta (tabela derivada) que invoca a função armazenada 'CalcularPercentagemPositiva (Viagem)'. Esta subconsulta agrupa os resultados por viagem e calcula a respetiva percentagem, sendo depois feito o join com a tabela 'Viagem' para filtrar apenas aquelas cuja percentagem seja superior ou igual ao parâmetro fornecido.

```
SELECT V.*, P.Percentagem FROM Viagem AS V
     INNER JOIN (
          SELECT Viagem, CalcularPercentagemPositiva(Viagem) AS
          Percentagem FROM Rel_Utilizador_Viagem AS RUV
               GROUP BY Viagem
          ) AS P ON V.ID = P.Viagem
               WHERE P.Percentagem >= ?
               ORDER BY P.Percentagem DESC;
```

O requisito **RM19** identifica as viagens mais populares (com mais visualizações). A consulta utiliza `COUNT()` sobre a relação 'Rel_Utilizador_Viagem'. O uso de `LEFT OUTER JOIN` é crucial para manter no sistema viagens que ainda não tenham visualizações. O resultado é ordenado de forma decrescente pela contagem.

```
SELECT V.*, COUNT(RUV.Viagem) AS Nr_Visualizações
FROM Viagem AS V
     LEFT OUTER JOIN Rel_Utilizador_Viagem AS RUV
     ON V.ID = RUV.Viagem
          GROUP BY V.ID
          ORDER BY Nr_Visualizações DESC;
```

Por fim, o requisito **RM20** permite consultar o feedback (avaliação e comentário) deixado pelos sócios numa viagem. Através de junções sucessivas, obtém-se o número do sócio autor da crítica e os detalhes da sua avaliação para a viagem parametrizada.

```
PREPARE RM20 FROM
'SELECT S.Nr_Sócio, RSV.Avaliação, RSV.Comentário
FROM Rel_Sócio_Viagem AS RSV
     INNER JOIN Sócio AS S ON S.Utilizador = RSV.Sócio
     WHERE Viagem = ?;';

SET @ViagemRM20 = 2;
EXECUTE RM20 USING @ViagemRM20;
DEALLOCATE PREPARE RM20;
```

Em suma, a tradução das expressões da álgebra relacional para a linguagem SQL, concretizada através das instruções apresentadas, permitiu cobrir a grande maioria dos requisitos de manipulação de dados do sistema BeLIUM Viagens. A utilização recorrente de instruções preparadas (`PREPARE` / `EXECUTE`) assegurou que as interrogações são não apenas funcionais, mas também flexíveis, permitindo a introdução de valores em tempo de execução sem comprometer a estrutura das consultas.

5.7. Indexação do Sistema de Dados

Um índice é uma estrutura de dados que organiza os valores de um ou mais atributos de uma tabela, permitindo localizar registos específicos sem necessidade de percorrer toda a relação. Quando corretamente utilizados, os índices podem melhorar significativamente o desempenho de uma base de dados. Esta otimização é particularmente útil em operações de filtragem, ordenação e junção, como as requeridas pelos requisitos **RM3** (listar paragens por viagem), **RM8** (filtrar viagens por localização) e **RM9** (filtrar por datas, custo e objetivo).

No entanto, a utilização de índices apresenta alguns inconvenientes: as operações de inserção, atualização e eliminação tornam-se mais lentas, pois exigem a manutenção das estruturas indexadas. Adicionalmente, os índices consomem espaço de armazenamento, o que pode ser crítico em bases de dados de grande dimensão.

Face ao reduzido volume de dados em BeLIUM_Viagens na fase inicial, a equipa de desenvolvimento optou por não criar índices, uma vez que todas as interrogações são executadas em frações de segundo. Contudo, foram desenvolvidos e documentados *scripts* de criação de índices para os atributos mais frequentemente utilizados nas consultas. Estes índices poderão ser ativados futuramente se o desempenho diminuir com o crescimento da base de dados.

O sistema de gestão de base de dados MySQL, com o motor de armazenamento InnoDB, cria automaticamente índices para garantir a integridade referencial e otimizar o desempenho das operações mais comuns. Em particular, os índices associados às chaves estrangeiras permitem acelerar tanto as verificações de integridade como as operações de `JOIN`.

Como tal, todos os atributos que seguem os seguintes critérios são indexados:

- Conjuntos de atributos que definem uma chave primária.
- Conjuntos de atributos que definem uma chave estrangeira.
- Atributos com a restrição `UNIQUE`.

Com base na análise dos inquéritos realizados à comunidade e na avaliação da frequência prevista de execução de cada interrogação, a equipa de desenvolvimento identificou os atributos que seriam mais beneficiados pela criação de índices adicionais, a serem implementados apenas se necessário devido ao crescimento do volume de dados. Esta decisão fundamenta-se no princípio de que, apesar de todos os requisitos funcionarem eficientemente com o volume atual de dados, alguns cenários de consulta poderão degradar-se com o aumento do número de registos.

Requisito	Descrição	Relação	Atributos Indexados
RM3	Data de chegada na paragem	Paragem	Data_Chegada
RM8	Localização das paragens	Paragem	País, Cidade
RM9	Detalhes das viagens	Viagem	Data_Início, Data_Término, Custo, Objetivo
RM17	Reações às viagens	Rel_Utilizador_Viagem	Reação

Tabela 39 - Descrição dos conjuntos de atributos a indexar.

Embora a chave primária composta (Nr, Viagem) já otimize a localização por viagem, a ordenação cronológica requerida por **RM3** implica o processamento completo das paragens. Um índice em 'Data_Chegada' permitirá ordenações eficientes sem recorrer a operações de *filesort*, otimizando significativamente a interrogação.

Em relação ao requisito **RM8**, as consultas de localização envolvem filtros sobre dois atributos textuais. Um índice composto nestas colunas acelerará significativamente as pesquisas por país e/ou cidade, especialmente quando combinadas.

Do mesmo modo, dada a natureza opcional e combinável dos filtros na interrogação do **RM9**, índices separados oferecem maior flexibilidade. O otimizador do MySQL poderá utilizar o índice mais seletivo conforme os parâmetros fornecidos em tempo de execução.

Finalmente, o requisito **RM17** implica o cálculo das percentagens de reações positivas de cada viagem, sendo necessário verificar se são reações positivas ou negativas. Este índice facilitará o cálculo destas estatísticas.

Para os restantes requisitos, a equipa deliberadamente não planeou índices adicionais, pelas seguintes razões:

- **RM4**, **RM6**, **RM15**, **RM20**: Estas interrogações SQL beneficiam integralmente dos índices automáticos criados pelas chaves primárias e estrangeiras. As operações de `JOIN` e filtragem já são otimizadas pelos índices existentes.
- **RM11**, **RM12**, **RM19**: As operações de `SUM()`, `AVG()` e `COUNT()` sobre as tabelas de junção utilizam eficientemente os índices de chave estrangeira. A criação de índices adicionais traria benefícios reduzidos para o volume de dados esperados.
- **RM6**, **RM11**, **RM12**, **RM13**, **RM16**: Considerando que o número de patrocinadores será reduzido (não se antecipam mais de algumas dezenas), a varredura sequencial destas tabelas permanecerá eficiente mesmo com o crescimento significativo do sistema.
- **RM18**: A complexidade desta interrogação (com *subquery* e cálculo percentual) tornaria qualquer índice de benefício limitado. A otimização deste requisito, se necessária, seria melhor alcançada através de materialização de resultados ou revisão do modelo de dados.

As instruções para a criação destes índices encontram-se a seguir:

```
CREATE INDEX IDX_Paragem_Data_Chegada
    ON Paragem (Data_Chegada);

CREATE INDEX IDX_Paragem_Localização
    ON Paragem (País, Cidade);

CREATE INDEX IDX_Viagem_Detalhes
ON Viagem (Data_Início, Data_Término, Custo, Objetivo);

CREATE INDEX IDX_Rel_Utilizador_Viagem_Reação
    ON Rel_Utilizador_Viagem (Reação);
```

A cláusula `CREATE INDEX` é responsável pela criação do índice. Após a sua nomeação, é indicada a tabela na qual o índice deve ser criado, assim como os atributos indexados.

A sua ativação seguirá um processo faseado:

- Monitorização Contínua: O desempenho das *queries* será monitorizado periodicamente, com especial atenção aos tempos de execução das operações relacionadas com os requisitos RM3, RM8, RM9 e RM17.
- Critérios de Ativação: Cada índice será implementado apenas quando o tempo médio de execução da *query* exceder 500 ms e o número de registos na tabela em questão exceder os 10.000.
- Testes de impacto: Antes da ativação, cada índice será testado em ambiente controlado para avaliar: a melhoria no tempo de execução das *queries* de leitura, o impacto nas operações de escrita (`INSERT`, `UPDATE`, `DELETE`) e o aumento do consumo de armazenamento.

Esta abordagem conservadora garante que o sistema mantém o equilíbrio ideal entre desempenho de leitura e eficiência de escrita, adaptando-se proativamente às necessidades evolutivas da base de dados BeLIUM_Viagens.

5.8. Implementação de procedimentos, funções e gatilhos

O MySQL oferece suporte nativo à definição e execução de procedimentos armazenados, funções e gatilhos (*triggers*), constituindo um ambiente de programação completo dentro do próprio sistema de gestão de bases de dados. Estes mecanismos permitem encapsular lógica de negócio e regras de validação diretamente no servidor de base de dados, reduzindo a complexidade da camada aplicacional e centralizando o controlo de operações críticas.

A principal vantagem destas construções reside na sua execução no servidor de base de dados, o que minimiza o tráfego de rede entre a aplicação e o SGBD, melhora o desempenho através da pré-

compilação, e assegura que as regras de negócio são aplicadas de forma consistente. Esta abordagem segue o princípio de colocar a lógica onde os dados residem, criando uma camada de abstração que simplifica o desenvolvimento aplicacional enquanto reforça a robustez e coerência do sistema de informação.

No entanto, estas operações avançadas são construídas como conjuntos de múltiplas instruções SQL, cada uma tradicionalmente terminada por ponto e vírgula (;). Esta convenção cria uma ambiguidade para o analisador sintático do MySQL, que interpretaria o primeiro ponto e vírgula dentro do corpo da rotina como o término do comando `CREATE` completo, resultando num erro de sintaxe.

Para resolver esta limitação, o MySQL oferece o comando `DELIMITER`, que permite redefinir temporariamente o carácter delimitador de instruções. Este comando instrui o servidor a utilizar uma sequência alternativa (normalmente `$$`) para identificar o final da declaração completa da rotina, enquanto os pontos e vírgulas internos são preservados como parte do corpo da rotina.

A sequência típica de operação é a seguinte:

1. Redefinir o delimitador para uma sequência não convencional.
2. Declarar a rotina completa (procedimento, função ou gatilho), utilizando pontos e vírgula internamente.
3. Encerrar a declaração com o novo delimitador.
4. Restaurar o delimitador padrão.

Esta sequência pode ser traduzida pela seguinte sintaxe SQL:

```
DELIMITER $$
-- | Procedimento, função ou gatilho $$
DELIMITER ;
```

Esta técnica de delimitadores foi aplicada a todas as rotinas definidas no sistema BeLIUM_Viagens, permitindo a criação de procedimentos complexos que encapsulam múltiplas operações SQL num único bloco lógico executado no servidor. Por este motivo, foi omitida nas instruções reveladas a seguir.

No âmbito da automatização de regras de negócio e garantia da integridade dos dados calculados, foi desenvolvido um gatilho (*trigger*) que reage automaticamente a alterações na base de dados. Este mecanismo assegura a consistência entre valores derivados e os dados fundamentais que os originam.

O gatilho, denominado 'AtualizaNrParagens', tem como objetivo específico manter a coerência entre o número real de registos na tabela 'Paragem' e o contador de paragens armazenado como atributo derivado na tabela 'Viagem'. O campo 'Nr_Paragens' representa um exemplo típico de dado calculado cuja exatidão depende da sincronização com operações de manipulação de dados subjacentes. O *trigger* foi implementado da seguinte forma:

```
CREATE TRIGGER AtualizaNrParagens
AFTER INSERT ON Paragem
FOR EACH ROW
BEGIN
     UPDATE Viagem
     SET Nr_Paragens = Nr_Paragens + 1
     WHERE ID = NEW.Viagem;
END
```

O gatilho é automaticamente acionado após cada operação de `INSERT` na tabela 'Paragem' (`AFTER INSERT`). É executado uma vez para cada linha afetada pela operação (`FOR EACH ROW`), garantindo a correta contabilização mesmo em inserções múltiplas, através da atualização (`UPDATE`) do campo 'Nr_Paragens' na tabela 'Viagem'. A cláusula `WHERE ID = NEW.Viagem` assegura que apenas a viagem associada à paragem recém-criada é atualizada. A referência `NEW.Viagem` acede ao valor da coluna 'Viagem' do registo inserido.

Seguidamente, foram desenvolvidas três funções para simplificar a escrita e padronizar a execução das interrogações relativas aos requisitos **RM11**, **RM12** e **RM18**. Estas funções encapsulam cálculos frequentemente necessários, permitindo a sua reutilização em múltiplos contextos e garantindo a consistência dos resultados em toda a aplicação.

As funções foram implementadas com as seguintes instruções:

```
CREATE FUNCTION CalcularInvestimentoTotal (ID_Patrocinador INT)
RETURNS DECIMAL(10,2) DETERMINISTIC
BEGIN
     RETURN (
          SELECT SUM(Valor) FROM Rel_Viagem_Patrocinador
          WHERE Patrocinador = ID_Patrocinador
     );
END

CREATE FUNCTION CalcularInvestimentoMédio (ID_Patrocinador INT)
RETURNS DECIMAL(10,2) DETERMINISTIC
     BEGIN
     RETURN (
          SELECT ROUND(AVG(Valor),2) FROM Rel_Viagem_Patrocinador
          WHERE Patrocinador = ID_Patrocinador
     );
END
```

```
CREATE FUNCTION CalcularPercentagemPositiva (ID_Viagem INT)
RETURNS INT DETERMINISTIC
     BEGIN
     RETURN (
          SELECT ROUND(100.0 *
          SUM(CASE WHEN Reação = 'Positiva' THEN 1 ELSE 0 END) /
          NULLIF(COUNT(*),0))
          FROM Rel_Utilizador_Viagem
          WHERE Viagem = ID_Viagem
     );
END
```

A primeira função, 'CalcularInvestimentoTotal', recebe como *input* o 'ID ' de um patrocinador e calcula o investimento total acumulado pelo mesmo em todas as viagens, agregando os valores de patrocínio associados e retornando a sua soma total. A cláusula `DETERMINISTIC` indica que, para o mesmo *input*, a função retorna sempre o mesmo resultado, permitindo otimizações por parte do otimizador de queries. Já a 'CalcularInvestimentoMédio' opera de forma semelhante, calculando a o investimento médio de um patrocinador (cujo 'ID' é dado como *input*) em vez do total. Adicionalmente, esta função arredonda a média às duas casas decimais através da função pré-definida `ROUND()`. Esta métrica é útil para análise do perfil de investimento, permitindo comparar a magnitude típica das contribuições de um patrocinador. Finalmente, a função 'CalcularPercentagemPositiva' computa a percentagem de reações positivas recebidas por uma viagem específica, recebendo o 'ID' da viagem como input. Para tal, conta o número total de reações e separa as positivas através de uma expressão `CASE`. De seguida, divide o número de reações positivas pelo total, multiplicando por 100 para obter a percentagem. A função pré-definida `NULLIF()` previne a divisão por zero, retornando `NULL` se não houver reações. Por fim, o resultado é arredondado para o inteiro mais próximo.

Esta abordagem de modularização através de funções armazenadas não só simplifica a implementação dos requisitos específicos, mas também estabelece uma base sólida para futuras extensões do sistema, promovendo boas práticas de desenvolvimento e manutenibilidade a longo prazo.

Como foi referido anteriormente, por motivos de segurança, certos requisitos implicavam a criação de procedimentos dedicados, mais especificamente para:

- Alterar passwords de utilizadores de forma segura.
- Gerir inserções, atualizações, remoções na tabela 'Rel_Utilizador_Viagem', garantindo a validação da identidade do utilizador.

Todos estes requisitos partilham uma característica comum: necessitam de verificar a autenticidade do utilizador antes de executar qualquer operação, confirmando que o email e palavra-passe fornecidos correspondem a um registo válido na base de dados. Esta verificação prévia é crucial para prevenir

acessos não autorizados e garantir que apenas utilizadores legítimos podem realizar operações sensíveis.

Consequentemente, concluiu-se que, para efeitos de modularidade e reutilização de código, seria apropriado criar um procedimento específico para a tarefa de autenticação. Este procedimento pode então ser invocado por outros procedimentos que necessitem de validar credenciais, evitando a duplicação de código e centralizando a lógica de autenticação num único ponto. Implementou-se o mesmo através da seguinte instrução:

```
CREATE PROCEDURE AutentificaUtilizador(
     IN p_email VARCHAR(100),
     IN p_password VARCHAR(100),
     OUT p_ID INT,
     OUT p_mensagem VARCHAR(100)
)
BEGIN
     DECLARE v_ID INT;
     DECLARE v_password VARCHAR(100);
     SET p_ID = NULL;
     SET p_mensagem = NULL;
     SELECT ID, Pass_Word INTO v_ID, v_password FROM Utilizador
          WHERE Email = p_email;
     IF v_ID IS NULL THEN
          SET p_mensagem = 'O email é inválido';
     ELSEIF v_password != p_password THEN
          SET p_mensagem = 'A password é inválida';
     ELSE SET p_ID = v_ID;
     END IF;
END
```

Para o seu funcionamento, necessita de quatro parâmetros:

- Dois parâmetros de entrada (`IN`):
    - 'p_email': endereço de email fornecido pelo utilizador.
    - 'p_password': password fornecida pelo utilizador.
- Dois parâmetros de saída (`OUT`):
    - 'p_ID': ID do utilizador, que será preenchido em caso de sucesso.
    - 'p_mensagem': Mensagem descritiva do resultado, preenchida apenas em caso de erro.

O funcionamento interno do procedimento divide-se em três fases:

- Declaração de variáveis auxiliares (`DECLARE`):
    - 'v_ID': armazena o ID do utilizador obtido na base de dados.
    - 'v_password': armazena a password obtida na base de dados.
- Inicialização dos parâmetros de saída:
    - 'p_ID' e 'p_password' são definidos como `NULL`, garantindo que, caso o procedimento não execute as instruções de atribuição, estes parâmetros têm um valor conhecido.
- Consulta à base de dados:
    - É executada uma instrução de `SELECT` na tabela 'Utilizador' para obter o 'ID' e a 'Pass_Word' correspondentes ao email fornecido.

Segue-se a implementação do procedimento 'MudarPassword', que permite a um utilizador alterar a sua palavra-passe, desde que conheça a atual. Este procedimento foi implementado através da seguinte instrução SQL:

```
CREATE PROCEDURE MudarPassword (
     IN p_email VARCHAR(100),
     IN p_password_antiga VARCHAR(100),
     IN p_password_nova VARCHAR(100)
)
BEGIN
     DECLARE v_ID INT;
     DECLARE v_mensagem VARCHAR(100);
     CALL AutentificaUtilizador (p_email, p_password_antiga, v_ID,
     v_mensagem);
     IF v_mensagem IS NOT NULL THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = v_mensagem;
     END IF;
     UPDATE Utilizador
          SET Pass_Word = p_password_nova
          WHERE ID = v_ID;
END
```

Como referido anteriormente, o procedimento 'AutentificaUtilizador' é reutilizado em vários contextos, sendo este um deles. O fluxo de execução do 'MudarPassword' pode ser descrito em três etapas principais:

1. Autentificação do utilizador através do procedimento realizado para o efeito.
2. Controlo de erros:

- Se `v_mensagem IS NOT NULL`, significa que a autentificação falhou. Neste caso, o procedimento é abortado através da instrução `SIGNAL SQLSTATE '45000'`, que levanta uma exceção com a mensagem de erro devolvida por 'AutentificaUtilizador'.
- Se `v_mensagem IS NULL`, a autentificação foi bem-sucedida e prossegue-se para a etapa seguinte.

3. Após as validações necessárias, é executada a instrução `UPDATE` na tabela 'Utilizador':

- A cláusula `SET Pass_Word = p_password_nova` atribui o novo valor de palavra-passe à coluna 'Pass_Word'.
- A cláusula `WHERE ID = v_ID` garante que a alteração afeta apenas o registo do utilizador autenticado.

Segue-se a criação dos procedimentos que permitem gerir as interações entre os utilizadores e as viagens, designadamente a inserção, atualização e remoção de visualizações (reações a viagens). Estes procedimentos foram desenvolvidos para garantir que apenas utilizadores devidamente autenticados possam realizar operações sobre os seus próprios registos, seguindo uma abordagem consistente de segurança e modularidade.

Primeiramente, o procedimento de inserção foi implementado da seguinte forma:

```
CREATE PROCEDURE InserirVisualização (
     IN p_email VARCHAR(100),
     IN p_password VARCHAR(100),
     IN p_viagem INT,
     IN p_reação ENUM('Negativa', 'Positiva')
)
BEGIN
     DECLARE v_ID INT;
     DECLARE v_mensagem VARCHAR(100);
     CALL AutentificaUtilizador (p_email, p_password, v_ID,
     v_mensagem);
     IF v_mensagem IS NOT NULL THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = v_mensagem;
     END IF;
     INSERT INTO Rel_Utilizador_Viagem (Utilizador, Viagem, Reação)
          VALUES (v_ID, p_viagem, p_reação);
END
```

Este procedimento recebe como parâmetros 'p_email' e 'p_password', para efeitos de autentificação, 'p_viagem', para identificar a viagem associada, e 'p_reação', que corresponde à reação do utilizador à

viagem. Após validar as credenciais através do procedimento 'AutentificaUtilizador', o procedimento insere um novo registo na tabela 'Rel_Utilizador_Viagem'. Se a autenticação falhar, o procedimento é abortado com a mensagem de erro apropriada.

Segundamente, foi implementado o procedimento de atualização:

```
CREATE PROCEDURE AtualizarVisualização (
     IN p_email VARCHAR(100),
     IN p_password VARCHAR(100),
     IN p_viagem INT,
     IN p_reação ENUM('Negativa', 'Positiva')
)
BEGIN
     DECLARE v_ID INT;
     DECLARE v_mensagem VARCHAR(100);
     DECLARE v_flag INT;
     CALL AutentificaUtilizador (p_email, p_password, v_ID,
     v_mensagem);
     IF v_mensagem IS NOT NULL THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = v_mensagem;
     END IF;
     SELECT COUNT(*) INTO v_flag FROM Rel_Utilizador_Viagem
          WHERE Utilizador = v_ID AND Viagem = p_viagem;
     IF v_flag = 0 THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'Registo inexistente';
     END IF;
     UPDATE Rel_Utilizador_Viagem
          SET Reação = p_reação
          WHERE Utilizador = v_ID AND Viagem = p_viagem;
END
```

O funcionamento deste procedimento é idêntico à do anterior, à exceção de que, em vez de uma inserção, apenas atualiza a reação do registo e que é necessária mais uma validação: o registo a atualizar já deve existir. De modo a garantir este fator, são realizadas as seguintes instruções adicionais:

1. Através da instrução `SELECT COUNT(*)`, determina o número de registos ('v_flag') que correspondem à chave primária ('Utilizador', 'Viagem').
2. Se não houver um registo a atualizar, o procedimento é abortado com a respetiva mensagem de erro. Caso contrário, prossegue-se a atualização do registo.

Terceiramente, foi executada a seguinte instrução para criar o procedimento de remoção:

```
CREATE PROCEDURE RemoverVisualização (
     IN p_email VARCHAR(100),
     IN p_password VARCHAR(100),
     IN p_viagem INT
)
BEGIN
     DECLARE v_ID INT;
     DECLARE v_mensagem VARCHAR(100);
     DECLARE v_flag INT;
     CALL AutentificaUtilizador (p_email, p_password, v_ID,
     v_mensagem);
     IF v_mensagem IS NOT NULL THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = v_mensagem;
     END IF;
     SELECT COUNT(*) INTO v_flag FROM Rel_Utilizador_Viagem
          WHERE Utilizador = v_ID AND Viagem = p_viagem;
     IF v_flag = 0 THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'Registo inexistente';
     END IF;
     DELETE FROM Rel_Utilizador_Viagem
          WHERE Utilizador = v_ID AND Viagem = p_viagem;
END
```

A nível de validações, este procedimento é idêntico ao anterior. No entanto, difere em dois aspetos. Como não necessita da reação para inserir/atualizar, não recebe o parâmetro 'p_reação'. Para além disso, como é óbvio, em vez de uma atualização (`UPDATE`) realiza uma remoção (`DELETE`).

Após a criação destes procedimentos armazenados, foi necessário configurar as permissões de execução para garantir que cada tipo de utilizador do sistema pudesse aceder apenas às funcionalidades adequadas ao seu perfil. Para tal, foram executadas as instruções `GRANT EXECUTE` referidas na secção 5.2.

De seguida, foi realizada a implementação dos procedimentos utilizados no povoamento da base de dados (conforme analisado em 5.3.). Todos os procedimentos seguem a mesma lógica e estrutura, à exceção do procedimento de inserção de paragens, ao qual foi adicionada uma verificação essencial. Por este motivo, serão apenas explicados os procedimentos de inserção em 'Utilizador' e em 'Paragem'. As instruções utilizadas para a sua criação encontram-se a seguir:

```
CREATE PROCEDURE InserirUtilizador (
     IN p_Nome VARCHAR(100),
     IN p_Telemovel VARCHAR(13),
     IN p_Email VARCHAR(100),
     IN p_Pass_Word VARCHAR(100)
)
BEGIN
     INSERT INTO Utilizador (Nome, Telemóvel, Email, Pass_Word)
          VALUES (p_Nome, p_Telemovel, p_Email, p_Pass_Word);
END

CREATE PROCEDURE InserirParagem (
     IN p_Nr INT UNSIGNED,
     IN p_Viagem INT UNSIGNED,
     IN p_Cidade VARCHAR(50),
     IN p_Pais VARCHAR(50),
     IN p_Data_Chegada DATETIME,
     IN p_Data_Partida DATETIME
)
BEGIN
     DECLARE v_flag INT;
     SELECT COUNT(*) INTO v_flag FROM Paragem
          WHERE Viagem = p_Viagem AND Data_Chegada <
          p_Data_Partida AND Data_Partida > p_Data_Chegada;
     IF v_flag > 0 THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'A paragem é inválida';
     END IF;
     INSERT INTO Paragem (Nr, Viagem, Cidade, País, Data_Chegada,
     Data_Partida)
          VALUES (p_Nr, p_Viagem, p_Cidade, p_Pais,
          p_Data_Chegada, p_Data_Partida);
     END
```

Tal como os restantes procedimentos de inserção, o 'InserirUtilizador' possui um funcionamento simples: recebe como parâmetros de entrada os valores do novo registo e realiza o registo com os mesmos. Já o 'InserirParagem', distingue-se por necessitar de validar a paragem quanto ao seu acontecimento: para uma dada viagem, os intervalos de datas das suas paragens não se podem intersetar. Por este motivo, foi adicionada uma consulta (`SELECT`) para encontrar paragens da mesma viagem cujo intervalo de datas se interseta com o intervalo da nova paragem. Caso haja alguma paragem, o procedimento é abortado, uma vez que é a nova paragem é considerada inválida.

Os procedimentos de inserção encontram-se no anexo **Erro! A origem da referência não foi encontrada.**.

É importante notar que, aquando da inserção de uma nova viagem, o campo 'Nr_Paragens' não deve ser definido (será 0 por defeito). Este será atualizado automaticamente pelo *trigger* 'AtualizaNrParagens' sempre que uma paragem for associada à viagem.

Para finalizar esta etapa, foi necessário implementar os procedimentos que permitem aos utilizadores realizar pesquisas avançadas sobre as viagens, conforme especificado pelos requisitos **RM8**, **RM9** e **RM18**. Estes procedimentos constituem a camada de interface entre a aplicação cliente e as consultas complexas de filtragem, encapsulando a lógica de negócio e proporcionando uma API uniforme e segura para o acesso aos dados. Foram definidas as seguintes instruções para criar estes procedimentos:

```
CREATE PROCEDURE FiltrarViagensPorLocalização (
     IN p_país VARCHAR(50),
     IN p_cidade VARCHAR(50)
)
BEGIN
     SELECT DISTINCT V.* FROM Viagem AS V
     INNER JOIN Paragem AS P ON V.ID = P.Viagem
          WHERE (p_país IS NULL OR P.País = p_país) AND (p_cidade
          IS NULL OR P.Cidade = p_cidade);
END

CREATE PROCEDURE FiltrarViagensPorDetalhes (
     IN p_data_min DATE,
     IN p_data_max DATE,
     IN p_custo_min DECIMAL(8,2),
     IN p_custo_max DECIMAL(8,2),
     IN p_objetivo ENUM('Pedagógico', 'Recreativo', 'Ambos')
)
BEGIN
     SELECT * FROM Viagem
     WHERE (p_data_min IS NULL OR Data_Início >= p_data_min) AND
     (p_data_max IS NULL OR Data_Término <= p_data_max) AND
     (p_custo_min IS NULL OR Custo >= p_custo_min) AND
     (p_custo_max IS NULL OR Custo <= p_custo_max) AND
     (p_objetivo IS NULL OR Objetivo = p_objetivo);
END
```

```
CREATE PROCEDURE FiltrarViagensPorReações (
     IN p_percentagem INT
)
BEGIN
     SELECT V.*, P.Percentagem FROM Viagem AS V
     INNER JOIN (
          SELECT Viagem, CalcularPercentagemPositiva(Viagem) AS
          Percentagem FROM Rel_Utilizador_Viagem AS RUV
               GROUP BY Viagem
     ) AS P ON V.ID = P.Viagem
          WHERE P.Percentagem >= p_percentagem
          ORDER BY P.Percentagem DESC;
END
```

Em suma, a implementação dos procedimentos armazenados no sistema BeLIUM_Viagens estabeleceu uma camada robusta e segura de interação com a base de dados. Através da criação de procedimentos específicos para operações sensíveis, foi possível garantir a integridade dos dados enquanto se oferece uma interface simplificada para a aplicação cliente. A configuração cuidadosa das permissões de execução, aliada à modularidade proporcionada pela reutilização do procedimento de autenticação, assegura que cada perfil de utilizador acede apenas às funcionalidades adequadas, cumprindo assim os requisitos de segurança e usabilidade estabelecidos. Esta abordagem não só satisfaz os requisitos funcionais identificados, mas também estabelece uma base sólida para futuras expansões e manutenção do sistema.

6. Conclusões e Trabalho Futuro

O presente projeto teve como objetivo primordial o desenvolvimento de um sistema de base de dados destinado a centralizar e preservar a memória histórica das atividades do núcleo. A solução visou colmatar a dispersão de informação identificada nas plataformas anteriores, seguindo rigorosamente o ciclo de vida de desenvolvimento de uma base de dados.

O processo iniciou-se com o levantamento de requisitos junto do corpo colaborativo e da comunidade alumni. No domínio da modelação, a estruturação conceptual e lógica garantiu a normalização até à Terceira Forma Normal (3FN), assegurando a integridade dos dados. A nível físico, a implementação no SGBD MySQL e a decisão arquitetural de armazenar caminhos de diretoria em vez de Binary Large Objects (BLOBs) revelaram-se cruciais para manter a base de dados leve e otimizada, pese embora a dependência externa criada ao nível do sistema de ficheiros. Adicionalmente, a centralização de operações críticas e o controlo de acessos mitigaram eficazmente as vulnerabilidades de segurança.

A verificação final indica que as metas de centralização, preservação da memória e otimização foram plenamente atingidas. A análise estrutural confirma que o sistema reúne as condições técnicas para os objetivos iniciais, projetando-se resultados quantitativos relevantes: uma redução de 40% na perda de informações e um aumento de 30% no retorno financeiro.

Em retrospetiva à gestão do projeto, a equipa de desenvolvimento reconhece que a planificação inicial revelou desvios face à realidade, decorrentes da inexperiência inicial relativamente à complexidade da implementação de uma Base de Dados. Uma análise comparativa entre o previsto e o executado evidencia o ajuste temporal estratégico, com o adiamento do desenvolvimento da etapa de implementação física. Esta janela temporal permitiu aos elementos aprofundarem os seus conhecimentos em SQL avançado, visando uma implementação mais eficiente.

Adicionalmente, verificou-se um maior paralelismo na execução das tarefas. Ao contrário da execução sequencial inicialmente prevista (conforme ilustrado no diagrama original), a realidade do projeto possibilitou um maior paralelismo na execução das tarefas. A independência entre determinados módulos permitiu que diferentes partes do trabalho avançassem em simultâneo, otimizando o tempo disponível.

Em relações aos pontos mais negativos, ao contrário da execução técnica, a fase de documentação inicial revelou lacunas, especificamente na definição dos dicionários de dados (obrigatoriedade de campos NULL vs NOT NULL), o que exigiu reajustes durante a implementação. Subsistem, no entanto, algumas limitações na escalabilidade geográfica devido à opção por campos textuais (VARCHAR) para localizações, o que poderá gerar inconsistências futuras e requer atenção.

Conclui-se que a equipa de desenvolvimento deve agora transitar para uma fase de manutenção e suporte ao núcleo. O foco deverá ser garantir a operacionalidade contínua da Base de Dados e implementar melhorias incrementais, baseadas no comentário crítico e na experiência real dos sócios e utilizadores.

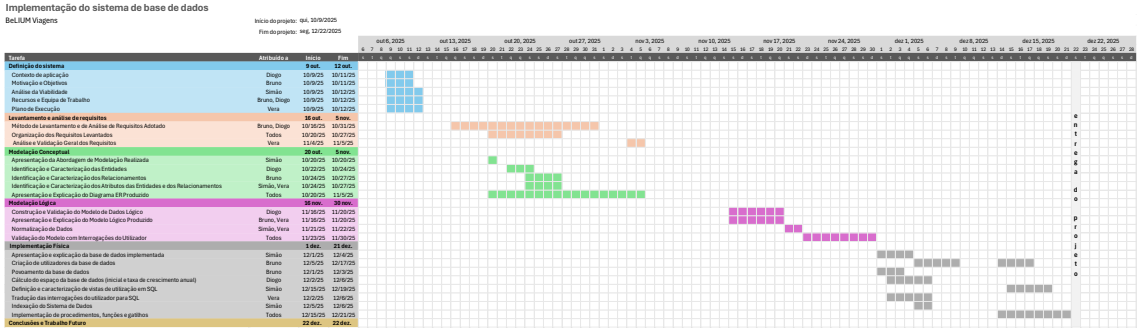


Figura 26 - Diagrama de Gantt (real).

Referências

- Cândido, C. (2020). brModelo. SIS4.com. http://www.sis4.com/brModelo/
- Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management (6th ed. – Global ed.). Pearson Education Limited.
- dbis-uibk. (2024a). RelaX – relational algebra calculator. https://dbis-uibk.github.io/relax/
- dbis-uibk. (2024b). RelaX – Help. https://dbis-uibk.github.io/relax/help
- Fowler, M. (s.d.). Software Architecture Guide. MartinFowler.com. https://martinfowler.com/architecture/
- Lucidchart. (s.d.). What is an Entity Relationship Diagram (ERD) https://www.lucidchart.com/pages/er-diagrams
- Microsoft. (2023). Manage the Size of the Transaction Log File. SQL Server. https://docs.microsoft.com/en-us/sql/relational-databases/databases/database-size-and-transaction-log-growth?view=sql-server-ver15
- Oracle. (2012). Oracle Database Administrator's Guide: Managing Space for Schema Objects. https://docs.oracle.com/cd/E11882_01/server.112/e41084/dbspace.htm
- Oracle. (2024a). MySQL 8.0 Reference Manual. MySQL. https://dev.mysql.com/doc/refman/8.0/en/
- Oracle. (2024b). MySQL 8.0 Reference Manual: InnoDB Storage Engine. MySQL. https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html
- Oracle. (2024c). MySQL 8.0 Reference Manual: Storage Requirements. MySQL. https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html
- Oracle. (2024d). MySQL Database Service. https://www.oracle.com/mysql/
- Oracle. (2024e). MySQL Workbench. MySQL. https://www.mysql.com/products/workbench/
- SQLBolt. (s.d.). Introduction to SQL. https://sqlbolt.com/
- Stack Overflow. (s.d.). Newest 'mysql' Questions. https://stackoverflow.com/questions/tagged/mysql
- Vertex42. (s.d.). Simple Gantt Chart Template. https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html
- W3Schools. (2024). SQL Tutorial. https://www.w3schools.com/sql/
- Winand, M. (s.d.). Use The Index, Luke! – SQL Indexing and Tuning. https://use-the-index-luke.com/

Lista de Siglas e Acrónimos

1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
3FN	Terceira Forma Normal
BD	Base de Dados
SBD	Sistema de Base de Dados
SGBD	Sistema de Gestão de Base de Dados
ER	Entidade-Relacionamento
RC	Requisito de Controlo
RD	Requisito de Descrição
RM	Requisito de Manipulação
SQL	*Structured Query Language*
CCP	Corpo Colaborativo do Projeto
ACID	Atomicity, Consistency, Isolation, Durability
CAOS	Centro de Apoio ao Open Source
CRUD	Create, Read, Update, Delete

Anexos

I. Requisitos de Descrição

Número	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista
RD1	20/10/2025	10:16	Uma viagem é identificada pelo seu ID. Cada viagem contém um objetivo, uma data inicial e uma data final, um custo associado, o número de participantes, o número de paragens e um ou mais tipos de deslocamento.	Utilizadores	Reunião do CCP	Desenvolvedores
RD2	20/10/2025	10:18	Um sócio pode publicar viagens. Cada viagem pode ou não ser publicada por um ou mais sócios.	Sócios	Reunião do CCP	Desenvolvedores
RD3	20/10/2025	10:20	Cada sócio é caracterizado pelo seu número de sócio e, opcionalmente, pelo estatuto dentro do núcleo e por uma foto de perfil.	Sócios	Reunião do CCP	Desenvolvedores
RD4	20/10/2025	10:32	Uma viagem possui pelo menos uma paragem. Cada paragem pertence obrigatoriamente a uma única viagem. A viagem deve, também, possuir o número de paragens associadas.	Utilizadores	Reunião do CCP	Desenvolvedores
RD5	20/10/2025	10:33	Uma paragem é identificada por um número único e contém uma localização (composta por país e cidade), assim como uma data de chegada e de partida e, opcionalmente, fotos tiradas no local.	Utilizadores	Reunião do CCP	Desenvolvedores
RD6	20/10/2025	10:41	Cada sócio corresponde a exatamente um utilizador. Um utilizador pode ou não ser sócio.	Utilizadores/Sócios	Reunião do CCP	Desenvolvedores
RD7	20/10/2025	10:49	Um utilizador é identificado pelo seu ID. Um utilizador possui ainda um nome, um email, um número de telemóvel e uma password para autentificação.	Utilizadores	Reunião do CCP	Desenvolvedores
RD8	20/10/2025	10:54	Um patrocinador patrocina uma ou mais viagens. Cada viagem pode ou não ser patrocinada por um ou mais patrocinadores.	Administração	Reunião do CCP	Desenvolvedores
RD9	20/10/2025	10:57	Um patrocinador é identificado por um ID único e possui um nome.	Administração	Reunião do CCP	Desenvolvedores
RD10	20/10/2025	10:59	Um patrocínio contém detalhes, como o valor investido e o(s) motivo(s) do apoio.	Administração	Reunião do CCP	Desenvolvedores
RD11	01/11/2025	15:20	Os sócios descrevem as viagens que publicam com uma avaliação. Podem, também, optar por deixar um comentário.	Sócios	Inquérito à Comunidade	Desenvolvedores
RD12	01/11/2025	15:52	Um utilizador pode visualizar zero ou mais viagens. Cada viagem pode ser visualizada por zero ou mais utilizadores. A visualização inclui a reação do utilizador à viagem.	Utilizadores	Inquérito à Comunidade	Desenvolvedores

II. Requisitos de Manipulação

Número	Data	Hora	Descrição	Vista de Utilização	Fonte	Analista
RM1	27/10/2025	14:10	Registar novos sócios e alterar os dados dos existentes.	Administração	Reunião do CCP	Desenvolvedores
RM2	27/10/2025	14:12	Registar novos utilizadores e alterar os dados dos existentes.	Administração	Reunião do CCP	Desenvolvedores
RM3	27/10/2025	14:16	Listar, por ordem, todas as paragens de uma dada viagem.	Utilizadores/ Administração	Reunião do CCP	Desenvolvedores
RM4	27/10/2025	14:20	Listar todos os sócios (número de sócio, nome e estatuto) que realizaram uma determinada viagem (por ordem alfabética).	Utilizadores	Reunião do CCP	Desenvolvedores
RM5	27/10/2025	14:22	Registar novas viagens.	Sócios	Reunião do CCP	Desenvolvedores
RM6	27/10/2025	14:22	Consultar os patrocínios (nome do patrocinador e valor) de uma dada viagem, ordenados pelo valor.	Administração	Reunião do CCP	Desenvolvedores
RM7	27/10/2025	14:24	Registar novas paragens.	Sócios	Reunião do CCP	Desenvolvedores
RM8	27/10/2025	14:27	Filtrar viagens pela localização das paragens.	Utilizadores	Reunião do CCP	Desenvolvedores
RM9	27/10/2025	14:29	Filtrar viagens pelas datas, pelo custo e pelo objetivo.	Utilizadores	Reunião do CCP	Desenvolvedores
RM10	27/10/2025	14:33	Registar novos patrocinadores e alterar os dados dos existentes.	Administração	Reunião do CCP	Desenvolvedores
RM11	27/10/2025	14:35	Calcular o investimento total de cada patrocinador.	Administração	Reunião do CCP	Desenvolvedores
RM12	27/10/2025	14:35	Calcular a média de investimento de um patrocinador por viagem.	Administração	Reunião do CCP	Desenvolvedores
RM13	27/10/2025	15:16	Editar o valor do patrocínio.	Administração	Reunião do CCP	Desenvolvedores
RM14	27/10/2025	15:35	Inserir comentários sobre viagens.	Sócios	Reunião do CCP	Desenvolvedores
RM15	27/10/2025	15:36	Listar as viagens realizadas por um sócio (através do número de sócio).	Sócios	Reunião do CCP	Desenvolvedores
RM16	27/10/2025	15:41	Consultar os motivos de um patrocínio.	Administração	Reunião do CCP	Desenvolvedores
RM17	01/11/2025	14:50	Consultar o número de viagens publicadas por cada sócio (ordenado por número de publicações).	Utilizadores	Inquérito à Comunidade	Desenvolvedores
RM18	01/11/2025	16:15	Filtrar viagens por percentagem de reações positivas (ordenado por percentagem).	Utilizadores	Inquérito à Comunidade	Desenvolvedores
RM19	01/11/2025	16:23	Consultar o top 10 de viagens mais populares, isto é, com mais visualizações.	Utilizadores	Inquérito à Comunidade	Desenvolvedores
RM20	01/11/2025	16:25	Consultar as opiniões dos sócios em relação a uma dada viagem.	Utilizadores	Inquérito à Comunidade	Desenvolvedores

III. Requisitos de Controlo

Número	Data	Hora	Descrição	Vista de utilização	Fonte	Analista
RC1	27/10/2025	14:39	Para se registar, um utilizador deve pedir diretamente a um administrador ou enviar um pedido, por email. Se o pedido for aceite, um administrador deve registá-lo no sistema e comunicá-lo do seu ID e da sua password.	Administração	Reunião do CCP	Desenvolvedores
RC2	27/10/2025	14:42	Um utilizador pode consultar todos os dados dos sócios, das viagens e das paragens no sistema.	Utilizadores	Reunião do CCP	Desenvolvedores
RC3	27/10/2025	14:46	Um utilizador pode atualizar a sua password, se souber a atual.	Utilizadores	Reunião do CCP	Desenvolvedores
RC4	27/10/2025	14:54	Não devem existir múltiplos registos de utilizadores com o mesmo email.	Utilizadores	Reunião do CCP	Desenvolvedores
RC5	27/10/2025	15:05	Utilizadores podem visualizar e reagir às viagens. Podem, também, atualizar ou remover a visualização.	Utilizadores	Reunião do CCP	Desenvolvedores
RC6	27/10/2025	15:08	Os sócios possuem todas as permissões dos utilizadores.	Sócios	Reunião do CCP	Desenvolvedores
RC7	27/10/2025	15:11	Os sócios podem inserir novas viagens, assim como as suas paragens. Podem, também, editá-las ou removê-las, caso necessário.	Sócios	Reunião do CCP	Desenvolvedores
RC8	27/10/2025	15:11	Os sócios podem consultar todos os dados no sistema, à excessão dos dados pessoais dos utilizadores.	Sócios	Reunião do CCP	Desenvolvedores
RC9	27/10/2025	15:11	Um sócio pode publicar uma viagem com os respetivos avaliação e comentário. Pode, ainda, editar ou eliminar esta publicação.	Sócios	Reunião do CCP	Desenvolvedores
RC10	27/10/2025	15:18	Os administradores (membros do CCP) podem realizar qualquer tipo de operação no sistema, incluindo alterações nucleares, caso necessário. São responsáveis por todas as operações fora do alcance dos sócios.	Administração	Reunião do CCP	Desenvolvedores
RC11	27/10/2025	15:25	Eventualmente, se necessário, sócios de confiança podem obter o estatuto de administrador do sistema (permissão atribuída por outros administradores).	Administração	Reunião do CCP	Desenvolvedores
RC12	27/10/2025	15:29	Ocasionalmente, devem ser armazenados backups dos dados.	Administração	Reunião do CCP	Desenvolvedores

IV. Relações RelaX

```
group: BeLIUM_Viagens

Utilizador = {
    Id:number, Nome:string, Telemovel:string, Email:string, Pass_Word:string
    1, BrunoMagalhaes, 351937426819, brunomagalhaes_cesium_1728_gmail.com, gA9rZ3tX7KpVL2mN
    2, VeraAlmeida, 351963817402, veraalmeida_cesium_1854_gmail.com, nB7qY4kP8sH2T6zG
    3, DiogoAzevedo, 351912345678, diogoazevedo_cesium_1912_gmail.com, pC6wR8vM9hL3K2sD
    4, SimaoSantos, 351934781256, simaosantos_cesium_2037_gmail.com, tD5eS7uN1jQ4H8mB
    5, JulioPinto, 351961482739, juliopinto_cesium_2149_gmail.com, vE4fT6sM2kR9H1pZ
    6, GustavoPereira, 351926573814, gustavopereira_cesium_2263_gmail.com, wF3gU5tN8jS1K9qY
    7, RuiAfonso, 351968240157, ruiafonso_cesium_2378_gmail.com, xG2hV4rM7lT0_P8wE
    8, MarioRodrigues, 351913857462, mariorodrigues_cesium_2441_gmail.com, yH1iW3_qN6mS9O7e
    9, HumbertoGomes, 351947162805, humbertogomes_cesium_1986_gmail.com, zI0jX2pM5nT8Q6tS
    10, GoncaloPombo, 351965028417, goncalopombo_cesium_2215_gmail.com, aJ9kY1oN4mU7R5uF
    11, JoaoSilva, 351912345679, joao.silva_email.com, bK8lZ0pO3nV6S4iG
    12, MariaSantos, 351934781257, maria.santos_email.com, cL7mY9qP2oW5T3hH
    13, PedroAlmeida, 351961482740, pedro.almeida_email.com, dM6nX8rQ1pV4U2jI
    14, AnaCosta, 351926573815, ana.costa_email.com, eN5oW7sR0qX3V1kJ
    15, RuiFerreira, 351968240158, rui.ferreira_email.com, fO4pV6tS9rY2W0lK
    16, SofiaMartins, 351913857463, sofia.martins_email.com, gP3qW5uT8sZ1X9mL
    17, TiagoLopes, 351947162806, tiago.lopes_email.com, hQ2rX4vU7tA0Y8nM
    18, InesRocha, 351965028418, ines.rocha_email.com, iR1sY3wV6uB9Z7oN
    19, CarlosPereira, 351932740681, carlos.pereira_email.com, jS0tZ2xW5_vC8A6pO
    20, BeatrizNunes, 351951864237, beatriz.nunes_email.com, kT9uY1_yX4wD7B5qP
}

Socio = {
    Nr_Socio:number, Utilizador:number, Foto_Perfil:string, Estatuto:string
    1728, 1, null, Direcao
    1854, 2, Imagens_FP1852.png, Direcao
    1912, 3, Imagens_FP1912.png, Direcao
    2037, 4, Imagens_FP2037.png, Direcao
    2149, 5, Imagens_FP2149.png, Direcao
    2263, 6, Imagens_FP2263.png, Direcao
    2378, 7, null, Alumni
    2441, 8, null, null
    1986, 9, Imagens_FP1986.png, null
    2215, 10, Imagens_FP2215.png, Direcao
}

Viagem = {
    Id:number, Data_Inicio:date, Data_Termino:date, Nr_Participantes:number, Custo:number,
    Objetivo:string
    1, 2025-04-10, 2025-04-15, 10, 1257.84, Ambos
    2, 2024-11-20, 2024-11-20, 1, 24.70, Recreativo
    3, 2025-02-14, 2025-02-14, 2, 120.79, Pedagogico
```

```
    4, 2025-03-05, 2025-03-05, 5, 237.28, Recreativo
    5, 2025-07-05, 2025-07-09, 4, 1297.58, Ambos
}

Deslocamento = {
    Tipo_Deslocamento:string, Viagem:number
    Aviao, 1
    Carro, 1
    Ape, 1
    Ape, 2
    Carro, 3
    Carro, 4
    Barco, 4
    Ape, 4
    Aviao, 5
    Autocarro, 5
    Ape, 5
}

Paragem = {
    Nr:number, Viagem:number, Cidade:string, Pais:string, Data_Chegada:date, Data_Partida:date
    1, 1, Lisboa, Portugal, 2025-04-10, 2025-04-10
    2, 1, Coimbra, Portugal, 2025-04-11, 2025-04-11
    3, 1, Porto, Portugal, 2025-04-12, 2025-04-13
    4, 1, Vigo, Espanha, 2025-04-14, 2025-04-15
    1, 2, Lisboa, Portugal, 2024-11-20, 2024-11-20
    2, 2, Sintra, Portugal, 2024-11-20, 2024-11-20
    1, 3, Braga, Portugal, 2025-02-14, 2025-02-14
    2, 3, Porto, Portugal, 2025-02-14, 2025-02-14
    1, 4, Evora, Portugal, 2025-03-05, 2025-03-05
    2, 4, Beja, Portugal, 2025-03-05, 2025-03-05
    3, 4, Faro, Portugal, 2025-03-05, 2025-03-05
    1, 5, Madrid, Espanha, 2025-07-05, 2025-07-05
    2, 5, Saragossa, Espanha, 2025-07-06, 2025-07-06
    3, 5, Barcelona, Espanha, 2025-07-07, 2025-07-07
    4, 5, Valencia, Espanha, 2025-07-08, 2025-07-08
    5, 5, Alicante, Espanha, 2025-07-09, 2025-07-09
}

Foto = {
    Foto:string, Paragem:number, Viagem:number
    Fotos_Viagem1_Lisboa_01.png, 1, 1
    Fotos_Viagem1_Coimbra_01.png, 2, 1
    Fotos_Viagem1_Porto_01.png, 3, 1
    Fotos_Viagem1_Vigo_01.png, 4, 1
    Fotos_Viagem2_Lisboa_01.png, 1, 2
    Fotos_Viagem2_Sintra_01.png, 2, 2
    Fotos_Viagem3_Braga_01.png, 1, 3
    Fotos_Viagem3_Porto_01.png, 2, 3
    Fotos_Viagem4_Evora_01.png, 1, 4
    Fotos_Viagem4_Beja_01.png, 2, 4
    Fotos_Viagem4_Faro_01.png, 3, 4
    Fotos_Viagem5_Madrid_01.png, 1, 5
    Fotos_Viagem5_Saragossa_01.png, 2, 5
    Fotos_Viagem5_Barcelona_01.png, 3, 5
    Fotos_Viagem5_Valencia_01.png, 4, 5
    Fotos_Viagem5_Alicante_01.png, 5, 5
}
```

```
Patrocinador = {
   Id:number, Nome:string
   1, YariLabs
   2, Accenture
   3, Uphold
}

Rel_Viagem_Patrocinador = {
   Viagem:number, Patrocinador:number, Valor:number
   1, 1, 1200.00
   1, 2, 600.00
}

Motivo = {
   Motivo:string, Patrocinador:number, Viagem:number
   ApoioaviagemcoletivadoCeSIUMparacoberturaparcialdecustosdetransporteealojamentoepromocaoatividade
   spedagogicasededivulgacao, 1, 1
   Patrociniodaexcursaorecreativadestinadoaapoiologisticoepromocaodamarcajuntodosparticipantes, 2, 1
}

Rel_Socio_Viagem = {
   Socio:number, Viagem:number, Avaliacao:string, Comentario:string
   1, 1, MuitoPositiva, Excelenteorganizacaoemuitoboainteracaoentreparticipantes
   2, 1, Positiva, null
   3, 1, Positiva, Conteudospedagogicosrelevanteseguiasmuitoprestaveis
   4, 1, Mediana, null
   5, 1, Positiva, null
   6, 1, MuitoPositiva, null
   7, 1, Negativa, null
   8, 1, Mediana, null
   9, 1, Positiva, null
   10, 1, MuitoPositiva, null
   1, 2, Positiva, Viagemcurtamasmuitointeressanteebemorganizada
   2, 3, MuitoPositiva, Conteudospedagogicosmuitorelevantesgosteibastante
   4, 4, Positiva, Boalogisticaeatividadesdivertidas
   1, 5, Positiva, null
   2, 5, MuitoPositiva, null
   3, 5, Positiva, null
   4, 5, MuitoPositiva, Aviagemfoiexcelente
}

Rel_Utilizador_Viagem = {
   Utilizador:number, Viagem:number, Reacao:string
   1, 1, Positiva
   2, 1, Positiva
   3, 1, Positiva
   4, 1, Positiva
   5, 1, Positiva
   6, 1, Positiva
   7, 1, Positiva
   8, 1, Positiva
   9, 1, Positiva
   10, 1, Positiva
   11, 1, Positiva
   12, 1, Positiva
   13, 1, Positiva
   14, 1, Negativa
   15, 1, Positiva
```

```
    16, 1, Positiva
    17, 1, Positiva
    18, 1, Positiva
    19, 1, Positiva
    1, 2, Positiva
    5, 2, Positiva
    8, 2, Negativa
    12, 2, Positiva
    15, 2, Positiva
    2, 3, Positiva
    4, 3, Positiva
    6, 3, Positiva
    9, 3, Negativa
    11, 3, Positiva
    13, 3, Positiva
    17, 3, Positiva
    1, 4, Positiva
    3, 4, Positiva
    5, 4, Positiva
    7, 4, Positiva
    10, 4, Positiva
    14, 4, Positiva
    18, 4, Positiva
    2, 5, Positiva
    4, 5, Positiva
    6, 5, Positiva
    8, 5, Negativa
    9, 5, Positiva
    11, 5, Positiva
    12, 5, Positiva
    13, 5, Negativa
    15, 5, Positiva
    17, 5, Positiva
    19, 5, Positiva
    20, 5, Positiva
}
```

V. Povoamento da base de dados

ID	Nome	Telemóvel	Email
1	Bruno Magalhães	+351937426819	brunomagalhaes_cesium_1728@gmail.com
2	Vera Almeida	+351963817402	veraalmeida_cesium_1854@gmail.com
3	Diogo Azevedo	+351912345678	diogoazevedo_cesium_1912@gmail.com
4	Simão Santos	+351934781256	simaosantos_cesium_2037@gmail.com
5	Júlio Pinto	+351961482739	juliopinto_cesium_2149@gmail.com
6	Gustavo Pereira	+351926573814	gustavopereira_cesium_2263@gmail.com
7	Rui Afonso	+351968240157	ruiafonso_cesium_2378@gmail.com
8	Mário Rodrigues	+351913857462	mariorodrigues_cesium_2441@gmail.com
9	Humberto Gomes	+351947162805	humbertogomes_cesium_1986@gmail.com
10	Gonçalo Pombo	+351965028417	goncalopombo_cesium_2215@gmail.com
11	João Silva	+351912345679	joao.silva@email.com
12	Maria Santos	+351934781257	maria.santos@email.com
13	Pedro Almeida	+351961482740	pedro.almeida@email.com
14	Ana Costa	+351926573815	ana.costa@email.com
15	Rui Ferreira	+351968240158	rui.ferreira@email.com
16	Sofia Martins	+351913857463	sofia.martins@email.com
17	Tiago Lopes	+351947162806	tiago.lopes@email.com
18	Inês Rocha	+351965028418	ines.rocha@email.com
19	Carlos Pereira	+351932740681	carlos.pereira@email.com
20	Beatriz Nunes	+351951864237	beatriz.nunes@email.com

Tabela 40 - Registos do povoamento inicial da relação 'Utilizador'.

Utilizador	Nr_Sócio	Foto_Perfil	Estatuto
1	1728	NULL	Direção
2	1854	Imagens/FP1854.png	Direção
3	1912	Imagens/FP1912.png	Direção
4	2037	Imagens/FP2037.png	Direção
5	2149	Imagens/FP2149.png	Direção
6	2263	Imagens/FP2263.png	Direção
7	2378	NULL	Alumni
8	2441	NULL	NULL
9	1986	Imagens/FP1986.png	NULL
10	2215	Imagens/FP2215.png	Direção

Tabela 41 - Registos do povoamento inicial da relação 'Sócio'.

ID	Data_Início	Data_Término	Nr_Paragens	Nr_Participantes	Custo	Objetivo
1	10/04/2025 09:00	15/04/2025 18:00	4	10	1257.84	Ambos
2	20/11/2024 08:30	20/11/2024 20:00	2	1	24.70	Recreativo
3	14/02/2025 10:00	14/02/2025 18:00	2	2	120.79	Pedagógico
4	05/03/2025 09:00	05/03/2025 17:30	3	5	237.28	Recreativo
5	05/07/2025 07:30	09/07/2025 21:00	5	4	1297.58	Ambos

Tabela 42 - Registos do povoamento inicial da relação 'Viagem'.

Tipo_Deslocamento	Viagem
Avião	1
Carro	1
A pé	1
A pé	2
Carro	3
Carro	4
Barco	4
A pé	4
Avião	5
Autocarro	5
A pé	5

Tabela 43 - Registos do povoamento inicial da relação 'Deslocamento'.

Nr	Viagem	País	Cidade	Data_Chegada	Data_Partida
1	1	Portugal	Lisboa	10/04/2025 09:00	10/04/2025 12:00
2	1	Portugal	Coimbra	11/04/2025 10:00	11/04/2025 16:00
3	1	Portugal	Porto	12/04/2025 09:30	13/04/2025 18:00
4	1	Espanha	Vigo	14/04/2025 09:00	15/04/2025 18:00
1	2	Portugal	Lisboa	20/11/2024 08:30	20/11/2024 11:00
2	2	Portugal	Sintra	20/11/2024 11:30	20/11/2024 20:00
1	3	Portugal	Braga	14/02/2025 10:00	14/02/2025 13:00
2	3	Portugal	Porto	14/02/2025 14:00	14/02/2025 18:00
1	4	Portugal	Évora	05/03/2025 09:00	05/03/2025 11:30
2	4	Portugal	Beja	05/03/2025 12:30	05/03/2025 15:00
3	4	Portugal	Faro	05/03/2025 15:30	05/03/2025 17:30
1	5	Espanha	Madrid	05/07/2025 10:00	05/07/2025 18:00
2	5	Espanha	Sevilha	06/07/2025 09:00	06/07/2025 17:00
3	5	Espanha	Barcelona	07/07/2025 09:00	07/07/2025 20:00
4	5	Espanha	Valência	08/07/2025 09:00	08/07/2025 19:00
5	5	Espanha	Alicante	09/07/2025 09:00	09/07/2025 21:00

Tabela 44 - Registos do povoamento inicial da relação 'Paragem'.

ID	Nome
1	Yari Labs
2	Accenture
3	Uphold

Tabela 45 - Registos do povoamento inicial da relação 'Patrocinador'.

Foto	Paragem	Viagem
Fotos/Viagem1_Lisboa_01.png	1	1
Fotos/Viagem1_Coimbra_01.png	2	1
Fotos/Viagem1_Porto_01.png	3	1
Fotos/Viagem1_Vigo_01.png	4	1
Fotos/Viagem2_Lisboa_01.png	1	2
Fotos/Viagem2_Sintra_01.png	2	2
Fotos/Viagem3_Braga_01.png	1	3
Fotos/Viagem3_Porto_01.png	2	3
Fotos/Viagem4_Evora_01.png	1	4
Fotos/Viagem4_Beja_01.png	2	4
Fotos/Viagem4_Faro_01.png	3	4
Fotos/Viagem5_Madrid_01.png	1	5
Fotos/Viagem5_Saragossa_01.png	2	5
Fotos/Viagem5_Barcelona_01.png	3	5
Fotos/Viagem5_Valencia_01.png	4	5
Fotos/Viagem5_Alicante_01.png	5	5

Tabela 46- Registos do povoamento inicial da relação 'Foto'.

Sócio	Viagem	Avaliação	Comentário
1	1	Muito Positiva	Excelente organização e muito boa interação entre participantes.
2	1	Positiva	NULL
3	1	Positiva	Conteúdos pedagógicos relevantes e guias muito prestáveis.
4	1	Mediana	NULL
5	1	Positiva	NULL
6	1	Muito Positiva	NULL
7	1	Negativa	NULL
8	1	Mediana	NULL
9	1	Positiva	NULL
10	1	Muito Positiva	NULL
1	2	Positiva	Viagem curta mas muito interessante e bem organizada.
2	3	Muito Positiva	Conteúdos pedagógicos muito relevantes, gostei bastante.
4	4	Positiva	Boa logística e atividades divertidas.
1	5	Positiva	NULL
2	5	Muito Positiva	NULL
3	5	Positiva	NULL
4	5	Muito Positiva	A viagem foi excelente, momentos inesquecíveis e muito bem organizada.

Tabela 47- Registos do povoamento inicial da relação 'Rel_Sócio_Viagem'.

Viagem	Patrocinador	Valor
1	1	1200.00
1	2	600.00

Tabela 48 - Registos do povoamento inicial da relação 'Rel_Viagem_Patrocinador'.

Motivo	Viagem	Patrocinador
Apoio à viagem coletiva do CeSIUM para cobertura parcial de custos de transporte e alojamento e promoção de atividades pedagógicas e de divulgação.	1	1
Patrocínio da excursão recreativa, destinado a apoio logístico e promoção da marca junto dos participantes.	1	2

Tabela 49 - Registos do povoamento inicial da relação 'Motivo'.

Utilizador	Viagem	Reação
1	1	Positiva
2	1	Positiva
3	1	Positiva
4	1	Positiva
5	1	Positiva
6	1	Positiva
7	1	Positiva
8	1	Positiva
9	1	Positiva
10	1	Positiva
11	1	Positiva
12	1	Positiva
13	1	Positiva
14	1	Negativa
15	1	Positiva
16	1	Positiva
17	1	Positiva
18	1	Positiva
19	1	Positiva
1	2	Positiva
5	2	Positiva
8	2	Negativa
12	2	Positiva
15	2	Positiva
2	3	Positiva
4	3	Positiva
6	3	Positiva
9	3	Negativa
11	3	Positiva
13	3	Positiva
17	3	Positiva
1	4	Positiva
3	4	Positiva
5	4	Positiva
7	4	Positiva
10	4	Positiva
14	4	Positiva
18	4	Positiva
2	5	Positiva
4	5	Positiva
6	5	Positiva
8	5	Negativa
9	5	Positiva
11	5	Positiva
12	5	Positiva
13	5	Negativa
15	5	Positiva
17	5	Positiva
19	5	Positiva
20	5	Positiva

Tabela 50 - Registos do povoamento inicial da relação 'Rel_Utilizador_Viagem'.

VI. Procedimentos de inserção

```
DELIMITER $$
CREATE PROCEDURE InserirUtilizador (
    IN p_Nome VARCHAR(100),
    IN p_Telemovel VARCHAR(13),
    IN p_Email VARCHAR(100),
    IN p_Pass_Word VARCHAR(100)
)
BEGIN
    INSERT INTO Utilizador (Nome, Telemóvel, Email, Pass_Word)
    VALUES (p_Nome, p_Telemovel, p_Email, p_Pass_Word);
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE InserirSocio (
    IN p_Nr_Socio INT UNSIGNED,
    IN p_Utilizador INT UNSIGNED,
    IN p_Foto_Perfil VARCHAR(100),
    IN p_Estatuto ENUM('Direção','Alumni')
)
BEGIN
    INSERT INTO Sócio (Nr_Sócio, Utilizador, Foto_Perfil, Estatuto)
    VALUES (p_Nr_Socio, p_Utilizador, p_Foto_Perfil, p_Estatuto);
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE InserirViagem (
    IN p_Data_Inicio DATETIME,
    IN p_Data_Termino DATETIME,
    IN p_Nr_Participantes INT,
```

```
    IN p_Custo DECIMAL(8,2),
    IN p_Objetivo ENUM('Pedagógico','Recreativo','Ambos')
)
BEGIN
    INSERT INTO Viagem (Data_Início, Data_Término, Nr_Participantes,
    Custo, Objetivo)
    VALUES (p_Data_Inicio, p_Data_Termino, p_Nr_Participantes,
    p_Custo, p_Objetivo);
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE InserirDeslocamento (
    IN p_Tipo_Deslocamento ENUM('Carro', 'Autocarro', 'Comboio',
    'Avião', 'Barco', 'Navio', 'Bicicleta', 'Mota', 'A pé', 'Outro'),
    IN p_Viagem INT UNSIGNED
)
BEGIN
    INSERT INTO Deslocamento (Tipo_Deslocamento, Viagem)
    VALUES (p_Tipo_Deslocamento, p_Viagem);
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE InserirParagem (
    IN p_Nr INT UNSIGNED,
    IN p_Viagem INT UNSIGNED,
    IN p_Cidade VARCHAR(50),
    IN p_Pais VARCHAR(50),
    IN p_Data_Chegada DATETIME,
    IN p_Data_Partida DATETIME
)
BEGIN
    DECLARE v_flag INT;
    SELECT COUNT(*) INTO v_flag FROM Paragem
    WHERE Viagem = p_Viagem AND Data_Chegada < p_Data_Partida AND
    Data_Partida > p_Data_Chegada;
    IF v_flag > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A paragem é inválida';
```

```
    END IF;
    INSERT INTO Paragem (Nr, Viagem, Cidade, País, Data_Chegada,
    Data_Partida)
    VALUES (p_Nr, p_Viagem, p_Cidade, p_Pais, p_Data_Chegada,
    p_Data_Partida);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirFoto (
    IN p_Foto VARCHAR(100),
    IN p_Paragem INT UNSIGNED,
    IN p_Viagem INT UNSIGNED
)
BEGIN
    INSERT INTO Foto (Foto, Paragem, Viagem)
    VALUES (p_Foto, p_Paragem, p_Viagem);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirRelSocioViagem (
    IN p_Socio INT UNSIGNED,
    IN p_Viagem INT UNSIGNED,
    IN p_Avaliacao ENUM('Muito Negativa', 'Negativa', 'Mediana',
    'Positiva', 'Muito Positiva'),
    IN p_Comentario VARCHAR(150)
)
BEGIN
    INSERT INTO Rel_Sócio_Viagem (Sócio, Viagem, Avaliação,
Comentário)
    VALUES (p_Socio, p_Viagem, p_Avaliacao, p_Comentario);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirPatrocinador (
    IN p_Nome VARCHAR(40)
)
BEGIN
```

```
    INSERT INTO Patrocinador (Nome)
    VALUES (p_Nome);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirRelViagemPatrocinador (
    IN p_Viagem INT UNSIGNED,
    IN p_Patrocinador INT UNSIGNED,
    IN p_Valor DECIMAL(8,2)
)
BEGIN
    INSERT INTO Rel_Viagem_Patrocinador (Viagem, Patrocinador, Valor)
    VALUES (p_Viagem, p_Patrocinador, p_Valor);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirMotivo (
    IN p_Motivo VARCHAR(200),
    IN p_Viagem INT UNSIGNED,
    IN p_Patrocinador INT UNSIGNED
)
BEGIN
    INSERT INTO Motivo (Motivo, Patrocinador, Viagem)
    VALUES (p_Motivo, p_Patrocinador, p_Viagem);
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE InserirRelUtilizadorViagem (
    IN p_Utilizador INT UNSIGNED,
    IN p_Viagem INT UNSIGNED,
    IN p_Reacao ENUM('Negativa','Positiva')
)
BEGIN
    INSERT INTO Rel_Utilizador_Viagem (Utilizador, Viagem, Reação)
    VALUES (p_Utilizador, p_Viagem, p_Reacao);
END $$
DELIMITER ;
```