



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2022/2023

Livraria Silva

Diogo Fernandes a87968

Filipe Azevedo a87969

João Vítor a83783

15 de janeiro, 2023

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Livraria Silva

Diogo Fernandes a87968

Filipe Azevedo a87969

João Vítor a83783

15 de janeiro, 2023

Resumo

Este relatório foi desenvolvido no âmbito do desenvolvimento de uma Base de Dados para a livraria Silva, que foca a sua atividade na venda de livros. Este software tem como objetivo facilitar e tornar mais eficiente a grande maioria dos processos inerentes ao funcionamento da livraria. Desde a gestão dos livros, das encomendas realizadas, gestão de funcionários e veículos.

Ao longo do relatório são apresentadas na íntegra todas as etapas do desenvolvimento desta base de dados até à implementação física da mesma.

Em primeiro lugar, foram estudadas todas as características do meio onde este software será inserido e analisamos o motivo pelo qual surgiu a necessidade desta base de dados e que objetivos esta precisava de cumprir.

Em segundo lugar, através de métodos de análise e em conjunto com o cliente, foram levantados e aprovados os requisitos para o sistema que guiaram todo o processo de desenvolvimento. No fim desta fase, foi iniciada a modulação conceptual, desenvolvida com recurso à ferramenta “**brModelo**”, e que foi posteriormente aprovada pelo dono da livraria em questão. A partir desta foi desenvolvido o modelo lógico, na ferramenta “**MySQL Workbench**” e foi implementada fisicamente garantindo que todo o trabalho anterior convergia para uma solução correta.

Para concluir, após a implementação da base de dados com o cliente, foi dado como terminado este projeto.

Área de Aplicação: Desenho, Arquitetura, Desenvolvimento e Implementação de Sistemas de Base de Dados.

Palavras-Chave: Bases de Dados, Análise de Requisitos, Entidades, Atributos, Relacionamentos, Modelo Conceptual, Modelo Lógico, Vistas de Utilização, *MySQL Workbench*, *SQL*

Índice

Resumo	<i>i</i>
Índice	<i>ii</i>
Índice de Figuras	<i>v</i>
Índice de Tabelas.....	<i>vii</i>
1. Definição do Sistema.....	<i>1</i>
1.1. Contexto de aplicação e fundamentação do sistema	<i>1</i>
1.2. Motivação e Objetivos do Trabalho	<i>2</i>
1.3. Análise de viabilidade do processo	<i>2</i>
1.4. Recursos e Equipa de Trabalho	<i>2</i>
1.5. Plano de Execução do Projeto.....	<i>3</i>
2. Levantamento e Análise de Requisitos.....	<i>4</i>
2.1. Método de levantamento e de análise de requisitos adotado	<i>4</i>
2.2. Requisitos levantados	<i>4</i>
2.2.1. Requisitos de descrição	<i>4</i>
2.2.2. Requisitos de exploração.....	<i>5</i>
2.2.3. Requisitos de controlo.....	<i>5</i>
2.3. Análise e validação geral dos requisitos	<i>5</i>
3. Modelação Conceptual.....	<i>6</i>
3.1. Apresentação da abordagem de modelação realizada	<i>6</i>
3.2. Identificação e caracterização das entidades	<i>7</i>
3.3. Identificação e caracterização dos relacionamentos	<i>7</i>

3.4.	Identificação e caracterização da associação dos atributos com as entidades e relacionamentos	10
3.5.	Detalhe ou generalização de entidades	11
3.6.	Apresentação e explicação do diagrama ER.....	13
4.	<i>Modelação Lógica</i>	14
4.1.	Construção e validação do modelo de dados lógico	14
4.2.	Desenho do Modelo Lógico.....	14
4.3.	Normalização de Dados	15
4.4.	Validação do modelo com interrogações do utilizador (Alguns exemplos. No ponto 5.3 encontram-se o resto na totalidade).	17
4.5.	Revisão do modelo lógico produzido.....	17
5.	<i>Implementação Física.....</i>	18
5.1.	Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	18
5.2.	Povoamento da Base de Dados.....	23
5.3.	Tradução das interrogações do utilizador para SQL (alguns exemplos)	28
5.4.	Estimativa do espaço em disco da base de dados e taxa de crescimento anual	33
5.5.	Definição e caracterização das vistas de utilização em SQL	35
5.6.	Definição e caracterização de triggers em SQL.....	37
5.7.	Plano de segurança e recuperação de dados	38
6.	<i>Conceção e Implementação de um Sistema de Dados em MongoDB</i>	39
6.1.	Definição do Esquema da Base de Dados	39
6.2.	Criação da Base de Dados e Coleções	40
6.3.	Processo de Migração de Dados.....	42
6.4.	Exploração de Dados em MongoDB	43
7.	<i>Conclusões e Trabalho Futuro</i>	46
8.	<i>Referências Bibliográficas</i>	47

<i>Lista de Siglas e Acrónimos</i>	<i>48</i>
---	------------------

Índice de Figuras

FIGURA 1: LIVROS -> ENCOMENDA	8
FIGURA 2: FORNECEDOR -> LIVROS.....	9
FIGURA 3: MODELO CONCEPTUAL	13
FIGURA 4: MODELO LÓGICO	14
FIGURA 5: TABELA CLIENTE	18
FIGURA 6: TABELA FUNCIONÁRIO	19
FIGURA 7: TABELA VEÍCULO	19
FIGURA 8: TABELA ENCOMENDA.....	20
FIGURA 9: TABELA LIVROS	21
FIGURA 10: TABELA ENCOMENDA_HAS_LIVROS	21
FIGURA 11: TABELA FORNECEDOR.....	22
FIGURA 12: TABELA FORNECIMENTO_HAS_LIVROS.....	22
FIGURA 13: FUNÇÃO DE INSERÇÃO DE UM NOVO CLIENTE	23
FIGURA 14: FUNÇÃO DE INSERÇÃO DE UMA ENCOMENDA ASSIM COMO O SEU CONTEÚDO (FEITO NUMA TRANSAÇÃO COM UM LIMITE MÁXIMO DE 3 LIVROS POR CADA ENCOMENDA QUE ATUALIZA DUAS TABELAS AO MESMO TEMPO: ENCOMENDA E ENCOMENDA_HAS_LIVROS)	25
FIGURA 15: FUNÇÃO DE INSERÇÃO NA TABELA ENCOMENDA_HAS_LIVROS (SUBSTITUÍDA PELA FUNÇÃO INSEREENCOMENDAFINAL)	25
FIGURA 16: FUNÇÃO DE INSERÇÃO DE UM FUNCIONÁRIO.....	25
FIGURA 17: FUNÇÃO DE INSERÇÃO DE UM LIVRO	25
FIGURA 18: FUNÇÃO DE INSERÇÃO DE UM VEÍCULO	25
FIGURA 19: FUNÇÃO DE INSERÇÃO DE UM FORNECEDOR.....	26
FIGURA 20: FUNÇÃO DE INSERÇÃO NA TABELA FORNECIMENTO_HAS_LIVROS (FEITA NUMA TRANSAÇÃO DEVIDO A INSERÇÃO DE 1 OU MAIS ELEMENTOS NUMA TABELA)	26
FIGURA 21: EXEMPLO DE COMO TODAS AS FUNÇÕES SÃO CHAMADAS.....	27
FIGURA 22: EXEMPLO DE INSERÇÃO ANTES DE SEREM FEITAS AS FUNÇÕES	27
FIGURA 23: VISUALIZAÇÃO DE UMA TABELA	28
FIGURA 24: PROCURA DO CARRO DE UM FUNCIONÁRIO NUM DIA	28
FIGURA 25: TOP 3 CLIENTES COM MAIS ENCOMENDAS	29

FIGURA 26: 2 VEÍCULOS MAIS UTILIZADOS	29
FIGURA 27: ENTREGAS FEITAS PELOS FUNCIONÁRIOS NUM CERTO DIA	30
FIGURA 28: ENCOMENDAS ENTREGUES POR UM FUNCIONÁRIO NUM CERTO INTERVALO DE TEMPO ...	30
FIGURA 29: LIVRO MAIS VENDIDO	31
FIGURA 30: CONTEÚDO DE UMA ENCOMENDA.....	31
FIGURA 31: ENCOMENDAS DE UM CLIENTE.....	32
FIGURA 32: REQUISITOS DE CONTROLO.....	32
FIGURA 33: VISTA 1 - MÉDIA DOS PREÇOS DOS LIVROS EM STOCK	35
FIGURA 34: VISTA 2 - PREÇO DE TODAS AS ENCOMENDAS.....	35
FIGURA 35: VISTA 3 - INFORMAÇÃO DE ENCOMENDA (SEM REPETIR INFORMAÇÃO)	36
FIGURA 36: TRIGGER 1 - ATUALIZAR STOCK SEMPRE QUE SÃO FORNECIDOS LIVROS.....	37
FIGURA 37: TRIGGER 2 - ATUALIZAR STOCK SEMPRE QUE SÃO ENCOMENDADOS LIVROS	38
FIGURA 38: BACKUP DE SEGURANÇA DA BD	38
FIGURA 39: ESQUEMA DA BD	40
FIGURA 40: MIGRAÇÃO DOS DADOS (TRANSIÇÃO DA INFORMAÇÃO)	42
FIGURA 41: MIGRAÇÃO DOS DADOS (ESCRITA PARA UM FICHEIRO JSON).....	42

Índice de Tabelas

TABELA 1: IDENTIFICAÇÃO E CARACTERIZAÇÃO DAS ENTIDADES.....	7
TABELA 2: IDENTIFICAÇÃO E CARACTERIZAÇÃO DOS ATRIBUTOS ASSOCIADOS ÀS ENTIDADES E RELACIONAMENTOS.....	10
TABELA 3: DETALHE OU GENERALIZAÇÃO DE ENTIDADES.	11
TABELA 4: ESTIMATIVA DO ESPAÇO EM DISCO DA BASE DE DADOS	33
TABELA 5: CRESCIMENTO ANUAL DA BASE DE DADOS	34

1. Definição do Sistema

Com o intuito de facilitar a organização da livraria na gestão das entregas e de contribuir para a evolução deste negócio, pretende-se criar um sistema de Base de Dados capaz de suportar diversas reservas e permitir uma melhor organização. Esta Base de Dados deverá conseguir guardar informações sobre a venda, todos os funcionários, veículos e clientes. Iniciaremos esta implementação pela breve contextualização da situação, descrevendo assim, com mais detalhe, os motivos que levaram à criação deste sistema.

1.1. Contexto de aplicação e fundamentação do sistema

A livraria Silva era uma pequena livraria localizada em Viana do Castelo. A empresa é gerida pelo Sr. Alberto, que é responsável por todo o tipo de transações e organização da livraria.

Mas com a recente evolução tecnológica, para um possível aumento de lucro decidiu transitar a sua livraria física para um sistema online com serviço de entrega. Devido a este fator, o número de encomendas aumentou consideravelmente o que se tornou obsoleto guardar os dados em formato físico. O Sr. Alberto decidiu então que seria necessário criar uma base de dados para uma melhor organização e controlar facilmente a informação relativa a cada venda e stock de livros. Devido a esta transição para um sistema online foi também preciso arranjar vários funcionários para que fosse possível uma entrega ao domicílio.

O facto de não termos acesso a uma base de dados anterior levou-nos a fazer uma pesquisa para saber o que será necessário para o funcionamento da mesma.

Esta implica guardar diversas informações de maneira a poder gerir o registo de cada cliente e os requisitos que o mesmo pede. Posto isto, é necessário organizar os dados para que o sistema de informação tenha conhecimento dos tais clientes, como a sua preferência desde número de funcionários ao serviço como veículos disponíveis. O sistema deve armazenar, recolher e processar informações, de forma a garantir pesquisas de forma eficaz e válida. Como tal, para tornar isto possível é necessário implementar uma base de dados, pois esta é uma ferramenta de recolha e organização de informações, nada mais do que aquilo que procuramos para responder as necessidades requeridas.

1.2. Motivação e Objetivos do Trabalho

Gerir a livraria não só é muito cansativo para o dono como tem impacto ambiental significativo devido ao uso de papel, assim que a transição para a nossa base de dados serve como um meio de responder a este aumento de trabalho sentido na Livraria Silva e um possível lucro. Temos como objetivo criar um sistema de armazenamento de informação que permita uma pesquisa mais eficaz para fazer com que haja um aumento de produção suficiente para igualar o aumento de encomendas, fazendo assim que consequentemente haja um aumento de lucro e um melhor serviço na Livraria Silva que justifique a criação deste projeto.

1.3. Análise de viabilidade do processo

A informação relativa a cada uma das encomendas é guardada sob a forma de um registo que o dono guarda numa capa. Isto implica que o dono tem de procurar em centenas de páginas sempre que quer ver os próximos serviços a realizar ou analisar alguma informação passado.

Devido à maneira antiquada que atualmente é guardada a informação, espera-se observar quase de imediato um beneficiamento na qualidade do serviço com a implementação da base de dados. A digitalização das reservas irá reduzir no tempo investido da análise das informações das encomendas e na quantidade de papel desperdiçado. Além disso, o dono poderá utilizar a base de dados para saber novas informações como “quais o livro mais vendido” de modo a centralizar os recursos disponíveis e melhorar assim a qualidade do serviço prestado adequadamente.

Podemos concluir que a implementação da base de dados trará benefícios tanto à gestão do tempo do dono como à velocidade e simplicidade da manutenção da livraria e que é por isso quase uma necessidade.

1.4. Recursos e Equipa de Trabalho

Para o desenvolvimento deste projeto foram contratados 3 administradores de base de dados que em conjunto pretendem criar de maneira simples uma base de dados.

A equipa que realizou este projeto é constituída por:

- 4 administradores de base de dados;
- O dono da livraria

1.5. Plano de Execução do Projeto

Dividiram-se inicialmente em duas equipas, uma equipa observou a informação física armazenada pelo cliente, enquanto a outra observou diretamente o funcionamento da livraria. Após essa pesquisa e posterior reunião com o dono começaram a desenvolver a base de dados, passando por:

- Criação do modelo conceptual
- Revisão do modelo conceptual
- Criação do modelo lógico
- Revisão do modelo lógico
- Implementação do modelo físico

2. Levantamento e Análise de Requisitos

Este capítulo refere-se aos requisitos necessários para a elaboração deste projeto. O projeto retrata uma livraria virtual com um sistema de entrega. Já explicado o caso de estudo foi importante perceber que informação deveria a nossa Base de Dados guardar. Para isto, passamos ao levantamento e análise de requisitos.

2.1. Método de levantamento e de análise de requisitos adotado

Os requisitos relacionados com a livraria surgiram após uma reunião com o dono, observação do funcionamento da livraria e recolha seguida de posterior análise de todos os documentos físicos guardados pelo dono de modo a retirar possível informação que se pretende guardar.

2.2. Requisitos levantados

2.2.1. Requisitos de descrição

1. O sistema deverá guardar encomendas realizadas, bem como o seu estado (paga ou entregue), data em que foi pedida, data de entrega, o local onde será entregue e o tipo de pagamento;
2. Cada cliente deverá ser guardado na base de dados, assim como os dados relativos como o NIF, o Nome, Morada, Rua, Localidade e Contacto;
3. Os funcionários também deverão ser guardados na base de dados junto com o seu nome e contacto, tendo cada um, um identificador único;
4. Os veículos da empresa também têm de ser guardados na base de dados, identificados pela matrícula e dados relevantes do mesmo, como sua marca e modelo;
5. Os livros serão guardados numa tabela assim como as suas características como nome, tema, autor, preço e quantidade em stock;

6. Os fornecedores vão ser armazenados numa tabela com o seu identificador e nome respetivamente.

2.2.2. Requisitos de exploração

1. Apresentar todos os clientes, funcionários, veículos, fornecedores e encomendas;
2. Apresentar as encomendas de um dado cliente;
3. Apresentar encomendas entregue numa certa data por um funcionário;
4. Apresentar o desempenho de um funcionário num certo dia;
5. Apresentar os veículos mais utilizados;
6. Top 3 clientes;
7. Livro mais vendido;
8. Conteúdo de uma encomenda;
9. Valor de cada encomenda.

2.2.3. Requisitos de controlo

1. O Sr. Silva dono da Livraria terá permissão para visualizar toda a informação armazenada, terá permissão para adicionar novas encomendas, clientes, veículos, funcionários, contactos;
2. Todos os sábados é necessário efetuar uma cópia de segurança.
3. Todos os funcionários, uma vez implementados na base de dados vão ter permissões para visualizar a tabela '**Encomenda**' e alterar o seu estado de '**Paga**' para '**Entregue**' como também o contacto do cliente.

2.3. Análise e validação geral dos requisitos

Prevemos um bom funcionamento da base de dados face aos requisitos levantados, visamos as necessidades tanto a nível de armazenamento da informação como também a nível de consulta. Da mesma forma a integridade e proteção dos dados também é devidamente endereçada.

3. Modelação Conceptual

Após a definição de todos os requisitos necessários para a implementação da Base de Dados, podemos agora criar um Modelo Conceptual para a representação da mesma. Este modelo é fundamental para o desenvolvimento de qualquer Base de Dados, pois fornece uma visão aproximada de como os utilizadores realmente visualizam os dados.

3.1. Apresentação da abordagem de modelação realizada

Após a referida análise de requisitos, devemos seguir os seguintes passos:

1. Identificação das Entidades;
2. Identificação dos Relacionamentos;
3. Identificação e associação dos Atributos às Entidades;
4. Detalhar e generalizar as entidades;
5. Apresentação e explicação do diagrama ER.

3.2. Identificação e caracterização das entidades

Tabela 1: Identificação e caracterização das entidades

Entidades	Descrição	Ações
Funcionário	Quem está encarregue das encomendas	Um funcionário usa um veículo para fazer a entrega da encomenda atribuída.
Cliente	Quem efetua uma encomenda	Um cliente efetua um pedido de uma encomenda.
Fornecedor	Empresa(s) que fornece(m) todo os livros	O fornecedor fornece livros à livraria.
Encomenda	Encomenda é constituída por livros	É encomendada.
Livros	Livros em stock	Vão constituir uma encomenda.
Veículo	São usados pelos funcionários	Vai ser escolhido pelo funcionário para distribuir a encomenda.

3.3. Identificação e caracterização dos relacionamentos

- Um funcionário está associado a cada encomenda que entrega. Esta relação é de 1 para N, pois um funcionário pode ter entregue 0 ou mais encomendas;
- O cliente está relacionado com cada encomenda que pede. Cada cliente pode pedir 0 a N encomendas, portanto é uma relação 1 para N;
- Os livros estão associados à encomenda. É uma relação N para M uma vez que numa encomenda podem se encontrar vários livros e o mesmo livro (tem vários em stock) pode estar em várias encomendas.
- O veículo está relacionado com a encomenda. Podem ser utilizados 0 a N veículos, daí é uma relação 1 para N;
- O fornecedor está relacionado aos livros. É uma relação N para M pois, o mesmo livro pode ser fornecido por vários fornecedores e um fornecedor pode distribuir vários livros diferentes. Este também pode não fornecedor no momento.

Relacionamentos N para M:

⇒ Livros → Encomenda

O relacionamento “tem” entre a entidade ‘**Livros**’ e a entidade ‘**Encomenda**’ significa que numa encomenda tem de ter 1 ou mais livros pois para uma encomenda existir tem de conter livros. No entanto, o mesmo livro (pois existem cópias) pode estar em várias encomendas.

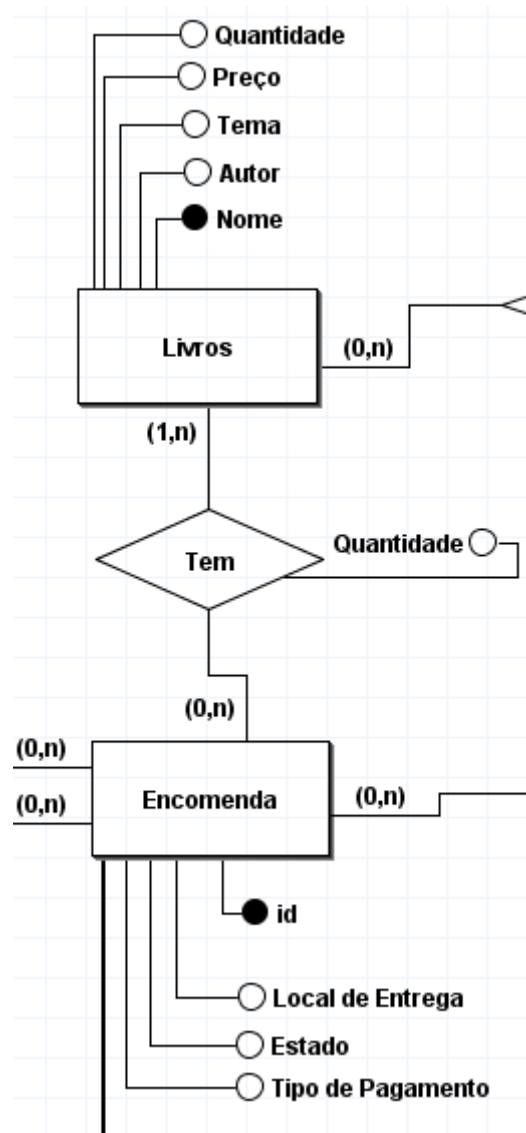


Figura 1: Livros -> Encomenda

⇒ **Fornecedor → Livros**

O relacionamento “fornece” entre a entidade ‘**Fornecedor**’ e ‘**Livros**’ representa a possibilidade de um fornecedor enviar livros para stock. Fornece 0 ou mais livros.

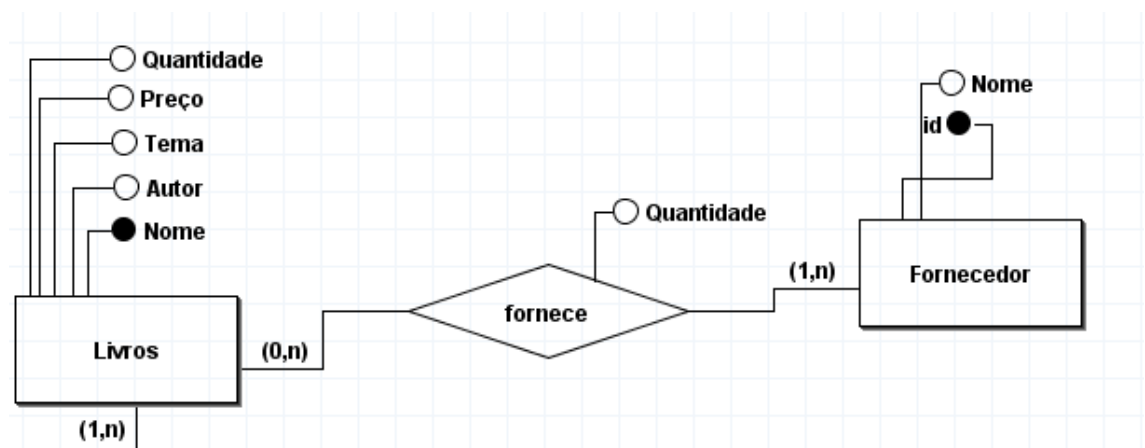


Figura 2: Fornecedor → Livros

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

Tabela 2: Identificação e caracterização dos atributos associados às entidades e relacionamentos.

Nome da entidade	Multiplicidade	Relação	Multiplicidade	Nome da entidade
Funcionário	(1, 1)	Entrega	(0, n)	Encomenda
Cliente	(1, 1)	Reserva	(0, n)	Encomenda
Encomenda	(0, n)	Tem	(1, n)	Livros
Veículo	(1, 1)	Utilizado	(0, n)	Encomenda
Fornecedor	(1, n)	Fornece	(0, n)	Livros

3.5. Detalhe ou generalização de entidades

Tabela 3: Detalhe ou generalização de entidades.

Nome da Entidade	Atributos	Descrição	Tipo de Dados	Null	Multivalorado	Derivado
Funcionário	ID(PK)	Identificador único do Funcionário	INT	Não	Não	Não
	Nome	Nome do Funcionário	VARCHAR(45)	Não	Não	Não
	Contacto	Contacto do funcionário	INT	Não	Não	Não
Cliente	NIF(PK)	Identificador do cliente. Número de identificação fiscal	INT	Não	Não	Não
	Nome	Nome do cliente	VARCHAR(45)	Não	Não	Não
	Contacto	Contacto do cliente	INT	Não	Não	Não
	Rua	Rua onde vive o cliente	VARCHAR(90)	Não	Não	Não
	Localidade	Localidade do cliente	VARCHAR(30)	Não	Não	Não
	Código de Postal	Código de postal do cliente	CHAR(8)	Não	Não	Não
Encomenda	ID(PK)	Identificador da encomenda	INT	Não	Não	Não
	Local de Entrega	Local de entrega da encomenda	VARCHAR(100)	Não	Não	Não
	Estado	Estado da encomenda	VARCHAR(15)	Não	Não	Não
	Tipo de pagamento	Tipo de pagamento	VARCHAR(15)	Não	Não	Não
	Data de Encomenda	Data em que foi encomendada	DATE	Não	Não	Não
	Data de Entrega	Data de Entrega	DATE	Não	Não	Não
	Matrícula (PK)	Identificador do veículo	CHAR(8)	Não	Não	Não

Veículo	Marca	Marca do veículo	VARCHAR(20)	Não	Não	Não
	Modelo	Modelo do veículo	VARCHAR(20)	Não	Não	Não
Livros	Nome(PK)	Nome do livro	VARCHAR(45)	Não	Não	Não
	Tema	Tema do livro	VARCHAR(45)	Não	Não	Não
	Autor	Autor do livro	VARCHAR(45)	Não	Não	
	PreçoL	Preço do livro	FLOAT	Não	Não	Não
	Quantidade	Quantidade em stock de um livro	INT	Não	Não	Não
Fornecedor	ID(PK)	Identificador do fornecedor	INT	Não	Não	Não
	Nome	Nome do fornecedor	VARCHAR(45)	Não	Não	Não

3.6. Apresentação e explicação do diagrama ER

Utilizando a ferramenta “**brModelo**” foi elaborado o diagrama ER abaixo apresentado.

Fazendo uma análise mais detalhada do diagrama podemos confirmar que todos os requisitos de exploração anteriormente mencionados conseguem serem respondidos através deste. Agora vamos falar de chaves candidatas.

Algumas entidades têm duas chaves candidatas, como por exemplo o **ID** e o **Nome**. Contudo, a escolha do ID para chave primaria é bastante óbvia para algumas entidades tendo em conta que, para todos os efeitos, terá um tamanho mais pequeno do que o nome de utilizador, levando a uma procura mais rápida e eficaz. A entidade ‘**Encomenda**’ já está classificada com a chave primária ID, pois é a única candidata. Outras entidades estão classificadas com um outro tipo de chave primária como o **NIF** no caso do cliente, a **Matrícula** no caso do veículo, e o **Nome** no caso do livro.

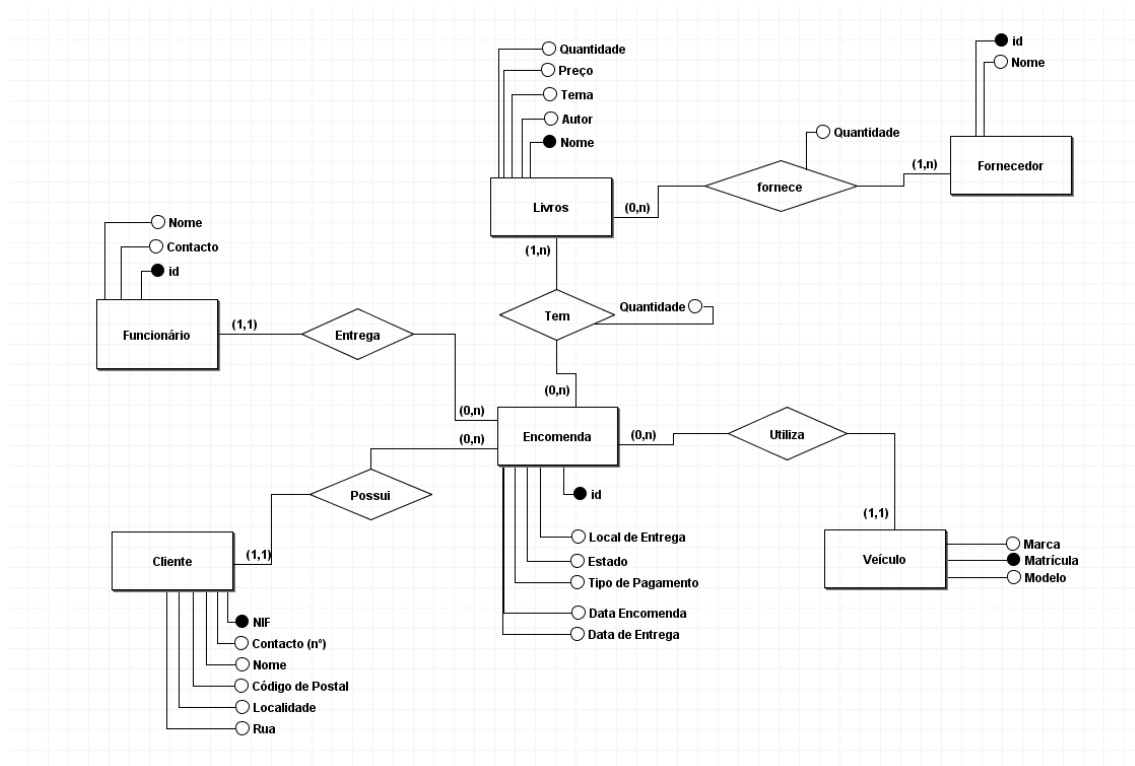


Figura 3: Modelo Conceptual

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Construindo o Modelo Conceptual, temos de passar então para a Construção do Modelo Lógico. Primeiro, começamos por criar as tabelas para cada entidade do Modelo anterior. De seguida, procedemos ao preenchimento de cada tabela com os respetivos atributos definidos no Modelo Conceptual.

4.2. Desenho do Modelo Lógico

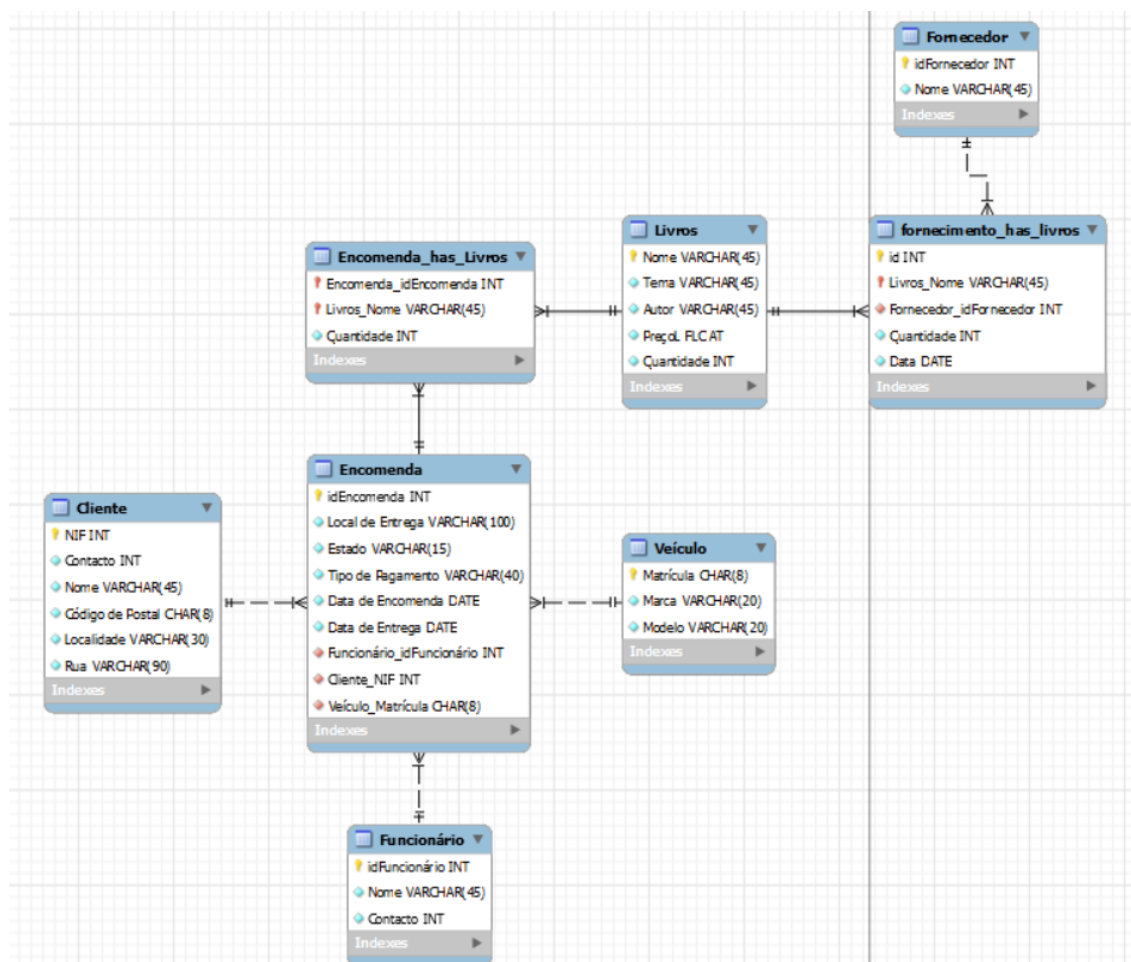


Figura 4: Modelo Lógico

4.3. Normalização de Dados

A normalização dos dados tem como objetivo eliminar a redundância dos dados, aumentando consequentemente o desempenho da mesma. Para se conseguir normalizar é necessário identificar as dependências entre os atributos.

- Id (Cliente) – Contacto, Nome, Código de Postal, Localidade, Rua;
- Id (Encomenda) – Local de Entrega, Estado, Tipo de Pagamento, Data de Encomenda, Data de Entrega, Funcionário, Cliente, Veículo;
- Id (Funcionário) – Nome, Contacto;
- Id (Veículo) – Marca, Modelo;
- Id (Livros) – Tema, Autor, Preço, Quantidade;
- Id (Fornecedor) – Nome
- Encomenda, Livros – Quantidade
- Fornecedor, Livros – Quantidade, Data, Livros, Fornecedor

Primeira Forma Normal (1ª FN):

1. Temos a 1ªFN presente, se todos os valores da tabela forem atômicos;
2. Se todos os atributos da relação não se encontram repetidos.

Segunda Forma Normal (2ª FN):

1. Precisa necessariamente de estar na 1ªFN;
2. Todos os atributos não primos, dependem da chave-primária.

Terceira Forma Normal (3ª FN):

1. Precisa necessariamente de estar na 2ªFN;
2. Todos os atributos não primos, dependem da chave-primária sem depender de nenhum outro que dependa da chave-primária.

Após o que foi apresentado acima, conclui-se que o modelo lógico respeita as **três** regras:

1ª FN - Em todas as tabelas, a chave-primária corresponde a uma única linha da tabela, verificando-se por exemplo na tabela '**Encomenda**', em que cada ID (chave-primária) é único. Nas tabelas em que há chave-primárias compostas, como por exemplo, a tabela '**Encomenda_has_livros**', temos a certeza que respeita a primeira forma, justificado pela transitividade, ou seja, como são provenientes de outra tabela, são a chave-primária e respeitam a **1ªFN**, então as chaves compostas também são únicas;

2ª FN - Não existe dependência parcial no modelo lógico;

3ª FN - Verificou-se que não havia dependências transitivas e sem redundância nas tabelas.

4.4. Validação do modelo com interrogações do utilizador (Alguns exemplos. No ponto 5.3 encontram-se o resto na totalidade).

- **Saber quais os clientes que efetuaram mais encomendas**

Para saber quantas encomendas foram feitas por um cliente, é necessário a junção das tabelas '*Cliente*' e '*Encomenda*'. Efetua-se um *count* e um agrupamento para saber quantas reservas efetuou o cliente com o NIF, comparando o atributo **Cliente** da tabela '*Encomenda*' com o atributo **NIF** da tabela '*Cliente*'.

- **Saber os livros em stock**

Para saber os livros que se encontram em stock basta retornar todas as entradas da tabela '*Livro*' cuja quantidade é maior que zero.

- **Saber quais os livros mais vendidos**

Para saber quantas vezes os livros foram requisitados na Livraria é necessário juntar as tabelas '*Livros*' e '*Encomenda_has_livros*'. Primeiro iremos à tabela '*Livros*' procurar o seu **Nome**, seguidamente vamos aceder à tabela '*Encomenda_has_livros*', e vamos fazer uma soma da quantidade de todos os livros, agrupando pelo nome, organizando de forma decrescente de forma a conseguir saber a quantidade total vendida.

4.5. Revisão do modelo lógico produzido

Concluída esta fase, foi organizada uma nova reunião para a validação deste modelo e para a autorização do avanço para a próxima fase. Depois de se verificar que todas as entidades do modelo conceptual estavam presentes no modelo lógico, também se verificou se todos os atributos estavam presentes e com o domínio adequado de maneira a conseguirmos armazenar toda a informação necessária.

Finalmente, foram efetuados alguns testes para termos certeza se a nossa base de dados correspondia ao que era pedido. Após alguns erros nas *queries* e devida correção, verificou-se que tudo estava a funcionar bem e que estavam a ser cumpridos os objetivos decidimos avançar para a próxima fase (**Implementação Física**).

5.Implementação Física

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

Utilizamos a ferramenta de desenho, desenvolvimento e administração de bases de dados **MySQL Workbench** de maneira a usufruirmos de todos os mecanismos que o MySQL oferece. Tendo em conta que o nosso modelo lógico foi criado nesta ferramenta, podemos utilizar o “**Forward Engineering**” que permite automatizar o processo *top-down* de construção de componentes de baixo nível (implementação no sistema de gestão de base de dados) através de uma abstração de alto nível (modelo lógico).

- Criação da Tabela Cliente

```
-- Table `LivrariaSilva`.`Cliente`  
-----  
CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Cliente` (  
  `NIF` INT NOT NULL,  
  `Contacto` INT NOT NULL,  
  `Nome` VARCHAR(45) NOT NULL,  
  `Código de Postal` CHAR(8) NOT NULL,  
  `Localidade` VARCHAR(30) NOT NULL,  
  `Rua` VARCHAR(90) NOT NULL,  
  PRIMARY KEY (`NIF`))  
ENGINE = InnoDB;
```

Figura 5: Tabela Cliente

- Criação da Tabela Funcionário

```
-----  
-- Table `LivrariaSilva`.`Funcionário`  
-----  
CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Funcionário` (  
  `idFuncionário` INT NOT NULL,  
  `Nome` VARCHAR(45) NOT NULL,  
  `Contacto` INT NOT NULL,  
  PRIMARY KEY (`idFuncionário`))  
ENGINE = InnoDB;
```

Figura 6: Tabela Funcionário

- Criação da Tabela Veículo

```
-----  
-- Table `LivrariaSilva`.`Veículo`  
-----  
CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Veículo` (  
  `Matrícula` CHAR(8) NOT NULL,  
  `Marca` VARCHAR(20) NOT NULL,  
  `Modelo` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`Matrícula`))  
ENGINE = InnoDB;
```

Figura 7: Tabela Veículo

- Criação da Tabela Encomenda

```
• CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Encomenda` (  
  `idEncomenda` INT NOT NULL,  
  `Local de Entrega` VARCHAR(100) NOT NULL,  
  `Estado` VARCHAR(15) NOT NULL,  
  `Tipo de Pagamento` VARCHAR(15) NOT NULL,  
  `Data de Encomenda` DATE NOT NULL,  
  `Data de Entrega` DATE NOT NULL,  
  `Funcionário_idFuncionário` INT NOT NULL,  
  `Cliente_NIF` INT NOT NULL,  
  `Veículo_Matrícula` CHAR(8) NOT NULL,  
  PRIMARY KEY (`idEncomenda`),  
  INDEX `fk_Encomenda_Funcionário1_idx` (`Funcionário_idFuncionário` ASC) VISIBLE,  
  INDEX `fk_Encomenda_Cliente1_idx` (`Cliente_NIF` ASC) VISIBLE,  
  INDEX `fk_Encomenda_Veículo1_idx` (`Veículo_Matrícula` ASC) VISIBLE,  
  CONSTRAINT `fk_Encomenda_Funcionário1`  
    FOREIGN KEY (`Funcionário_idFuncionário`)  
    REFERENCES `LivrariaSilva`.`Funcionário` (`idFuncionário`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Encomenda_Cliente1`  
    FOREIGN KEY (`Cliente_NIF`)  
    REFERENCES `LivrariaSilva`.`Cliente` (`NIF`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Encomenda_Veículo1`  
    FOREIGN KEY (`Veículo_Matrícula`)  
    REFERENCES `LivrariaSilva`.`Veículo` (`Matrícula`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Figura 8: Tabela Encomenda

- Criação da Tabela Livros

```

-----
-- Table `LivrariaSilva`.`Livros`
-----

CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Livros` (
  `Nome` VARCHAR(45) NOT NULL,
  `Tema` VARCHAR(45) NOT NULL,
  `Autor` VARCHAR(45) NOT NULL,
  `PreçoL` FLOAT NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`Nome`))
ENGINE = InnoDB;

```

Figura 9: Tabela Livros

- Criação da Tabela Encomenda_has_livros

```

-----
-- Table `LivrariaSilva`.`Encomenda_has_Livros`
-----

CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Encomenda_has_Livros` (
  `Encomenda_idEncomenda` INT NOT NULL,
  `Livros_Nome` VARCHAR(45) NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`Encomenda_idEncomenda`, `Livros_Nome`),
  INDEX `fk_Encomenda_has_Livros_Livros1_idx` (`Livros_Nome` ASC) VISIBLE,
  INDEX `fk_Encomenda_has_Livros_Encomenda1_idx` (`Encomenda_idEncomenda` ASC) VISIBLE,
  CONSTRAINT `fk_Encomenda_has_Livros_Encomenda1`
    FOREIGN KEY (`Encomenda_idEncomenda`)
      REFERENCES `LivrariaSilva`.`Encomenda` (`idEncomenda`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Encomenda_has_Livros_Livros1`
    FOREIGN KEY (`Livros_Nome`)
      REFERENCES `LivrariaSilva`.`Livros` (`Nome`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 10: Tabela Encomenda_has_Livros

- Criação da Tabela Fornecedor

```

-----
-- Table `LivrariaSilva`.`Fornecedor`
-----
> CREATE TABLE IF NOT EXISTS `LivrariaSilva`.`Fornecedor` (
  `idFornecedor` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idFornecedor`))
ENGINE = InnoDB;

```

Figura 11: Tabela Fornecedor

- Criação da Tabela fornecimento_has_livros

```

CREATE TABLE IF NOT EXISTS `Livraria`.`fornecimento_has_livros` (
  `id` INT NOT NULL,
  `Livros_Nome` VARCHAR(45) NOT NULL,
  `Fornecedor_idFornecedor` INT NOT NULL,
  `Quantidade` INT NOT NULL,
  `Data` DATE NOT NULL,
  PRIMARY KEY (`id`, `Livros_Nome`),
  INDEX `fk_Livros_has_Fornecedor_Fornecedor1_idx` (`Fornecedor_idFornecedor` ASC) VISIBLE,
  INDEX `fk_Livros_has_Fornecedor_Livros1_idx` (`Livros_Nome` ASC) VISIBLE,
  CONSTRAINT `fk_Livros_has_Fornecedor_Livros1`
    FOREIGN KEY (`Livros_Nome`)
      REFERENCES `Livraria`.`Livros` (`Nome`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Livros_has_Fornecedor_Fornecedor1`
    FOREIGN KEY (`Fornecedor_idFornecedor`)
      REFERENCES `Livraria`.`Fornecedor` (`idFornecedor`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 12: Tabela fornecimento_has_livros

5.2. Povoamento da Base de Dados

Inicialmente fomos povoando a base de dados à mão com valor a valor, mas depois para uma utilização mais rápida e simples definimos funções para conseguirmos adicionar valores com uma melhor otimização. Optámos também por duas transações, uma para inserção de informação na encomenda e também na relação onde está presente o seu conteúdo, a outra foi feita para inserir um fornecimento com vários livros.

```
DELIMITER $$  
CREATE PROCEDURE `insereCliente` (IN NIF INT, IN Contacto INT, IN Nome VARCHAR(45), IN `Código de Postal` CHAR(8), IN Localidade VARCHAR(30), IN Rua VARCHAR(90))  
BEGIN  
    INSERT INTO cliente (NIF, Contacto, Nome, `Código de Postal`, Localidade, Rua)  
    VALUES (NIF, Contacto, Nome, `Código de Postal`, Localidade, Rua);  
END $$
```

Figura 13: Função de inserção de um novo cliente

Tanto as funções **'insereEncomendaFinal'** e **'insereFornecimentoFinal'** foram feitas numa forma de transação pois ambas elas vão adicionar informação a várias tabelas (entidades) com um a vários livros.

A primeira, respetivamente, verifica o stock dos livros e compara com os livros encomendados. De seguida vai colocar informação tanto na entidade **'Encomenda'** como na entidade **'Encomenda_has_livros'**. Se o livro não estiver disponível, este não vai fazer qualquer alteração e não vai adicionar à encomenda.

A segunda vai permitir a um fornecedor fazer um fornecimento de vários livros que vão ser adicionados na tabela **'fornecimento_has_livros'**. Em caso de problema esta operação vai ser cancelada com um **'rollback'**.


```

DELIMITER $$

CREATE PROCEDURE `insereEncomendaFinal` (IN idEncomenda INT,
IN `Local de Entrega` VARCHAR(100), IN Estado VARCHAR(15),
IN `Tipo de Pagamento` VARCHAR(40), IN `Data de Encomenda` DATE,
IN `Data de Entrega` DATE, IN Funcionário_idFuncionário INT, IN Cliente_NIF INT, IN Veiculo_Matricula CHAR(8),
IN Livro1 VARCHAR(40), IN Qnt1 INT, IN Livro2 VARCHAR(40), IN Qnt2 INT, IN Livro3 VARCHAR(40), IN Qnt3 INT)

BEGIN

DECLARE var1 INT DEFAULT 0;
DECLARE var2 INT DEFAULT 0;
DECLARE var3 INT DEFAULT 0;

DECLARE ErroTransacao BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET ErroTransacao = 1;

START TRANSACTION;

SELECT livros.Quantidade from livros
WHERE livros.Nome = Livro1 Into var1;

SELECT livros.Quantidade from livros
WHERE livros.Nome = Livro2 Into var2;

SELECT livros.Quantidade from livros
WHERE livros.Nome = Livro3 Into var3;

IF (Livro1 != '0' AND Qnt1 != 0 AND Livro2 = '0' AND Qnt2 = 0 AND Livro3 = '0' AND Qnt3 = 0 AND var1 >= Qnt1)
THEN INSERT INTO encomenda(idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`, `Data de Encomenda`,
`Data de Entrega`, Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula)
VALUE (idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`, `Data de Encomenda`, `Data de Entrega`,
Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula);

INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro1, Qnt1);

ELSE IF (Livro1 != '0' AND Qnt1 != 0 AND Livro2 != '0' AND Qnt2 != 0 AND Livro3 = '0' AND Qnt3 = 0 AND var1 >= Qnt1 AND var2 >= Qnt2)
THEN INSERT INTO encomenda(idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`, `Data de Encomenda`,
`Data de Entrega`, Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula)
VALUE (idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`, `Data de Encomenda`,
`Data de Entrega`, Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula);

INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro1, Qnt1);

INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro2, Qnt2);

ELSE IF (Livro1 != '0' AND Qnt1 != 0 AND Livro2 != '0' AND Qnt2 != 0 AND Livro3 != '0' AND Qnt3 != 0 AND var1 >= Qnt1 AND var2 >= Qnt2 AND var3 >= Qnt3)
THEN INSERT INTO encomenda(idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`,
`Data de Encomenda`, `Data de Entrega`, Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula)
VALUE (idEncomenda, `Local de Entrega`, Estado, `Tipo de Pagamento`, `Data de Encomenda`,
`Data de Entrega`, Funcionário_idFuncionário, Cliente_NIF, Veiculo_Matricula);

INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro1, Qnt1);
INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro2, Qnt2);
INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
VALUE (idEncomenda, Livro3, Qnt3);

END IF;
END IF;
END IF;
IF ErroTransacao THEN
ROLLBACK;
ELSE
COMMIT;
END IF;

END $$

```

Figura 14: Função de inserção de uma encomenda assim como o seu conteúdo (Feito numa transação com um limite máximo de 3 livros por cada encomenda que atualiza duas tabelas ao mesmo tempo: Encomenda e Encomenda_has_livros)

```
DELIMITER $$
CREATE PROCEDURE `insereEncomenda_has_livros` (IN Encomenda_idEncomenda INT, IN Livros_Nome VARCHAR(45), IN Quantidade INT)
BEGIN
    INSERT INTO encomenda_has_livros (Encomenda_idEncomenda, Livros_Nome, Quantidade)
    VALUE (Encomenda_idEncomenda, Livros_Nome, Quantidade);
END $$
```

Figura 15: Função de inserção na tabela Encomenda_has_Livros (Substituída pela função insereEncomendaFinal)

```
DELIMITER $$
CREATE PROCEDURE `insereFuncionário` (IN idFuncionário INT, IN Nome VARCHAR(45), IN Contacto INT)
BEGIN
    INSERT INTO funcionário (idFuncionário, Nome, Contacto)
    VALUE (idFuncionário, Nome, Contacto);
END $$
```

Figura 16: Função de inserção de um funcionário

```
DELIMITER $$
CREATE PROCEDURE `insereLivro` (IN Nome VARCHAR(45), IN Tema VARCHAR(45), IN Autor VARCHAR(45), IN PreçoL FLOAT, IN Quantidade INT)
BEGIN
    INSERT INTO livros (Nome, Tema, Autor, PreçoL, Quantidade)
    VALUE (Nome, Tema, Autor, PreçoL, Quantidade);
END $$
```

Figura 17: Função de inserção de um livro

```
DELIMITER $$
CREATE PROCEDURE `insereVeículo` (IN Matrícula CHAR(8), IN Marca VARCHAR(20), IN Modelo VARCHAR(20))
BEGIN
    INSERT INTO veículo (Matrícula, Marca, Modelo)
    VALUE (Matrícula, Marca, Modelo);
END $$
```

Figura 18: Função de inserção de um veículo

```

DELIMITER $$
CREATE PROCEDURE `insereFornecedor` (IN idFornecedor INT, IN Nome VARCHAR(45))
BEGIN
    INSERT INTO fornecedor (idFornecedor, Nome)
    VALUE (idFornecedor, Nome);
END $$

```

Figura 19: Função de inserção de um fornecedor

```

DELIMITER $$
CREATE PROCEDURE `insereFornecimentoFinal` (IN id INT, IN Livro1 VARCHAR(45), IN Livro2 VARCHAR(45), IN Livro3 VARCHAR(45),
IN Qnt1 INT, IN Qnt2 INT, IN Qnt3 INT, IN Fornecedor_idFornecedor INT, IN `Data` DATE)
BEGIN

    DECLARE ErroTransacao BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET ErroTransacao = 1;

    START TRANSACTION;

    IF (Livro1 != '0' AND Livro2 = '0' AND Livro3 = '0' AND Qnt1 > 0)
    THEN
        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro1, Fornecedor_idFornecedor, Qnt1, `Data`);

    ELSE IF (Livro1 != '0' AND Livro2 != '0' AND Livro3 = '0' AND Qnt1 > 0 AND Qnt2 > 0)
    THEN
        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro1, Fornecedor_idFornecedor, Qnt1, `Data`);

        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro2, Fornecedor_idFornecedor, Qnt2, `Data`);

    ELSE IF (Livro1 != '0' AND Livro2 != '0' AND Livro3 != '0' AND Qnt1 > 0 AND Qnt2 > 0 AND Qnt3 > 0 )
    THEN
        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro1, Fornecedor_idFornecedor, Qnt1, `Data`);

        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro2, Fornecedor_idFornecedor, Qnt2, `Data`);

        INSERT INTO fornecimento_has_livros (id, Livros_Nome, Fornecedor_idFornecedor, Quantidade, `Data`)
        VALUE (id, Livro3, Fornecedor_idFornecedor, Qnt3, `Data`);

    END IF;
END IF;
IF ErroTransacao THEN
    ROLLBACK;
ELSE
    COMMIT;
END IF;
END $$

```

Figura 20: Função de inserção na tabela fornecimento_has_livros (Feita numa transação devido a inserção de 1 ou mais elementos numa tabela)

```

#Função de inserção de uma encomenda assim como o seu conteúdo
#Recebe: (ID,Rua,Estado,TipoPagamento,DataquefoiEncomendada,DataEntrega,idFuncionario,NifCliente,Matricula,Li
#Se a encomenda tiver um livro, colocar no Livro2,Qnt2,Livro3,Qnt3, 0 como parametro
#Se a encomenda tiver dois livros, colocar no Livro3,Qnt3, 0 como parametro
#Se a encomenda tiver tres livros, preencher todos os parametros normalmente
#Limite maximo 3 livros por encomenda, implementado para controlar melhor e organizar
call insereEncomendaFinal(24,'Rua do Amongo','Entregue','Dogecoin','2000-07-22','2000-07-23', 2,126434219 , 'I

#Função de inserção de fornecimentos antiga(Funciona mas nao se deve usar para evitar problemas)
call insereFornecimentoHasLivros(10,'Chainsaw Man', 2, 1, '2023-01-10');

#Função de inserção de um cliente
call inserecliente(123456789, 123456789, 'teste','1234-124','teste','teste');

#Função de inserção de uma encomenda antiga(Funciona mas nao se deve usar para evitar problemas)
call insereEncomenda(14,'Rua do Sésamo','Paga','MBway','2000-07-21','2000-07-23', 4, 254876589, 'UO-57-12');

#Função para inserir o conteúdo de uma encomenda antiga(Nao usar para evitar problemas)
call insereEncomenda_has_livros(14,'Chainsaw Man',1);

```

Figura 21: Exemplo de como todas as funções são chamadas

```

INSERT INTO livros
(NOME, Tema, Autor, PreçoL, Quantidade)
VALUES
('Chainsaw Man', 'Ação/Drama', 'Tatsuki Fujimoto', 9.99, 21),
('Dune', 'Ficção Científica', 'Frank Herbert', 13.99, 50),
('Little Women', 'Ficção Doméstica', 'Louisa May Alcott', 15.99, 35),
('Neon Genesis Evangelion', 'Drama Psicológico/Apocalíptico', 'Akio Satsukawa', 20, 105),
('Harry Potter - Phoenix', 'Drama/Fantasia', 'J. K. Rowling', 15.99, 200),
('Harry Potter - The Squid Master', 'Comédia/Fantasia', 'J. K. Rowling', 9.99, 10),
('The Rabbit Hole', 'Fantasia', 'John Lennon Jr.', 11.45, 32),
('Suspect', 'Drama/Mistério', 'João Vermelho', 12.42, 20);

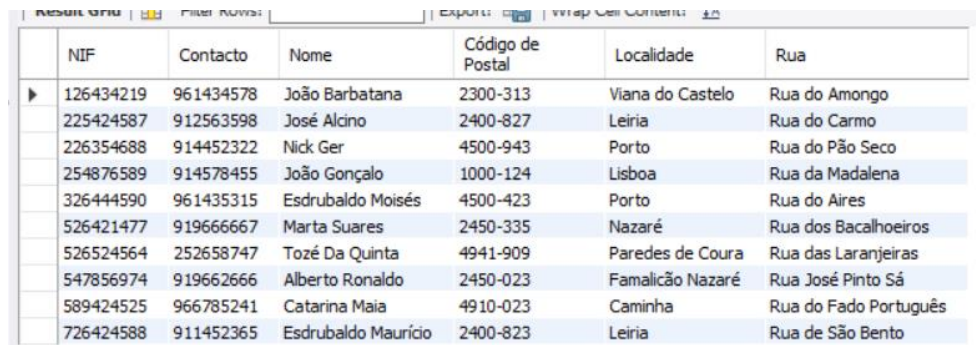
```

Figura 22: Exemplo de inserção antes de serem feitas as funções

5.3. Tradução das interrogações do utilizador para SQL (alguns exemplos)

- Exemplo de visualização de uma tabela

```
DELIMITER $$  
CREATE PROCEDURE `visualizarClientes`()  
BEGIN  
    SELECT * FROM cliente;  
END $$  
  
call visualizarClientes();
```

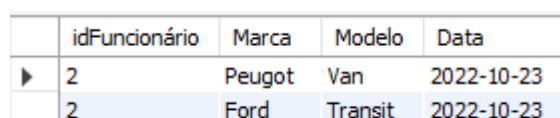


	NIF	Contacto	Nome	Código de Postal	Localidade	Rua
▶	126434219	961434578	João Barbatana	2300-313	Viana do Castelo	Rua do Amongo
	225424587	912563598	José Alcino	2400-827	Leiria	Rua do Carmo
	226354688	914452322	Nick Ger	4500-943	Porto	Rua do Pão Seco
	254876589	914578455	João Gonçalo	1000-124	Lisboa	Rua da Madalena
	326444590	961435315	Esdrubaldo Moisés	4500-423	Porto	Rua do Aires
	526421477	919666667	Marta Suares	2450-335	Nazaré	Rua dos Bacalhoeiros
	526524564	252658747	Tozé Da Quinta	4941-909	Paredes de Coura	Rua das Laranjeiras
	547856974	919662666	Alberto Ronaldo	2450-023	Famalicão Nazaré	Rua José Pinto Sá
	589424525	966785241	Catarina Maia	4910-023	Caminha	Rua do Fado Português
	726424588	911452365	Esdrubaldo Maurício	2400-823	Leiria	Rua de São Bento

Figura 23: Visualização de uma Tabela

- Procurar o carro que um funcionário utilizou num certo dia

```
DELIMITER $$  
Create procedure verCarroFuncionarioNumaData (IN id INT, IN `Data` DATE)  
BEGIN  
    SELECT encomenda.Funcionário_idFuncionário as idFuncionário, veículo.Marca as Marca, veículo.Modelo as Modelo, encomenda.`Data de Entrega` as `Data` from encomenda  
    inner join veículo  
    on encomenda.Veículo_Matricula = veículo.Matricula  
    WHERE `Data`= encomenda.`Data de Entrega`  
    AND id = encomenda.Funcionário_idFuncionário;  
END $$  
  
call verCarroFuncionarioNumaData(2,'2022-10-23')
```



	idFuncionário	Marca	Modelo	Data
▶	2	Peugot	Van	2022-10-23
	2	Ford	Transit	2022-10-23

Figura 24: Procura do carro de um funcionário num dia

- Verificar o top 3 clientes com mais encomendas

```
DELIMITER $$
CREATE PROCEDURE `top3clientes`()
BEGIN
    SELECT cliente.NIF, cliente.Nome, count(encomenda.Cliente_NIF) as 'Encomendas Feita' from cliente
    INNER join encomenda
    ON cliente.NIF = encomenda.Cliente_NIF
    GROUP BY cliente.NIF
    ORDER BY 3 DESC
    LIMIT 3;
END $$
```

	NIF	Nome	Encomendas Feita
▶	254876589	João Gonçalo	3
	726424588	Esdrubaldo Maurício	2
	526421477	Marta Suares	2

Figura 25: Top 3 clientes com mais encomendas

- Verificar os 2 veículos mais utilizados

```
DELIMITER $$
CREATE PROCEDURE `veiculosMaisUtilizados` ()
) BEGIN
    SELECT encomenda.Veículo_Matrícula, count(*) AS NrEntregas FROM encomenda
    GROUP BY encomenda.Veículo_Matrícula
    ORDER BY 2 DESC
    LIMIT 2;
- END $$
```

	Veículo_Matrícula	NrEntregas
▶	UO-57-12	6
	25-KL-69	5

Figura 26: 2 veículos mais utilizados

- Quantas entregas foram feitas num dia pelos funcionários num certo dia

```
DELIMITER $$
CREATE PROCEDURE `entregasDosFuncionarios` (IN dia DATE)
BEGIN
    SELECT encomenda.Funcionário_idFuncionário, count(*) AS NrEntregas FROM encomenda
    WHERE encomenda.`Data de Entrega` = dia
    GROUP BY encomenda.Funcionário_idFuncionário
    ORDER BY 1;
END $$
```

	Funcionário_idFuncionário	NrEntregas
▶	1	1
	4	2

Figura 27: Entregas feitas pelos funcionários num certo dia

- Encomendas entregues por um funcionário num certo intervalo de tempo

```
DELIMITER $$
CREATE PROCEDURE `encomendasEntreguesPorFuncionario` (IN idFuncionario INT, IN dataInicial DATE, IN dataFinal DATE)
BEGIN
    SELECT * FROM encomenda
    WHERE encomenda.Funcionário_idFuncionário=idFuncionario
    AND encomenda.Estado='Entregue'
    AND dataInicial <= encomenda.`Data de Entrega`
    AND dataFinal >= encomenda.`Data de Entrega`;
END $$
```

	idEncomenda	Local de Entrega	Estado	Tipo de Pagamento	Data de Encomenda	Data de Entrega	Funcionário_idFuncionário	Cliente_NIF	Veículo_Matricula
▶	7	Rua do Fado Português	Entregue	Bitcoin	2022-10-21	2022-10-22	1	589424525	UO-57-12
	11	Rua do Pão Seco	Entregue	Cartão de Crédito	2022-10-20	2022-10-21	1	226354688	UO-57-12

Figura 28: Encomendas entregues por um funcionário num certo intervalo de tempo

- Livro mais vendido

```
DELIMITER $$
CREATE PROCEDURE livroMaisVendido()
BEGIN
    SELECT livros.Nome, sum(encomenda_has_livros.Quantidade) as Quantidade from Livros
    INNER JOIN encomenda_has_livros ON encomenda_has_livros.Livros_Nome = Livros.Nome
    GROUP by livros.Nome
    ORDER by 2 DESC
    LIMIT 1;
END $$
```

	Nome	Quantidade
▶	Neon Genesis Evangelion	152

Figura 29: Livro mais vendido

- Conteúdo de uma encomenda

```
DELIMITER $$
CREATE PROCEDURE livrosNumaEncomenda (IN encomenda INT)
BEGIN
    SELECT encomenda_has_livros.Encomenda_idEncomenda,
    encomenda_has_livros.Livros_Nome, encomenda_has_livros.Quantidade
    FROM encomenda_has_livros
    WHERE Encomenda = encomenda_has_livros.Encomenda_idEncomenda;
END $$
```

	Encomenda_idEncomenda	Livros_Nome	Quantidade
▶	1	Chainsaw Man	2
	1	Dune	1

Figura 30: Conteúdo de uma encomenda

- Encomendas por um dado cliente

```

DELIMITER $$
CREATE PROCEDURE `encomendasDeCliente` (IN NIF INT)
BEGIN
    SELECT * FROM encomenda
    WHERE encomenda.Cliente_NIF=NIF
    ORDER BY 1 DESC;
END $$

```

	idEncomenda	Local de Entrega	Estado	Tipo de Pagamento	Data de Encomenda	Data de Entrega	Funcionário
▶	13	Rua do Sésamo	Entregue	MBway	2000-07-21	2000-07-23	4
	2	Rua da Madalena	Paga	Cartão de Crédito	2022-10-22	2022-10-23	1
	1	Rua da Madalena	Paga	Paysafecard	2022-10-22	2022-10-23	1

Figura 31: Encomendas de um cliente

- Requisitos de Controlo

Foram adicionadas certas permissões, tanto para o Sr. Alberto como os funcionários. O Sr. Alberto vai ter poder em toda a base de dados, sendo esta procurar ou alterar informação. Os funcionários apenas vão ter permissões para alterar um **‘Estado’** de uma encomenda e ver as informações do cliente caso ocorra algum problema durante o transporte.

```

CREATE USER 'SrAlberto'@'localhost' IDENTIFIED BY 'livrariasilva';
CREATE USER 'funcionario'@'localhost' IDENTIFIED BY 'funcionariosilva';

GRANT ALL PRIVILEGES ON livraria.* TO 'SrAlberto'@'localhost';

GRANT UPDATE (`Estado`) ON livraria.encomenda TO 'funcionario'@'localhost';

GRANT EXECUTE ON PROCEDURE visualizarEncomendas TO 'funcionario'@'localhost';

GRANT SELECT (Contacto) ON livraria.cliente TO 'funcionario'@'localhost';
GRANT SELECT (Nome) ON livraria.cliente TO 'funcionario'@'localhost';
GRANT SELECT (NIF) ON livraria.cliente TO 'funcionario'@'localhost';
GRANT SELECT (`Código de Postal`) ON livraria.cliente TO 'funcionario'@'localhost';
GRANT SELECT ON livraria.cliente TO 'funcionario'@'localhost';
flush privileges;

```

Figura 32: Requisitos de controlo

5.4. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Tabela 4: Estimativa do espaço em disco da base de dados

Entidade	Atributos	Tipo de Dados	Tamanho	Ocorrências	SubTotal
Funcionário	ID	INT	4	4	216
	Nome	VARCHAR (45)	46		
	Contacto	INT	4		
Cliente	NIF	INT	4	11	2497
	Contacto	VARCHAR (45)	46		
	Nome	VARCHAR (45)	46		
	Código de Postal	CHAR (8)	9		
	Localidade	VARCHAR (30)	31		
	Rua	VARCHAR (90)	91		
Encomenda	ID	INT	4	14	2352
	Local de Entrega	VARCHAR (100)	101		
	Estado	VARCHAR (15)	16		
	Tipo de Pagamento	VARCHAR (40)	41		
	Data de Encomenda	DATE	3		
	Data de Entrega	DATE	3		
Fornecedor	ID	INT	4	3	150
	Nome	VARCHAR (45)	46		
Veículo	Matrícula	CHAR (8)	9	3	153
	Marca	VARCHAR (20)	21		
	Modelo	VARCHAR (20)	21		
Encomenda has livros	Quantidade	INT	4	19	76
Livros	Nome	VARCHAR (45)	46	8	1168
	Tema	VARCHAR (45)	46		
	Autor	VARCHAR (45)	46		
	Preço	FLOAT	4		
	Quantidade	INT	4		
	ID	INT	4		

Fornecimento	Quantidade	INT	4	7	77
	Data	DATE	3		
				Total	6689

$$E(F,C,E, FN,V,EL,L,FNL)=54F + 227C + 168E + 50FN + 51V + 4EL + 146L + 11FNL$$

(Cálculo para o espaço utilizado: Ocorrências x Espaço Ocupado por cada atributo)

Assumindo um crescimento anual de **10%**, estima-se que para nos próximos 5 anos os valores sejam os seguintes:

Tabela 5: Crescimento anual da base de dados

Ano	Tamanho (Bytes)
0	6689
1	7357,9
2	8093,69
3	8903,05
4	9793,35
5	10772,68

5.5. Definição e caracterização das vistas de utilização em SQL

Vistas são uma mais-valia na implementação de uma base de dados. Estas permitem um aumento significativo na segurança, uma vez que o utilizador tem apenas acesso a uma tabela temporária e não à tabela onde são armazenados os dados.

- Ver a média dos preços dos livros em stock

```
CREATE VIEW vw_mediaPreço AS
SELECT AVG(livros.PreçoL) AS MediaPreço, AVG(livros.Quantidade) AS MediaQuantidade FROM livros;

SELECT * FROM vw_mediaPreço
```

	MediaPreço	MediaQuantidade
▶	12.338888751135933	66.3333

Figura 33: Vista 1 - Média dos preços dos livros em stock

- Ver o preço de todas as encomendas

```
CREATE VIEW vw_preço AS
SELECT encomenda_has_livros.Encomenda_idEncomenda as id, ROUND(SUM((encomenda_has_livros.Quantidade * livros.PreçoL))) as Preço from encomenda_has_livros
inner join livros
on encomenda_has_livros.Livros_Nome = livros.Nome
GROUP BY 1
ORDER BY 1;
```

	id	Preço
▶	1	34
	2	16
	3	26
	4	60
	5	16
	6	14
	7	12
	8	39
	9	20
	10	920
	11	10
	12	26

Figura 34: Vista 2 - Preço de todas as encomendas

- Informação relativamente a uma encomenda sem repetir informação

```
CREATE VIEW vw_encomenda AS
SELECT encomenda.idEncomenda, funcionário.Nome as 'funcionário', veículo.Modelo, cliente.Nome as 'cliente', '-' as 'livro' from encomenda
INNER JOIN funcionário on encomenda.Funcionário_idFuncionário = funcionário.idFuncionário
INNER JOIN veículo on encomenda.Veículo_Matricula = veículo.Matricula
INNER JOIN cliente on encomenda.Cliente_NIF = cliente.NIF
INNER JOIN encomenda_has_livros ON encomenda.idEncomenda = encomenda_has_livros.Encomenda_idEncomenda
INNER JOIN livros on encomenda_has_livros.Livros_Nome = livros.Nome
UNION
SELECT encomenda_has_livros.Encomenda_idEncomenda, '-', '-', '-', livros.Nome As 'livro' from encomenda_has_livros
INNER JOIN livros on encomenda_has_livros.Livros_Nome = livros.Nome
ORDER by 1;
```

	idEncomenda	funcionário	Modelo	cliente	livro
►	1	Filipe Azevedo	Transit	João Gonçalo	-
	1	-	-	-	Chainsaw Man
	1	-	-	-	Dune
	2	Filipe Azevedo	Transit	João Gonçalo	-
	2	-	-	-	Harry Potter - Phoenix
	3	João Vítor	Van	Alberto Ronaldo	-
	3	-	-	-	Harry Potter - Phoenix
	3	-	-	-	Harry Potter - The Squid Master
	4	Diogo Fernandes	Chiron	Marta Suares	-
	4	-	-	-	Chainsaw Man
	4	-	-	-	Neon Genesis Evangelion
	5	João Vítor	Transit	Marta Suares	-
	5	-	-	-	Little Women

Figura 35: Vista 3 - Informação de encomenda (sem repetir informação)

5.6. Definição e caracterização de triggers em SQL

Triggers são objetos da base de dados que são associados a uma tabela. Um trigger é acionado quando é executado uma ação numa tabela para não ser preciso de alterar uma tabela manualmente.

- **Atualizar o stock sempre que são fornecidos livros**

```
# adicionar ao stock de livros sem que e fornecido
DELIMITER $$
CREATE TRIGGER addStock
  AFTER INSERT ON fornecimento_has_livros
  FOR EACH ROW
BEGIN
  UPDATE livros
    SET livros.Quantidade = livros.Quantidade + NEW.Quantidade
    WHERE livros.Nome = NEW.Livros_Nome;
END $$
```

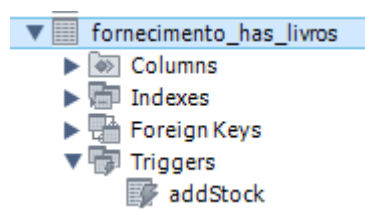


Figura 36: Trigger 1 - Atualizar stock sempre que são fornecidos livros

- Retirar do stock sempre que é feito uma encomenda

```

DELIMITER $$
CREATE TRIGGER takeStock
  AFTER INSERT ON encomenda_has_livros
  FOR EACH ROW
  BEGIN
    UPDATE livros

      SET livros.Quantidade = livros.Quantidade - NEW.Quantidade

      WHERE livros.Nome = NEW.Livros_Nome;
  END $$

```

Figura 37: Trigger 2 - Atualizar stock sempre que são encomendados livros

5.7. Plano de segurança e recuperação de dados

Para manter e/ou reforçar a segurança na base de dados, pretende-se fazer cópias de segurança regularmente prevenindo assim a perda minimalista de informação e ter uma palavra-passe forte e difícil de decifrar, tentando impedir o acesso de outras pessoas (outro método é usar o comando *mysqldump* no terminal com um *path* e vai criar uma cópia nesse caminho).

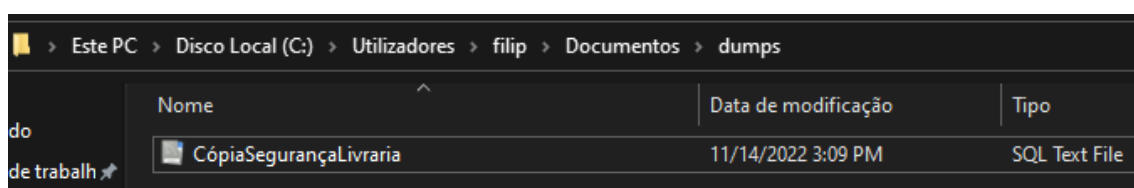
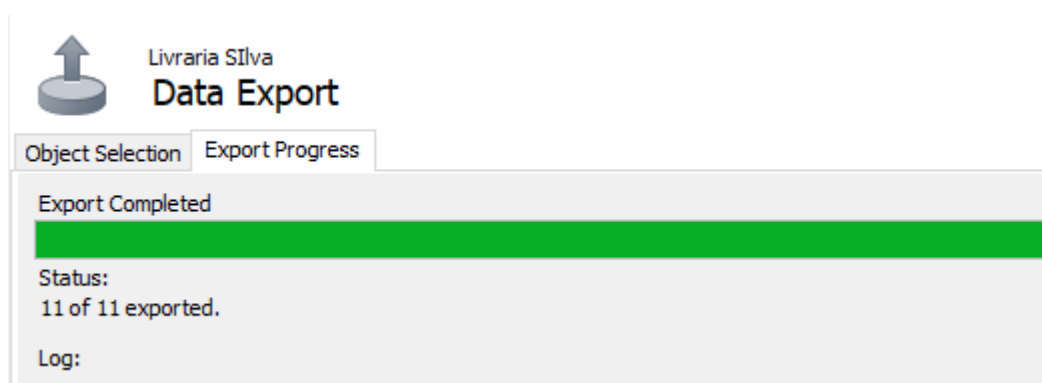
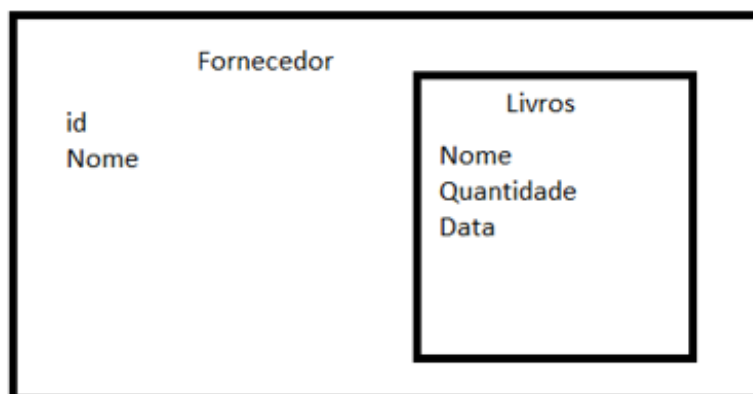
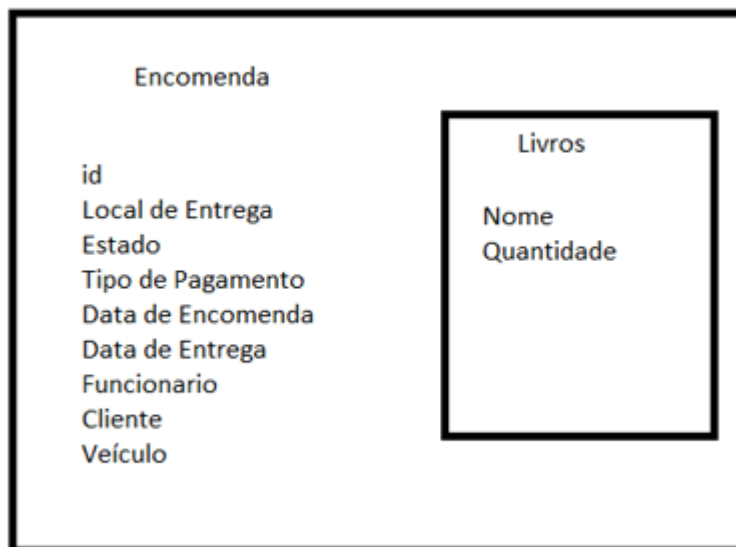


Figura 38: Backup de segurança da BD

6. Conceção e Implementação de um Sistema de Dados em MongoDB

6.1. Definição do Esquema da Base de Dados



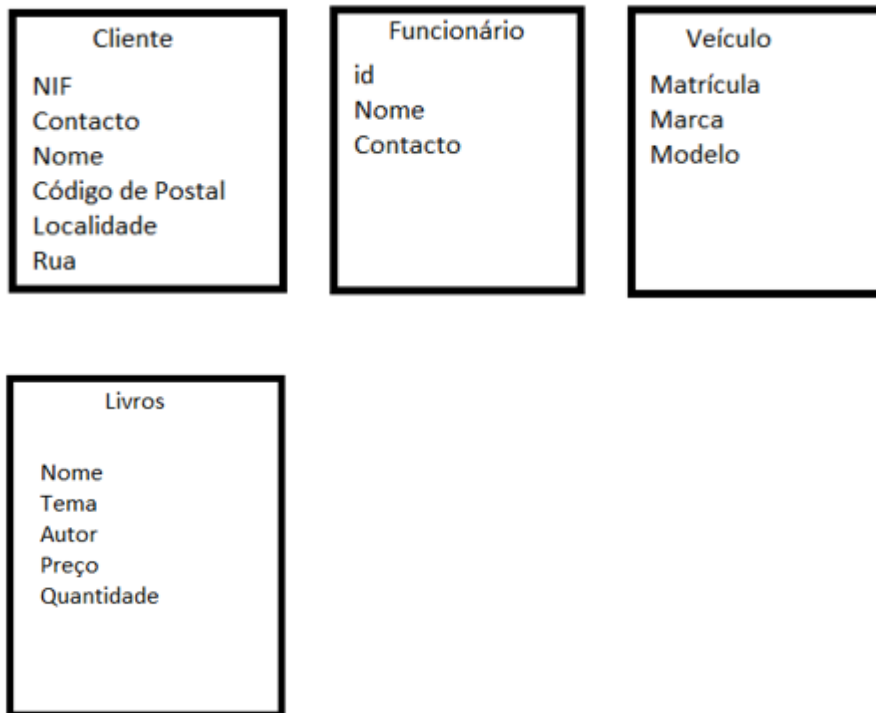


Figura 39: Esquema da BD

6.2. Criação da Base de Dados e Coleções

- **Encomenda:**

```

{
  "_id": ID da encomenda
  "Local de Entrega": Local de Entrega da encomenda
  "Estado": Estado da Encomenda
  "Tipo de Pagamento": Tipo de Pagamento da encomenda
  "Data de encomenda": Data em que foi encomendada
  "Data de Entrega": Data em que foi entregue
  "Funcionário": Funcionário atribuído à encomenda
  "Cliente": Cliente que encomendou
  "Veículo": Veículo usado para transporte
  "Livros": Lista que contém nomes e quantidade de livros na encomenda
}
  
```

- **Fornecedor:**

```

{
  "_id": ID do fornecedor
}
  
```

“**Nome**”: Nome do fornecedor
“**Livros**”: Lista de Livros que contém nomes, quantidade e data de fornecimento
}

- **Cliente:**
{
 “**_id**”: ID do cliente
 “**NIF**”: Nif do cliente
 “**Contacto**”: Contacto do cliente
 “**Nome**”: Nome do cliente
 “**Código de Postal**”: Código de postal do cliente
 “**Localidade**”: Localidade do cliente
 “**Rua**”: Rua do cliente
}

- **Funcionário**
{
 “**_id**”: ID do funcionário
 “**Nome**”: Nome do funcionário
 “**Contacto**”: Contacto do Funcionário
}

- **Veículo**
{
 “**_id**”: ID do veículo
 “**Matrícula**”: Matrícula do veículo
 “**Modelo**”: Modelo do veículo
 “**Marca**”: Marca do veículo
}

- **Livros**
{
 “**_id**”: ID do livro
 “**Nome**”: Nome do livro
 “**Tema**”: Tema do livro
 “**Autor**”: Autor do livro
 “**Preço**”: Preço do livro
 “**Quantidade**”: Quantidade do livro em stock
}

6.3. Processo de Migração de Dados

Após serem definidas as coleções, procedemos à transição da informação toda da base de dados em SQL para a estrutura acima criada. Este processo foi realizado com o recurso ao comando `JSON_ARRAYAGG` que recebe um conjunto de valores JSON ou SQL e retorna um array JSON e o comando `JSON_OBJECT` que recebe uma lista de pares chave-valor e retorna um objeto JSON que contém esses pares.

```
SELECT JSON_ARRAYAGG(JSON_OBJECT('Nome', Nome, 'Livros',
  (SELECT JSON_ARRAYAGG(JSON_OBJECT('id', LF.id, 'Nome', L.Nome, 'Quantidade', LF.Quantidade, 'Data', LF.Data))
    FROM fornecedor as F
    INNER JOIN fornecimento_has_livros as LF
      ON fornecedor.idFornecedor = LF.Fornecedor_idFornecedor
    INNER JOIN livros AS L
      ON LF.Livros_Nome = L.Nome
    WHERE F.idFornecedor = LF.Fornecedor_idFornecedor)))
  FROM fornecedor INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Fornecedor.json';
```

Figura 40: Migração dos dados (transição da informação)

Estes dados, depois de traduzidos, são escritos para um ficheiro JSON que depois é diretamente importado para a plataforma MongoDBCompass.

LivrariaSilva.Encomenda

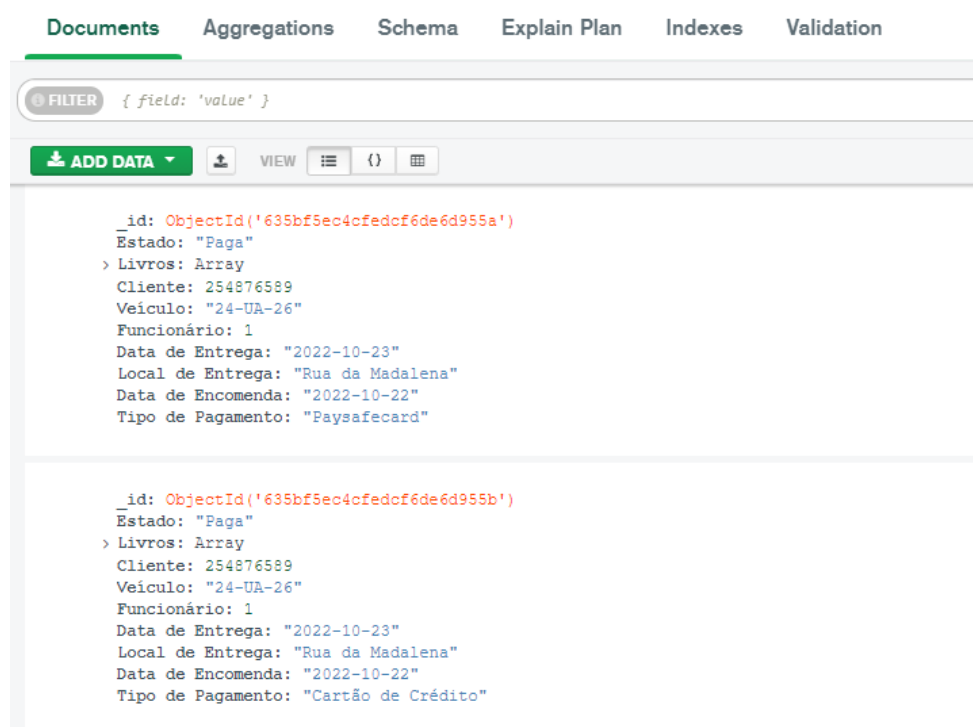


Figura 41: Migração dos dados (escrita para um ficheiro JSON)

6.4. Exploração de Dados em MongoDB

Nesta secção será apresentado o código MongoDB que permite obter resposta a cada requisito de exploração e exemplos da resposta com o povoamento atual.

- **Top 3 clientes com mais encomendas ordenados pelo nome**

```
db.Cliente.aggregate([
  {
    $lookup:{
      from: "Encomenda",
      localField: "NIF",
      foreignField: "Cliente",
      as: "EncomendasCliente"
    },
  },
  {$unwind: "$EncomendasCliente"},
  {
    $group: {
      _id: "$_id",
      NIF: {$first: "$NIF"},
      Nome: {$first: "$Nome"},
      EncomendasCliente: {$sum: 1 },
    },
  },
  {$unwind: "$EncomendasCliente"},
  {$sort: {"EncomendasCliente": -1, "Nome": 1}},
  {$limit: 3},
])
```

```
{
  _id: ObjectId("635bf9064cfedcf6de6d956f"),
  NIF: 254876589,
  Nome: 'João Gonalo',
  EncomendasCliente: 2
},
{
  _id: ObjectId("635bf9064cfedcf6de6d9571"),
  NIF: 526421477,
  Nome: 'Marta Suares',
  EncomendasCliente: 2
},
{
  _id: ObjectId("635bf9064cfedcf6de6d9573"),
  NIF: 547856974,
  Nome: 'Alberto Ronaldo',
  EncomendasCliente: 1
}
```

- Livro mais vendido

```
db.Encomenda.aggregate([
  { $unwind: "$Livros" },
  {
    $group: {
      _id: "$Livros.Nome",
      number: { $sum: "$Livros.Quantidade" }
    }
  },
  {
    $group: {
      _id: 0,
      Livros: { $push: { Nome: "$_id", number: "$number" } }
    }
  },
  {
    $project: { Livros: 1, _id: 0 }
  },
  { $unwind: "$Livros" },
  { $sort: { "Livros.number": -1 } },
  { $limit: 1 },
])
```

```
[ { Livros: { Nome: 'Neon Genesis Evangelion', number: 29 } } ]
```

- Saber quantas encomendas estão entregues e pagas

```
db.Encomenda.aggregate({$group:{_id:"$Estado",count:{$sum:1}}}).sort({count: -1});
```

```
[ { _id: 'Entregue', count: 7 }, { _id: 'Paga', count: 5 } ]
```

- Ver a encomenda com a informação do cliente

```
db.Encomenda.aggregate([
  $lookup: {
    from: "Cliente",
    localField: "Cliente",
    foreignField: "NIF",
    as: "Cliente", }, },
]).pretty()
```

```

_id: ObjectId("635bf5ec4cfedcf6de6d9565"),
Estado: 'Entregue',
Livros: [
  { Nome: 'Dune', Quantidade: 1 },
  { Nome: 'Suspect', Quantidade: 1 }
],
'Veículo': '25-KL-69',
'Funcionário': 4,
'Data de Entrega': '2022-10-21',
'Local de Entrega': 'Rua do Amongo',
'Data de Encomenda': '2022-10-20',
'Tipo de Pagamento': 'Bitcoin',
Cliente: [
  {
    _id: ObjectId("635bf9064cfedcf6de6d956c"),
    NIF: 126434219,
    Rua: 'Rua do Amongo',
    Nome: 'João Barbatana',
    Contacto: 961434578,
    Localidade: 'Viana do Castelo',
    'Código de Postal': '2300-313'
  }
]

```

- Ver o conteúdo das encomendas já pagas pelo cliente

```

db.Encomenda.aggregate([
  $match: {
    $or : [{
      Estado: "Paga"
    }],
  },
  $group: {
    _id: "$Cliente",
    data: { $push: "$Livros" },
  },
])

```

```

{
  _id: 254876589,
  data: [
    [
      { Nome: 'Chainsaw Man', Quantidade: 2 },
      { Nome: 'Dune', Quantidade: 1 }
    ],
    [ { Nome: 'Harry Potter - Phoenix', Quantidade: 1 } ]
  ]
}

```

7. Conclusões e Trabalho Futuro

Ao finalizar este trabalho podemos concluir que todos os requisitos propostos pelo cliente foram implementados na base de dados. Verificamos também que é possível a criação da base de dados tanto em modelo relacional quanto em não relacional, visto que estes modelos têm as suas vantagens e desvantagens será necessária uma análise das mesmas para decidir qual se adaptaria melhor ao nosso problema.

Numa base de dados não relacional teríamos uma melhor performance em bases de dados maiores e uma maior simplicidade na consulta de informação quando comparada com o modelo relacional, porém tem como desvantagens uma possível repetição de dados visto que não existem relacionamentos entre documentos o que provém de uma falta de garantia na consistência de dados visto que cabe ao administrador manter esta consistência, considerando que apenas recentemente a Livraria Silva passou para um sistema online esta falta de consistência poderá ser um ponto decisivo a favor da base de dados relacional. Enquanto existem mais vantagens e desvantagens para cada um dos modelos, nenhuma das que ainda não foram mencionadas teriam valor o suficiente para afetar a decisão, com isto, visto a necessidade de existir um controlo de dados efetuado não apenas pelo administrador, mas também pelo próprio sistema consideramos melhor a abordagem com um modelo relacional.

A implementação de alguma funcionalidade como por exemplo indexação poderia ajudar a performance ao executar *queries*, também podia-se desenvolver novos gatilhos mais complexos para automatizar melhor a base de dados e também desenvolver a parte conceptual poderia ser implementado um sistema que guardasse a faturação da livraria.

8.Referências Bibliográficas

Connolly, Thomas and Begg, Carolyn(2005). Database Systems: A Practical Approach To Design, Implementation And Management: 6th edition.

Belo, O., “Bases de Dados Relacionais: Implementação com MySQL”.

Mysql [Online] Website: <https://dev.mysql.com/doc/workbench/en/>

[Online] Website: <https://www.w3schools.com/sql/default.asp>

[Online]Website:<https://www.tripwire.com/state-of-security/featured/database-security-best-practices-you-should-know/>

Lista de Siglas e Acrônimos

BD	Base de Dados
UC	Unidade Curricular
PK	Primary Key
SQL	Structured Query Language
ACID	Atomicidade, Consistência, Isolamento e Durabilidade