



Universidade do Minho

Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2022/2023

Online Bookstore Database

Rodrigo Monteiro, Diogo Abreu, Luís Figueiredo, e Afonso Pimenta
junho, 2023

Data de recepção	
Responsável	
Avaliação	
Observações	

Online Bookstore Database

A100706, A100646, A100549, A100648

junho, 2023

Resumo

Este projeto tem como objetivo desenvolver uma base de dados para uma livraria online, proporcionando um sistema eficiente e organizado para a gestão de livros, clientes e encomendas. A implementação da base de dados visa resolver os problemas enfrentados numa livraria física e possibilitar um funcionamento mais escalável e adequado para um ambiente online.

O trabalho é dividido em etapas, iniciando-se pelo levantamento e análise dos requisitos essenciais do sistema. Utilizando a ferramenta brModelo, é desenvolvido o modelo conceptual, identificando as entidades, relacionamentos e atributos relevantes para a livraria. Em seguida, no MySQL Workbench, é construído o modelo lógico, levando em consideração as regras de mapeamento ER e as técnicas de normalização de dados.

A implementação física da base de dados é realizada, traduzindo o modelo lógico para o MySQL, utilizando SQL. Além disso, são aplicadas técnicas de indexação para otimizar o desempenho do sistema. O dbForge Data Generator for MySQL é utilizado para inserir dados de teste e garantir o funcionamento adequado.

Para o sistema de recolha de dados, são utilizados JavaScript, Node.js e o pacote npm mysql2. É criado um parser de ficheiros json e um servidor com Express.js, permitindo a inserção de dados através de uma API com solicitações POST ou por meio de uma interface frontend. Isso proporciona uma experiência interativa e amigável para os usuários.

Por fim, é implementado o sistema de painéis de análise utilizando o Tableau. Isso permite a visualização e análise dos dados da livraria de forma intuitiva, apresentando dashboards com informações relevantes para a gestão.

Este projeto oferece uma abordagem abrangente para o desenvolvimento de uma base de dados para uma livraria online, destacando a importância de uma estrutura sólida para a gestão eficiente dos livros, clientes e finanças.

Área de Aplicação: Estruturação e implementação de uma base de dados

Palavras-Chave: Base de Dados, SQL, modelação, integridade, segurança, tabelas, manutenção, evolução, brModelo, Tableau, MySQL, MySQL Workbench, análise de requisitos, modelo conceptual, modelo lógico, modelo físico, vistas, taxa de crescimento

Índice

Lista de Siglas e Acrónimos.....	6
1 Definição do sistema.....	7
1.1 Contexto de aplicação do sistema.....	7
1.2 Objetivos do trabalho.....	7
1.3 Análise da viabilidade do processo.....	8
1.4 Recursos e equipa de trabalho.....	8
1.5 Plano de execução do projeto.....	9
2 Levantamento e análise de requisitos.....	9
2.1 Método de levantamento e de análise de requisitos adotado.....	9
2.2 Organização dos requisitos levantados.....	10
2.3 Análise e validação geral dos requisitos.....	12
3 Modelação Conceptual.....	12
3.1 Apresentação da abordagem de modelação realizada.....	12
3.2 Identificação e caracterização das entidades.....	12
3.3 Identificação e caracterização dos relacionamentos.....	13
3.4 Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.....	14
3.5 Apresentação e explicação do diagrama ER produzido.....	19
4. Modelação Lógica.....	20
4.1 Construção e validação do modelo de dados lógico.....	20
4.2 Normalização de Dados.....	21
4.3 Apresentação e explicação do modelo lógico produzido.....	21
4.4 Validação do modelo com interrogações do utilizador.....	23
5. Implementação Física.....	23
5.1 Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL.....	23
5.2 Tradução das interrogações do utilizador para SQL (alguns exemplos).....	28
5.3 Definição e caracterização das vistas de utilização em SQL (alguns exemplos)....	30
5.4 Cálculo do espaço da bases de dados (inicial e taxa de crescimento anual).....	30
5.5 Indexação do Sistema de Dados.....	32
5.6 Procedimentos Implementados.....	32
5.7 Plano de segurança e recuperação de dados.....	35
6. Implementação do Sistema de Recolha de Dados.....	35
6.1 Apresentação e modelo do sistema.....	35
6.2 Implementação do sistema de recolha.....	36
6.3 Funcionamento do sistema.....	40
7. Implementação do Sistema de Painéis de Análise.....	42
7.1 Definição e caracterização da vista de dados para análise.....	42
7.2 Povoamento das estruturas de dados para análise.....	42
7.3 Apresentação e caracterização dos dashboards implementados.....	43
8. Conclusões e Trabalho Futuro.....	45
9. Referências Bibliográficas.....	45

Índice de Figuras

Figura 1 - Diagrama de Gantt.....	9
Figura 2 - Diagrama ER.....	20
Figura 3 - Entidade-relacionamento encomenda.....	20
Figura 4 - Modelo Lógico do MySQL Workbench.....	22
Figura 5 - Query 1.....	28
Figura 6 - Query 2.....	28
Figura 7 - Query 3.....	29
Figura 8 - Query 4.....	29
Figura 9 - Memória necessária ao longo do tempo.....	32
Figura 10 - Exemplo das tabelas no dbForge.....	35
Figura 11 - Pasta “model” com os ficheiros JSON.....	40
Figura 12 - Comando para executar o script de povoamento.....	40
Figura 13 - Inicialização do server.....	40
Figura 14 - Post request com o Thunder Client.....	41
Figura 15 - Interface front-end (1).....	41
Figura 16 - Interface front-end (2).....	42
Figura 17 - Conexão entre Tableau e a base de dados MySQL (bookstore).....	43
Figura 18 - Exemplo de relacionamentos no Tableau.....	43
Figura 19 - Tabela atualizada no Tableau.....	43
Figura 20 - Dashboard do Tableau.....	44

Índice de Tabelas

Tabela 1 - Recursos e equipa de trabalho.....	8
Tabela 2 - Requisitos de descrição.....	11
Tabela 3 - Requisitos de manipulação.....	11
Tabela 4 - Requisitos de controlo.....	12
Tabela 5 - Identificação e caracterização das entidades.....	13
Tabela 6 - Identificação e caracterização dos relacionamentos.....	13
Tabela 7 - Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.....	19
Tabela 8 - Espaço ocupado na base de dados por cada tipo.....	30
Tabela 9 - Espaço ocupado pelas tabelas.....	31

Lista de Siglas e Acrónimos

SGBD	Sistema de Gestão de Bases de Dados
ER	<i>Entity-Relationship</i>
SQL	<i>Structured Query Language</i>
JSON	<i>JavaScript Object Notation</i>
NPM	<i>Node Package Manager</i>
ID	Identificador
IVA	Imposto sobre o Valor Acrescentado
ISBN	<i>International Standard Book Number</i>
NIF	Número de Identificação Fiscal

1 Definição do sistema

1.1 Contexto de aplicação do sistema

A meio do ano de 2021, Artur perdeu o seu trabalho numa livraria local. A livraria fechou, tal como muitos outros negócios ao longo do ano, devido à crise que a pandemia *Covid 19* provocou no país. Artur gostava daquele trabalho. Desde criança sempre adorou livros, tinha uma boa relação com os seus dois colegas de trabalho Maria e Pedro, e, para além disso, o local da livraria era bastante perto de onde vivia.

A livraria tinha um ambiente acolhedor, com prateleiras de madeira que exibiam uma grande variedade de livros de diferentes géneros e autores. A primeira pessoa que os clientes encontravam ao entrar na livraria era a Sra. Maria, a proprietária. A Sra. Maria era a responsável pela gestão geral da livraria, incluindo o atendimento ao cliente, a organização de eventos literários e a gestão financeira.

Artur era o vendedor, responsável por ajudar os clientes a encontrar o que procuravam, por dar sugestões com base nos seus interesses, e pela organização da livraria. Ficava predominantemente na área da literatura clássica, enquanto que o Pedro, que exercia as mesmas funções, ficava nas secções de literatura contemporânea.

Depois de a livraria fechar, Artur encontrou-se por diversos dias descontente e insatisfeito até surgir a ideia de reabrir a livraria, mas desta vez online, juntamente com a Sra. Maria e o Pedro. Após considerações, pesquisas e planeamento, percebeu que seria necessário a implementação de uma base de dados.

Desse modo, bastantes problemas que foram sentidos na antiga livraria seriam resolvidos, levando a um funcionamento mais eficiente e organizado da gestão dos livros, dos clientes e da gestão financeira. A manutenção de um registo manual dos livros disponíveis, dos seus preços e quantidades, e de vendas, era eficaz com o inventário relativamente limitado e um número gerenciável de vendas da livraria. No entanto, com uma livraria online, é bastante mais prático a utilização de uma base de dados, e, para além disso, é uma abordagem muito mais escalável para futuros inventários e volumes de vendas maiores.

1.2 Objetivos do trabalho

1. Melhorar a gestão do inventário de livros - monitorizar quais são os livros que precisam de ser reabastecidos, a quantidade de livros novos que são comprados, a quantidade que é vendida; ampliar a secção de livros disponíveis - oferecer uma maior variedade de opções
2. Tornar fácil o acesso às informações precisas e atualizadas de cada livro - título, autor, avaliação, editora, sinopse, preço, formato (*ebook* - .epub, .kepub (para o *kobo*), .mobi (para o *kindle*), - ou físico - capa dura, livro de bolso, *etc.*) - portes, ISBN, idioma, dimensões, encadernação, nº de páginas, classificação temática/ género, data de publicação, comentários/ opiniões dos leitores, acerca do autor, acerca da editora.
3. Gerar um perfil que contém várias informações sobre cada cliente.
4. Facilitar a gestão de encomendas: rastrear todas as encomendas recebidas, processá-las de maneira eficiente e atualizar o status das mesmas, de modo a que seja possível acompanhar o processo de entrega.

1.3 Análise da viabilidade do processo

O Artur, a Sra. Maria e o Pedro acreditam que a incorporação de uma base de dados para a nova livraria online, conseguirá:

1. Aumentar o número de vendas, e o número de clientes
2. Melhorar a experiência dos clientes, oferecendo uma navegação e pesquisa mais rápida e eficaz
3. Melhorar a estratégia de *marketing* e aumentar o número de promoções - sabendo os padrões e histórico de compra dos clientes
4. Disponibilizar de um maior número de livros
5. Reduzir os custos - previsão de demanda por livros, levando a compras mais eficientes

1.4 Recursos e equipa de trabalho

Recursos		Equipa de trabalho	
Recursos Humanos	Recursos Materiais	Pessoal Interno	Pessoal Externo
- Funcionários da loja; - Equipa a contratar de desenvolvimento da base de dados; - Equipa a contratar de desenvolvimento da aplicação; - Equipa a contratar de desenvolvimento do <i>website</i> .	- <i>Hardware</i> (1 servidor); - <i>Software</i> (SGBD, Aplicação de gerenciamento de <i>stock</i> , <i>Website</i> da livraria).	- Artur, Maria, Pedro (Manutenção de <i>stock</i> , <i>marketing</i> , vendas)	- Arquiteto e engenheiro de bases de dados da empresa a contratar - Desenvolvedor <i>web</i> a contratar - Desenvolvedor de aplicações a contratar

Tabela 1 - Recursos e equipa de trabalho

Recursos humanos

- Funcionários da loja (Artur, Maria e Pedro), e equipa a contratar de desenvolvimento da base de dados.

Recursos materiais

- *Hardware* (1 servidor, ...)
- *Software* (SGBD, ...)

Equipa de trabalho

- Pessoal Interno: Artur, Maria, Pedro - gerenciamento da manutenção de *stock*, de vendas e encomendas, *marketing*, etc.
- Pessoal Externo: Arquiteto e engenheiro de bases de dados da empresa a contratar - levantamento de requisitos, modelação do sistema, implementação do sistema;
- Outros: clientes selecionados - inquéritos de opinião e avaliação da qualidade dos serviços prestados;

1.5 Plano de execução do projeto

Para definir e planear a forma como o processo de desenvolvimento do SGBD iria ser realizado, os funcionários da livraria e a equipa da empresa contratada para o seu desenvolvimento, estabeleceram um plano concreto de trabalhos, bem como o seu cronograma de execução.

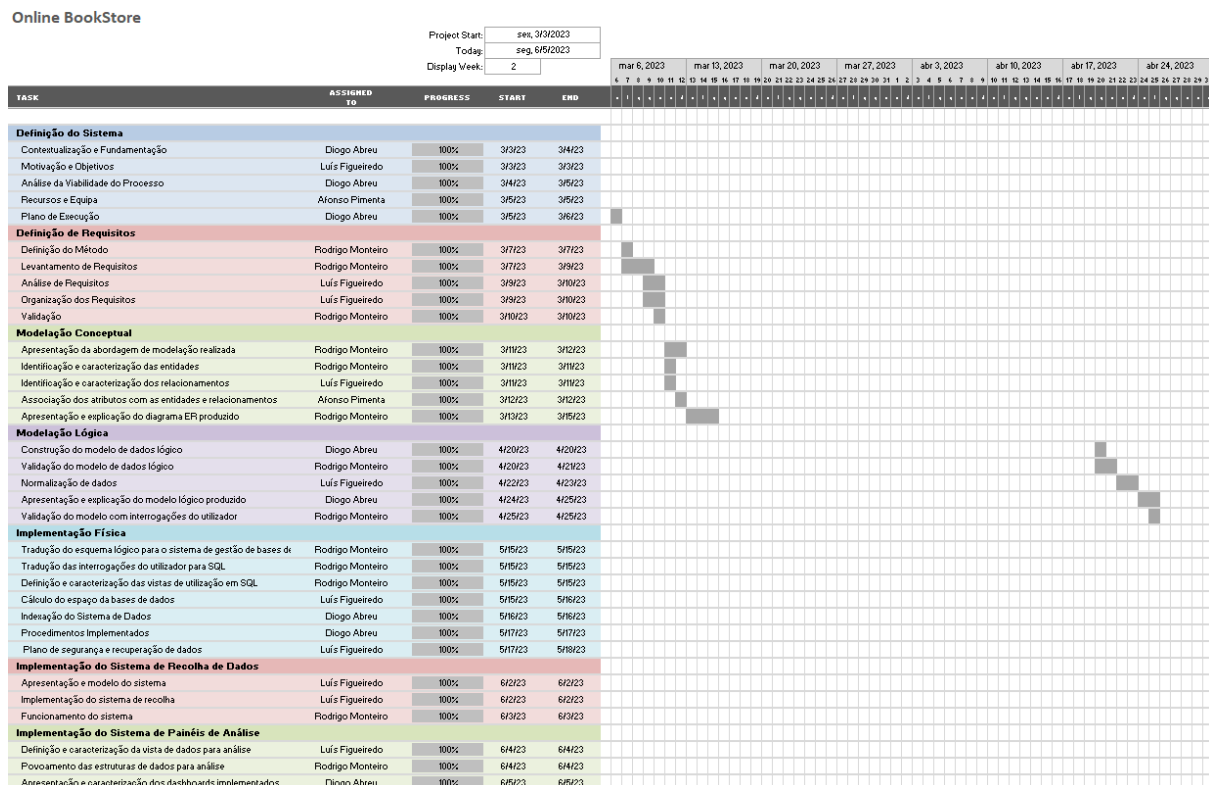


Figura 1 - Diagrama de Gantt

Em geral, foi possível cumprir o planeamento feito.

2 Levantamento e análise de requisitos

2.1 Método de levantamento e de análise de requisitos adotado

- Reuniões entre os funcionários da livraria
- Análise dos registos de livros, de vendas e de clientes, e das parcerias com empresas e editoras
- Investigação/ pesquisa acerca de outras livrarias online, e dos seus métodos
- Inquéritos aos clientes regulares da livraria, acerca, por exemplo, do que poderia ter sido melhorado/ opinião acerca do funcionamento da livraria e da disponibilidade de livros

2.2 Organização dos requisitos levantados

Requisitos de descrição

Nº	Data	Descrição	Fonte	Analista
RD1	11/03/2023	Cada cliente deve ser registado com: id/número sequencial, nome, email, morada(s) de envio e de faturação, contactos (número de telemóvel e email), NIF	Maria	R.M.
RD2	11/03/2023	Cada livro deve ser registado com um título, autor, avaliação, editora, <i>slogan</i> , sinopse, preço, formato (<i>ebook</i> : .epub, .kepub, .mobi; ou físico), portes, ISBN, idioma, dimensões, número de páginas, data de publicação, nº de edição, comentários/ opiniões dos leitores, quantidade em <i>stock</i> , desconto e IVA	Artur	R.M.
RD3	11/03/2023	Cada autor deve ser registado com: nome, avaliação média dos seus livros, descrição/ pequena biografia, bibliografia/ lista de livros e um ID próprio	Maria	R.M.
RD4	11/03/2023	Todos os livros devem pertencer a uma ou mais categorias temáticas / géneros	Maria	R.M.
RD5	11/03/2023	Todos os clientes possuem um histórico de compras, contendo as encomendas feitas até ao momento.	Maria	R.M.
RD6	11/03/2023	Encomendas são registadas com um ID, modo de envio, valor, método de pagamento, modo de envio, e estado (pendente, a ser processada, enviada, entregue, cancelada, devolvida - como endereço de entrega incorreto ou destinatário ausente), data da encomenda (data de entrega, envio e de pagamento)	Artur	R.M.
RD7	11/03/2023	Cada editora deve ser registada com o nome, um ID próprio, endereço da sede, contactos (email e números de telefone), site e descrição/ informações sobre o catálogo de livros	Artur	R.M.
RD8	11/03/2023	Cada género/ categoria temática deve ser registado com: nome, ID próprio, descrição	Maria	R.M.
RD9	11/03/2023	Códigos promocionais podem ser utilizados sobre encomendas e possuem um código próprio, data de início e data de fim, valor da promoção, e número de vezes que a promoção foi usada	Maria	R.M.
RD10	23/03/2023	<i>Reviews</i> são feitas pelos clientes em relação a livros e são registadas com um ID próprio, uma data de publicação, uma avaliação e um comentário (o cliente pode escolher se a <i>review</i> é pública ou privada).	Artur	R.M.

RD11	23/03/2023	A quantidade de livros disponível é fornecida por fornecedores, que são registados com um ID próprio, nome, descrição	Artur	R.M.
------	------------	---	-------	------

Tabela 2 - Requisitos de descrição

Requisitos de manipulação

Nº	Data	Descrição	Fonte	Analista
RM1	11/03/2023	Deve ser possível obter a informação correspondente a um livro a partir do seu ID.	Artur	R.M.
RM2	11/03/2023	Deve ser possível listar todos os livros disponíveis de um dado género ou subgénero.	Pedro	R.M.
RM3	11/03/2023	Deve ser possível obter a descrição de um autor e a lista de livros correspondente (por exemplo, com base na data de publicação, avaliação, etc.)	Pedro	R.M.
RM4	11/03/2023	Deve ser possível obter a lista de livros que uma editora fornece.	Pedro	R.M.
RM5	11/03/2023	Deve ser possível listar o histórico de compras de um cliente.	Pedro	R.M.
RM6	11/03/2023	Deve ser possível obter a informação relativa a um cliente a partir do seu ID.	Maria	R.M.
RM7	23/03/2023	Deve ser possível listar todos os fornecedores especializados num dado género literário.	Maria	R.M.
RM8	23/03/2023	Deve ser possível listar as <i>reviews</i> de um livro, ou feitas por um cliente.	Maria	R.M.
RM9	11/03/2023	Deve ser possível listar todas as atuais promoções.	Maria	R.M.
RM10	23/03/2023	Deve ser possível obter todos os livros disponíveis num dado idioma.	Artur	R.M.
RM11	23/03/2023	Deve ser possível obter a lista de livros fornecida por um dado fornecedor e a sua quantidade.	Artur	R.M.

Tabela 3 - Requisitos de manipulação

Requisitos de controlo

Nº	Data	Descrição	Fonte	Analista
RC1	11/03/2023	Os três funcionários devem estar registados no sistema com as suas informações e credenciais de acesso.	Maria	R.M.
RC2	11/03/2023	Os três funcionários podem aceder a qualquer tipo de consulta de dados.	Maria	R.M.

RC2	11/03/2023	O sistema opera 24/7.	Maria	R.M.
RC3	11/03/2023	Apenas a Sra. Maria pode remover, modificar e adicionar livros, géneros e editoras.	Maria	R.M.
RC5	11/03/2023	Todos os dias o sistema produz um relatório que contém o número de encomendas e <i>reviews</i> feitas, o número de códigos promocionais usados, o número de novos clientes, a quantidade de livros adicionada caso haja.	Maria	R.M.

Tabela 4 - Requisitos de controlo

2.3 Análise e validação geral dos requisitos

Com o levantamento efetuado através dos métodos adotados, foram então organizados e analisados todos os requisitos identificados, assegurando que não existem erros, inconsistências, redundância, ou ambiguidades e que o documento final fique completo.

Por fim, numa reunião agendada, foi discutida a análise e organização de requisitos presente no documento de requisitos final por todos os elementos, com o fim de proceder à sua validação e aprovação.

3 Modelação Conceptual

3.1 Apresentação da abordagem de modelação realizada

Na modelação conceptual, foi considerada apenas uma vista global. De modo a produzir o esquema, foram identificadas e caracterizadas as entidades, relacionamentos e atributos, e foi utilizada a notação Chen.

3.2 Identificação e caracterização das entidades

Designação	Descrição	Sinónimos	Ocorrência
Cliente	Dados pessoais e identificadores do utilizador, inclui também dados financeiros opcionais.	Utilizador	Cada cliente tem um identificador, número próprio e sequencial.
Encomenda	Uma entidade-relacionamento com informações das datas, estado e modo de envio.	Compra	Cada encomenda possui um ID próprio.
Código Promocional	Um código que pode ser utilizado antes de se efetuar o pagamento de modo a aplicar um desconto no valor da encomenda.	Promoção	Promoções são identificadas através do seu código, que pode ser utilizado pelos clientes.

Livro	Dados de identificação, e de descrição. É o único tipo de produto disponível para venda. Catalogação de todo o tipo de livros é necessária. Esta entidade não identifica um livro em específico, mas sim uma dada quantidade de livros em <i>stock</i> que partilham uma série de atributos.	-	Possui um ID próprio. O ISBN também poderia ser utilizado como um identificador.
Fornecedor	Fornece a quantidade de livros disponíveis em <i>stock</i> , pode ter especialização num ou mais géneros literários (por exemplo, ficção científica).	-	Possui um ID próprio.
Autor	Dados estatísticos, biográficos e de identificação.	Escritor	Possui um ID próprio.
Editora	Dados acerca da editora que publica os livros disponíveis para venda.	-	Possui um ID próprio.
Género	Categoria/ género literário serve como uma forma de organização dos livros com base em características temáticas e estruturais (poesia, filosofia, ficção histórica, etc.).	Categoria	Possui um ID próprio.
Idioma	Linguagem em que um livro é escrito (Inglês, Português, etc.).	-	Possui um ID próprio.

Tabela 5 - Identificação e caracterização das entidades

3.3 Identificação e caracterização dos relacionamentos

Entidade	Relacionamento	Cardinalidade	Participação	Entidade
Cliente	encomenda	N:N	P:P	Livro
Cliente	<i>review</i>	N:N	P:P	Livro
Livro	fornecido	N:N	P:T	Fornecedor
Fornecedor	especialização	N:N	P:P	Género
Livro	escrito por	N:N	T:T	Autor
Livro	editado por	N:1	P:T	Editora
Livro	pertence	N:N	P:T	Género
Livro	escrito em	N:1	P:T	Idioma
Código Promocional	aplicado	1:N	P:P	Encomenda

Tabela 6 - Identificação e caracterização dos relacionamentos

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

Entidade/ Relacionam ento	Atributo	Tipo de Atributo	Descrição	Domínio e tamanho	Opcional	Exemplo
Cliente	Nome	Simples	Nome completo	VARCHAR(45)	N	Manuel Ribeiro
	ID	Chave	Identificador do cliente	INT	N	706
	Moradas(1,n)	Multivalor e composto	Lista de moradas do cliente	VARCHAR(45)	N	Avenida Guerra Junqueiro nº4, Casal Do Basílio, Lisboa Avenida Guerra Junqueiro nº4, Casal Do Basílio, Lisboa
	- Morada de envio		Morada para a qual se pode realizar o envio		N	
	- Morada de faturação		Morada que se usa ao realizar a faturação		N	
	Contactos	Composto	Formas de contactar do cliente	VARCHAR(45)	N	exemplo@g mail.com 911111111
	- Email		Endereço de email		N	
	- Telemóvel		Número de telemóvel	CHAR(9)	N	
Encomenda	NIF	Simples	Número de identificação fiscal	VARCHAR(9)	S	000000000
	ID	Chave	Identificador da encomenda	INT	N	706
	Data de entrega	Simples	Data de entrega prevista da encomenda	DATETIME	S	01/01/2025
Encomenda	Data de envio	Simples	Data em que a encomenda vai ser enviada	DATETIME	S	01/01/2025

	Data de Pagamento	Simples	Data em que foi realizado o pagamento	DATETIME	S	01/01/2025
	Método de pagamento	Simples	Forma de como foi paga a encomenda	ENUM(...)	N	Dinheiro
	Estado	Simples	Estado atual da encomenda	ENUM(...)	N	Entregue
	Valor	Simples	Custo da encomenda	DECIMAL(6,2)	N	21,10
	Modo de Envio	Simples	Método de transporte e entrega da encomenda	VARCHAR(45)	N	Normal
Livro	Nº da edição	Simples	Número da versão atual do livro	INT	N	1
	ID	Chave	Identificador do livro	INT	N	706
	Sinopse	Simples	Resumo breve da obra	VARCHAR(400)	N	
	Data de publicação	Simples	Data da publicação do livro	DATE	N	01/01/2000
	Título	Simples	Título do livro	VARCHAR(45)	N	Era uma vez
	Preço	Simples	Custo da compra de um livro	DECIMAL(5,2)	N	21,10
	ISBN	Simples	Código numérico de 13 dígitos que identifica cada livro a nível internacional	VARCHAR(20)	N	978-3-16-148410-0

	Dimensões	Simples	Tamanho do livro (em centímetros caso este esteja em formato físico ou em MegaBytes caso esteja formato digital)	VARCHAR(45)	S	11 cm * 18 cm * 1,5 cm
	Formato	Simples	Forma como o conteúdo do livro é apresentado (físico, digital ou audiobook)	VARCHAR(45)	N	Digital
	Nº de páginas	Simples	Número de páginas de um livro	INT	S	100
	Desconto	Simples	Percentagem de desconto sobre o preço do livro	DECIMAL(2,0)	S	21,10 (%)
	IVA	Simples	Imposto correspondente a uma percentagem sobre o preço do livro	DECIMAL(2,0)	N	23,00 (%)
	Quantidade	Simples	Quantidade de cópias disponíveis	INT	N	129
Idioma	ID	Chave	Identificador do idioma	INT	N	706
	Nome	Simples	Nome do idioma	VARCHAR(45)	N	Inglês
Gênero	Nome	Simples	Categoria literária em que o livro se encaixa	VARCHAR(100)	N	Romance
	ID	Chave	Identificador do gênero	INT	N	706
	Descrição	Simples	Breve descrição do gênero de livros	VARCHAR(350)	N	Romance- história de amor

Editora	ID	Chave	Identificador da editora	INT	N	706
	Nome	Simples	Nome da editora	VARCHAR(75)	N	Editora Fixe
	Endereço	Simples	Morada da editora	VARCHAR(150)	N	Avenida Guerra Junqueiro nº4, Casal Do Basílio, Lisboa
	Descrição	Simples	Breve descrição da editora	VARCHAR(350)	S	Editora de livros
	Contactos		Lista de formas de contactar a editora		N	
	-Email	Composto	Endereço de email	VARCHAR(100)	N	exemplo@gmail.com
	-Telefone		Número de telefone	VARCHAR(9)	N	911111111
	Website	Simples	Website pertencente à editora	VARCHAR(100)	S	www.editora.pt
Autor	Nome	Simples	Nome do Autor	VARCHAR(45)	N	José Autor Manuel
	ID	Chave	Identificador do autor	INT	N	706
	Avaliação média	Simples	Média das avaliações das obras do autor	DECIMAL(3,2)	N	7,10
	Biografia	Simples	Pequeno texto sobre a vida do autor	VARCHAR(500)	S	Este autor nasceu a 18/02/2004 e escreveu uma obra chamada "Era uma vez"
	Data de nascimento	Simples	Data em que o autor nasceu	DATE	N	01/01/2003

Código Promocional	Código	Chave	Identificador da promoção	VARCHAR(50)	N	706
	Valor	Simples	Valor percentual de desconto numa compra	INT	N	21,10 (%)
	Data início	Simples	Data a partir da qual começa uma promoção	DATETIME	N	01/01/2025
	Data fim	Simples	Data a partir da qual a promoção deixa de existir	DATETIME	N	01/01/2025
Review	Visibilidade	Simples	Se a review é visível por outros utilizadores (pública ou privada)	ENUM(...)	N	Pública
	Comentário	Simples	Comentário sobre um livro	VARCHAR(200)	N	Bom livro, gostei.
	Data	Simples	Data da publicação da review	DATETIME	N	01/01/2025
	ID	Chave	Identificador da review	INT	N	706
	Avaliação	Simples	Avaliação de um livro numa escala de 0 a 10	ENUM(...)	N	5
Fornecido	Valor	Simples	Custo dos produtos fornecidos	DECIMAL(6,2)	N	21,10
	ID	Chave	Identificador do fornecido	INT	N	706
	Data	Simples	Data do fornecimento de livros	DATE	N	01/01/2025
	Quantidade	Simples	Quantidade de livros fornecidos	INT	N	87

Fornecedor	ID	Chave	Identificador do fornecedor	INT	N	706
	Nome	Simple	Nome do fornecedor	VARCHAR(45)	N	José Fornecedor da Silva
	Descrição	Simple	Breve descrição do fornecedor	VARCHAR(45)	S	Fornecedor de marcadores de páginas
	Contactos		Lista de formas de contactar o fornecedor		N	
	-Email	Composto	Endereço de email	VARCHAR(45)	N	exemplo@g mail.com
	-Telefone		Número de telefone	VARCHAR(9)	N	911111111
	Endereço	Simple	Morada do fornecedor	VARCHAR(45)	N	Avenida Guerra Junqueiro nº4, Casal Do Basílio, Lisboa

Tabela 7 - Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

3.5 Apresentação e explicação do diagrama ER produzido

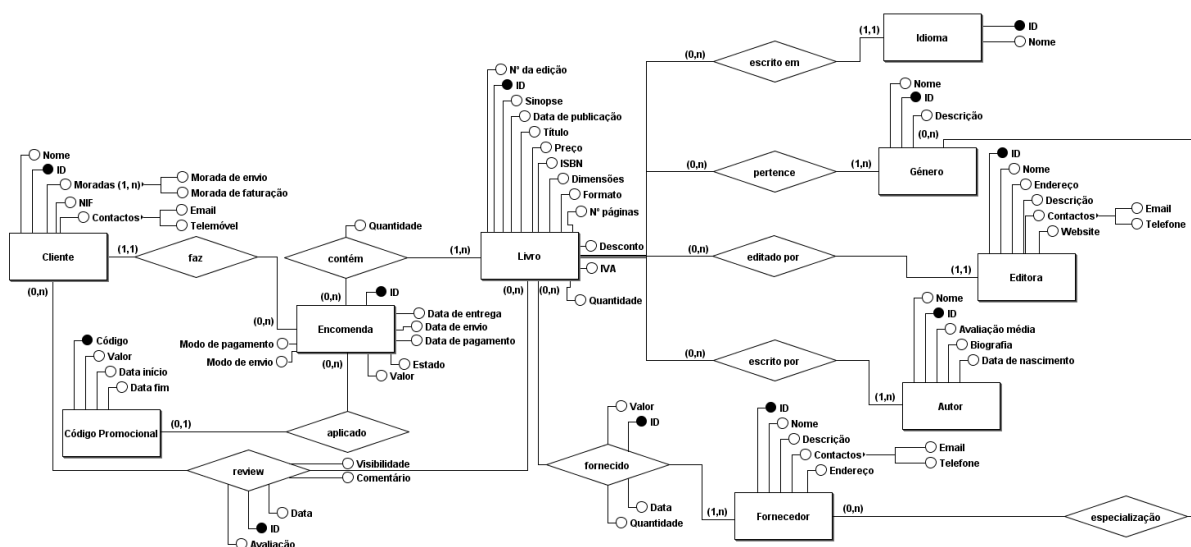


Figura 2 - Diagrama ER

Depois do levantamento e organização de requisitos, e da definição de entidades e relacionamentos, foi possível construir o diagrama ER apresentado acima. Este diagrama foi construído utilizando o software *brModelo*, que utiliza uma notação muito similar à notação Chen (difere na apresentação visual em alguns aspetos).

Para além disso, é importante destacar algumas decisões tomadas: inicialmente, consideramos *encomenda* uma entidade-relacionamento, visto que a esta pode ser aplicado um código promocional, e deste modo o modelo ficaria mais inteligível, no entanto, desse modo o modelo não descreveria a possibilidade de se encomendar, por exemplo, mais do que um livro igual, portanto decidimos utilizar mais relacionamentos para descrever a compra de uma encomenda; existem alguns "ciclos" no modelo, no entanto consideramos estes ciclos necessários para cumprir os requisitos e não uma manifestação de redundância.

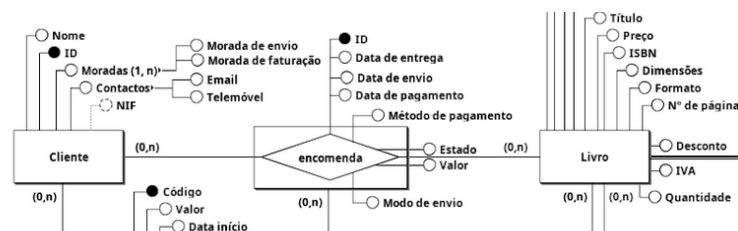


Figura 3 - Entidade-relacionamento encomenda

Assim, passamos a explicar brevemente o raciocínio por detrás dos relacionamentos e entidades.

Um cliente pode comprar um ou mais livros fazendo uma encomenda, a qual tem as datas correspondentes às mudanças de estado (pagamento, envio e entrega); ao fazer uma encomenda, pode aplicar um código promocional, com valor, por exemplo, de 20%, diminuindo o valor final da encomenda; também pode fazer *reviews* de qualquer livro, mesmo que não o tenha comprado, as quais podem ser públicas ou privadas.

Um livro possui um ou mais idiomas, um ou mais géneros literários, um ou mais autores e uma editora.

A quantidade de um livro específico é obtida através de fornecimentos feitos por fornecedores, que podem ter especializações em um ou mais géneros literários.

4. Modelação Lógica

4.1 Construção e validação do modelo de dados lógico

Começamos por definir cada uma das entidades do modelo conceptual, adicionando os respectivos atributos: atributos compostos são vistos como atributos simples, e atributos multivalorados possuem uma tabela própria que possui como chave estrangeira a chave primária da entidade; os relacionamentos 1:N também foram tratados da mesma forma. Os relacionamentos N:M deram origem a uma tabela com duas chaves estrangeiras, tal como é o caso, por exemplo, do relacionamento *Review* - como são relacionamentos N:M, estes podem ter atributos.

Consideramos o modelo produzido válido, uma vez que foi produzido com base no modelo conceptual, e validado novamente por verificação da satisfação dos requisitos levantados previamente.

4.2 Normalização de Dados

O modelo lógico está normalizado com verificação até à 3ª forma normal: todos os atributos das tabelas são atômicos e as dependências funcionais são elementares e diretas.

4.3 Apresentação e explicação do modelo lógico produzido

A entidade cliente deu origem a outra tabela, ClienteMoradas, devido ao seu atributo multivalorado Moradas, e o atributo composto Contactos é representado na tabela Cliente pelos seus atributos constituintes.

A tabela Encomenda guarda a chave estrangeira Cliente, pois cada encomenda tem apenas um cliente associado. Decidimos que esta chave estrangeira pode ser nula, apesar de um cliente ter de participar obrigatoriamente no ponto de vista do modelo conceptual, pois, assim, caso um cliente seja apagado, a informação sobre a encomenda continua a existir na base de dados, e a chave estrangeira Cliente fica nula devido à restrição imposta na criação da tabela Encomenda, *on delete set null*.

Uma encomenda possui um ou mais livros com diferentes quantidades de cada livro, sendo uma relação N:M, por isso é criada a tabela EncomendaLivro com as chaves estrangeiras Livro e Encomenda.

O livro é a tabela com mais colunas, mas três delas podem ter valores nulos nas linhas: Desconto, Idioma e Editora. Idioma e Editora podem ser nulos, pois, assim, caso um Idioma ou Editora seja apagado, o livro continua a existir, mas com esses parâmetros nulos, devido à restrição mencionada anteriormente, *on delete set null*.

Um cliente também pode fazer *reviews* de qualquer tipo de livro, mesmo que não o tenha comprado, e, caso um cliente seja apagado, a *Review* é apagada devido à restrição *on delete cascade*, mas, caso o livro alvo da *Review* seja apagado, a *Review* não é apagada e valor Livro fica *null*.

Um livro pode ter vários fornecedores, sendo uma relação N:M que guarda a quantidade do fornecimento, e por isso é criada uma tabela FornecedorLivro, e também pode ter vários géneros, sendo criada a tabela GéneroLivro.

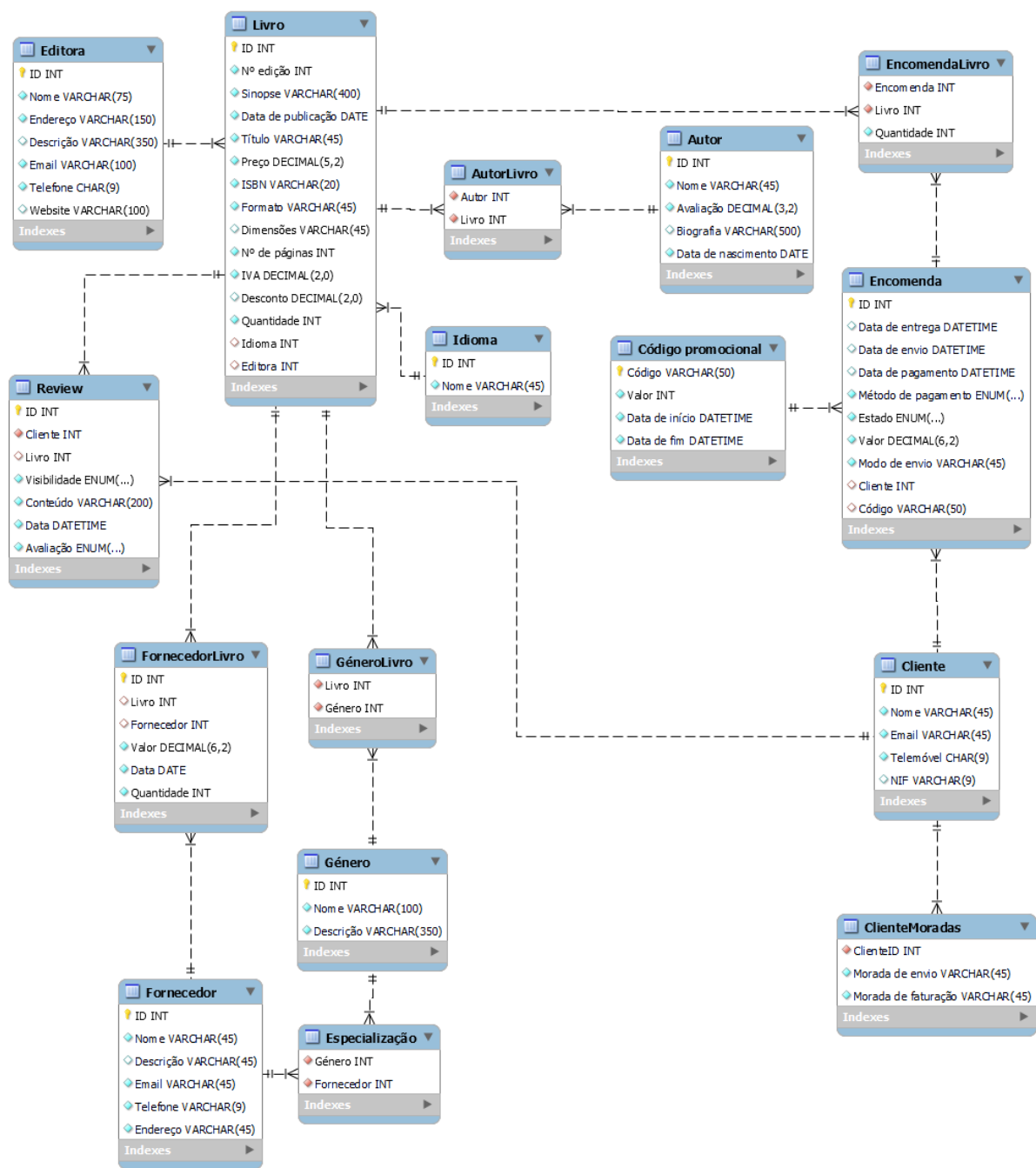


Figura 4 - Modelo Lógico do MySQL Workbench

4.4 Validação do modelo com interrogações do utilizador

De forma a verificar se o modelo é válido, levantamos algumas interrogações que terão de ser corretamente respondidas:

- Livros com melhor avaliação média

É possível obter os livros com melhor avaliação média, fazendo uma junção entre a tabela *Review* e a tabela *Livro* de acordo com o ID do livro (pode ser uma junção interna, ou externa à direita caso se queira obter livros que possivelmente não tenham *reviews*).

- Autores com livros mais comprados

É possível obter os autores com livros mais comprados fazendo uma junção interna entre as tabelas *EncomendaLivro* e *AutorLivro* de acordo com o ID do livro, e *AutorLivro* e *Autor* de acordo com o ID do autor, agrupando de acordo com o ID do autor e, por fim, ordenando a tabela.

- Livros que ninguém encomendou

É possível obter uma listagem dos livros que ninguém encomendou obtendo os livros da tabela *Livro* cujo ID não existe na tabela *EncomendaLivro*.

- Quantidade de livros (únicos) por género literário

É possível obter a quantidade de livros (únicos) por género literário fazendo uma série de junções internas: *Livro* com *GéneroLivro* de acordo com o ID do livro, e *GéneroLivro* e *Género* de acordo com o ID do género; e agrupando por género.

5. Implementação Física

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

Como o processo de construção do modelo lógico foi realizado utilizando *MySQL Workbench*, com recurso à opção *Forward Engineering*, obtemos um *script* base, que depois foi modificado de acordo com algumas decisões estabelecidas e restrições necessárias.

```
create schema if not exists `bookstore`;  
use `bookstore` ;  
  
create table if not exists `bookstore`.`Cliente` (  
  `ID` int not null auto_increment,  
  `Nome` varchar(45) not null,  
  `Email` varchar(45) not null,  
  `Telemóvel` char(9) not null,  
  `NIF` char(9) null,  
  primary key (`ID`))  
engine = InnoDB;
```

```

create table if not exists `bookstore`.`ClienteMoradas` (
  `ClienteID` int not null,
  `Morada de envio` varchar(45) not null,
  `Morada de faturação` varchar(45) not null,
  index `ClienteID_idx` (`ClienteID` asc) visible,
  constraint `ClienteMoradas`
    foreign key (`ClienteID`)
      references `bookstore`.`Cliente` (`ID`)
      on delete cascade
      on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`Idioma` (
  `ID` int not null auto_increment,
  `Nome` varchar(45) not null,
  primary key (`ID`))
engine = InnoDB;

create table if not exists `bookstore`.`Editora` (
  `ID` int not null auto_increment,
  `Nome` varchar(75) not null,
  `Endereço` varchar(150) not null,
  `Descrição` varchar(350) null,
  `Email` varchar(100) not null,
  `Telefone` char(9) not null,
  `Website` varchar(100) null,
  primary key (`ID`))
engine = InnoDB;

create table if not exists `bookstore`.`Livro` (
  `ID` int not null auto_increment,
  `Nº edição` int unsigned not null,
  `Sinopse` varchar(400) not null,
  `Data de publicação` date not null,
  `Título` varchar(45) not null,
  `Preço` decimal(5,2) not null default 0.00,
  `ISBN` varchar(20) not null,
  `Formato` varchar(45) not null,
  `Dimensões` varchar(45) null,
  `Nº de páginas` int unsigned not null,
  `IVA` decimal(2,0) not null default 0,
  `Desconto` decimal(2,0) default 0,
  `Quantidade` int unsigned not null default 0,
  `Idioma` int null,
  `Editora` int null,
  constraint `CheckPreçoLivro`
    check(`Preço` >= 0.00),
  constraint `CheckDescontoLivro`
    check(`Desconto` >= 0 and `Desconto` <= 100),
  constraint `CheckIVALivro`
    check(`IVA` >= 0 and `IVA` <= 100),
  primary key (`ID`),
  index `IdiomaID_idx` (`Idioma` asc) visible,
  index `EditoraID_idx` (`Editora` asc) visible,
  constraint `IdiomaLivro`
    foreign key (`Idioma`)
      references `bookstore`.`Idioma` (`ID`)
      on delete set null
      on update cascade,
  constraint `EditoraLivro`
    foreign key (`Editora`)
      references `bookstore`.`Editora` (`ID`)
      on delete set null
      on update cascade
)

```



```

engine = innodb;

create table if not exists `bookstore`.`Código promocional` (
  `Código` varchar(50) not null,
  `Valor` int unsigned not null default 0,
  `Data de início` datetime not null,
  `Data de fim` datetime not null,
  constraint `CheckValor`
    check(`Valor` >= 0 and `Valor` <= 100),
  constraint `CheckDatasCodigo`
    check (`Data de início` < `Data de fim`),
  primary key (`Código`))
engine = InnoDB;

create table if not exists `bookstore`.`Encomenda` (
  `ID` int not null auto_increment,
  `Data de entrega` datetime null,
  `Data de envio` datetime null,
  `Data de pagamento` datetime null,
  `Método de pagamento` enum ('Cartão de Crédito', 'Transferência Bancária', 'PayPal')
not null,
  `Estado` enum ('Finalizada', 'Enviada', 'Entregue') not null,
  `Valor` decimal(6,2) not null default 0.00,
  `Modo de envio` varchar(45) not null,
  `Cliente` int null,
  `Código` varchar(50) null, -- código promocional
  primary key (`ID`),
  index `CódigoID_idx` (`Código` asc) visible,
  index `ClienteID_idx` (`Cliente` asc) visible,
  constraint `ClienteEncomenda`
    foreign key (`Cliente`)
      references `bookstore`.`Cliente` (`ID`)
      on delete set null
      on update cascade,
  constraint `CódigoEncomenda`
    foreign key (`Código`)
      references `bookstore`.`Código promocional` (`Código`)
      on delete set null
      on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`EncomendaLivro` (
  `Encomenda` int not null,
  `Livro` int not null,
  `Quantidade` int unsigned not null,
  index `LivroID_idx` (`Livro` asc) visible,
  index `EncomendaID_idx` (`Encomenda` asc) visible,
  constraint `CheckQuantidadeEncomenda`
    check(`Quantidade` > 0 and `Quantidade` <= 50),
  constraint `ELEncomendaID`
    foreign key (`Encomenda`)
      references `bookstore`.`Encomenda` (`ID`)
      on delete cascade
      on update cascade,
  constraint `ELLivroID`
    foreign key (`Livro`)
      references `bookstore`.`Livro` (`ID`)
      on delete cascade
      on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`Review` (
  `ID` int not null auto_increment,
  `Cliente` int not null,
  `Livro` int null,

```

```

`Visibilidade` enum ('Público', 'Privado') not null,
`Conteúdo` varchar(200) not null,
`Data` datetime not null,
`Avaliação` enum ('1', '2', '3', '4', '5') not null,
primary key (`ID`),
index `ClienteID_idx` (`Cliente` asc) visible,
index `LivroID_idx` (`Livro` asc) visible,
constraint `ReviewClientID`
    foreign key (`Cliente`)
        references `bookstore`.`Cliente` (`ID`)
        on delete cascade
        on update cascade,
constraint `ReviewLivroID`
    foreign key (`Livro`)
        references `bookstore`.`Livro` (`ID`)
        on delete set null
        on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`Gênero` (
    `ID` int not null auto_increment,
    `Nome` varchar(100) not null,
    `Descrição` varchar(350) not null,
    primary key (`ID`))
engine = InnoDB;

create table if not exists `bookstore`.`Autor` (
    `ID` int not null auto_increment,
    `Nome` varchar(45) not null,
    `Avaliação` decimal(3,2) not null default 0.00,
    `Biografia` varchar(500) null,
    `Data de nascimento` date not null,
    constraint `CheckAvaliação`
        check(`Avaliação` >= 0 and `Avaliação` <= 5),
    primary key (`ID`))
engine = InnoDB;

create table if not exists `bookstore`.`GêneroLivro` (
    `Livro` int not null,
    `Gênero` int not null,
    index `LivroID_idx` (`Livro` asc) visible,
    index `GêneroID_idx` (`Gênero` asc) visible,
    constraint `GêneroLivroLivroID`
        foreign key (`Livro`)
            references `bookstore`.`Livro` (`ID`)
            on delete cascade
            on update cascade,
    constraint `GêneroLivroGêneroID`
        foreign key (`Gênero`)
            references `bookstore`.`Gênero` (`ID`)
            on delete cascade
            on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`AutorLivro` (
    `Autor` int not null,
    `Livro` int not null,
    index `LivroID_idx` (`Livro` asc) visible,
    index `AutorID_idx` (`Autor` asc) visible,
    constraint `LivroAutorID`
        foreign key (`Livro`)
            references `bookstore`.`Livro` (`ID`)
            on delete cascade
            on update cascade,
    constraint `AutorLivroID`

```

```

        foreign key (`Autor`)
        references `bookstore`.`Autor` (`ID`)
        on delete cascade
        on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`Fornecedor` (
  `ID` int not null auto_increment,
  `Nome` varchar(45) not null,
  `Descrição` varchar(45) null,
  `Email` varchar(45) not null,
  `Telefone` varchar(9) not null,
  `Endereço` varchar(45) not null,
  primary key (`ID`))
engine = InnoDB;

create table if not exists `bookstore`.`FornecedorLivro` (
  `ID` int not null auto_increment,
  `Livro` int null,
  `Fornecedor` int null,
  `Valor` decimal(6,2) not null default 0.00,
  `Data` date not null,
  `Quantidade` int unsigned not null default 0.00,
  index `FornecedorID_idx` (`Fornecedor` asc) visible,
  index `LivroID_idx` (`Livro` asc) visible,
  primary key (ID),
  constraint `CheckValorLivro`
    check(`Valor` >= 0),
  constraint `FLFornecedorID`
    foreign key (`Fornecedor`)
    references `bookstore`.`Fornecedor` (`ID`)
    on delete set null
    on update cascade,
  constraint `FLLivroID`
    foreign key (`Livro`)
    references `bookstore`.`Livro` (`ID`)
    on delete set null
    on update cascade)
engine = InnoDB;

create table if not exists `bookstore`.`Especialização` (
  `Gênero` int not null,
  `Fornecedor` int not null,
  index `GêneroID_idx` (`Gênero` asc) visible,
  index `FornecedorID_idx` (`Fornecedor` asc) visible,
  constraint `GêneroID`
    foreign key (`Gênero`)
    references `bookstore`.`Gênero` (`ID`)
    on delete cascade
    on update cascade,
  constraint `FornecedorID`
    foreign key (`Fornecedor`)
    references `bookstore`.`Fornecedor` (`ID`)
    on delete cascade
    on update cascade)
engine = InnoDB;

```

5.2 Tradução das interrogações do utilizador para SQL (alguns exemplos)

- Livros com melhor avaliação média (de ficção)

```
-- 10 livros de ficção com melhor avaliação média (nem todos os livros têm reviews)
select L.Título, avg(R.Avaliação) as `Avaliação Média`
  from Review as R
       right join Livro as L on R.Livro = L.ID
       left join GéneroLivro as GL on GL.Livro = L.ID
 where GL.Género in (
       select ID
         from Género
        where Nome like '%ficção%'
     )
 group by L.ID
 order by `Avaliação Média` desc, L.Título
 limit 10;
```

	Título	Avaliação Média
►	Gray Moby Fahrenheit Tale to Bones	4.5
	of Hunger Bones Book Tale The Thrones Rye	4
	To of with Stars Bell The and Also Jungle	3
	Clockwork Man Cristo Finn The	NULL
	Dragon Lovely The The Shining Tattoo	NULL
	Karamazov Catcher	NULL
	Kill Heights Gone The Don Little of	NULL
	Prince Game Book The the Thief Mockingbird	NULL
	The Giver	NULL
	Vinci the with The Prejudice To with	NULL

Figura 5 - Query 1

- Autores com livros mais comprados

```
select A.Nome, sum(EL.Quantidade) as Total
  from EncomendaLivro as EL
       inner join AutorLivro as AL on EL.Livro = AL.Livro
       inner join Autor as A on AL.Autor = A.ID
 group by A.ID
 order by Total desc
 limit 10;
```

	Nome	Total
►	Selinda Keller	183
	Orthey Vogt	154
	California Zabel	149
	Wilbrecht Ebel	141
	Clemendina Tischler	96
	Fredegar chultz	87
	Conny Dippel	78
	Lisa-Maria Marschner	72
	Balte Joseph	65
	Heiderose Klotz	59

Figura 6 - Query 2

- Livros que ninguém encomendou

```
select ID, Titulo, Quantidade
  from Livro
   where ID not in (
     select distinct Livro
       from EncomendaLivro);
```

	ID	Título	Quantidade
►	2	Hunger the Rings of Great Two Alchemist	63
	4	The Bride Gone Runner	20
	6	of Hunger Bones Book Tale The Thrones Rye	86
	11	of in Letter The The	47
	14	of and Little	90
	16	The in in Cities Finn and Handmaid's	90
	18	Jungle of a of Lord Punishment	33
	21	Dragon Lovely The The Shining Tattoo	5
	26	of Frankenstein Two Letter Divine Solitude	56
	27	Eden The Sun Great Tattoo The Eyre	64
	28	of Guide and Dick Crime Thief The The Purple	1
	30	451 Kite	78

Figura 7 - Query 3

Também se pode obter o mesmo resultado da seguinte maneira:

```
select L.*
  from Livro as L
   left join EncomendaLivro as EL on EL.Livro = L.ID
 group by L.ID
 having count(EL.Encomenda) = 0 or count(EL.Encomenda) is null;
```

- Quantidade de livros (únicos) por género literário

```
select G.Nome as `Género literário`, count(G.ID) as Quantidade
  from Livro as L
   inner join GéneroLivro as GL on L.ID = GL.Livro
   inner join Género as G on GL.Género = G.ID
 group by G.Nome
 order by Quantidade desc;
```

	Género literário	Quantidade
►	Poesia	4
	Autoajuda	3
	Ficção	3
	Literatura de viagem	2
	Religião e espiritualidade	2
	Filosofia	2
	Conto	2
	Suspense	2
	Mistério	2
	Literatura de ficção científica pós-apocalíptica	2
	Mistério policial	2
	Literatura de humor	1

Figura 8 - Query 4

5.3 Definição e caracterização das vistas de utilização em SQL (alguns exemplos)

- View que mostra os melhores códigos ativos na data atual

```
create view MelhoresCódigosAtivos as
select Código
  from `Código promocional`
 where `Data de fim` > curdate()
 order by Valor;
```

- View que mostra a quantidade de livros comprados na data atual

```
create view QuantidadeDeLivrosCompradosHoje as
select count(EL.Quantidade)
  from Encomenda as E
       inner join EncomendaLivro as EL on EL.Encomenda = E.ID
 where date(E.`Data de pagamento`) = curdate();
```

- View que mostra os livros com baixa quantidade (e que possivelmente terão de aumentar nos próximos fornecimentos)

```
create view LivrosBaixaQuantidade as
select * from Livro where Quantidade < 5;
```

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

Valores de referência: <https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html>

<i>varchar</i>	<i>L + 1 bytes if column values require 0 – 255 bytes, L + 2 bytes if values may require more than 255 bytes</i>
<i>int</i>	<i>4 bytes</i>
<i>enum</i>	<i>1 or 2 bytes, depending on the number of enumeration values (65,535 values maximum)</i>

Tabela 8 - Espaço ocupado na base de dados por cada tipo

<i>Tabela</i>	<i>Espaço (bytes)</i>	<i>Nº de entradas</i>	<i>Total (bytes)</i>
Editora	793	5	3965
Livro	594	50	29700
Review	224	30	7840

Autor	559	35	19565
AutorLivro	8	50	400
Encomenda	135	40	5400
Cliente	116	50	5800
ClienteMoradas	96	50	4800
Código Promocional	71	10	710
EncomendaLivro	12	50	600
Idioma	50	10	5000
AutorLivro	8	50	400
FornecedorLivro	23	50	1150
Fornecedor	198	10	1980
Género	457	50	22850
GéneroLivro	8	50	400
Especialização	8	10	80
			110640

Tabela 9 - Espaço ocupado pelas tabelas

Assim, a base de dados atual precisa de 110640 bytes, ou 0,11064 megabytes.

Consideramos que o número de utilizadores por ano é dado pela fórmula $50 + 200 \log(x)$, que são adicionados em média 70 livros por ano, não havendo necessidade de ter em conta a adição de novas editoras, géneros, idiomas ou fornecedores, visto que o número seria relativamente pequeno, e que são adicionados em média 20 autores por ano.

O que implica que também se terão de guardar mais moradas, mais entradas nas tabelas GéneroLivro, AutorLivro e FornecedorLivro (cada livro recebe em média 1.5 fornecimentos).

Supondo que um cliente faz em média 2.7 reviews por ano, então, por ano, existem mais $\approx n^{\circ} \text{ clientes} \times 2.70 \text{ reviews}$. Supondo que um cliente faz em média 3.3 encomendas por ano, cada uma em média com 1.6 livros, então existem mais $\approx n^{\circ} \text{ clientes} \times 3.3$

encomendas, e mais $\approx (n^{\circ} \text{ clientes} \times 3.3) \times 1.6$ entradas na tabela EncomendaLivros.

Também se supõe que são adicionados 20 códigos promocionais por ano. Assim a fórmula fica:

$$f(x) = (50 + 200 \times \log_2(x)) \times (116 + 96 + (3.3 \times (135 + 1.6 \times 12))) + 1.7 \times 224) + 20 \times 559 + 70 \times (594 + 8 + 8 + 23 \times 1.5) + 20 \times 71, x > 1.$$

Portanto, no próximo ano, a base de dados terá ≈ 247670 bytes, ou 0.247670 Mb.

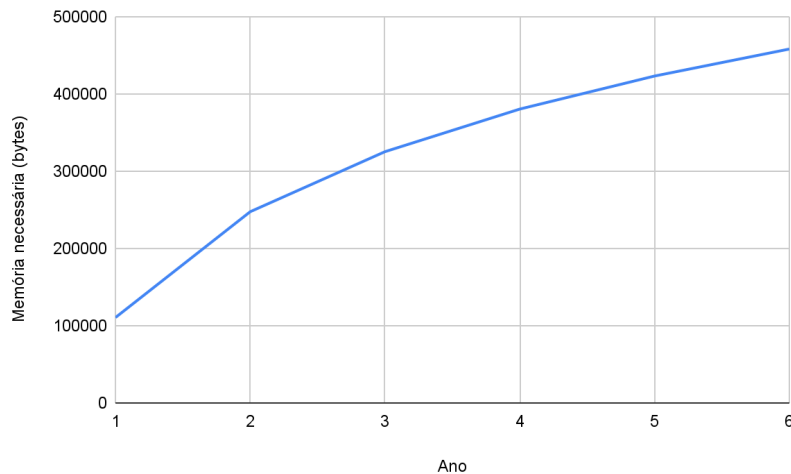


Figura 9 - Memória necessária ao longo do tempo

5.5 Indexação do Sistema de Dados

Todas as chaves estrangeiras têm um índice de modo a melhorar o desempenho das operações de junção de tabelas. Para além dos índices relacionados às chaves estrangeiras, achamos que poderia ser benéfico adicionar mais um índice para o Título na tabela Livro, visto pode tornar a pesquisa de livros por título mais rápida:

```
create index LivroTítulo_idx
on Livro (Título);
```

5.6 Procedimentos Implementados

- Inserir novo fornecimento

```
DELIMITER $$
create procedure novoFornecimento
(in LivroID int, in FornecedorID int, in Valor decimal(6,2),
in QuantidadeFornecimento int)
begin
declare erroTransação bool default 0;
declare continue handler for sqlexception set erroTransação = 1;

start transaction;
insert into FornecedorLivro (Livro, Fornecedor, Valor, Data, Quantidade)
values
(LivroID, FornecedorID, Valor, now(), QuantidadeFornecimento);
update Livro
set Quantidade = Quantidade + QuantidadeFornecimento
where ID = LivroID;

if erroTransação then rollback;
else commit;
end if;
end $$
```



```
DELIMITER ;
```

Neste procedimento, é adicionado um novo fornecimento de um dado livro por um dado fornecedor, com uma certa quantidade e valor. Portanto, é responsável por inserir essa informação na tabela FornecedorLivro, e por atualizar a quantidade desse livro. Caso algum erro aconteça, é feito um *rollback*.

Exemplo de utilização: `call novoFornecimento (1, 1, 129.49, 20);`

- Calcular o preço de uma encomenda

```
DELIMITER $$
create procedure calcularPreçoEncomenda (in EncomendaID int)
begin
    declare result decimal (6,2);
    declare eID int;
    declare codigoValor int;
    declare correçãoDePreço decimal(6, 2);

    declare erroTransação bool default 0;
    declare continue handler for sqlexception set erroTransação = 1;

    start transaction;

    select E.ID, sum(L.Preço * (1 - coalesce(L.Desconto/100.0, 0)) *
coalesce(EL.Quantidade, 0)), coalesce(C.Valor, 0) into eID, result, codigoValor
    from Encomenda as E
        left join `Código promocional` as C on E.Código = C.Código
        inner join EncomendaLivro as EL on E.ID = EL.Encomenda
        inner join Livro as L on EL.Livro = L.ID
    where E.ID = EncomendaID
    group by E.ID;

    set correçãoDePreço = result * (1 - (codigoValor / 100.0));
    select eID as Encomenda, result as `Preço total`, concat(codigoValor, '%') as
'Promoção', correçãoDePreço as `Correção de preço`;

    update Encomenda
        set Valor = correçãoDePreço
        where ID = EncomendaID;

    if erroTransação then rollback;
    else commit;
    end if;
end $$
DELIMITER ;
```

Este procedimento calcula e atualiza o preço de uma dada encomenda, sendo que recebe o respectivo identificador. Primeiro são declaradas um conjunto de variáveis, cujos valores serão calculados pelo “*select ... into*”. Para isso é necessário juntar as tabelas Encomenda com `Código promocional` (junção externa visto que uma encomenda não tem necessariamente de ter um código promocional associado), Encomenda com EncomendaLivro e EncomendaLivro com Livro (junções internas) de modo a obter a quantidade comprada de cada livro, o preço e o desconto de cada livro. De seguida, são mostrados os valores obtidos, preço total, valor da promoção e a correção de preço, e, por fim, é feita uma atualização do valor da encomenda com base na correção de preço calculada. Caso algum erro aconteça numa das fases, é feito um *rollback*.

Exemplo de utilização: `call calcularPreçoEncomenda (5);`

- Dar códigos promocionais

```
DELIMITER $$
create procedure darCodigosPromocionais (in Semanas int, in MinSum decimal(6,2),
in CódigoPromocional varchar(50), out Avisos varchar(5000))
begin
    declare Fim integer default 0;
    declare NomeCliente char(75);
    declare AvisosConcat varchar(10000) default '';
    declare cursorClientes cursor for
        select C.Nome
            from Encomenda as E
            inner join Cliente as C on E.Cliente = C.ID
            where E.`Data de pagamento` >= date_sub(
                curdate(), interval Semanas week)
            group by C.ID
            having sum(E.Valor) > MinSum;

    declare continue handler for not found set fim = 1;
    open cursorClientes;
    fazAvisos : loop
        fetch cursorClientes into NomeCliente;
        if Fim = 1 then leave fazAvisos;
        end if;
        set AvisosConcat = concat (AvisosConcat, 'Parabéns, ', NomeCliente, ',
ganhou o código promocional: ', CódigoPromocional, '\n\n');
    end loop fazAvisos;
    close cursorClientes;
    set Avisos = AvisosConcat;
end $$
DELIMITER ;
```

Este procedimento constrói uma string com avisos que contêm um código promocional direcionados aos clientes que gastaram mais do que uma dada quantia nas últimas n semanas. Este procedimento declara uma definição de um cursor, para que seja possível fazer uma travessia de todos os clientes (de todas as linhas do cursor), construindo a string ao longo do loop.

Exemplo de utilização: `call darCodigosPromocionais(4, 300, '05CTNM9913W', @AvisosCodigos);`

- Outros (exemplos)

```
call emailCliente(25);
call melhoresCódigosPromocionais(curdate(), 5);
call especializaçõesFornecedor(2);
call livrosDeEncomenda(5);
call avisoATodosOsUtilizadores('Promoção 20% em livros de Ficção', @Avisos);
select @Avisos;
```

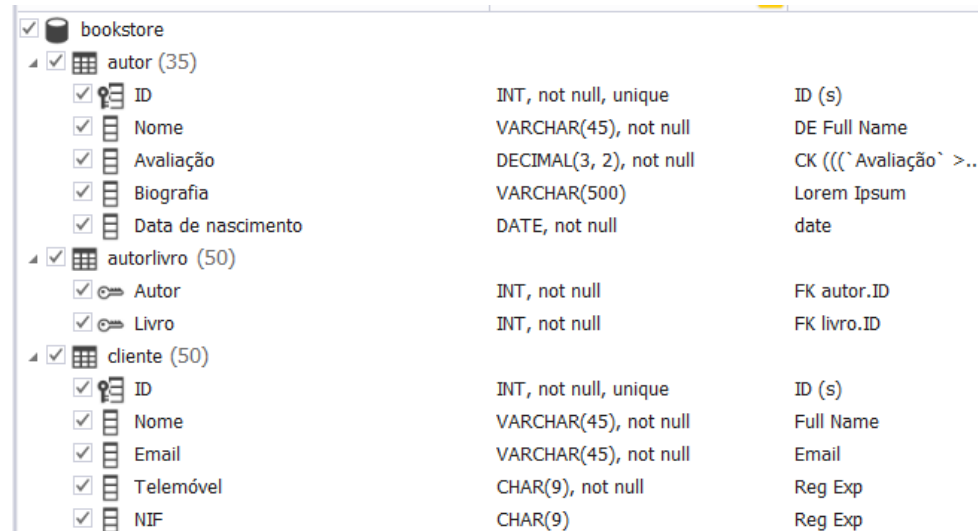
5.7 Plano de segurança e recuperação de dados

Para garantir a segurança e a possibilidade de recuperação de dados da base de dados, achamos necessário a realização regular de *backups*, gestão de acesso de acordo com os requisitos de controlo, e a implementação de testes de integridade e segurança.

6. Implementação do Sistema de Recolha de Dados

6.1 Apresentação e modelo do sistema

Para obter dados semi-aleatórios para teste da base de dados, utilizamos o *software dbForge Data Generator for MySQL*, e foi a partir desses dados que estimamos o espaço da base de dados anteriormente.



The screenshot shows the dbForge Data Generator interface for a database named 'bookstore'. It displays a tree view on the left with the following structure:

- bookstore
 - autor (35)
 - ID (INT, not null, unique)
 - Nome (VARCHAR(45), not null)
 - Avaliação (DECIMAL(3, 2), not null)
 - Biografia (VARCHAR(500))
 - Data de nascimento (DATE, not null)
 - autorlivro (50)
 - Autor (INT, not null, FK autor.ID)
 - Livro (INT, not null, FK livro.ID)
 - cliente (50)
 - ID (INT, not null, unique)
 - Nome (VARCHAR(45), not null)
 - Email (VARCHAR(45), not null)
 - Telemóvel (CHAR(9), not null)
 - NIF (CHAR(9))

The right pane shows the column details for each table, including data types, constraints, and sample values.

Figura 10 - Exemplo das tabelas no *dbForge*

Com este software, obtemos um *script* de povoamento *.sql* com instruções *insert* para cada tabela. Exemplo:

```
INSERT INTO cliente (ID, Nome, Email, Telemóvel, NIF) VALUES
(1, 'Scot Acker', 'Adam.Acker24@example.com', '958078790', 'K1'),
(2, 'Albert Mclean', 'byiswceo_myqdc@example.com', '960225224', NULL),
(3, 'Adelaida Bergman', 'LealU19@nowhere.com', '925059547', '8T'),
(4, 'Abram Patten', 'kkhx3854@example.com', '907807076', 'Q'),
(5, 'Edgar Adler', 'owpk8399@example.com', '962916149', '71'),
(6, 'Edward Jarvis', 'Jennings@example.com', '956514165', 'R1'),
(7, 'Wesley Ramirez', 'Gee@nowhere.com', '928258762', '27Y9'),
(8, 'Gregg Fulmer', 'PhilRichey8@example.com', '904531941', '6A0CED6HJ'),
(9, 'Karly Aguirre', 'Babin@example.com', '919818460', '6'),
(10, 'Rueben Sell', 'MargaritoAnthony@example.com', '978770683', 'F'),
...
(50, 'Rolande Price', 'Spann93@example.com', '964498372', NULL);
```

Para além disso, também implementamos um sistema de recolha de dados a partir de ficheiros *.json* com a linguagem de programação *JavaScript*, *Node.js* e o *npm package mysql2*. Também criamos um *server* com *Express.js*, de modo a que seja possível inserir dados através de uma *API* com *post requests*, ou através de uma interface da *front-end*.

6.2 Implementação do sistema de recolha

Para a recolha de dados a partir de ficheiros *.json*, primeiro é criada a conexão:

```
const pool = mysql.createPool({
  host: process.env.HOST,
  user: process.env.USER,
  password: process.env.PASSWD,
  database: process.env.DATABASE
}).promise();
```

De seguida, definimos um *parser* genérico, que possa ser usado para inserir os dados de diversas tabelas:

```
const genericParser = async (fileName, queryFn) => {
  const data = JSON.parse(
    fs.readFileSync(path.join(__dirname, "model", fileName + ".json"), 'utf-8'));
  for (const elem of data[fileName]) {
    try {
      await pool.query(queryFn(elem));
    } catch (err) {
      throw err;
    }
  }
}
```

Exemplo de utilização relativamente à tabela `Códigos promocionais`:

```
await genericParser("codigosPromocionais", (código) => {
  return `INSERT INTO \`Código promocional\` (Código, Valor, \`Data de
início\`, \`Data de fim\`) VALUES ('${código.Código}', ${código.Valor},
'${código["Data de início"]}', '${código["Data de fim"]}');`
});
```

Ficheiro *JSON*:

```
{
  "codigosPromocionais": [
    {
      "Código": "3FG6EG391H6",
      "Valor": 73,
      "Data de início": "2022-01-16 16:02:35",
      "Data de fim": "2023-01-03 03:46:18"
    },
    // ...
  ]
}
```

A única tabela em que não é utilizado o *parser* genérico é a tabela *Cliente*, pois o ficheiro *json* de clientes, inclui também as moradas:

```
{
  "clientes": [
    {
      "nome": "John Doe",
      "email": "johndoe@example.com",
      "phone": "912056093",
      "nif": "123456789",
      "moradas": [
        {
```

```

        "Morada de envio": "Morada envio exemplo, Cidade",
        "Morada de faturação": "Morada faturamento exemplo, Cidade"
    }, {
        "Morada de envio": "Morada envio exemplo 2, Cidade",
        "Morada de faturação": "Morada faturamento exemplo 2, Cidade"
    }
  ]
}
// ...
]
}

```

Sendo assim, o parser utilizado para clientes é o seguinte:

```

const parseClientes = () => {
  const data = JSON.parse(fs.readFileSync(path.join(__dirname, "model",
    "clientes.json"), 'utf-8'));
  data.clientes.forEach(async cliente => {
    try {
      let [result] = await pool.query(`
        insert into Cliente (Nome, Email, Telemóvel, NIF)
        values (?, ?, ?, ?)`,
        [cliente.nome, cliente.email, cliente.phone, cliente.nif]);

      cliente.moradas.forEach(morada => {
        pool.query(`
          insert into ClienteMoradas (ClienteID, \`Morada de envio\`,
          \`Morada de faturação\`) values (?, ?, ?)`,
          [result.insertId,
            morada["Morada de envio"], morada["Morada de faturação"]]);
      })
    }
    catch (err) {
      throw err;
    }
  })
}

```

Função de *parsing* completa:

```

const processParsing = async () => {
  parseClientes();
  await genericParser("autores", (autor) => {
    return `INSERT INTO Autor (Nome, Avaliação, Biografia, \`Data de nascimento\`)
    VALUES ('${autor.nome}', ${autor.avaliação}, '${autor.biografia}', '${autor["data de
    nascimento"]}');`;
  })

  await genericParser("codigosPromocionais", (código) => {
    return `INSERT INTO \`Código promocional\` (Código, Valor, \`Data de início\`,
    \`Data de fim\`) VALUES ('${código.Código}', ${código.Valor}, '${código["Data de
    início"]}'), '${código["Data de fim"]}');`;
  })

  await genericParser("editoras", (editora) => {
    return `INSERT INTO Editora (Nome, Endereço, Descrição, Email, Telefone,
    Website) VALUES ('${editora.Nome}', '${editora.Endereço}', '${editora.Descrição}',
    '${editora.Email}', '${editora.Telefone}', '${editora.Website}');`;
  })
}

```

```

    await genericParser("fornecedores", (fornecedor) => {
        return `INSERT INTO Fornecedor (Nome, Descrição, Email, Telefone, Endereço)
VALUES ("${fornecedor.Nome}", "${fornecedor.Descrição}", "${fornecedor.Email}",
"${fornecedor.Telefone}", "${fornecedor.Endereço}")`; })

    await genericParser("generos", (genero) => {
        return `INSERT INTO Género (Nome, Descrição) VALUES ("${genero.Nome}",
"${genero.Descrição}")`; })

    await genericParser("idiomas", (idioma) => {
        return `INSERT INTO Idioma (Nome) VALUES ("${idioma.Nome}")`; })

    await genericParser("encomendas", (encomenda) => {
        const valor = encomenda.Valor ? encomenda.Valor : 0;
        let query = `INSERT INTO Encomenda (\`Data de entrega\`, \`Data de envio\`,
\`Data de pagamento\`, \`Método de pagamento\`, Estado, Valor, \`Modo de envio\`,
Cliente, Código) VALUES ("${encomenda["Data de entrega"]}", "${encomenda["Data de
envio"]}", "${encomenda["Data de pagamento"]}", "${encomenda["Método de pagamento"]}",
"${encomenda.Estado}", ${valor}, "${encomenda["Modo de envio"]}", ${encomenda.Cliente})`;

        if (encomenda.Código) query += `, "${encomenda.Código}"`;
        else query += `, NULL`;
        return query;
    });

    await genericParser("livros", (livro) => {
        return `INSERT INTO Livro (\`Nº edição\`, Sinopse, \`Data de publicação\`,
Título, Preço, ISBN, Formato, Dimensões, \`Nº de páginas\`, IVA, Desconto, Quantidade,
Idioma, Editora) VALUES (${livro["Nº edição"]}, "${livro.Sinopse}", "${livro["Data de
publicação"]}", "${livro.Título}", ${livro.Preço}, "${livro.ISBN}", "${livro.Formato}",
"${livro.Dimensões}", ${livro["Nº de páginas"]}, ${livro.IVA}, ${livro.Desconto},
${livro.Quantidade}, ${livro.Idioma}, ${livro.Editora})`; })

    await genericParser("autorLivro", (autorLivro) => {
        return `INSERT INTO AutorLivro (Autor, Livro) VALUES (${autorLivro.Autor},
${autorLivro.Livro})`; })

    await genericParser("encomendaLivro", (encomendaLivro) => {
        return `INSERT INTO EncomendaLivro (Encomenda, Livro, Quantidade) VALUES
(${encomendaLivro.Encomenda}, ${encomendaLivro.Livro}, ${encomendaLivro.Quantidade})`;
    })

    await genericParser("especializacao", (especializacao) => {
        return `INSERT INTO Especialização (Género, Fornecedor) VALUES
(${especializacao.Género}, ${especializacao.Fornecedor})`; })

    await genericParser("fornecedorLivro", (fornecedorLivro) => {
        return `INSERT INTO FornecedorLivro (Livro, Fornecedor, Valor, Data, Quantidade)
VALUES (${fornecedorLivro.Livro}, ${fornecedorLivro.Fornecedor},
${fornecedorLivro.Valor}, "${fornecedorLivro.Data}", ${fornecedorLivro.Quantidade})`; })

    await genericParser("generoLivro", (generoLivro) => {
        return `INSERT INTO GéneroLivro (Livro, Género) VALUES (${generoLivro.Livro},
${generoLivro.Género})`; })

    await genericParser("reviews", (review) => {
        return `INSERT INTO Review (Cliente, Livro, Visibilidade, Conteúdo, Data,
Avaliação) VALUES (${review.Cliente}, ${review.Livro}, "${review.Visibilidade}",
"${review.Conteúdo}", "${review.Data}", "${review.Avaliação}")`; })
}

```

No final do ficheiro, a função é chamada da seguinte maneira:

```

(async () => {
    try {

```

```

    await processParsing();
    console.log('Data parsing and insertion completed successfully.');
```

```

  } catch (err) {
    console.error('Error occurred during data parsing and insertion:', err);
  } finally { pool.end(); }
}());
```

Para a adição de dados a partir de uma *rest API*, foi criado um *server* simples:

```

const PORT = process.env.PORT || 3500;
const app = express();
app.use(express.json());
app.use('/', express.static(path.join(__dirname, '/public')));

app.get('/createCliente', (req, res) => {
  res.sendFile(path.join(__dirname, "views", "createCliente.html"));
})
// ...
app.post('/insertCliente', async (req, res) => {
  const { nome, email, telemovel } = req.body;
  if (!nome || !email || !telemovel)
    return res
      .status(400)
      .json({ message: "Name, email and phone are required." });
  try {
    const result = await insertCliente(nome, email, telemovel);
    res.status(201).json({ success: `New user '${result.id}' created.` });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
})
// ...
app.listen(PORT, () => { console.log(`Server running on port: ${PORT}.`); })
```

Utilização da rota criada:

```

const register = async (username, email, phone) => {
  const data = {
    nome: username,
    email: email,
    telemovel: phone,
  };

  try {
    const response = await fetch("/insertCliente", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        credentials: "include",
      },
      body: JSON.stringify(data),
    });
  }
  // ...
}
```

6.3 Funcionamento do sistema

Para executar o *script* de povoamento *.js* (com *node*), é necessário ter alguns módulos instalados (*mysql2*, *fs*, *path*, *dotenv*), e os seguintes ficheiros json numa pasta “*model*”:

```
\MySQL\js\model>tree /F
Folder PATH listing
Volume serial number is CA92-3ADC
C:
  autores.json
  autorLivro.json
  clientes.json
  codigosPromocionais.json
  editoras.json
  encomendaLivro.json
  encomendas.json
  especializacao.json
  fornecedores.json
  fornecedorLivro.json
  generoLivro.json
  generos.json
  idiomas.json
  livros.json
  reviews.json

No subfolders exist
```

Figura 11 - Pasta “*model*” com os ficheiros JSON

```
C:\Users\                                     \MySQL\js>node parser.js
Data parsing and insertion completed successfully.
```

Figura 12 - Comando para executar o *script* de povoamento

Para além disso, é possível inserir dados através da *API* das seguintes formas:
Primeiro, é necessário inicializar o *server*.

```
\MySQL\js>npm run dev

> bookstore@1.0.0 dev
> nodemon server

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server running on port: 3500.
```

Figura 13 - Inicialização do server

De seguida, pode-se utilizar uma ferrameta como o *Thunder Client* para enviar a post request:

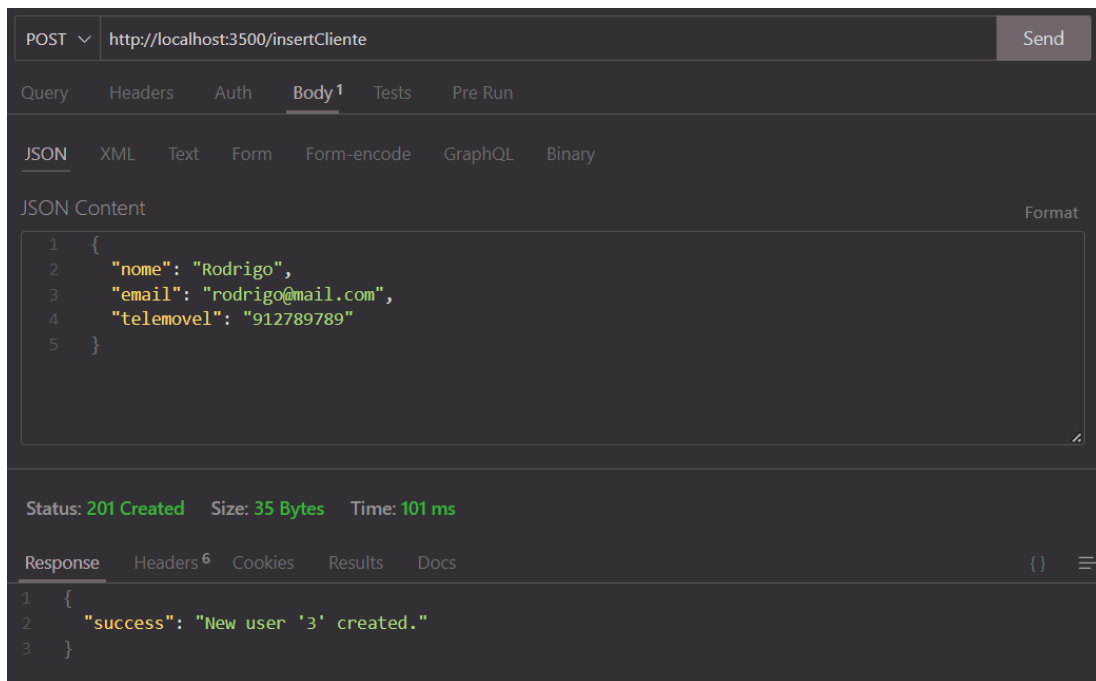


Figura 14 - Post request com o *Thunder Client*

Também se pode utilizar uma interface da *front-end* para inserir dados:

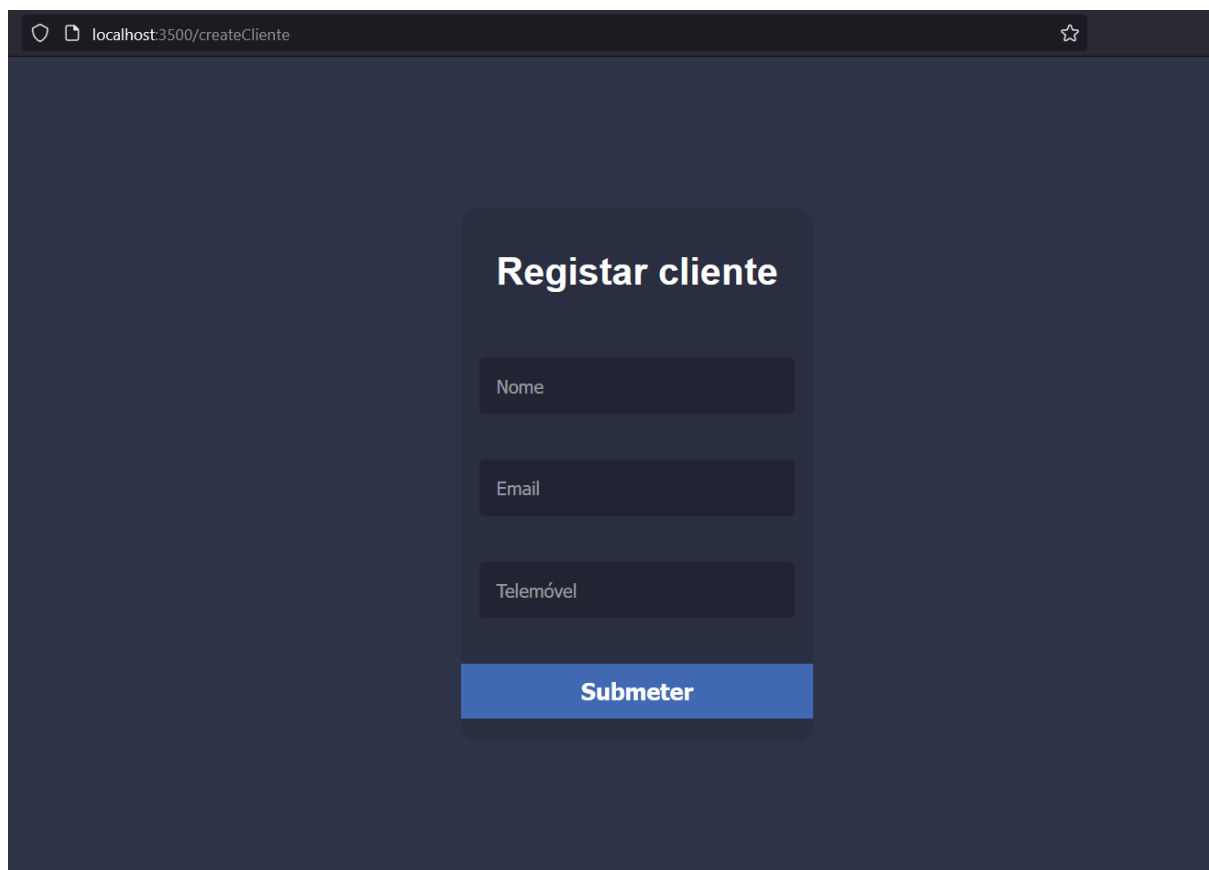


Figura 15 - Interface *front-end* (1)

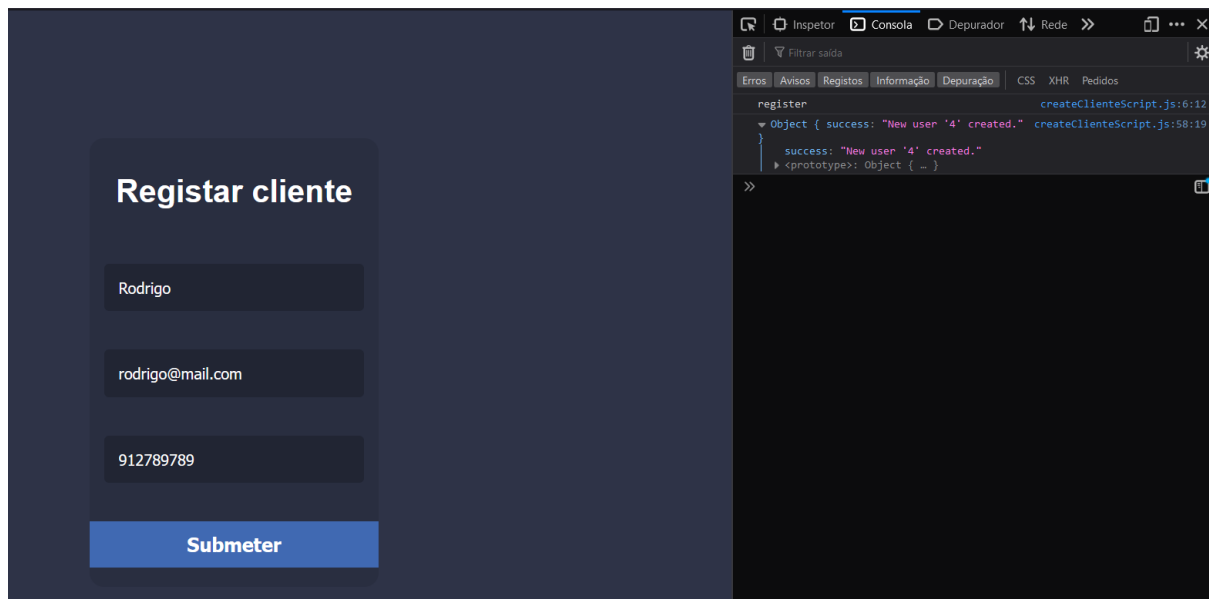


Figura 16 - Interface *front-end* (2)

7. Implementação do Sistema de Painéis de Análise

7.1 Definição e caracterização da vista de dados para análise

O *software* utilizado para a implementação do sistema de painéis de análise foi o *Tableau*, semelhante ao *Power BI* da Microsoft. Decidimos implementar o sistema com o objetivo de obter dashboards com diversas vistas/ representações visuais de dados úteis para o funcionamento da loja e tomada de decisões: número de encomendas feitas ao longo do tempo; livros mais comprados; quantidades de cada livro; fornecimentos (quantidade e valor) dos fornecedores ao longo do tempo. Deste modo, é possível, por exemplo, decidir de forma mais fácil quais são os fornecimentos de livros necessários (os que têm menos quantidade, ou os mais comprados) e as respectivas quantidades, como também quais são os melhores fornecedores para esses fornecimentos.

7.2 Povoamento das estruturas de dados para análise

O povoamento das estruturas de dados para análise foi realizado importando os dados das tabelas da base de dados *SQL*.

MySQL

Geral

SQL inicial

Servidor

localhost

Porta

3306

Banco de dados

bookstore

Nome de usuário

root

Senha

☐ Exigir SSL

Figura 17 - Conexão entre Tableau e a base de dados MySQL (bookstore)

De seguida, atualizamos cada tabela e fizemos as ligações necessárias de acordo com o objetivo estabelecido acima, descrevendo os relacionamentos como junções internas. Exemplo:

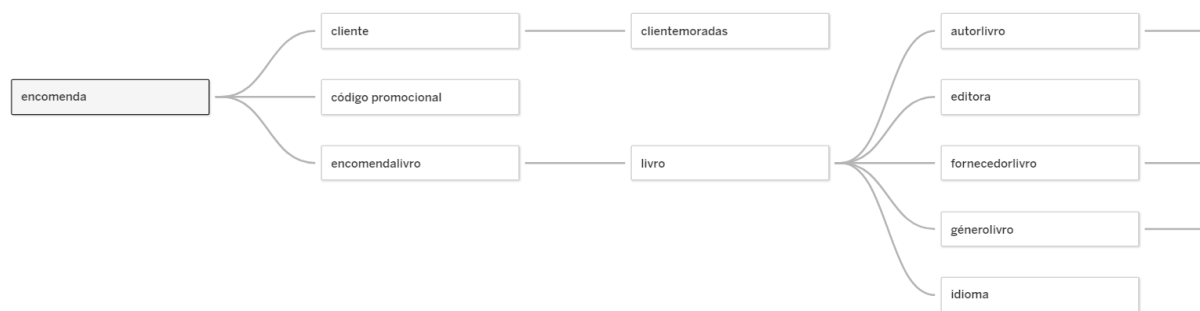


Figura 18 - Exemplo de relacionamentos no *Tableau*

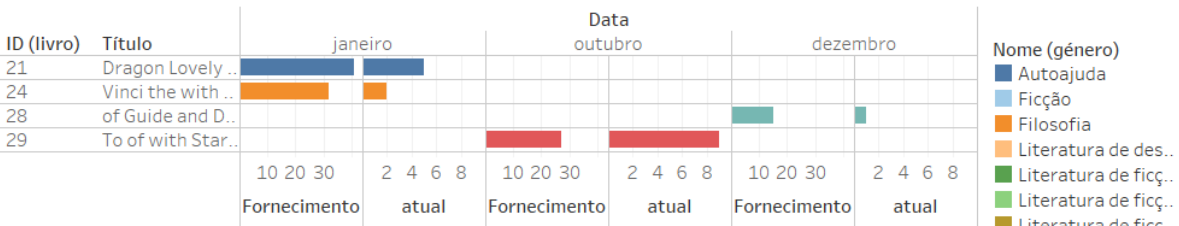
Nome				#	encomenda	encomenda	encomenda	encomenda	Abc	encomenda	encomenda
encomenda				ID	Data de entrega	Data de envio	Data de pagamento	Método de pagamento	Estado		
Campos											
Tipo	Nome de campo	Tabela física	Nome de ...								
#	ID	encomenda	ID	1	11/06/2023 01:51:14	28/05/2023 00:00:55	11/05/2023 20:56:39	PayPal	Finalizada		
				2	11/06/2023 00:00:18	04/06/2023 04:05:46	01/05/2023 00:05:20	Transferência Bancária	Entregue		
				3	12/06/2023 02:01:07	10/06/2023 05:54:47	22/05/2023 04:45:40	Transferência Bancária	Entregue		
				4	20/06/2023 00:29:20	08/06/2023 04:21:24	01/05/2023 00:00:47	PayPal	Entregue		
				5	16/06/2023 11:16:52	02/06/2023 00:30:21	01/05/2023 00:01:34	PayPal	Enviada		
				6	12/06/2023 11:14:57	30/05/2023 06:40:07	22/05/2023 04:47:44	PayPal	Entregue		

Figura 19 - Tabela atualizada no *Tableau*

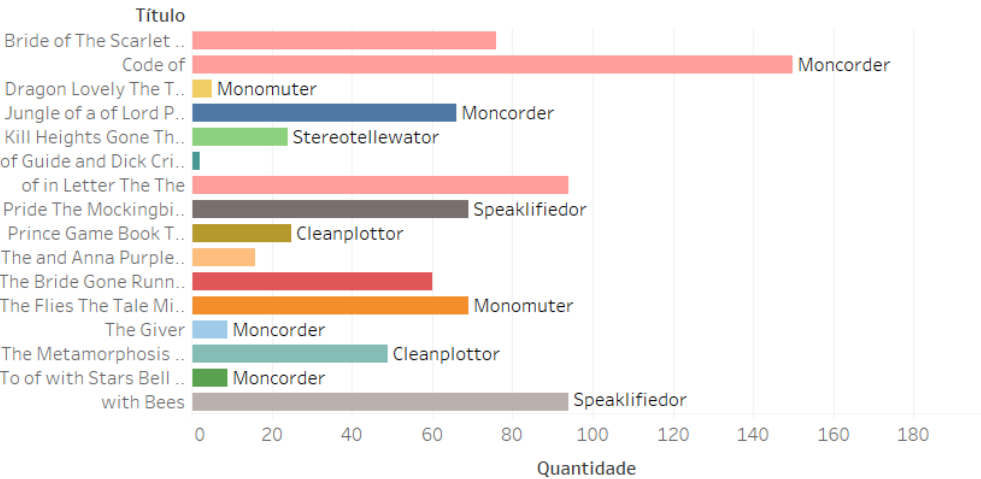
7.3 Apresentação e caracterização dos dashboards implementados

Dashboard de análise de encomendas e fornecimentos:

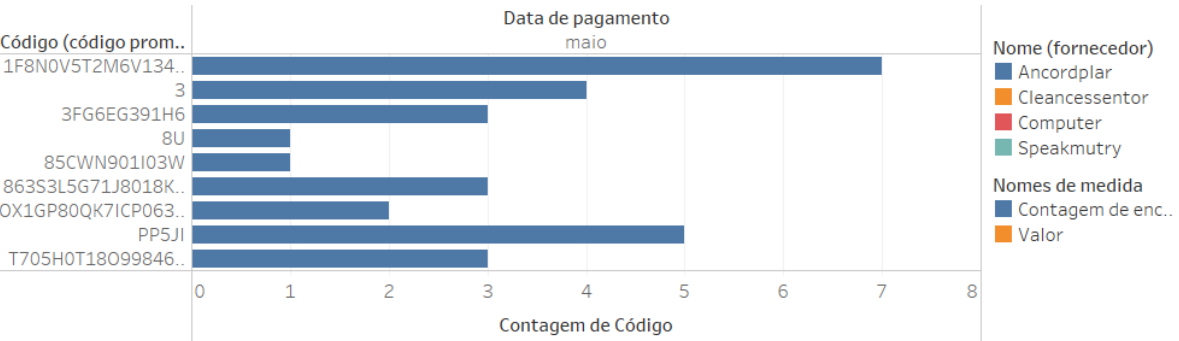
Livros com baixa quantidade: data e quantidade dos fornecimentos



Quantidade total encomendada por livro



Quantidade de códigos promocionais usados no mês de maio



Dinheiro ganho com encomendas no mês de maio

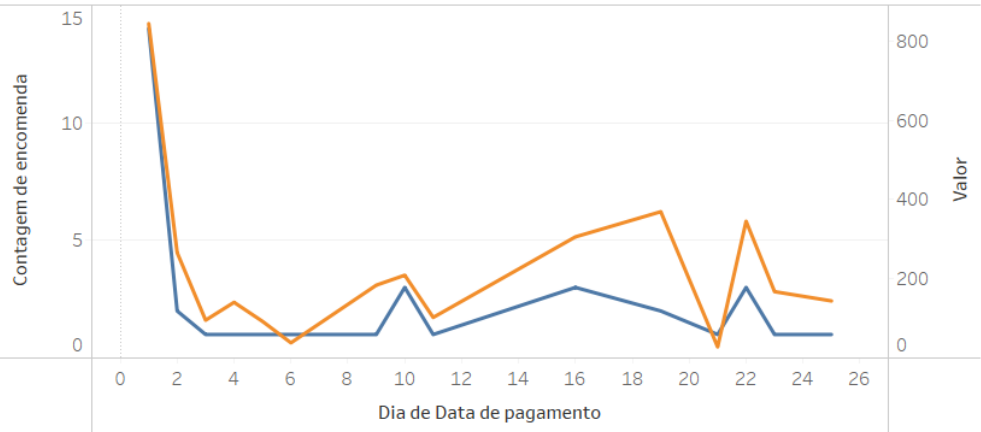


Figura 20 - Dashboard do Tableau

8. Conclusões e Trabalho Futuro

Após a realização deste projeto, foi nos possível aprofundar melhor os conhecimentos adquiridos ao longo do semestre nesta unidade curricular, transportando-os da parte teórica para a parte prática.

Durante a realização do mesmo, as principais dificuldades obtidas foram a conciliação de horários entre os elementos do grupo e com os outros projetos e avaliações das outras cadeiras, que nos obrigaram a um maior esforço e dedicação para obter este resultado final e também a utilização do power bi em que surgia sempre um erro aquando da sua utilização.

Em suma, pensamos ter correspondido bem aos objetivos do trabalho e, consequentemente, conseguimos concluir o trabalho com sucesso.

9. Referências Bibliográficas

- mysql2 - npm: <https://www.npmjs.com/package/mysql2>
- MySQL documentation - Storage Requirements:
<https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html>
- Tableau documentation - Dashboards:
<https://help.tableau.com/current/pro/desktop/en-us/dashboards.htm>