Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

AppGo

Inês Alves (A81368)
João Nuno Lopes(A80397)
João Pedro Gomes (A82428)
Pedro Barros (A81953)

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

AppGo

Inês Alves
João Nuno Lopes
João Pedro Gomes
Pedro Barros

Novembro, 2018

Resumo

No âmbito da Unidade Curricular de Bases de Dados, foi realizada uma base de dados relacional cujo tema escolhido pelo grupo foi uma Loja de Aplicações, denominada “AppGo”.

Ao longo deste relatório serão apresentados todos os passos para criação do sistema em causa, desde o Modelo Concetual até ao Modelo Físico.

Numa primeira fase de contextualização, são apresentados os requisitos levantados, assim como a apresentação de um Modelo ER, utilizando a ferramenta brModelo. É também detalhado e apresentado um modelo lógico utilizando o MySQL Workbench.

Finalmente, é apresentada a implementação física da base de dados, representando então a passagem do modelo lógico para o modelo físico, assim como a passagem dos requisitos estabelecidos anteriormente pelo utilizador e pelo publicador para a linguagem SQL.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados no âmbito de uma aplicação de transferências de produtos via smartphone

Palavras-Chave: Bases de Dados Relacionais, Modelo Concetual, Modelo Lógico, Modelo Físico, MySQL Workbench, SQL, Análise de Requisitos, Atributos, Entidades, Relacionamentos

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

A fim de melhorar a qualidade de vida das pessoas e contribuir para a evolução tecnológica mundial, pretende-se criar um sistema de Base de Dados capaz de suportar diversas transferências de produtos úteis ao dia-a-dia da população. Esta Base de Dados deverá conseguir guardar que produtos o utilizador transferiu, assim como quando é que essa transferência foi efetuada, e que produtos foram publicados pelo Publicador. Iniciaremos esta implementação pela breve contextualização da situação, descrevendo assim, com mais detalhe, os motivos que levaram à criação deste sistema.

1.1. Contextualização

Em 2005, a Maria acabou a Licenciatura em Engenharia Informática, na Universidade do Minho. Com o aparecimento dos smartphones, reparou que a transferência de aplicações era um processo muito complicado para o cidadão comum, isto é, mais distante da área das tecnologias. Posto isto, teve a ideia de criar uma loja de aplicações, de seu nome “AppGo”, com o objetivo de facilitar o acesso a estas transferências.

No entanto, enfrentou muitos obstáculos. No início, a maioria das empresas sentia-se relutante a esta nova evolução. Após a ideia ter sido exposta e esclarecida, foi apoiada por algumas empresas de renome. Começou assim, uma jornada em direção à melhoria da qualidade de vida dos seus clientes.

Com o passar do tempo, a loja sofreu um processo evolutivo relevante. No princípio, possuía apenas jogos e atividades lúdicas. Com a evolução referida, a Maria pretendia acrescentar filmes, livros e música, de forma a que os seus clientes tivessem hipótese de expandir a sua área cultural.

1.2. Fundamentação da implementação da Base de Dados

Tendo em conta os problemas identificados pela Maria no que toca à falta de evolução da área tecnológica e, sabendo que uma das áreas com maior taxa de crescimento é esta, a personagem aproveitou esta oportunidade para trazer um conforto diferente ao dia-a-dia das pessoas e resolver diferentes problemas de eficiência quanto à transferência de aplicações.

Sendo que a AppGo se trata de uma aplicação de telemóvel que armazena um grande número de produtos, torna-se fundamental a implementação de uma base de dados. Sem uma base de dados torna-se muito trabalhoso realizar operações de busca de informações relativas às entidades intervenientes, por exemplo, saber quantos utilizadores é que transferiram um produto, quantos produtos de uma certa categoria foram publicados por um publicador, quais os produtos mais caros, autenticação de um utilizador, etc. Através da base de dados, a solução a todos estes problemas é-nos apresentada de uma forma quase instantânea o que permite a sua consulta para, por exemplo, fins estatísticos, muito mais prática. Para além disso, também a Maria vê aqui uma oportunidade de ascender na sua carreira e de se tornar um ícone na área tecnológica.

Em suma, uma base de dados é um elemento fundamental de qualquer serviço, tanto físico como digital, já que, atualmente, é necessário a consulta e o armazenamento de um grande número de dados e a sua eficiência é muito superior à que nos é apresentada em registos físicos.

1.3. Análise da Viabilidade do Processo

A AppGo é uma entidade que, apesar de recente, pretende ser bastante útil pelos seus utilizadores e trazer uma melhoria na qualidade de vida destes. Para isto, é necessária uma análise mais aprofundada da mesma, de modo a garantir que esta não caminhe em direção à falência.

Com o intuito de combater esta hipótese, é importante ter em conta os custos associados à criação da aplicação e em quanto tempo esta trará lucro para quem a administra. Ou seja, para dirigir esta aplicação, inicialmente apenas a Maria era suficiente. No entanto, com o crescer da mesma, foi necessário empregar mais pessoas cuja área de interesse fosse a mesma. Posto isto, podemos concluir que o investimento feito pela Maria, foi rapidamente recuperado, uma vez que, melhorou de tal modo o dia-a-dia dos seus utilizadores, que necessitou de mais empregados para a ajudar na administração da AppGo. É de salientar ainda que, quanto maior for a quantidade de utilizadores da aplicação, mais empregados serão necessários contratar, fazendo esta aplicação render cada vez.

Para além disso, com esta nova aplicação permite uma resposta rápida às mais diversas necessidades dos seus utilizadores, isto é, apenas com uma transferência, estes podem obter a aplicação que desejam. Também para os publicadores há, obviamente, vantagens: divulgam o seu trabalho e lucram com o mesmo.

Outra vantagem deste sistema é a possibilidade de vários utilizadores/publicadores acederem ao mesmo tempo à aplicação, assim como o facto desta ser flexível, ou seja, por exemplo, mesmo que um publicador altere algo no seu produto publicado, nada afeta outros produtos, podendo os mesmos serem acedidos de igual forma.

A partir desta breve observação, é seguro afirmarmos que a AppGo é viável e uma mais valia na vida dos seus utilizadores.

2. Análise e Levantamento de requisitos

Este capítulo refere-se aos requisitos necessários para a elaboração deste projeto. O projeto retrata uma Loja de Aplicações. 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 AppGo foram previstos após a realização de um inquérito (Inquérito 1 disponível na secção Anexos) a pessoas, em formato anónimo, onde nos foi possível identificar de que maneira poderia esta aplicação ser de fácil utilização para toda a população, procurando facilitar a vida quotidiana. Posto isto, os requisitos levantados perante esta abordagem são descritos a seguir.

2.2. Requisitos Levantados

2.2.1. Requisitos de Descrição

1. Quando um utilizador se regista na AppGo tem que fornecer alguns dados obrigatoriamente: Nome de utilizador, Email, Morada (que é composta pela Rua e sua localidade), Data de nascimento e também escolher uma password. Após registo, o sistema atribui a este novo utilizador o seu ID;

2. Quando um publicador se regista na AppGo tem que fornecer o nome e uma descrição sua. O sistema atribui-lhe o seu ID;

3. Cada produto publicado tem que conter obrigatoriamente o seu nome, uma descrição, o seu preço e a sua categoria (jogo/aplicação/livro/filme/música). O sistema atribui a este um ID, associa-lhe o ID do seu publicador e ainda uma classificação que é nula enquanto não for avaliado por ninguém. Este produto pode ter comentários associados se forem feitos por algum utilizador. Cada comentário tem guardado o seu conteúdo, assim como a data em que foi feito. É-lhe também associado o ID do seu autor e o ID do produto respetivo;

4. Quando um utilizador faz a transferência de um produto, a data desta fica guardada, assim como o ID do utilizador e do produto.

2.2.2. Requisitos de Exploração

Do ponto de vista do Utilizador:

1. Dado o ID de um produto, saber informação sobre o seu publicador;
2. Dado o ID de um produto, saber a designação da categoria desse produto;
3. Saber qual o produto com melhor classificação;
4. Saber quais são os produtos gratuitos;
5. Dado o ID de um utilizador, saber quais são os produtos transferidos por esse utilizador;
6. Dado o ID de um utilizador, saber o comentário mais recente, do último produto transferido por esse utilizador.

Do ponto de vista do Publicador:

1. Dado o ID de um produto, saber que utilizadores o transferiram e a data dessa transferência;
3. Dado o ID de um produto, saber que comentários foram feitos a esse produto;
4. Dado o ID de um publicador, saber qual o seu produto com melhor classificação;

Ponto de vista geral da Aplicação:

1. Top 10 melhores Produtos presentes na Loja;
2. Média das idades dos utilizadores registados na Loja;
3. Média do tamanho das aplicação existentes na Loja.

2.2.3. Requisitos de Controlo

1. Um utilizador pode saber os produtos de um publicador, assim como o seu nome e descrição;
2. Um publicador pode saber o número de utilizadores que descarregaram o seu produto;
3. Um utilizador e um publicador podem aceder a toda a informação de uma aplicação com a exceção do seu ID;
4. Um utilizador pode transferir produtos, avaliar e comentar;
5. Um publicador pode inserir mais produtos na aplicação;
6. Um publicador pode alterar dados de um produto que lhe pertença.

2.3. Análise Geral de Requisitos

Em suma, é possível utilizar a aplicação sendo um utilizador, um publicador ou um administrador.

→ O utilizador pode transferir, comentar e avaliar produtos da loja.

→ O publicador pode publicar novos produtos.

→ O administrador pode fazer alterações na loja, assim como aceder a todos os dados que estão presentes nela sobre todos os que utilizam a loja.

3. Modelação Concetual

Após definição de todos os requisitos necessários para a implementação da Base de Dados, podemos criar um Modelo Concetual para representação da mesma. Este modelo é fundamental para o desenvolvimento de qualquer Base de Dados. 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 e/ou relacionamentos;
4. Determinação do domínio dos Atributos;
5. Identificação de chaves primárias, candidatas e estrangeiras;
6. Detalhar e generalizar (ou não) as entidades;
7. Apresentação e explicação do diagrama ER;
8. Validação do Modelo de dados com o utilizador

3.2. Identificação e caracterização das entidades

- **Utilizador**: entidade que representa um utilizador que interage com a loja apenas descarregando, avaliando e comentando produtos;
- **Publicador**: esta entidade representa todos os que publicam produtos na loja. A sua interação com a loja apenas passa por publicar novos produtos em seu nome;
- **Produto**: define todos os produtos diferentes que estão na loja para serem descarregados. Estes produtos são postos à venda pelos publicadores.

Entidades	Descrição	Aliases	Ações
Utilizador	Quem efetua transferências e comentários	Consumidor	Um Utilizador transfere um produto e faz comentários
Publicador	Quem coloca o produto na Loja para posteriormente serem transferidos	Empresas de aplicações Escritores Artistas	Um Publicador publica os seus produtos.
Produto	Artigo à disposição do utilizador	Aplicação Multimédia	O Produto representa o conteúdo disponível na loja
Categoria	Tipo de produto na loja	Aplicação Filme Música Livro Jogo	A categoria está associada a um produto e representa o tipo da mesma
Morada	Localidade do utilizador	Cidade	A morada está associada a um utilizador e representa o local onde este vive

Table 1: Identificação e caracterização das entidades

3.3. Identificação e caracterização dos relacionamentos

- Um publicador está associado ao Produto que publica. Esta relação é de 1 para N, pois um publicador pode publicar múltiplos produtos na loja;
- Um produto está relacionado com o utilizador que o transfere. Esta ligação é de N para N uma vez que muitos utilizadores podem descarregar o número de produtos que quiserem. É ainda guardada a data da transferência e também é feito um segundo relacionamento de N para N que vai constituir um comentário. Este guarda o conteúdo e a data deste comentário;
- Podem existir vários produtos da mesma categoria;
- Cada utilizador tem associada uma única morada, mas uma morada pode ter vários utilizadores.

⇒ **Relacionamento Produto – Publicador**

O relacionamento "publica" entre a entidade Produto e a entidade Publicador representa a possibilidade de 1 publicador poder publicar N produtos. De salientar que pode haver publicadores registados que não tenham, até ao momento, publicado qualquer produto.

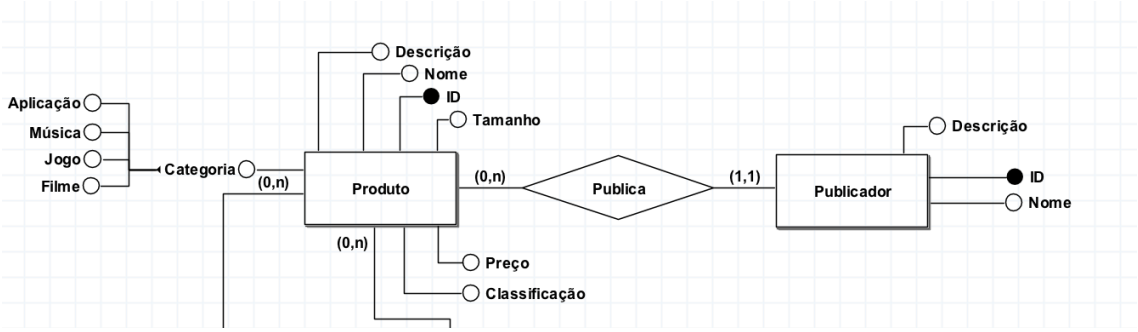

Figure 1: Relacionamento Produto - Publicador

⇒ **Relacionamento Produto – Utilizador**

Como podemos verificar na figura seguinte, existem 2 relacionamentos entre a Entidade Produto e a Entidade Utilizador. O relacionamento "transfere" possui um atributo Data, que indica em que data foi efetuada a transferência de um determinado produto. Para além disso, na nossa implementação, foi assumido que haveria pelo menos 1 utilizador incluído na base de dados. Tal como anteriormente, é possível concluir que 1 utilizador, apesar de estar registado na aplicação, pode não ter transferido qualquer produto até ao momento.

Já relativamente ao relacionamento "comenta", temos dois atributos: conteúdo, que nos indica o que foi escrito; e o atributo data, que nos dá informação sobre a data em que o comentário foi efetuado pelo utilizador. Novamente, um utilizador registado pode não ter feito nenhum comentário ao produto, daí a cardinalidade (0,n).

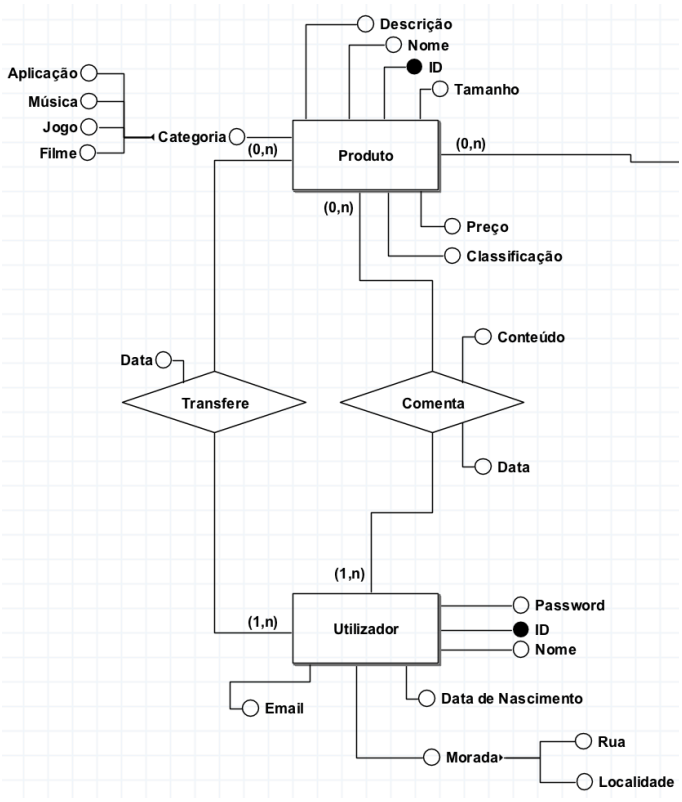

Figure 2: Relacionamentos Produto - Utilizador

3.4. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

- Um utilizador tem como chave primária o seu ID que é um identificador. Tem como atributos simples: nome (nome do utilizador), data de nascimento, email e password. Como atributo composto tem a sua morada que é composta pela rua e a localidade;
- O produto tem como chave primária o seu ID para identificação. Como atributos simples tem: o tamanho (do produto em MBytes), o seu nome a sua descrição, o seu preço e a sua classificação (de 0 a 5 valores). Como atributo composto tem a sua categoria, esta pode ser: Aplicação, Jogo, Livro, Filme e Música;
- O publicador tem como chave primária o seu identificador tal como as outras entidades. Como atributos simples tem o seu nome e a sua descrição.

Nome da entidade	Multiplicidade	Relação	Atributos	Descrição	Multiplicidade	Nome da Entidade
Utilizador	(1,n)	Transfere	Data	Data da transferência do produto	(0,n)	Produto
Utilizador	(1,n)	Comenta	Conteúdo Data	Conteúdo do comentário efetuado; Data em que foi efetuado o comentário	(0,n)	Produto
Publicador	(1,1)	Publica	-		(0,n)	Produto

Table 2: Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

3.5. Detalhe ou generalização de entidades

Nome da Entidade	Atributos	Descrição	Tipo de Dados	Null	Multivalorado	Derivado
Utilizador	IdUtilizador	Identificador único do utilizador	INT	Não	Não	Não
	Email	Email do utilizador	VARCHAR(45)	Não	Não	Não
	Nome	Nome de utilizador	VARCHAR(45)	Não	Não	Não
	Data_Nascimento	Data de Nascimento do utilizador	DATE	Não	Não	Não
	Password	Palavra-passe do utilizador	VARCHAR(45)	Não	Não	Não
Produto	IdProduto	Identificador único do produto	INT	Não	Não	Não
	Nome	Nome do produto	VARCHAR(45)	Não	Não	Não
	Tamanho	Tamanho do produto	INT	Não	Não	Não
	Descrição	Descrição do produto	VARCHAR(45)	Não	Não	Não
	Preço	Preço do produto	INT	Não	Não	Não
	Classificação	Classificação do produto	INT	Não	Não	Não
Publicador	IdPublicador	Identificador único do publicador	INT	Não	Não	Não
	Nome	Nome do publicador	VARCHAR(45)	Não	Não	Não
	Descrição	Descrição do publicador	VARCHAR(100)	Não	Não	Não

Morada	IdMorada	Identificador único da morada	INT	Não	Não	Não
	Rua	Rua específica da morada	VARCHAR(100)	Não	Não	Não
	Localidade	Localidade específica da morada	VARCHAR(45)	Não	Não	Não
Categoria	IdCategoria	Identificador único da categoria	INT	Não	Não	Não
	Designação	Designação da categoria	VARCHAR(45)	Não	Não	Não

Table 3: Detalhe e generalização de entidades

3.6. Apresentação e explicação do diagrama ER

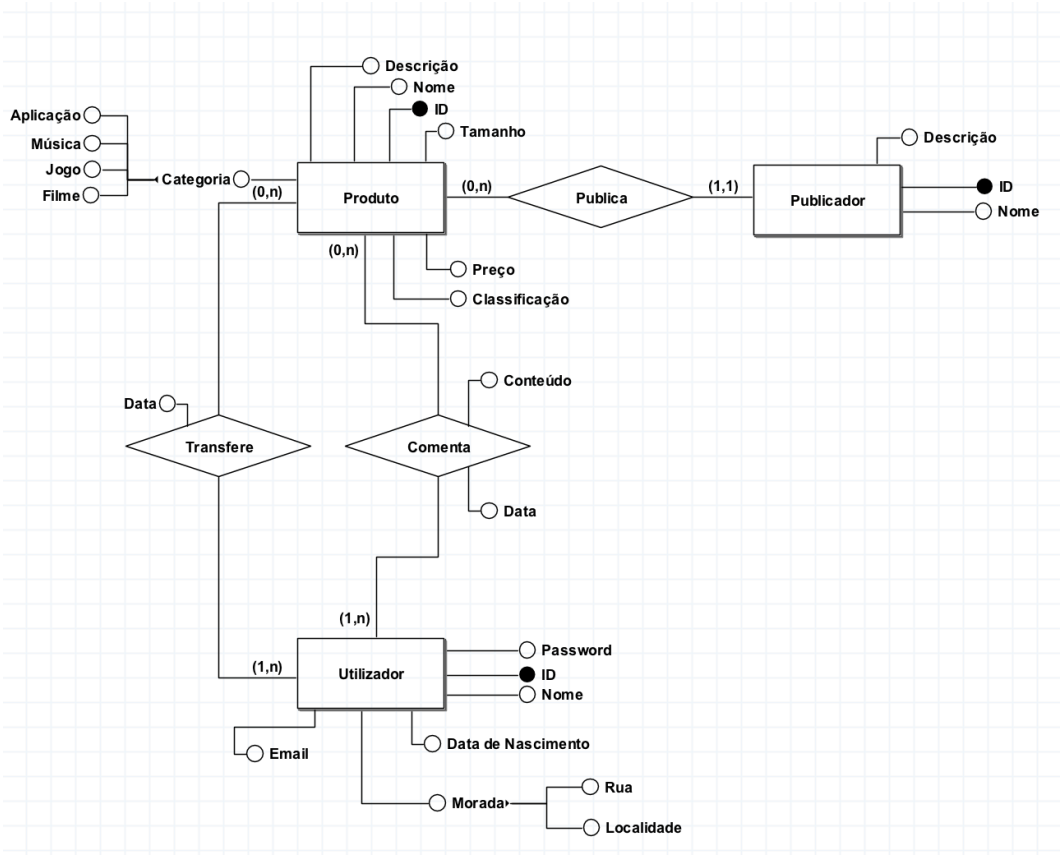

Figure 3: Modelo Concetual

Utilizando a ferramenta brModelo foi elaborado o diagram ER acima apresentado.

Fazendo uma análise mais detalhada do diagrama podemos confirmar que todos os requisitos de exploração anteriormente mencionados conseguem respondidos através deste.

No entanto, neste ponto, é importante falar de chaves candidatas.

- **Entidade Utilizador**

Esta entidade possui como chaves candidatas o seu ID, o seu nome e o seu email. Contudo, a escolha do ID para chave primária é óbvia, uma vez que, para todos os efeitos, terá um tamanho mais pequeno do que o nome de utilizador, levando a uma procura mais rápida. Já o email, apesar de ser uma chave candidata, pode ser alterado, deixando de ser uma hipótese para PK.

- **Entidade Produto**

Esta entidade possui como chaves candidatas o seu ID e o seu nome, no entanto, a razão para o ID ser a sua PK é a mesma que foi usada na entidade acima.

- **Entidade Publicador**

Esta entidade possui as mesmas chaves candidatas que a entidade produto: o seu ID e o seu nome, logo a razão da escolha da PK é a mesma utilizada em ambas as entidades acima referidas.

3.7. Validação do modelo de dados com o utilizador

Com o intuito de verificar se o objetivo principal da base de dados estava em meios de ser cumprido, numa fase de pré-lançamento da aplicação, escolhemos duas pessoas aleatórias para a testar. Uma delas numa faixa etária entre os 10-20 anos e outra entre os 50-70. Ambas realizaram o mesmo inquérito (Inquérito 2, disponível na secção Anexos).

Como ambas as avaliações foram positivas, ou seja, ambas as pessoas consideraram fácil aceder à aplicação e realizar transferências a partir da mesma, concluímos que a criação da aplicação estava num bom caminho e que, portanto, estava validada por parte do utilizador.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Construído o Modelo Concetual, passamos para a construção do Modelo Lógico. Começou-se por criar uma tabela para cada entidade do Modelo Concetual. No entanto, sabemos que, quando existem relações entre tabelas com multiplicidade N:M, é necessária também a criação de uma tabela para essa relação, como é o caso da relação Utilizador-Produto.

Procedeu-se ao preenchimento de cada tabela com os respetivos atributos definidos no Modelo Concetual. Note-se que o atributo Morada é um atributo Composto, pelo que criamos uma tabela própria para este atributo.

4.2. Desenho do modelo lógico

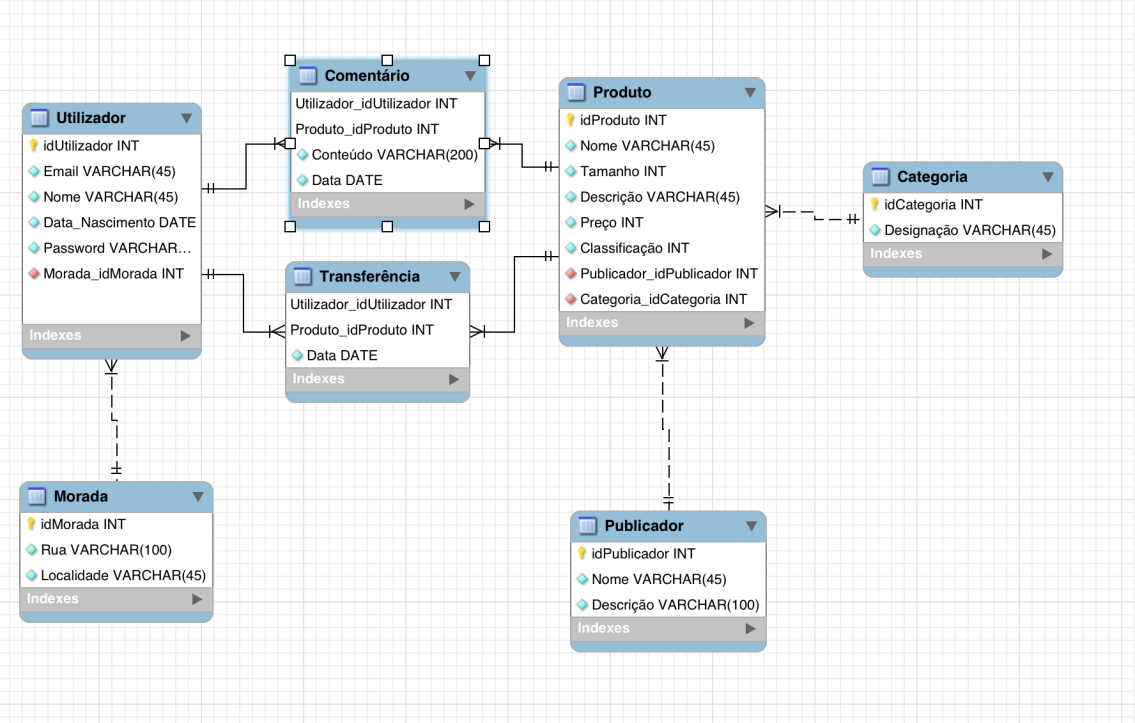

Figure 4: Modelo Lógico

4.3. Validação do modelo através da normalização

O objetivo da normalização é identificar um conjunto adequado de relações que suportem os requisitos da aplicação. Para fazer uso da normalização é necessário que em cada relacionamento, se identifiquem as dependências funcionais existentes entre os atributos:

- **Utilizador**
  IdUtilizador→Email, Nome, Data_Nascimento, Password, Morada_idMorada
- **Produto**
  IdProduto→Nome, Tamanho, Descrição, Preço, Classificação, Publicador_idPublicador, Categoria_idCategoria
- **Publicador**
  IdPublicador→Nome, Descrição
- **Morada**
  IdMorada→Rua, Localidade
- **Categoria**
  IdCategoria→Designação
- **Comentário**
  Utilizador_idUtilizador, Produto_idProduto→Conteúdo, Data
- **Transferência**
  Utilizador_idUtilizador, Produto_idProduto→Data

⇒ Primeira Forma Normal (1FN)

Uma relação está na 1ª Forma Normal se:

- Cada atributo contém apenas valores atómicos (ou seja, não podem ser decompostos);
- Não há conjuntos de atributos repetidos descrevendo a mesma característica.

⇒ Segunda Forma Normal (2FN)

Uma relação está na 2.ª Forma Normal se:

- Pertencer à 1FN;
- Todos os atributos que não são chaves, dependem completamente da chave primária.

⇒ **Terceira Forma Normal (3FN)**

Uma relação está na 3.ª Forma Normal se:

- Pertencer à 2FN;
- Todos os atributos que não são chaves, dependem completamente da chave primária e não dependem de um outro atributo que, por sua vez, dependesse da chave primária.

Posto isto, sabemos que o nosso Modelo Lógico segue as três primeiras regras de normalização:

1FN – Confirmamos que, em todas as tabelas, cada valor da sua chave primária corresponde a uma única linha da respetiva tabela. Podemos verificar isto nas tabelas Utilizador e Publicador, onde cada um tem o seu ID, que é único. De notar que um Publicador, apesar de poder transferir um produto publicado por si, não vai ser a partir do mesmo ID, uma vez que para ter acesso às permissões de Publicador tem que se autenticar de forma diferente.

Já, por exemplo, nas tabelas Comentário e Transferência, têm chaves compostas constituídas por chaves estrangeiras. Tendo em conta que sabemos que as chaves estrangeiras destas são únicas, sabemos, então, por transitividade, que as chaves compostas destas tabelas também são únicas.

2FN – Verificou-se que não existem dependências parciais no Modelo Lógico. Dados que seriam repetidos nas tabelas têm a sua própria tabela (e.g.: tabela Morada).

3FN – Verificou-se que o modelo não tem dependências transitivas, isto é, não há atributos redundantes nas tabelas (e.g.: não existe o atributo NomeUtilizador na tabela Transferência uma vez que o nome do utilizador pode ser derivado do seu ID).

Entidade	Identificador único da Entidade	Dependências
Utilizador (A)	IdUtilizador (a1)	Email (a2)
		Nome (a3)
		Data_Nascimento (a4)
		Password (a5)
		Morada_idMorada (d1)
Produto (B)	IdProduto (b1)	Nome (b2)
		Tamanho (b3)
		Descrição (b4)
		Preço (b5)
		Classificação (b6)
		Publicador_idPublicador (c1)
		Categoria_idCategoria (e2)
Publicador (C)	IdPublicador (c1)	Nome (c2)
		Descrição (c3)

Morada (D)	IdMorada (d1)	Rua (d2)
		Localidade (d3)
Categoria (E)	IdCategoria (e1)	Designação (e2)
Comentário (F)	Utilizador_idUtilizador (a1) Produto_idProduto (b1)	Conteúdo (f1) Data (f2)
Transferência (G)	Utilizador_idUtilizador (a1) Produto_idProduto (b1)	Data (g1)

Table 4: Validação do modelo através da normalização

A = {a1, a2, a3, a4, a5, d1}

B = {b1, b2, b3, b4, b5, b6, c1, e2}

C = {c1, c2, c3}

D = {d1, d2, d3}

E = {e1, e2}

F = {a1, b1, f1, f2}

G = {a1, b1, g1}

4.4. Validação do modelo com interrogações do utilizador

Considere-se que, sempre que é referido o ID de uma entidade, está-se a referir à chave primária dessa mesma entidade.

- **Saber qual é o publicador de um certo produto**

Para saber qual o publicador de um certo produto é necessário juntar as tabelas Publicador e Produto. Com esta junção, procuramos que Publicador tem um ID igual à chave estrangeira do Produto (Publicador_idPublicador). Após esta pesquisa, dando o nome do produto que queremos encontrar, obtemos o resultado.

- **Saber a designação da categoria de um produto**

Para saber que designação tem uma dada categoria de um dado produto é necessário juntar as tabelas Produto e Categoria. Com esta junção, procuramos que Categoria tem um ID igual à chave estrangeira do Produto (Categoria_idCategoria). Após esta pesquisa, dando o nome do produto que queremos encontrar, obtemos o resultado.

- **Saber qual o produto com melhor classificação**

Para dar resposta a este requisito é apenas necessário ordenar de forma decrescente o atributo Classificação da entidade Produto e limitar o número de resultados a 1 valor. Deste modo, só aparece um resultado da ordenação.

- **Saber quais são os produtos gratuitos**

Neste requisito a resposta é simples. Basta procurar em toda a tabela Produto, que produtos têm preço nulo.

- **Saber quais são os produtos transferidos por um utilizador**

Para dar resposta a este requisito é necessário fazer uma junção das tabelas Utilizador, Transferência e Produto. Após esta junção, procuramos que Utilizador tem um ID igual à chave estrangeira de Transferência (Utilizador_idUtilizador) e ainda que Produto tem um ID igual a esta mesma chave estrangeira de Transferência. Assim, dado o nome do utilizador que pretendemos, obtemos o resultado.

- **Saber o comentário mais recente, do último produto transferido por um utilizador**

Para saber o comentário mais recente, do último produto transferido por um dado utilizador, faz-se a junção das tabelas Comentário, Utilizador, Produto e Transferência. Depois da junção ser feita, procuramos que Utilizador tem um ID igual à chave estrangeira de Comentário (Utilizador_idUtilizador) e que Produto tem um ID igual a esta mesma chave estrangeira de Comentário. Verificamos ainda que Utilizador tem um ID igual à chave estrangeira de Transferência (Utilizador_idUtilizador) e damos o nome do Utilizador que pretendemos. Após estas confirmações, ordenamos as transferências e os comentários por Data, por ordem decrescente e limitamos a quantidade de resultado a 1 valor, uma vez que só nos interessa um único comentário: o mais recente.

4.5. Validação do modelo com as transações estabelecidas

Este capítulo refere-se à inserção de dados na base de dados, a fim de efetuar uma gestão dos inputs recebidos, ou seja, caso ocorra algum erro, a inserção não se realiza.

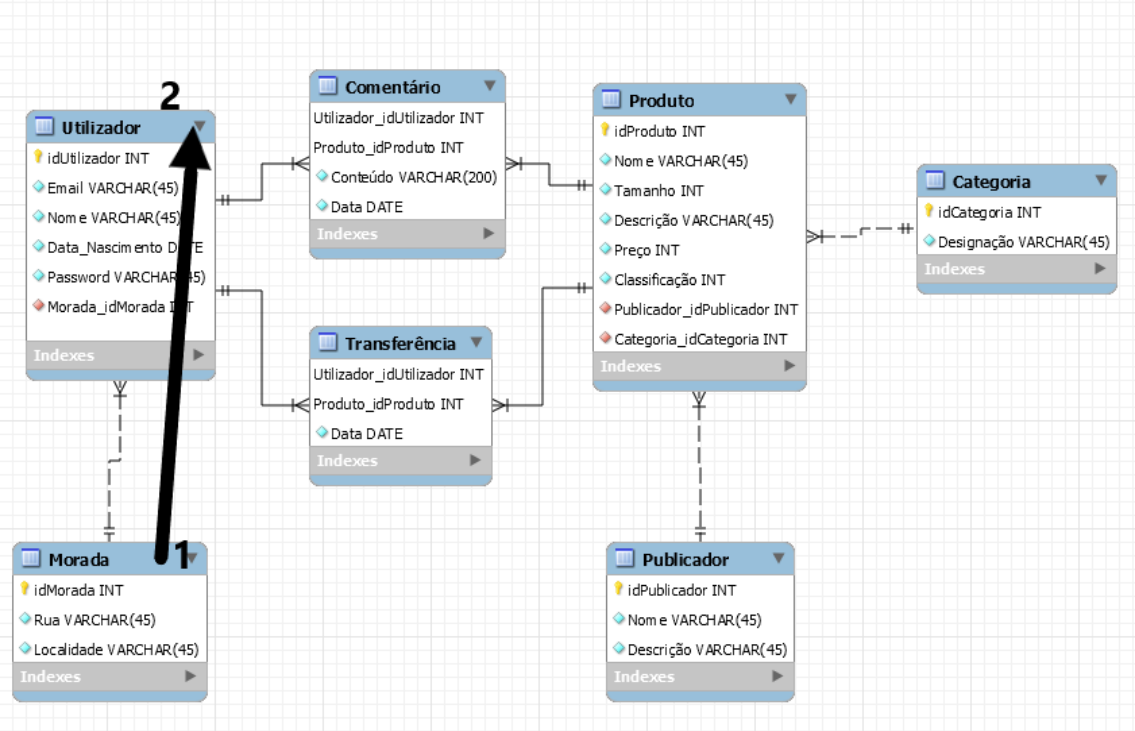

Figure 5: Transação que descreve a inserção de uma nova morada

1 - Inserção de uma morada na base de dados;

2 – Registo de utilizador e associação entre utilizador e a sua morada correspondente.

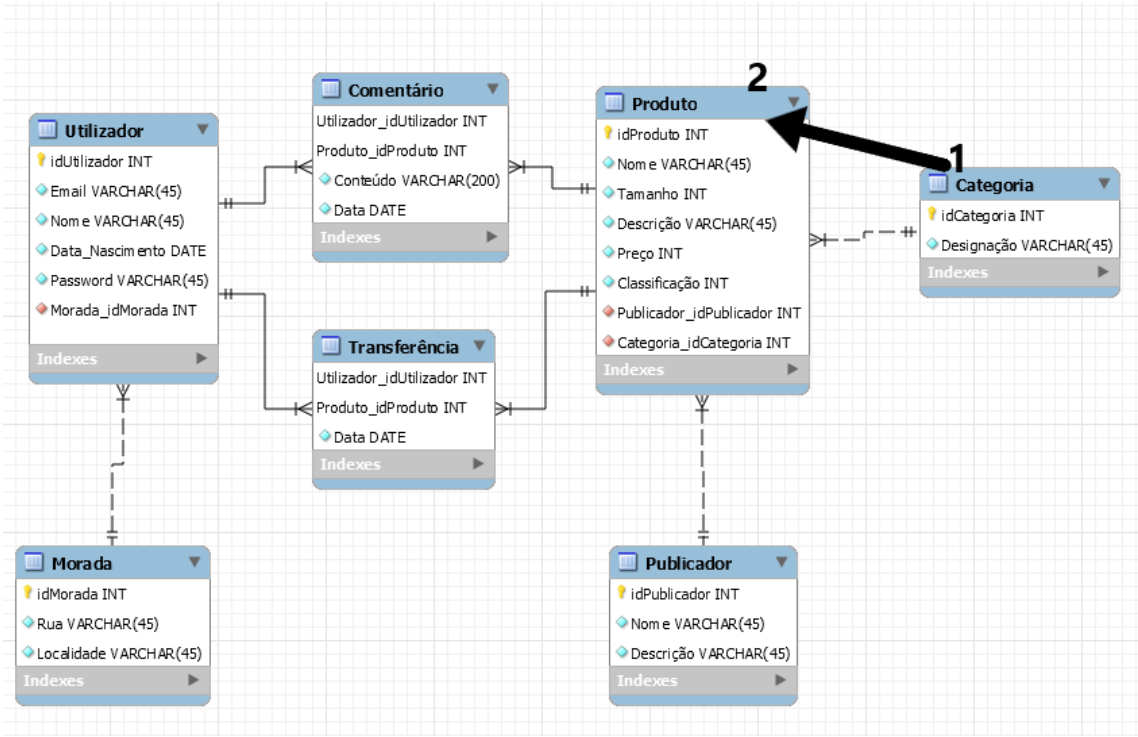

Figure 6: Transação que descreve a inserção de uma categoria

1 – Inserção de uma categoria na base de dados;

2 – Registo de um produto e associação entre produto e a sua categoria correspondente.

4.6. Revisão do modelo lógico com o utilizador

Tal como anteriormente, a nossa aplicação foi posta à prova por utilizadores escolhidos ao acaso, a fim de perceber se as suas necessidades estavam a ser respeitadas. Partindo do princípio que o Utilizador não tem conhecimentos sobre Bases de Dados ou Sistemas de Gestão de Bases de Dados, procuramos não envolver demasiados termos e conceitos técnicos, de modo a facilitar a compreensão da aplicação para o mesmo. Deste modo, o Modelo Lógico foi mostrado e explicado detalhadamente, em linguagem corrente e pessoalmente, aos dois utilizadores previamente escolhidos para avaliar a aplicação, possibilitando também que todas as dúvidas que surgissem fossem devidamente esclarecidas.

Posto isto, após todas as explicações necessárias, concluímos que o modelo foi aprovado pelos utilizadores e superou até as suas expectativas.

5. Implementação Física

5.1. Seleção do Sistema de gestão de bases de dados

Nesta parte do relatório fazemos a migração do Modelo Lógico para o Modelo Físico.

A Seleção do Sistema de gestão de Bases de Dados foi feita pelo grupo que realizou o trabalho, uma vez o utilizador não possuía conhecimento suficiente para efetuar esta escolha. Sendo assim, o grupo escolheu utilizar a ferramenta MySQL Workbench, utilizada nas aulas da Unidade Curricular de Bases de Dados. No entanto, para além da familiarização desta ferramenta, foi também importante perceber que existem algumas vantagens na utilização da mesma.

Para além disso, existem algumas vantagens na utilização desta, tais como o facto de ser um *software open-source*, suportar uma grande quantidade de motores de bases de dados e, ainda assim, uma ferramenta poderosa, no entanto bastante simples de usar.

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

Sendo já conhecido o Modelo Lógico da nossa aplicação, proceder para a implementação do Modelo Físico foi um processo bastante simples. Utilizamos a opção "Forward Engeneering" disponibilizada pelo SGBD escolhido, gerando assim o Modelo Físico.

- Criação da Tabela Publicador

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Publicador` (
  `idPublicador` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Descrição` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idPublicador`))
ENGINE = InnoDB;
```

Figure 7: Modelo Físico - Tabela do Publicador

- Criação da Tabela Categoria

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Categoria` (
  `idCategoria` INT UNSIGNED NOT NULL,
  `Designação` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idCategoria`))
ENGINE = InnoDB;
```

Figure 8: Modelo Físico - Tabela da Categoria

- Criação da Tabela Produto

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Produto` (
  `idProduto` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Tamanho` INT UNSIGNED NOT NULL,
  `Descrição` VARCHAR(45) NOT NULL,
  `Preço` INT UNSIGNED NOT NULL,
  `Classificação` INT UNSIGNED NOT NULL,
  `Publicador_idPublicador` INT UNSIGNED NOT NULL,
  `Categoria_idCategoria` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`idProduto`),
  INDEX `fk_Produto_Publicador1_idx` (`Publicador_idPublicador` ASC) VISIBLE,
  INDEX `fk_Produto_Categoria1_idx` (`Categoria_idCategoria` ASC) VISIBLE,
  CONSTRAINT `fk_Produto_Publicador1`
    FOREIGN KEY (`Publicador_idPublicador`)
    REFERENCES `trabalho`.`Publicador` (`idPublicador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Produto_Categoria1`
    FOREIGN KEY (`Categoria_idCategoria`)
    REFERENCES `trabalho`.`Categoria` (`idCategoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figure 9: Modelo Físico - Tabela do Produto

- Criação da Tabela Morada

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Morada` (
  `idMorada` INT UNSIGNED NOT NULL,
  `Rua` VARCHAR(100) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMorada`))
ENGINE = InnoDB;
```

Figure 10: Modelo Físico - Tabela da Morada

- Criação da Tabela Utilizador

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Utilizador` (
  `idUtilizador` INT UNSIGNED NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Data_Nascimento` DATE NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Morada_idMorada` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`idUtilizador`),
  INDEX `fk_Utilizador_Morada1_idx` (`Morada_idMorada` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_Morada1`
    FOREIGN KEY (`Morada_idMorada`)
    REFERENCES `trabalho`.`Morada` (`idMorada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figure 11: Modelo Físico - Tabela do Utilizador

- Criação da Tabela Comentário

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Comentário` (
  `Utilizador_idUtilizador` INT UNSIGNED NOT NULL,
  `Produto_idProduto` INT UNSIGNED NOT NULL,
  `Conteúdo` VARCHAR(200) NOT NULL,
  `Data` DATE NOT NULL,
  PRIMARY KEY (`Utilizador_idUtilizador`, `Produto_idProduto`),
  INDEX `fk_Utilizador_has_Produto_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
  INDEX `fk_Utilizador_has_Produto_Utilizador1_idx` (`Utilizador_idUtilizador` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_has_Produto_Utilizador1`
    FOREIGN KEY (`Utilizador_idUtilizador`)
    REFERENCES `trabalho`.`Utilizador` (`idUtilizador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Utilizador_has_Produto_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `trabalho`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figure 12: Modelo Físico - Tabela do Comentário

- Criação da Tabela Transferência

```
CREATE TABLE IF NOT EXISTS `trabalho`.`Transferência` (
  `Utilizador_idUtilizador` INT UNSIGNED NOT NULL,
  `Produto_idProduto` INT UNSIGNED NOT NULL,
  `Data` DATE NOT NULL,
  PRIMARY KEY (`Utilizador_idUtilizador`, `Produto_idProduto`),
  INDEX `fk_Utilizador_has_Produto1_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
  INDEX `fk_Utilizador_has_Produto1_Utilizador1_idx` (`Utilizador_idUtilizador` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_has_Produto1_Utilizador1`
    FOREIGN KEY (`Utilizador_idUtilizador`)
    REFERENCES `trabalho`.`Utilizador` (`idUtilizador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Utilizador_has_Produto1_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `trabalho`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figure 13: Modelo Físico - Tabela da Transferência

5.3. Tradução das interrogações do utilizador para SQL

⇒ **Do ponto de vista do Utilizador:**

- Saber qual é o publicador de um certo produto

```
DELIMITER //
CREATE PROCEDURE obterpublicador (IN idProduto INT)
   BEGIN
      SELECT Publicador.*, Produto.idProduto, Produto.Nome FROM Produto
      INNER JOIN Publicador on Publicador.idPublicador = Produto.Publicador_idPublicador
      WHERE Produto.idProduto = idProduto;
   END //
DELIMITER //
```

Figure 14: Procedimento referente à obtenção do publicador de um determinado produto

- Saber a designação da categoria de um produto

```
DELIMITER //
CREATE PROCEDURE obterdesignação (IN idProduto INT)
   BEGIN
      SELECT Produto.Nome, Categoria.Designação From Produto
      INNER JOIN Categoria on Categoria.idCategoria = Produto.Categoria_idCategoria
      WHERE Produto.idProduto = idProduto;
   END //
DELIMITER //
```

Figure 15: Procedimento referente à obtenção da designação de um determinado produto

- Saber qual o produto com melhor classificação

```
DELIMITER //
CREATE PROCEDURE obterproduto_maior_classificação ()
   BEGIN
      SELECT Produto.Nome, Produto.Classificação from Produto
      ORDER BY Classificação DESC
      LIMIT 1;
   END //
DELIMITER //
```

Figure 16: Procedimento referente à obtenção do produto com melhor classificação

- Saber quais são os produtos gratuitos

```
DELIMITER //
CREATE PROCEDURE obterprodutos_gratuitos ()
   BEGIN
      SELECT * FROM Produto
      WHERE Preço = 0;
   END //
DELIMITER //
```

Figure 17: Procedimento referente à obtenção de todos os produtos gratuitos

- Saber quais são os produtos transferidos por um utilizador

```
DELIMITER //
CREATE PROCEDURE obterprodutos_utilizador (IN idUtilizador INT)
   BEGIN
      SELECT Produto.Nome from Utilizador
      INNER JOIN Transferência ON Utilizador.idUtilizador = Transferência.Utilizador_idUtilizador
      INNER JOIN Produto ON Produto.idProduto = Transferência.Produto_idProduto
      WHERE Utilizador.idUtilizador = idUtilizador;
   END //
DELIMITER //
```

Figure 18: Procedimento referente à obtenção de todos os produtos transferidos por um determinado utilizador

- Saber o comentário mais recente, do último produto transferido por um utilizador

```
DELIMITER //
CREATE PROCEDURE obtercomentário_maisrecente_produto_maisrecente (IN idUtilizador INT)
   BEGIN
      SELECT Comentário.Conteúdo, Produto.Nome, Transferência.Data, Comentário.Data from Comentário
      INNER JOIN Utilizador ON Utilizador.idUtilizador = Comentário.Utilizador_idUtilizador
      INNER JOIN Produto ON Produto.idProduto = Comentário.Produto_idProduto
      INNER JOIN Transferência ON Transferência.Utilizador_idUtilizador = Utilizador.idUtilizador
      WHERE Utilizador.idUtilizador = idUtilizador
      ORDER BY Transferência.Data DESC,
            Comentário.Data DESC
      LIMIT 1;
   END //
DELIMITER //
```

Figure 19: Procedimento referente à obtenção do comentário mais recente feito no último produto transferido por um determinado utilizador

⇒ **Do Ponto de Vista do Publicador:**

- Saber que utilizadores transferiram um produto que lhe pertence e em que data foi feita essa transferência

```
DELIMITER //
CREATE PROCEDURE obterproduto_utilizadores (IN idProduto INT)
   BEGIN
      SELECT Produto.Nome, Utilizador.Nome, Publicador.Nome, Transferência.Data FROM Publicador
      INNER JOIN Produto ON Produto.Publicador_idPublicador = Publicador.idPublicador
      INNER JOIN Transferência ON Transferência.Produto_idProduto = Produto.idProduto
      INNER JOIN Utilizador ON Transferência.Utilizador_idUtilizador = Utilizador.idUtilizador
      WHERE Produto.idProduto = idProduto;
   END //
DELIMITER //
```

Figure 20: Procedimento referente à obtenção de todos os utilizadores que transferiram produtos pertencentes a um dado Publicador e em que data isso ocorreu

- Saber que comentários foram feitos a um determinado produto que lhe pertence

```
DELIMITER //
CREATE PROCEDURE obterproduto_comentários (IN idProduto INT)
   BEGIN
      SELECT Produto.nome, Publicador.Nome, Comentário.Conteúdo, Utilizador.Nome FROM Publicador
      INNER JOIN Produto on Produto.Publicador_idPublicador = Publicador.idPublicador
      INNER JOIN Comentário ON Comentário.Produto_idProduto = Produto.idProduto
      INNER JOIN Utilizador ON Comentário.Utilizador_idUtilizador = Utilizador.idUtilizador
      WHERE Produto.idProduto = idProduto;
   END //
DELIMITER //
```

Figure 21: Procedimento referente à obtenção de todos os comentários que foram feitos num determinado produto

- Saber qual o seu produto com melhor classificação

```
DELIMITER //
CREATE PROCEDURE obterproduto_maiorclassificação (IN idPublicador INT)
   BEGIN
      SELECT Produto.Nome, Produto.Classificação, Publicador.Nome from Produto
      INNER JOIN Publicador ON Publicador.idPublicador = Produto.Publicador_idPublicador
      WHERE Publicador.idPublicador = idPublicador
      ORDER BY Produto.Classificação DESC
      LIMIT 1;
   END //
DELIMITER //
```

Figure 22: Procedimento referente à obtenção do produto de um dado Publicador melhor classificado

5.4. Tradução das transações estabelecidas para SQL

- Inserir novo utilizador

```
DELIMITER //
CREATE PROCEDURE insere_utilizador (IN id INT, IN email VARCHAR(45), IN nome VARCHAR(45), IN data DATE,
                        IN password VARCHAR(45), IN morada INT)
   BEGIN
      DECLARE exit handler for SQLEXCEPTION
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      DECLARE exit handler for SQLWARNING
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      START TRANSACTION;
         INSERT INTO Utilizador
            (idUtilizador, Email, Nome, Data_Nascimento,Password,Morada_idMorada)
            VALUES
            (id,email,nome,data,password,morada);
      COMMIT;
   END //
```

Figure 23: Inserção de novo utilizador na BD

- Inserir novo publicador

```
CREATE PROCEDURE insere_publicador (IN id INT, IN nome VARCHAR(45), IN descrição VARCHAR(100))
    BEGIN
        DECLARE exit handler for SQLEXCEPTION
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        DECLARE exit handler for SQLWARNING
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        START TRANSACTION;
            INSERT INTO Publicador
                (idPublicador, Nome, Descrição)
                VALUES
                (id,nome,descrição);
        COMMIT;
    END //
```

Figure 24: Inserção de novo Publicador na BD

- Inserir novo produto

```
CREATE PROCEDURE insere_produto (IN id INT, IN nome VARCHAR(45), IN tamanho INT,IN descrição VARCHAR(100),
                    IN preço INT, IN classificação INT, IN idpublicador INT, IN idcategoria INT)
    BEGIN
        DECLARE exit handler for SQLEXCEPTION
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        DECLARE exit handler for SQLWARNING
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        START TRANSACTION;
            INSERT INTO Publicador
                (idProduto, Nome, Tamanho , Descrição , Preço, Classificação, Publicador_idPublicador, Categoria_idCategoria)
                VALUES
                (id,nome,tamanho,descrição,preço,classificação,idpublicador,idcategoria);
        COMMIT;
    END //
```

Figure 25: Inserção de novo produto na BD

- Inserir nova transferência

```
CREATE PROCEDURE insere_transferencia (IN idutilizador INT, idproduto INT, IN data DATE)
  BEGIN
    DECLARE exit handler for SQLEXCEPTION
      BEGIN
        SHOW ERRORS LIMIT 1;
        SHOW WARNINGS;
        ROLLBACK;
      END;

    DECLARE exit handler for SQLWARNING
      BEGIN
        SHOW ERRORS LIMIT 1;
        SHOW WARNINGS;
        ROLLBACK;
      END;

    START TRANSACTION;
      INSERT INTO Publicador
        (Utilizador_idUtilizador,Produto_idProduto,Data)
        VALUES
        (idutilizador,idproduto,data);
    COMMIT;
  END //
```

Figure 26: Inserção de nova transferência na BD

- Inserir novo comentário

```
CREATE PROCEDURE insere_comentario (IN idutilizador INT, idproduto INT, IN data DATE, IN conteudo VARCHAR(200))
  BEGIN
    DECLARE exit handler for SQLEXCEPTION
      BEGIN
        SHOW ERRORS LIMIT 1;
        SHOW WARNINGS;
        ROLLBACK;
      END;

    DECLARE exit handler for SQLWARNING
      BEGIN
        SHOW ERRORS LIMIT 1;
        SHOW WARNINGS;
        ROLLBACK;
      END;

    START TRANSACTION;
      INSERT INTO Comentário
        (Utilizador_idUtilizador,Produto_idProduto,Conteúdo,Data)
        VALUES
        (idutilizador,idproduto,data,conteudo);
    COMMIT;
  END //
```

Figure 27: Inserção de novo comentário na BD

- Inserir nova categoria

```
CREATE PROCEDURE insere_categoria (IN id INT, IN designação VARCHAR(45))
   BEGIN
      DECLARE exit handler for SQLEXCEPTION
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      DECLARE exit handler for SQLWARNING
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      START TRANSACTION;
         INSERT INTO Categoria
            (idCategoria , Designação)
            VALUES
            (id,designação);
      COMMIT;
   END //
```

Figure 28: Inserção de nova categoria na BD

- Inserir nova morada

```
CREATE PROCEDURE insere_morada (IN id INT,IN rua VARCHAR(100), IN localidade VARCHAR(45))
   BEGIN
      DECLARE exit handler for SQLEXCEPTION
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      DECLARE exit handler for SQLWARNING
         BEGIN
            SHOW ERRORS LIMIT 1;
            SHOW WARNINGS;
            ROLLBACK;
         END;

      START TRANSACTION;
         INSERT INTO Morada
            (idMorada, Rua, Localidade)
            VALUES
            (id,rua,localidade);
      COMMIT;
   END //
```

Figure 29: Inserção de nova morada na BD

5.5. Escolha, definição e caracterização de índices em SQL

- **Porquê criar índices?**

Os índices aceleram a visualização de dados. Por exemplo, imaginemos que temos um livro com 1000 páginas que não tem um índice, reportando o seu conteúdo. Fazer uma procura neste livro não seria muito preocupante, no entanto, fazer várias procuras de diferentes conteúdos, demoraria muito mais tempo. Por outro lado, se o livro tivesse um índice, fossem necessárias muitas ou poucas procuras no mesmo, a procura seria quase instantânea, uma vez que erámos informados imediatamente da página onde deveríamos procurar o conteúdo que desejamos.

Os índices das bases de dados funcionam da mesma maneira. Se a base de dados tiver um índice não precisa de ver todas as linhas da tabela.

Vejamos o seguinte exemplo:

Queremos procurar na tabela seguinte o registo em que Código = 3.

Figure 30: Exemplo de procura sem índices

Para realizar esta procura, tivemos que percorrer a tabela toda. Neste exemplo, havia apenas 3 registos, o que não era preocupante, mas em tabelas com muitos mais registos, trata-se, de facto, uma procura lenta. Para melhorar estas procuras utilizamos índices.

De notar que a utilização de índices acelera as cláusulas “WHERE” e “ORDER BY”, mas abranda as cláusulas “INSERT” e “UPDATE”.

- **Porque não criar índices?**

Os índices são muito bons no que toca a melhorar a performance da base de dados, otimizando as procuras, mas, por outro lado, ocupam muito espaço de disco.

Posto isto, nesta primeira fase não nos pareceu necessária a implementação de índices, para além dos criados automaticamente pelo InnoDB, uma vez que a base de dados criada é relativamente pequena. O InnoDB é um mecanismo de armazenamento que faz corresponder a chave primária da tabela a um “índice especial” deste, ou seja, quando é criada uma chave primária, o InnoDB faz com que esta seja um desses índices.

No entanto, com o crescimento da base de dados, principalmente no setor dos produtos, índices podem vir a ser necessários por uma questão de eficiência.

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

Entidade	Atributo	Tipo de dados	Tamanho (em bytes)	Ocorrências	SubTotal (em bytes)
Utilizador	idUtilizador	INT	4	21	3150
	Email	VARCHAR(45)	46		
	Nome	VARCHAR(45)	46		
	Data	Date	3		
	Password	VARCHAR(45)	46		
	Morada_idMorada	INT	4		
Produto	idProduto	INT	4	64	7424
	Nome	VARCHAR (45)	46		
	Tamanho	INT	4		
	Descrição	VARCHAR(45)	46		
	Preço	INT	4		
	Classificação	INT	4		
	Publicador_idPublicador	INT	4		
	Categoria_idCategoria	INT	4		
Comentário	Utilizador_idUtilizador	INT	4	65	13780
	Produto_idProduto	INT	4		
	Conteúdo	VARCHAR(200)	201		
	Data	Date	3		
Transferência	idUtilizador	INT	4	34	374
	idProduto	INT	4		
	Data	Date	3		
Publicador	idPublicador	INT	4	22	3322
	Nome	VARCHAR(45)	46		
	Descrição	VARCHAR(100)	101		
Categoria	idCategoria	INT	4	5	250
	Designação	VARCHAR(45)	46		
Morada	idMorada	INT	4	17	2567
	Rua	VARCHAR(100)	101		
	Localidade	VARCHAR(45)	46		
				Total	30867

Table 5: Tabela auxiliar para estimar o espaço em disco da BD

Tendo em conta a tabela acima podemos calcular, aproximadamente, o espaço ocupado pela base de dados com a seguinte função:

E (U, P, C, T, PUB, CAT, M) = 150U + 116P + 212C + 11T + 151PUB + 50CAT +151M

Logo, assumindo um crescimento anual de 5%, os valores estimados para os próximos anos são os seguintes:

Ano	Tamanho (em bytes)
0	30528
1	32054.4
2	33657.12
3	35339.76
4	37106.75
5	38962.09

Table 6: Tabela representativa do crescimento anual

5.7. 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 realmente armazenados os dados. Em contrapartida, a criação desta tabela temporária provoca uma redução na velocidade de execução das *queries* mas, pelo nosso ponto de vista, é compensada pelo aumento da segurança.

- Top 10 produtos melhor classificados

```
CREATE VIEW Top10Produtos AS
   SELECT Produto.Nome, Produto.Classificação  from Produto
      ORDER BY Produto.CLassificação DESC
   LIMIT 10;
```

Figure 31: Vista referente ao top 10 produtos melhor classificados

- Média das idades dos utilizadores da Loja

```
CREATE VIEW mediaIdades AS
   Select AVG(TIMESTAMPDIFF(YEAR, Data_Nascimento, CURDATE())) from Utilizador;
```

Figure 32: Vista referente à média das idades dos utilizados da Loja

- Média dos tamanhos dos produtos que se encontram na base de dados

```
CREATE VIEW avgSize AS
  SELECT AVG(Tamanho) FROM Produto;
```

Figure 33: Vista referente à média dos tamanhos dos produtos que se encontram na BD

5.8. Definição e caracterização dos mecanismos de segurança em SQL

As BD requerem uma segurança rigorosa e que garanta que o seu conteúdo está protegido de ataques inesperados.

No contexto da nossa BD existem três tipos de utilizadores da mesma: Administrador, Utilizador e Publicador.

O Administrador tem permissão para realizar qualquer operação na BD.

O Utilizador pode inserir e alterar um utilizador na BD, isto é, pode fazer o seu registo e alterá-lo, posteriormente. De notar que é obrigatório que insira uma morada para efetuar o seu registo. Pode ainda selecionar um produto que queira transferir e inserir comentários no mesmo. Para isso realiza uma transferência.

Por fim, o Publicador pode selecionar, inserir e alterar a entidade Publicador, ou seja, tem permissão para fazer o seu registo e alterá-lo, à posteriori, assim como selecionar um publicador para saber informações sobre ele. Para além disso, pode ainda escolher, inserir e alterar um produto que lhe pertença e ainda selecionar um produto que não lhe pertença a fim de obter informações sobre o mesmo.

```
-- Administrador
CREATE USER 'administrador'@'localhost' IDENTIFIED BY 'administrador';
GRANT ALL ON trabalho.* TO 'administrador'@'localhost';

-- Utilizador
CREATE USER 'utilizador'@'localhost' IDENTIFIED BY 'utilizador';
GRANT INSERT, UPDATE ON trabalho.Utilizador to 'utilizador'@'localhost';
GRANT SELECT ON trabalho.Produto to 'utilizador'@'localhost';
GRANT INSERT ON trabalho.Comentario to 'utilizador'@'localhost';
GRANT INSERT ON trabalho.Transferência to 'utilizador'@'localhost';
GRANT INSERT ON trabalho.Morada to 'utilizador'@'localhost';

-- Publicador
CREATE USER 'publicador'@'localhost' IDENTIFIED BY 'publicador';
GRANT SELECT, INSERT, UPDATE ON trabalho.Publicador to 'publicador'@'localhost';
GRANT SELECT, INSERT, UPDATE ON trabalho.Produto to 'publicador'@'localhost';
```

Figure 34: Definição dos mecanismos de segurança

5.9. Revisão do sistema implementado com o utilizador

Uma vez finalizada a implementação da base de dados, foi realizada a apresentação da mesma ao utilizador. Esta apresentação é elaborada com o intuito de perceber se todas as necessidades do utilizador foram saciadas e se os requisitos levantados foram, de facto, cumpridos.

Concluímos que o produto final correspondeu às expectativas do utilizador, levando a uma aprovação da nossa base de dados.

6. Conclusões e Trabalho Futuro

Tal como foi referido, após todo o processo de implementação da base de dados, estruturação e desenvolvimento da mesma, foi feita a apresentação da mesma aos utilizadores que se mostraram agradados com o resultado final, deixando-nos, consequentemente, com um sentimento de missão cumprida.

De um modo geral, todas as etapas foram realizadas com o devido cuidado, procurando sempre a maneira mais simples para se realizarem os objetivos propostos, ou seja, tornar a aplicação uma mais valia para a população, mas também fácil de manusear por parte dos seus utilizadores, tendo em conta que alguns podem estar mais à vontade que outros quanto a utilizar novas tecnologias.

Quanto ao trabalho futuro, espera-se que o número de produtos disponíveis aumente cada vez mais, assim como o número de transferências, tornando a base de dados mais complexa, mas também mais completa e com mais hipóteses de escolha. Com isto, será, provavelmente, necessária a utilização de índices devido ao crescimento da base de dados.

Para além disso, outra questão que é importante analisar é a criação de novas categorias de produtos. Com o aumento de pessoas a utilizar esta aplicação, aumenta também a probabilidade destas quererem novas categorias para além das que estão disponíveis, a fim de aumentar ainda mais o seu conforto e qualidade de vida. Novamente aqui vemos a importância de usar índices, caso se pretenda criar muitas mais categorias, uma vez que mais categorias dão origem a uma "área de manobra" maior por parte dos publicadores.

Concluindo, dada a extensão da base de dados não foram implementados todos os aspetos que conduzissem à otimização da mesma, como é o caso da utilização de índices. No entanto, como vimos, estes não devem ser descartados num caso futuro, caso a extensão da base de dados aumente.

7. Referências Bibliográficas

MySQL, 2018-11-24. MySQL™ Workbench Reference Manual. [online] Disponível em:
<https://dev.mysql.com/doc/workbench/en/> [Acedido em 24 de novembro de 2018].

Joel Rodrigues. Índices MySQL: Otimização de Consultas. [online] Disponível em:
< http://www.linhadecodigo.com.br/artigo/3620/indices-mysql-otimizacao-de-consultas.aspx /> [Acedido em 20 de novembro de 2018]

Thomas Connolly, Carolyn Begg. Fourth edition published 2005. Database Systems - A Pratical Approach to Design, Implement, and Management. Edinburgh Gate, Harlow, Inglaterra: Pearson Education Limited

8. Bases de Dados NoSQL

Antes de começarmos a explicar o porquê da utilização deste sistema no âmbito do nosso trabalho é importante percebermos as características de um sistema NoSQL.

O constante crescimento da tecnologia traduziu-se numa necessidade de reavaliar como armazenar e manter a grande quantidade de dados a que temos acesso, nos dias que correm.

NoSQL (Not Only SQL) é uma denominação para Bases de Dados Não Relacionais, o que não significa que não possuem relacionamentos, mas sim que não são orientados a tabelas. Existem 4 tipos de Bases de Dados NoSQL:

- **Chave-Valor:** um determinado valor é acedido através de uma chave identificadora única sendo esta, então, a que suporta maior quantidade de dados;
- **Grafos:** os dados são armazenados em grafos, ou seja, em vértices e arestas;
- **Colunas:** armazena os dados em linhas particulares de uma tabela no disco, podendo suportar várias linhas e colunas e, permitindo também, sub-colunas;
- **Documentos:** os dados são armazenados como documentos, onde um documento pode ser um dado no formato chave-valor (exemplo: formato JSON)

8.1. Principais Características

1. **Escalabilidade Horizontal**
   A escalabilidade horizontal consiste em aumentar o número de máquinas disponíveis, o que não seria viável em Bases de Dados Relacionais devido a problemas de concorrência. É também importante referir que, SGDB Relacionais revelam problemas na sua performance quando executam “joins”. Como o NoSQL evita “joins” naturalmente, a sua performance mantém-se alta.

2. **Flexibilidade**
   Estruturas de dados intuitivas e flexíveis atraem developers que trabalham em equipas de rápido desenvolvimento. Isto permite uma fácil aplicação da escalabilidade e também um aumento na disponibilidade dos dados.

3. **Replicação**
   Possibilita o armazenamento de dados e backups independentemente do local físico onde os dados estão armazenados, levando a uma diminuição do tempo gasto para a recuperação de informações.

4. **Disponibilidade**
   A indisponibilidade de uma Base de Dados pode causar sérios prejuízos, como por

exemplo, perda de clientes. Tendo em conta que a maioria das Bases de Dados NoSQL oferecem eficientes arquiteturas de replicação de dados, é seguro afirmarmos que isto proporciona a estas Bases de Dados maior disponibilidade. Ou seja, se um ou mais servidores falha, um outro estará apto para continuar o trabalho automaticamente sem perda de dados.

5. **Raízes Open-Source**
   Muitas Bases de Dados NoSQL têm raízes na "comunidade" Open-Source, o que se traduziu num rápido crescimento do seu uso e popularidade.

6. **API Simples**
   Para que o acesso às informações seja feito da forma mais rápida possível, APIs são desenvolvidas para que qualquer aplicação possa ter acesso aos mesmos. APIs simples são uma boa maneira de garantir esta facilidade.

7. **Baixo custo operacional**
   Devido ao peso do Open-Source no NoSQL, o custo para iniciar a utilização dessas Bases de Dados torna-se muito baixo ou zero. É comum ouvir-se dizer que a transição de uma Base de Dados Relacional para uma Base de Dados Não Relacional diminuiu muito os custos mantendo um desempenho melhor ou igual à anterior.

8. **Armazenamento de dados estruturados, semi-estruturados e não estruturados**
   Permite uma fácil aplicação da escalabilidade, assim como um aumento na escalabilidade de dados. De notar que, na ausência de dados estruturados, não há garantia de existir integridade de dados.

8.2. SQL vs NoSQL

	SQL	NoSQL
Armazenamento de Dados	O armazenamento é feito num modelo relacional, em tabelas, em que as linhas são as entidades e as colunas os atributos das respetivas entidades	Abrange várias bases de dados, cada uma com a sua estrutura de armazenamento de dados
Estrutura e Flexibilidade	Alterações que sejam feitas ao nível estrutural implicam alterar toda a base de dados	Possui estruturas dinâmicas, ou seja, informações podem ser adicionadas facilmente
Escalabilidade	A escalabilidade é vertical. Uma maior quantidade de dados implica um servidor maior, o que é dispendioso	A escalabilidade é horizontal, tornando-se menos dispendioso
Propriedades ACID	A grande maioria das bases de dados são compatíveis com ACID	Muitas bases de dados NoSQL sacrificam a sua compatibilidade ACID para fornecer um maior desempenho e escalabilidade

8.3. MongoDB

Tal como já foi referido, uma das formas de armazenar dados é fazê-lo através de documentos, onde cada registo e os seus dados são considerados documentos. Na elaboração deste trabalho iremos utilizar a base de dados MongoDB, sendo, por este motivo, importante explorarmos as suas características.

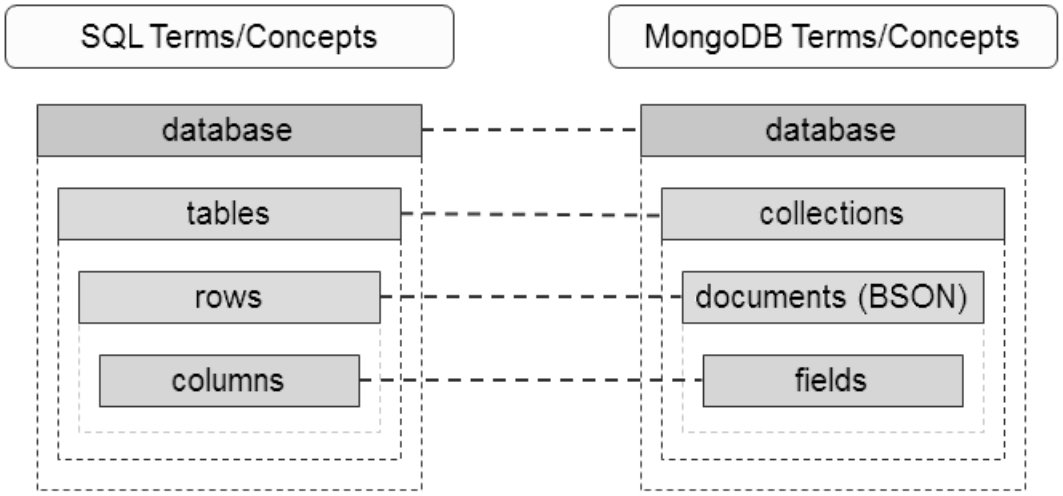

Figure 35: SQL vs NoSQL

O MongoDB, como já foi dito, armazena dados em documentos flexíveis, como é o caso de documentos JSON, o que significa que os campos podem variar de documento para documento e a estrutura de dados pode ser alterada ao longo do tempo.

- **Vantagens**

→ Utilizando MongoDB conseguimos atingir uma melhor performance, visto que uma única consulta retorna tudo o que precisamos de saber sobre o documento;

→ Simplifica bastante as consultas, uma vez que não existem transações nem "joins", ou seja, as consultas são mais fáceis de escrever e de ajustar;

→ Escalonamento horizontal com Sharding muito bem implementado. Sharding é utilizado quando temos muitos dados e estamos no limite do disco. Dividimos esses dados por várias máquinas, tendo assim mais rendimento e maior capacidade de armazenamento em disco;

→ Como a linguagem de consulta SQL é fácil de ser convertida para MongoDB temos uma maior facilidade para migração de organizações que atualmente usam bases de dados relacionais;

→ Possui uma funcionalidade chamada GridFS que é responsável por armazenar arquivos de grandes dimensões. Isso significa que vídeos, arquivos, etc, podem ser armazenados diretamente na base de dados.

- **Desvantagens**

→ quando queremos alterar todos os registros relacionados a uma unidade semântica, é preciso tratar um a um;

→ Não existem relacionamentos entre os documentos, o que leva a uma grande repetição de dados;

→ Não garante a consistência dos dados, o que significa que cabe ao programador da aplicação garantir que todas as inserções ou alterações nos dados os mantêm consistentes.

8.4. Utilização de um sistema NoSQL na AppGo

A Maria, tendo em conta o seu orçamento limitado, começou a reparar num crescimento insustentável da sua base de dados, o que levou a uma velocidade muito reduzida e começou a afetar o funcionamento do seu negócio. Assim, a Maria optou por procurar outras soluções que salvassem o seu investimento inicial e decidiu fazer a migração da sua base de dados para um sistema NoSQL, mais propriamente, MongoDB. Logo após esta mudança, a Maria notou um grande aumento de performance na sua aplicação.

Apesar de perceber que os Sistemas de Gestão de Bases de Dados Relacionais serem bastante mais utilizados do que os Sistemas de Gestão de Bases de Dados Não Relacionais devido às propriedades ACID[1], esta opção revelou-se muito mais rentável. Conseguimos então perceber que, por este motivo, estes novos SGBD vieram melhorar a disponibilidade, escalabilidade e performance dos mesmos.

8.5. Processo de migração

Sabendo que já existia uma SGBD Relacional previamente feita, começamos por efetuar a migração de dados, tornando possível uma simulação mais realista do caso de estudo. Inicialmente, este processo de migração foi feito utilizando a ferramenta *mongify,* no entanto, deparamo-nos com diversos problemas, tornando o resultado final diferente do que era o nosso objetivo. Com este entrave, foi decidido que seria viável a realização de um script utilizando a linguagem de programação JAVA, uma vez que era a linguagem com que nos encontrávamos mais familiarizados, tendo sido esta mesma utilizada no ano anterior em Programação Orientada a Objeto e, neste mesmo ano, na Unidade Curricular de Desenvolvimento de Sistemas de

[1] Atomicidade, Consistência, Isolamento e Durabilidade

Software. Este script permitiu-nos realizar todo o processo de migração sem nunca haver a necessidade de desligar os servidores e fazendo com que a informação ficasse organizada como objetivamos.

É importante referir que a principal característica do esquema anterior para MongoDB é a remoção dos relacionamentos entre tabelas. Sendo assim, é lógico que não seria possível não ter uma collection para guardar informações sobre os produtos, uma vez que são eles a base do nosso sistema, ou seja, toda a nossa aplicação é baseada nestes. Isto acontece porque todos os dias nos é possível observar novos produtos que publicadores inserem, levando a uma quantidade de inserções enorme. O mesmo acontece com a *collection* Publicador, já que são estes que fazem as inserções dos produtos e com a *collection* Utilizador porque, como podemos perceber, são estes que realizam transferências da aplicação e fazem uso dos produtos.

Com a identificação das *collections* necessárias para este trabalho, podemos agora passar para a ilustração visual e mais pormenorizada das mesmas, assim como da estrutura dos documentos.

8.6. Esquema das coleções

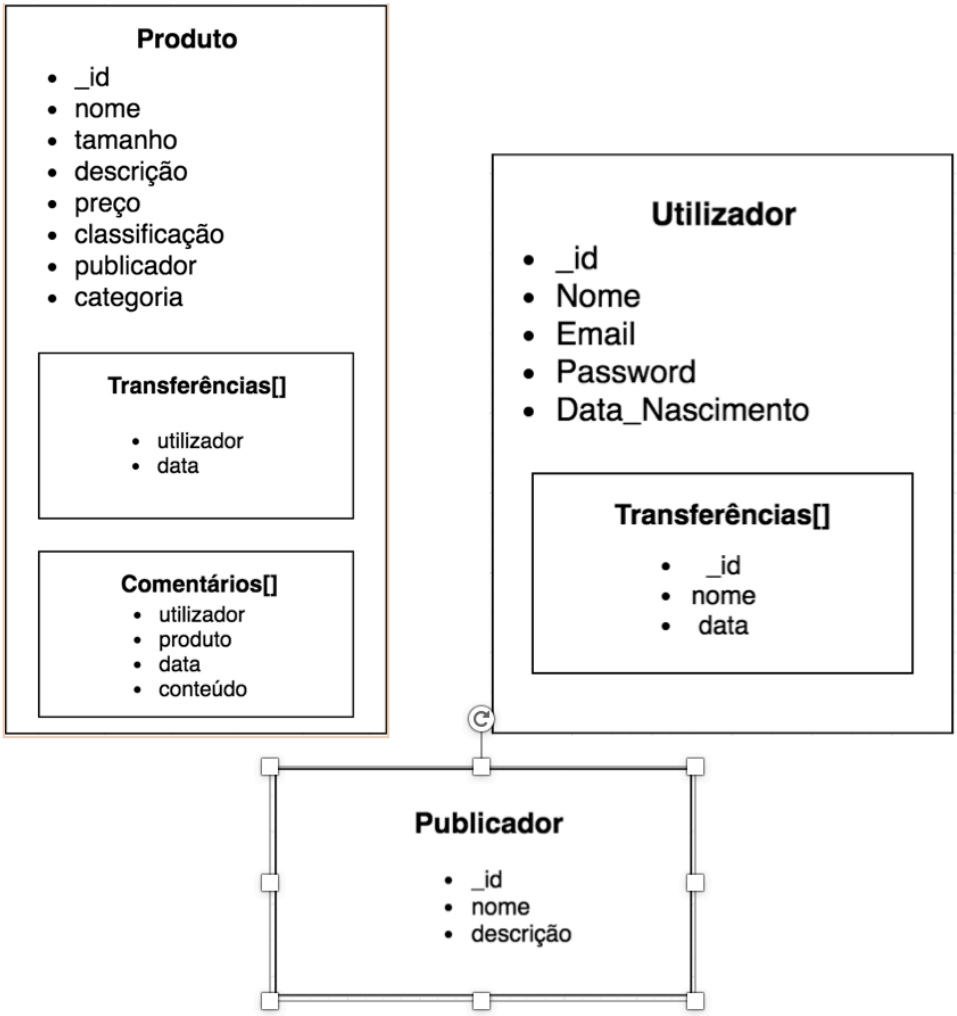

Figure 36: Esquema das coleções

8.7. Estrutura dos Documentos

- **Utilizador:**

{

"_id": ID do utilizador

"Nome": Nome do utilizador

"Email": Email do utilizador

"Password": Password do utilizador

"Data_Nascimento": Data de nascimento do utilizador

"Localidade": Distrito a que pertence a rua onde o utilizador habita

"Rua": Rua da morada do utilizador

"Transferências": Lista que contém os nomes dos produtos transferidos pelo utilizador

}

- **Produto:**

{

"_id": ID do produto

"Nome": Nome do produto

"Tamanho": Espaço ocupado pelo produto

"Descrição": Descrição do produto

"Preço": Preço do produto (0 se for gratuito)

"Classificação": Classificação do produto numa escala de 0 a 5

"Publicador": Nome do publicador deste produto

"Categoria": Categoria a que pertence o produto (Aplicação, Jogo, Música, Filme, Livro)

"Comentários": Lista que contém os comentários feitos pelos utilizadores a este produto

"Transferências": Lista que contém as transferências feitas deste produto

}

- **Publicador:**

{

"_id": ID do publicador

"Nome": Nome do publicador

"Descrição": Descrição do publicador

}

8.8. Queries

⇒ Saber qual o produto com melhor classificação

```
1.  > db.Produto.find().sort({Classificação:-1}).limit(1)
```

⇒ Saber a designação da categoria de um produto

```
1.  > db.Produto.find({Categoria: "Aplicação"}).pretty()
```

⇒ Saber quais são os produtos gratuitos

```
1.  > db.Produto.find({Preço: 0}).pretty()
```

⇒ Saber qual o produto, de um dado publicador, com melhor classificação

```
1.  > db.Produto.find({Publicador: "Donut Games"}).sort({Classificação: -
    1}).limit(1)
```

⇒ Saber quais são os produtos transferidos por um utilizador

```
1.  > db.Utilizador.find({_id:1}, {transferencias:1}).pretty()
```

⇒ Saber que comentários foram feitos a um determinado produto que lhe pertence

```
1.  > db.Produto.find({_id:1}, {comentarios:1}).pretty()
```

8.9. Conclusão

Tal como já foi identificado anteriormente, as Bases de Dados Não Relacionais têm ganho, cada vez mais, uma importância acrescida devido ao massivo aumento de dados com que lidamos na atualidade, sendo que estes são importantes guardar. Assim, é comum que soluções como o MongoDB ganhem popularidade, permitindo aos seus utilizadores adquirirem gradualmente uma maior confiança nestes sistemas.

Sendo assim, é possível afirmarmos que a implementação deste sistema se revelou uma mais valia para o nosso trabalho.

Após uma análise crítica de todo o nosso trabalho, ou seja, tanto quanto ao Modelo Relacional como o Modelo Não Relacional, concluímos que ambos têm as suas vantagens e desvantagens, como já era de prever. A mais importante de referir, pensamos que seja o facto da consistência de dados. Ora, quanto à base de dados relacional, esta característica é assegurada, algo que é bastante importante numa base de dados em que os publicadores podem adicionar e modificar informações ou, neste caso, produtos. Esta característica já não é observada no modelo não relacional. Então, observamos aqui um dilema: optando por um modelo relacional conseguimos reduzir o problema da redundância de dados observado no modelo não relacional, no entanto, com a enorme quantidade de dados a que a nossa aplicação está sujeita, como já foi referido inúmeras vezes neste mesmo relatório, um modelo não relacional torna-se muito mais viável. Isto deve-se à fraca escalabilidade verificada no modelo relacional.

Com o intuito de respeitar as propriedades ACID já enunciadas, um SGBD Relacional, não consegue processar grandes quantidades de dados, uma vez que possui um limite de memória do CPU. Uma vez que, para a AppGo temos expectativas altas e contamos ter uma grande aderência por parte dos utilizadores e, consequentemente, dos publicadores, um SGBD Não Relacional pareceu-nos o mais adequado. Optando por este sistema, garantimos que todas as características essenciais que procuramos alcançar são satisfeitas, podendo a nossa aplicação crescer indefinidamente através de sistemas distribuídos e da organização em documentos. Mesmo assim, com esta hipótese, temos a desvantagem já identificada da redundância de dados. A fim de ser diminuída, esta desvantagem só depende do grupo que realiza a sua estruturação, sendo importante manter em consideração todas as inserções feitas anteriormente.

9. Bibliografia NoSQL

NoSQL, BASE vs ACID, Teorema CAP, 2013. [online] Disponível em: https://pt.slideshare.net/Celio12/nosql-base-vs-acid-e-teorema-cap

MongoDB, 2015. NoSQL Databases Explained [online] Disponível em: https://www.mongodb.com/nosql-explained

MongoDB, 2015. What is a Mongo DB? [online] Disponível em: https://www.mongodb.com/what-is-mongodb

Lista de Siglas e Acrónimos

BD	Base de Dados
1FN	*Primeira Forma Normal*
2FN	*Segunda Forma Normal*
3FN	*Terceira Forma Normal*
PK	*Primary Key*
FK	Foreign Key
U	Utilizador
P	Produtor
C	Comentário
T	Transferência
PUB	Publicador
CAT	Categoria
M	Morada
SGBD	Sistema de Gestão de Bases de Dados

Anexos

I. Inquérito 1, disponível em: https://www.survio.com/survey/d/L3A0G1L3C2X7U1C6V?preview=1

II. Inquérito 2, disponível em: https://www.survio.com/survey/d/E7O9T4N8Q6C0M3K8W?preview=1

III. Script Modelo Físico

```
1.  -- MySQL Workbench Forward Engineering
2.
3.  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4.  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
5.  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES
    ,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTIO
    N';
6.
7.  -- -----------------------------------------------------
8.  -- Schema trabalho
9.  -- -----------------------------------------------------
10.
11. -- -----------------------------------------------------
12. -- Schema trabalho
13. -- -----------------------------------------------------
14. CREATE SCHEMA IF NOT EXISTS `trabalho` DEFAULT CHARACTER SET utf8 ;
15. USE `trabalho` ;
16.
17.
18. -- -----------------------------------------------------
19. -- Table `trabalho`.`Publicador`
20. -- -----------------------------------------------------
21. CREATE TABLE IF NOT EXISTS `trabalho`.`Publicador` (
22.   `idPublicador` INT UNSIGNED NOT NULL,
23.   `Nome` VARCHAR(45) NOT NULL,
24.   `Descrição` VARCHAR(100) NOT NULL,
25.   PRIMARY KEY (`idPublicador`))
26. ENGINE = InnoDB;
27.
28.
29. -- -----------------------------------------------------
30. -- Table `trabalho`.`Categoria`
31. -- -----------------------------------------------------
32. CREATE TABLE IF NOT EXISTS `trabalho`.`Categoria` (
33.   `idCategoria` INT UNSIGNED NOT NULL,
34.   `Designação` VARCHAR(45) NOT NULL,
35.   PRIMARY KEY (`idCategoria`))
36. ENGINE = InnoDB;
37.
38.
39. -- -----------------------------------------------------
40. -- Table `trabalho`.`Produto`
41. -- -----------------------------------------------------
42. CREATE TABLE IF NOT EXISTS `trabalho`.`Produto` (
43.   `idProduto` INT UNSIGNED NOT NULL,
44.   `Nome` VARCHAR(45) NOT NULL,
45.   `Tamanho` INT UNSIGNED NOT NULL,
46.   `Descrição` VARCHAR(45) NOT NULL,
47.   `Preço` INT UNSIGNED NOT NULL,
48.   `Classificação` INT UNSIGNED NOT NULL,
49.   `Publicador_idPublicador` INT UNSIGNED NOT NULL,
50.   `Categoria_idCategoria` INT UNSIGNED NOT NULL,
51.   PRIMARY KEY (`idProduto`),
52.   INDEX `fk_Produto_Publicador1_idx` (`Publicador_idPublicador` ASC) VISIBLE,
53.   INDEX `fk_Produto_Categoria1_idx` (`Categoria_idCategoria` ASC) VISIBLE,
54.   CONSTRAINT `fk_Produto_Publicador1`
55.     FOREIGN KEY (`Publicador_idPublicador`)
56.     REFERENCES `trabalho`.`Publicador` (`idPublicador`)
57.     ON DELETE NO ACTION
58.     ON UPDATE NO ACTION,
59.   CONSTRAINT `fk_Produto_Categoria1`
60.     FOREIGN KEY (`Categoria_idCategoria`)
```

```
    REFERENCES `trabalho`.`Categoria` (`idCategoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `trabalho`.`Morada`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `trabalho`.`Morada` (
  `idMorada` INT UNSIGNED NOT NULL,
  `Rua` VARCHAR(100) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMorada`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `trabalho`.`Utilizador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `trabalho`.`Utilizador` (
  `idUtilizador` INT UNSIGNED NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Data_Nascimento` DATE NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Morada_idMorada` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`idUtilizador`),
  INDEX `fk_Utilizador_Morada1_idx` (`Morada_idMorada` ASC) VISIBLE,
  CONSTRAINT `fk_Utilizador_Morada1`
    FOREIGN KEY (`Morada_idMorada`)
    REFERENCES `trabalho`.`Morada` (`idMorada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `trabalho`.`Comentário`
-- -----------------------------------------------------
      CREATE TABLE IF NOT EXISTS `trabalho`.`Comentário` (
        `Utilizador_idUtilizador` INT UNSIGNED NOT NULL,
        `Produto_idProduto` INT UNSIGNED NOT NULL,
        `Conteúdo` VARCHAR(200) NOT NULL,
        `Data` DATE NOT NULL,
        PRIMARY KEY (`Utilizador_idUtilizador`, `Produto_idProduto`),
        INDEX `fk_Utilizador_has_Produto_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
        INDEX `fk_Utilizador_has_Produto_Utilizador1_idx` (`Utilizador_idUtilizador` ASC) VISIBLE,
        CONSTRAINT `fk_Utilizador_has_Produto_Utilizador1`
          FOREIGN KEY (`Utilizador_idUtilizador`)
          REFERENCES `trabalho`.`Utilizador` (`idUtilizador`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION,
        CONSTRAINT `fk_Utilizador_has_Produto_Produto1`
          FOREIGN KEY (`Produto_idProduto`)
          REFERENCES `trabalho`.`Produto` (`idProduto`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION)
      ENGINE = InnoDB;


      -- -----------------------------------------------------
      -- Table `trabalho`.`Transferência`
      -- -----------------------------------------------------
```

```
125.       CREATE TABLE IF NOT EXISTS `trabalho`.`Transferência` (
126.         `Utilizador_idUtilizador` INT UNSIGNED NOT NULL,
127.         `Produto_idProduto` INT UNSIGNED NOT NULL,
128.         `Data` DATE NOT NULL,
129.         PRIMARY KEY (`Utilizador_idUtilizador`, `Produto_idProduto`),
130.         INDEX `fk_Utilizador_has_Produto1_Produto1_idx` (`Produto_idProduto`
    ASC) VISIBLE,
131.         INDEX `fk_Utilizador_has_Produto1_Utilizador1_idx` (`Utilizador_idUti
    lizador` ASC) VISIBLE,
132.         CONSTRAINT `fk_Utilizador_has_Produto1_Utilizador1`
133.           FOREIGN KEY (`Utilizador_idUtilizador`)
134.           REFERENCES `trabalho`.`Utilizador` (`idUtilizador`)
135.           ON DELETE NO ACTION
136.           ON UPDATE NO ACTION,
137.         CONSTRAINT `fk_Utilizador_has_Produto1_Produto1`
138.           FOREIGN KEY (`Produto_idProduto`)
139.           REFERENCES `trabalho`.`Produto` (`idProduto`)
140.           ON DELETE NO ACTION
141.           ON UPDATE NO ACTION)
142.       ENGINE = InnoDB;
143.
144.
145.       SET SQL_MODE=@OLD_SQL_MODE;
146.       SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
147.       SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

IV. Script do Povoamento

```
-- Povoamento Categoria

INSERT INTO Categoria
    (idCategoria , Designação)
    VALUES
        (1,'Jogo'),
        (2,'Aplicação'),
        (3,'Música'),
        (4,'Livro'),
        (5,'Filme');

-- Povoamento Morada

INSERT INTO MORADA
    (idMorada, Rua, Localidade)
    VALUES
        (1,'Rua do raio','Braga'),
        (2,'Rua dos chãos','Braga'),
        (3,'Rua da invicta','Porto'),
        (4,'Rua de cima','Porto'),
        (5,'Rua do lado','Évora'),
        (6,'Rua da direita','Lisboa'),
        (7,'Rua da esquerda','Lisboa'),
        (8,'Rua superior','Lisboa'),
        (9,'Rua inferior','Leiria'),
        (10,'Rua interior','Coimbra'),
        (11,'Rua exterior','Faro'),
        (12,'Rua vasta','Vila Real'),
        (13,'Rua da alegria','Aveiro'),
        (14,'Rua da boa sorte','Aveiro'),
        (15,'Rua da juventude','Setúbal'),
        (16,'Rua das casas','Setúbal'),
        (17,'Bairro do povo','Braga');

-- Povoamento Publicadores

INSERT INTO Publicador
    (idPublicador, Nome, Descrição)
    VALUES
        (1,'SuperCell','Empresa de jogos mega brutal'),
        (2,'Nitrome','Empresa de jogos arcade'),
        (3,'Logisk','Empresa de jogos puzzle'),
        (4,'Ketchapp','Empresa de jogos'),
        (5,'Donut Games','Empresa de jogos arcade'),
        (6,'InShot','Empresa de aplicações de edição de imagem'),
        (7,'TripAdvisor','Empresa de aplicações direcionadas a turismo'),
        (8,'Maple Media','Empresa de aplicações de gestão de armazenamento'),

        (9,'Shanga','Empresa de aplicações de wallpapers'),
        (10,'Google LLC','Empresa de várias aplicações para uso no diário'),
        (11,'Muse','Banda inglesa de rock'),
        (12,'Queen','Banda de rock britânica'),
        (13,'Coldplay','Banda britânica de rock alternativo'),
        (14,'Arcade Fire','Banda de indie rock'),
        (15,'Linkin Park','Banda de rock dos Estados Unidos'),
        (16,'Marvel','Estúdio de filmes de super heróis'),
        (17,'DC','Estúdio de filmes de super heróis'),
        (18,'Disney','Estúdio de Filmes de animação'),
        (19,'Pixar','Estúdio de filmes de animação'),
        (20,'Lucasfilm','Estúdio de filmes de ficção científica'),
        (21,'J.K. Rowling','Escritora de livros de fantasia'),
        (22,'Christopher Paolini','Escritor de livros de fantasia');
```

```
62.
63. -- Povoamento Produto
64.
65. INSERT INTO Produto
66.     (idProduto, Nome, Tamanho , Categoria_idCategoria,Descrição , Preço, Class
    ificação, Publicador_idPublicador)
67.     VALUES
68.         (1,'Clash Royale',500,1,'Jogo de Estratégia online mega brutal', 0, 5,
     1),
69.         (2,'Clash of clans',200,1,'Jogo de Estratégia online',0,4,1),
70.         (3,'Hay Day',160,1,'Jogo de evoluir a sua fazenda',1,3,1),
71.         (4,'Boom Beach',250,1,'Jogo de Estratégia online',0,4,1),
72.         (5,'Redundeon',200,1,'Jogo de aventura',1,5,2),
73.         (6,'Icebreaker',160,1,'Jogo puzzle',2,4,2),
74.         (7,'Tower Fortress',210,1,'Jogo de estratégia',0,4,2),
75.         (8,'Leap Day',110,1,'Jogo de ação',2,3,2),
76.         (9,'Hexio',120,1,'Jogo puzzle',2,4,3),
77.         (10,'Horizon',200,1,'Jogo arcade',0,4,4),
78.         (11,'Knife Hit',180,1,'Jogo arcade',0,5,4),
79.         (12,'Rider',220,1,'Jogo arcade',0,5,4),
80.         (13,'Mr Gun',120,1,'Jogo arcade',0,4,4),
81.         (14,'Stairs',110,1,'Jogo arcade',3,4,4),
82.         (15,'Knife Hit',180,1,'Jogo arcade',0,5,4),
83.         (16,'Micro Battles',120,1,'Jogo arcade multiplayer',1,4,5),
84.         (17,'Trafic Rush',90,1,'Jogo arcade multiplayer',0,4,5),
85.         (18,'Rat on the scooter',130,1,'Jogo arcade multiplayer',1,5,5),
86.         (19,'Editor de vídeo',120,2,'Aplicação de edição de vídeo',0,5,6),
87.         (20,'Editor de fotos',70,2,'Aplicação de edição de fotos',3,5,6),
88.         (21,'Colagem de fotos',80,2,'Aplicação de edição de vida',0,4,6),
89.         (22,'TripAdvisor',90,2,'Aplicação de review de restaurantes e hotéis',
    0,4,7),
90.         (23,'SeatGuru',70,2,'Aplicação para marcação',3,4,7),
91.         (24,'PropAppAlug',110,2,'Aplicação de aluguel',2,3,7),
92.         (25,'Gerenciador de arquivos',70,2,'Aplicação de gestão de armazenamen
    to',0,5,8),
93.         (26,'Wallpapers',90,2,'Aplicação de wallpapers',2,5,9),
94.         (27,'Google Tradutor',90,2,'Aplicação de tradução',0,5,10),
95.         (28,'Google Earth',150,2,'Aplicação de mapa Mundo',4,5,10),
96.         (29,'Google Notícias',70,2,'Aplicação de Notícias',0,4,10),
97.         (30,'Google',60,2,'Aplicação de motor de busca',0,5,10),
98.         (31,'Pressure',7,3,'Música dos Muse',1,4,11),
99.         (32,'Something Human',9,3,'Música dos Muse',1,4,11),
100.        (33,'Algorithm',8,3,'Música dos Muse',1,5,11),
101.        (34,'The Void',9,3,'Música dos Muse',2,4,11),
102.        (35,'Dollar',9,3,'Música dos Queen',2,5,12),
103.        (36,'Heart Quake',9,3,'Música dos Queen',2,5,12),
104.        (37,'Keep Yourself Alive',8,3,'Música dos Queen',2,5,12),
105.        (38,'Fix you',7,3,'Música dos ColdPlay',2,5,13),
106.        (39,'The Scientist',7,3,'Música dos ColdPlay',2,4,13),
107.        (40,'Yellow',7,3,'Música dos ColdPlay',2,4,13),
108.        (41,'Viva La vida',9,3,'Música dos ColdPlay',2,5,13),
109.        (42,'Rebellion',7,3,'Música dos Arcade Fire',1,4,14),
110.        (43,'Ready to Start',7,3,'Música dos Arcade Fire',1,5,14),
111.        (44,'Wake up',8,3,'Música dos Arcade Fire',1,4,14),
112.        (45,'Fix you',7,3,'Música dos Arcade Fire',1,5,14),
113.        (46,'Numb',8,3,'Música dos Linkin Park',1,5,15),
114.        (47,'In the end',10,3,'Música dos Linkin Park',1,5,15),
115.        (48,'Burn It Down',9,3,'Música dos Linkin Park',1,5,15),
116.        (49,'Iron Man',1024,5,'Filme de super-heróis',5,5,16),
117.        (50,'Iron Man 2',1024,5,'Filme de super-heróis',5,5,16),
118.        (51,'Avengers',1024,5,'Filme de super-heróis',8,5,16),
119.        (52,'Avengers 2',1024,5,'Filme de super-heróis',8,5,16),
120.        (53,'Suicide Squad',1024,5,'Filme de super-heróis',7,5,17),
121.        (54,'Batman',1024,5,'Filme de super-heróis',7,5,17),
122.        (55,'O Rei Leão',1024,5,'Filme de animação',9,5,18),
123.        (56,'A bela e o Monstro',1024,5,'Filme de animação',9,4,18),
```

```
        (57,'O Aladino',1024,5,'Filme de animação',9,5,18),
        (58,'Toy Story',1024,5,'Filme de animação',9,5,19),
        (59,'Monstros e Companhia',1024,5,'Filme de animação',5,5,19),
        (60,'Star Wars',1024,5,'Filme de ficção científica',5,5,20),
        (61,'Harry Potter e a pedra filosofal',200,4,'Livro de fantasia',5,5,2
    1),
        (62,'Harry Potter e a Camâra dos segredos',200,4,'Livro de fantasia',5
    ,5,21),
        (63,'Eragon',200,4,'Livro de fantasia',5,5,22),
        (64,'Brisingr',200,4,'Livro de fantasia',5,5,22);

    -- Povoamento Utilizadores

    INSERT INTO Utilizador
        (idUtilizador, Email, Nome, Data_Nascimento,Password,Morada_idMorad
    a)
        VALUES
            (1,'soaresgon@gmail.com','soaresgon', '1990-12-
    12','pass1',1),
            (2,'nmdo.p@gmail.com','nmdo2','1992-06-23','pass2',2),
            (3,'jose2000@hotmail.com','josef','1981-09-12','pass3',3),
            (4,'americo_33@gmail.com','mero95','1995-12-15','pass4',4),
            (5,'ines@gmail.com','ines','1993-03-22','pass5',5),
            (6,'dant@hotmail.com','danny','1993-01-08','pass6',5),
            (7,'peterex@hotmail.com','peterex1','1993-06-08','pass7',6),
            (8,'diosnes@hotmail.com','diosnes','1998-11-26','pass8',7),
            (9,'crocoftw@gmail.com','nmfo','1998-12-30','pass9',8),
            (10,'mail3@gmail.com','montras','1990-11-01','pass10',9),
            (11,'rybur4@hotmail.com','rybur','1999-02-08','pass11',10),
            (12,'ruca12@gmail.com','ruii','1999-05-01','pass12',10),
            (13,'filas3@gmail.com','lili','1994-11-28','pass13',11),
            (14,'jnuno@hotmail.com','jnuno','1992-10-10','pass14',12),
            (15,'vicas111@gmail.com','vicente18','1998-03-
    11','pass15',12),
            (16,'oliba33@gmail.com','liba321','1985-09-21','pass16',12),
            (17,'must13@gmail.com','dududu','1985-10-29','pass17',13),
            (18,'dot1@hotmail.com','qua1234','1999-10-15','pass18',14),
            (19,'rest1np@gmail.com','rip','2000-11-30','pass19',15),
            (20,'memes911@gmail.com','memergod','2000-10-
    23','pass20',16),
            (21,'hugo112@gmail.com','paihugo','1992-07-09','pass21',17);

    -- Povoamento Comentário

    INSERT INTO Comentário
         (Utilizador_idUtilizador,Produto_idProduto,Conteúdo,Data)
         VALUES
            (1,1,'Muito Bom','2016-10-11'),
            (2,1,'First','2015-12-08'),
            (3,2,'Recomendo Muito Bom !!!','2017-12-07'),
            (10,3,'O melhor','2017-01-02'),
            (10,4,'O meu jogo favorito','2018-06-04'),
            (10,5,'Recomendo','2017-10-10'),
            (10,6,'Muito viciante','2017-11-11'),
            (10,7,'Estou viciado','2018-02-01'),
            (4,8,'A ideia é boa mas tem muitos bugs','2017-01-30'),
            (10,9,'Muito bom para todo o mundo!!','2018-02-25'),
            (10,10,'Estou adorando','2017-03-03'),
            (10,11,'Completamente colado','2018-01-31'),
            (5,12,'Adorei','2016-02-27'),
            (10,23,'Maravilhoso','2017-12-12'),
            (10,24,'Maravilindo','2018-03-10'),
            (10,25,'Incrível','2017-04-24'),
            (10,21,'Lindo','2017-03-03'),
            (6,22,'Vai dar muito jeito','2016-03-20'),
            (7,26,'Pessimo','2018-04-06'),
```

```
                (8,26,'Horrivel','2018-05-09'),
                (10,27,'Meu favorito','2018-02-21'),
                (10,28,'Detestável','2017-10-10'),
                (10,29,'Muito mal optimizado','2016-12-23'),
                (10,30,'Não é de grande utilidade','2017-09-29'),
                (9,31,'Maravilha','2017-03-10'),
                (10,32,'Gostei muito','2017-11-29'),
                (10,33,'Lindo','2017-10-20'),
                (10,34,'Demais','2017-10-09'),
                (10,35,'Que mimo','2017-08-08'),
                (10,36,'Do melhor','2017-01-04'),
                (10,37,'Não gostei','2017-06-21'),
                (10,38,'Muito horrível','2017-03-20'),
                (10,39,'Mais ou menos bom','2017-09-10'),
                (10,40,'Uma porcaria','2016-11-28'),
                (10,41,'Das minhas favoritas','2017-01-07'),
                (10,42,'Surpreendeu pela positiva','2017-05-17'),
                (11,42,'Incrivel','2017-06-01'),
                (10,43,'Apaixonado','2017-07-02'),
                (10,44,'Linda','2017-08-29'),
                (12,51,'Muito útil','2016-08-02'),
                (13,52,'Adoro','2017-09-09'),
                (14,53,'Lindo','2018-03-20'),
                (15,54,'Meu filme favorito','2016-10-29'),
                (16,55,'Simplesmente lindo!!','2016-12-31'),
                (17,56,'Fantástico','2017-05-16'),
                (18,57,'Odiei','2017-06-20'),
                (19,58,'Mesmo bom','2018-01-01'),
                (20,59,'Destestei','2017-03-07'),
                (1,60,'Magnífico','2018-06-08'),
                (2,60,'Adorei','2017-07-23'),
                (3,13,'Uma porcaria','2016-11-30'),
                (4,14,'Cheio de bugs','2017-02-22'),
                (5,14,'Horrível','2018-01-23'),
                (6,15,'Amei','2017-12-02'),
                (7,16,'Muito bom para jogar com amigos','2017-09-13'),
                (8,17,'Estou viciado','2018-02-12'),
                (9,18,'Dá para estar','2017-07-13'),
                (10,19,'Muito útil','2017-12-21'),
                (11,20,'Muito bom e fácil de usar','2018-10-20'),
                (12,45,'Optimo','2017-12-15'),
                (13,46,'Meu favorito','2016-11-29'),
                (14,47,'Uma maravilha','2017-08-22'),
                (15,48,'Gosto muito!!!','2016-12-02'),
                (21,1,'100% o melhor jogo de todos','2018-04-05'),
                (21,13,'Muito divertido','2018-05-05');

        -- Povoamento Transferência

        INSERT INTO Transferência
             (Utilizador_idUtilizador,Produto_idProduto,Data)
             VALUES
             (1,1,'2017-11-01'),
             (1,40,'2017-11-10'),
             (2,7,'2017-11-20'),
             (3,13,'2017-11-30'),
             (4,20,'2017-01-01'),
             (5,32,'2017-03-01'),
             (6,33,'2017-04-01'),
             (7,18,'2017-05-01'),
             (8,18,'2016-06-02'),
             (8,55,'2016-07-02'),
             (9,64,'2016-10-02'),
             (10,61,'2016-10-02'),
             (11,60,'2016-10-05'),
             (12,60,'2016-10-20'),
```

```
250.            (12,2,'2016-10-22'),
251.            (12,1,'2016-10-12'),
252.            (13,25,'2016-05-14'),
253.            (14,26,'2016-06-16'),
254.            (15,27,'2016-08-02'),
255.            (16,31,'2016-09-20'),
256.            (16,32,'2018-02-27'),
257.            (16,38,'2018-01-20'),
258.            (17,39,'2018-12-19'),
259.            (17,8,'2018-12-17'),
260.            (18,46,'2018-10-12'),
261.            (18,47,'2018-05-02'),
262.            (19,47,'2018-05-22'),
263.            (19,50,'2018-08-13'),
264.            (20,50,'2018-11-11'),
265.            (20,51,'2018-03-19'),
266.            (20,61,'2018-01-18'),
267.            (21,1,'2017-02-13'),
268.            (21,54,'2016-11-21'),
269.            (21,13,'2016-12-13');
```

V. Script das Transações estabelecidas para SQL

```
1.         DELIMITER //
2.  CREATE PROCEDURE insere_utilizador (IN id INT, IN email VARCHAR(45), IN nome V
    ARCHAR(45), IN data DATE,
3.                                          IN password VARCHAR(45), IN morada INT)
4.      BEGIN
5.          DECLARE exit handler for SQLEXCEPTION
6.              BEGIN
7.                  SHOW ERRORS LIMIT 1;
8.                  SHOW WARNINGS;
9.                  ROLLBACK;
10.             END;
11.
12.         DECLARE exit handler for SQLWARNING
13.             BEGIN
14.                 SHOW ERRORS LIMIT 1;
15.                 SHOW WARNINGS;
16.                 ROLLBACK;
17.             END;
18.
19.         START TRANSACTION;
20.             INSERT INTO Utilizador
21.                 (idUtilizador, Email, Nome, Data_Nascimento,Password,Morada_id
    Morada)
22.                 VALUES
23.                 (id,email,nome,data,password,morada);
24.         COMMIT;
25.     END //
26.
27. CREATE PROCEDURE insere_publicador (IN id INT, IN nome VARCHAR(45), IN descriç
    ão VARCHAR(100))
28.     BEGIN
29.         DECLARE exit handler for SQLEXCEPTION
30.             BEGIN
31.                 SHOW ERRORS LIMIT 1;
32.                 SHOW WARNINGS;
33.                 ROLLBACK;
34.             END;
35.
36.         DECLARE exit handler for SQLWARNING
37.             BEGIN
38.                 SHOW ERRORS LIMIT 1;
39.                 SHOW WARNINGS;
40.                 ROLLBACK;
41.             END;
42.
43.         START TRANSACTION;
44.             INSERT INTO Publicador
45.                 (idPublicador, Nome, Descrição)
46.                 VALUES
47.                 (id,nome,descrição);
48.         COMMIT;
49.     END //
50.
51. CREATE PROCEDURE insere_produto (IN id INT, IN nome VARCHAR(45), IN tamanho IN
    T,IN descrição VARCHAR(100),
52.                                     IN preço INT, IN classificação INT, IN idpubl
    icador INT, IN idcategoria INT)
53.     BEGIN
54.         DECLARE exit handler for SQLEXCEPTION
55.             BEGIN
56.                 SHOW ERRORS LIMIT 1;
```

```
57.                 SHOW WARNINGS;
58.                 ROLLBACK;
59.             END;
60.
61.         DECLARE exit handler for SQLWARNING
62.             BEGIN
63.                 SHOW ERRORS LIMIT 1;
64.                 SHOW WARNINGS;
65.                 ROLLBACK;
66.             END;
67.
68.         START TRANSACTION;
69.             INSERT INTO Publicador
70.                 (idProduto, Nome, Tamanho , Descrição , Preço, Classificação,
    Publicador_idPublicador, Categoria_idCategoria)
71.                 VALUES
72.                 (id,nome,tamanho,descrição,preço,classificação,idpublicador,id
    categoria);
73.         COMMIT;
74.     END //
75.
76. CREATE PROCEDURE insere_transferencia (IN idutilizador INT, idproduto INT, IN
    data DATE)
77.     BEGIN
78.         DECLARE exit handler for SQLEXCEPTION
79.             BEGIN
80.                 SHOW ERRORS LIMIT 1;
81.                 SHOW WARNINGS;
82.                 ROLLBACK;
83.             END;
84.
85.         DECLARE exit handler for SQLWARNING
86.             BEGIN
87.                 SHOW ERRORS LIMIT 1;
88.                 SHOW WARNINGS;
89.                 ROLLBACK;
90.             END;
91.
92.         START TRANSACTION;
93.             INSERT INTO Publicador
94.                 (Utilizador_idUtilizador,Produto_idProduto,Data)
95.                 VALUES
96.                 (idutilizador,idproduto,data);
97.         COMMIT;
98.     END //
99.
100.        CREATE PROCEDURE insere_comentario (IN idutilizador INT, idproduto INT,
     IN data DATE, IN conteudo VARCHAR(200))
101.            BEGIN
102.                DECLARE exit handler for SQLEXCEPTION
103.                    BEGIN
104.                        SHOW ERRORS LIMIT 1;
105.                        SHOW WARNINGS;
106.                        ROLLBACK;
107.                    END;
108.
109.                DECLARE exit handler for SQLWARNING
110.                    BEGIN
111.                        SHOW ERRORS LIMIT 1;
112.                        SHOW WARNINGS;
113.                        ROLLBACK;
114.                    END;
115.
116.                START TRANSACTION;
117.                    INSERT INTO Comentário
```

```
                        (Utilizador_idUtilizador,Produto_idProduto,Conteúdo,Dat
    a)
                        VALUES
                        (idutilizador,idproduto,data,conteudo);
                COMMIT;
            END //

        CREATE PROCEDURE insere_categoria (IN id INT, IN designação VARCHAR(45)
    )
            BEGIN
                DECLARE exit handler for SQLEXCEPTION
                    BEGIN
                        SHOW ERRORS LIMIT 1;
                        SHOW WARNINGS;
                        ROLLBACK;
                    END;

                DECLARE exit handler for SQLWARNING
                    BEGIN
                        SHOW ERRORS LIMIT 1;
                        SHOW WARNINGS;
                        ROLLBACK;
                    END;

                START TRANSACTION;
                    INSERT INTO Categoria
                        (idCategoria , Designação)
                        VALUES
                        (id,designação);
                COMMIT;
            END //

        CREATE PROCEDURE insere_morada (IN id INT,IN rua VARCHAR(100), IN local
    idade VARCHAR(45))
            BEGIN
                DECLARE exit handler for SQLEXCEPTION
                    BEGIN
                        SHOW ERRORS LIMIT 1;
                        SHOW WARNINGS;
                        ROLLBACK;
                    END;

                DECLARE exit handler for SQLWARNING
                    BEGIN
                        SHOW ERRORS LIMIT 1;
                        SHOW WARNINGS;
                        ROLLBACK;
                    END;

                START TRANSACTION;
                    INSERT INTO Morada
                        (idMorada, Rua, Localidade)
                        VALUES
                        (id,rua,localidade);
                COMMIT;
            END //

        DELIMITER //
```

VI. Script das Queries

```
1.
    -- Do ponto de vista do Utilizador:
2.      -- 1. Saber qual é o publicador de um certo produto;
3.  DELIMITER //
4.  CREATE PROCEDURE obterpublicador (IN idProduto INT)
5.      BEGIN
6.          SELECT Publicador.*, Produto.idProduto, Produto.Nome FROM Produto
7.          INNER JOIN Publicador on Publicador.idPublicador = Produto.Publicador_
    idPublicador
8.          WHERE Produto.idProduto = idProduto;
9.      END //
10. DELIMITER //
11.
12. SET @idProduto = 3;
13. CALL obterpublicador (@idProduto);
14. DROP PROCEDURE obterpublicador;
15.
16.     -- 2. Saber a designação da categoria de um produto;
17. DELIMITER //
18. CREATE PROCEDURE obterdesignação (IN idProduto INT)
19.     BEGIN
20.         SELECT Produto.Nome, Categoria.Designação From Produto
21.         INNER JOIN Categoria on Categoria.idCategoria = Produto.Categoria_idCa
    tegoria
22.         WHERE Produto.idProduto = idProduto;
23.     END //
24. DELIMITER //
25.
26. CALL obterdesignação (@idProduto);
27. DROP PROCEDURE obterdesignação;
28.
29.     -- 3. Saber qual o produto com melhor classificação;
30. DELIMITER //
31. CREATE PROCEDURE obterproduto_maior_classificação ()
32.     BEGIN
33.         SELECT Produto.Nome, Produto.Classificação from Produto
34.         ORDER BY Classificação DESC
35.         LIMIT 1;
36.     END //
37. DELIMITER //
38.
39. CALL obterproduto_maior_classificação();
40. DROP PROCEDURE obterproduto_maior_classificação;
41.
42.     -- 4. Saber quais são os produtos gratuitos;
43. DELIMITER //
44. CREATE PROCEDURE obterprodutos_gratuitos ()
45.     BEGIN
46.         SELECT * FROM Produto
47.         WHERE Preço = 0;
48.     END //
49. DELIMITER //
50.
51. CALL obterprodutos_gratuitos();
52. DROP PROCEDURE obterprodutos_gratuitos;
53.
54.     -- 5. Saber quais são os produtos transferidos por um utilizador;
55. DELIMITER //
56. CREATE PROCEDURE obterprodutos_utilizador (IN idUtilizador INT)
57.     BEGIN
58.         SELECT Produto.Nome from Utilizador
59.         INNER JOIN Transferência ON Utilizador.idUtilizador = Transferência.Ut
    ilizador_idUtilizador
```

```
60.         INNER JOIN Produto ON Produto.idProduto = Transferência.Produto_idProd
    uto
61.         WHERE Utilizador.idUtilizador = idUtilizador;
62.     END //
63. DELIMITER //
64.
65. SET @idUtilizador = 5;
66. CALL obterprodutos_utilizador(@idUtilizador);
67. DROP PROCEDURE obterprodutos_utilizador;
68.
69.     -
    - 6. Saber o comentário mais recente, do último produto transferido por um uti
    lizador;
70. DELIMITER //
71. CREATE PROCEDURE obtercomentário_maisrecente_produto_maisrecente (IN idUtiliza
    dor INT)
72.     BEGIN
73.         SELECT Comentário.Conteúdo, Produto.Nome, Transferência.Data, Comentár
    io.Data from Comentário
74.         INNER JOIN Utilizador ON Utilizador.idUtilizador = Comentário.Utilizad
    or_idUtilizador
75.         INNER JOIN Produto ON Produto.idProduto = Comentário.Produto_idProduto

76.         INNER JOIN Transferência ON Transferência.Utilizador_idUtilizador = Ut
    ilizador.idUtilizador
77.         WHERE Utilizador.idUtilizador = idUtilizador
78.         ORDER BY Transferência.Data DESC,
79.                  Comentário.Data DESC
80.         LIMIT 1;
81.     END //
82. DELIMITER //
83.
84. CALL obtercomentário_maisrecente_produto_maisrecente(@idUtilizador);
85. DROP PROCEDURE obtercomentário_maisrecente_produto_maisrecente;
86.
87. -- Do ponto de vista do Publicador:
88.     -
    - 1. Saber que utilizadores transferiram um produto que lhe pertence e em que
    data foi feita essa transferência;
89. DELIMITER //
90. CREATE PROCEDURE obterproduto_utilizadores (IN idProduto INT)
91.     BEGIN
92.         SELECT Produto.Nome, Utilizador.Nome, Publicador.Nome, Transferência.D
    ata FROM Publicador
93.         INNER JOIN Produto ON Produto.Publicador_idPublicador = Publicador.idP
    ublicador
94.         INNER JOIN Transferência ON Transferência.Produto_idProduto = Produto.
    idProduto
95.         INNER JOIN Utilizador ON Transferência.Utilizador_idUtilizador = Utili
    zador.idUtilizador
96.         WHERE Produto.idProduto = idProduto;
97.     END //
98. DELIMITER //
99.
100.        SET @idProduto = 12;
101.        CALL obterproduto_utilizadores (@idProduto);
102.        DROP PROCEDURE obterproduto_utilizadores;
103.
104.            -
    - 2. Saber que comentários foram feitos a um determinado produto que lhe perte
    nce;
105.        DELIMITER //
106.        CREATE PROCEDURE obterproduto_comentários (IN idProduto INT)
107.            BEGIN
108.                SELECT Produto.nome, Publicador.Nome, Comentário.Conteúdo, Util
    izador.Nome FROM Publicador
```

```
109.                INNER JOIN Produto on Produto.Publicador_idPublicador = Publica
    dor.idPublicador
110.                INNER JOIN Comentário ON Comentário.Produto_idProduto = Produto
    .idProduto
111.                INNER JOIN Utilizador ON Comentário.Utilizador_idUtilizador = U
    tilizador.idUtilizador
112.                WHERE Produto.idProduto = idProduto;
113.            END //
114.        DELIMITER //
115.
116.        CALL obterproduto_comentários(@idProduto);
117.        DROP PROCEDURE obterproduto_comentários;
118.
119.            -- 3. Saber qual o seu produto com melhor classificação;
120.        DELIMITER //
121.        CREATE PROCEDURE obterproduto_maiorclassificação (IN idPublicador INT)

122.            BEGIN
123.                SELECT Produto.Nome, Produto.Classificação, Publicador.Nome fro
    m Produto
124.                INNER JOIN Publicador ON Publicador.idPublicador = Produto.Publ
    icador_idPublicador
125.                WHERE Publicador.idPublicador = idPublicador
126.                ORDER BY Produto.Classificação DESC
127.                LIMIT 1;
128.            END //
129.        DELIMITER //
130.
131.        SET @idPublicador = 5;
132.        CALL obterproduto_maiorclassificação(@idPublicador);
133.        DROP PROCEDURE obterproduto_maiorclassificação;
```

VII. Script dos Mecanismos de Segurança

```
1.  -- Administrador
2.  CREATE USER 'administrador'@'localhost' IDENTIFIED BY 'administrador';
3.  GRANT ALL ON trabalho.* TO 'administrador'@'localhost';
4.
5.  -- Utilizador
6.  CREATE USER 'utilizador'@'localhost' IDENTIFIED BY 'utilizador';
7.  GRANT INSERT, UPDATE ON trabalho.Utilizador to 'utilizador'@'localhost';
8.  GRANT SELECT ON trabalho.Produto to 'utilizador'@'localhost';
9.  GRANT INSERT ON trabalho.Comentario to 'utilizador'@'localhost';
10. GRANT INSERT ON trabalho.Transferência to 'utilizador'@'localhost';
11. GRANT INSERT ON trabalho.Morada to 'utilizador'@'localhost';
12.
13. -- Publicador
14. CREATE USER 'publicador'@'localhost' IDENTIFIED BY 'publicador';
15. GRANT SELECT, INSERT, UPDATE ON trabalho.Publicador to 'publicador'@'localhost
    ';
16. GRANT SELECT, INSERT, UPDATE ON trabalho.Produto to 'publicador'@'localhost';
```

VIII. Script das Views

```
CREATE VIEW Top10Produtos AS
    SELECT Produto.Nome, Produto.Classificação  from Produto
        ORDER BY Produto.CLassificação DESC
    LIMIT 10;

CREATE VIEW mediaIdades AS
    Select AVG(TIMESTAMPDIFF(YEAR, Data_Nascimento, CURDATE())) from Utilizado
r;

CREATE VIEW avgSize AS
    SELECT AVG(Tamanho) FROM Produto;
```

IX. Script da Migração

```
package com.example.java;
import com.mongodb.BasicDBObject;
import com.mongodb.BulkWriteOperation;
import com.mongodb.BulkWriteResult;
import com.mongodb.Cursor;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import com.mongodb.MongoClient;
import com.mongodb.ParallelScanOptions;
import com.mongodb.ServerAddress;

import java.lang.reflect.Array;
import java.sql.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Set;

import static java.util.concurrent.TimeUnit.SECONDS;


class Publicador {
    int idPublicador;
    String nome;
    String descricao;
    List<Produto> produtos = new ArrayList();

}
class Comentario {
    int idUtilizador, idProduto;
    String conteudo;
    Date data;
}
class Transferencia {
    int idProduto;
    String designacao;
    Date data;
}

class TransferenciaProd {
```

```
46.     int idUtilizador;
47.     Date data;
48. }
49. 
50. class Produto {
51.     int idProduto;
52.     String nome;
53.     int tamanho;
54.     String descricao;
55.     int preco;
56.     int classificacao;
57.     int publicador;
58.     int categoria;
59.     List<Comentario> comentarios = new ArrayList();
60.     List<TransferenciaProd> transfs = new ArrayList();
61. 
62. }
63. class Categoria {
64.     int idCategoria;
65.     String designacao;
66. 
67. }
68. 
69. class Utilizador {
70.     int idUtilizador;
71.     String email, nome, password;
72.     Date data;
73.     String localidade , rua ;
74.     List<Transferencia> trans;
75. 
76. }
77. class Morada {
78.     int idMorada;
79.     String rua,localidade;
80. }
81. 
82. 
83. public class Main {
84.     static MongoClient mongoClient = new MongoClient("localhost", 27017);
85. 
86.     public static void enviaProduto(Produto p){
87.         DB db = mongoClient.getDB("trabalho");
88.         DBCollection coll = db.getCollection("Produto");
89. 
90.         List<BasicDBObject> transferencias = new ArrayList<>();
91.         List<BasicDBObject> comentarios = new ArrayList<>();
92.         for(TransferenciaProd a : p.transfs){
93.             BasicDBObject obj = new BasicDBObject()
94.                     .append("utilizador",a.idUtilizador)
95.                     .append("data",a.data);
96.             transferencias.add(obj);
97.         }
98. 
99.         for(Comentario c : p.comentarios){
100.                 BasicDBObject obj = new BasicDBObject()
101.                         .append("utilizador",c.idUtilizador)
102.                         .append("produto",c.idProduto)
103.                         .append("data",c.data)
104.                         .append("conteudo",c.conteudo);
105.                 comentarios.add(obj);
106.             }
107. 
108.             BasicDBObject doc = new BasicDBObject("_id",p.idProduto)
109.                     .append("nome",p.nome)
110.                     .append("tamanho",p.tamanho)
111.                     .append("descricao",p.descricao)
```

```
                        .append("preço",p.preco)
                        .append("classificacao",p.classificacao)
                        .append("publicador",p.publicador)
                        .append("categoria",p.categoria)
                        .append("transferencias",transferencias)
                        .append("comentarios",comentarios);

                coll.insert(doc);

            }

            public static void carregaProdutos() throws SQLException{
                Connection con = DriverManager.getConnection("jdbc:mysql://localhost/trabalho", "root", "ineslucasalves");

                PreparedStatement aux = con.prepareStatement("SELECT * from Produto");
                ResultSet st = aux.executeQuery();
                while (st.next()){
                    Produto p = new Produto();
                    p.idProduto = st.getInt("idProduto");
                    p.nome = st.getString("nome");
                    p.tamanho = st.getInt("tamanho");
                    p.descricao = st.getString("Descrição");
                    p.preco = st.getInt("Preço");
                    p.classificacao = st.getInt("Classificação");
                    p.publicador = st.getInt("Publicador_idPublicador");
                    p.categoria = st.getInt("Categoria_idCategoria");

                    List<Comentario> comments = new ArrayList<>();
                    List<TransferenciaProd> trans = new ArrayList<>();

                    PreparedStatement aux2 = con.prepareStatement("select * from Comentário where Produto_idProduto = ?");
                    aux2.setInt(1,p.idProduto);
                    ResultSet st2 = aux2.executeQuery();

                    while (st2.next()){
                        Comentario c = new Comentario();
                        c.data = st2.getTimestamp("Data");
                        c.conteudo = st2.getString("Conteúdo");
                        c.idProduto = p.idProduto;
                        c.idUtilizador = st2.getInt("Utilizador_idUtilizador");

                        comments.add(c);
                    }

                    PreparedStatement aux3 = con.prepareStatement("select * from Transferência where Produto_idProduto = ?");
                    aux3.setInt(1,p.idProduto);
                    ResultSet st3 = aux3.executeQuery();

                    while (st3.next()){
                        TransferenciaProd t = new TransferenciaProd();
                        t.data = st3.getTimestamp("Data");
                        t.idUtilizador = st3.getInt("Utilizador_idUtilizador");

                        trans.add(t);
                    }

                    p.comentarios = comments;
                    p.transfs = trans;

                    enviaProduto(p);
```

```
172.                 }
173.             }
174.
175.             public static void enviaUtilizadores(Utilizador u){
176.                 DB db = mongoClient.getDB("trabalho");
177.                 DBCollection coll = db.getCollection("Utilizador");
178.
179.                 List<BasicDBObject> transferencias = new ArrayList<>();
180.                 for(Transferencia a : u.trans){
181.                     BasicDBObject obj = new BasicDBObject("_id",a.idProduto)
182.                                             .append("nome",a.designacao)
183.                                             .append("data",a.data);
184.                     transferencias.add(obj);
185.                 }
186.
187.                 BasicDBObject doc = new BasicDBObject("_id",u.idUtilizador)
188.                                                 .append("nome",u.nome)
189.                                                 .append("email",u.email)
190.                                                 .append("password",u.password)
191.                                                 .append("data_nascimento",u.dat
    a)
192.                                                 .append("transferencias",transf
    erencias);
193.
194.                 coll.insert(doc);
195.
196.             }
197.
198.             public static void carregaUtilizadores() throws SQLException{
199.                 Connection con = DriverManager.getConnection("jdbc:mysql://loca
    lhost/trabalho", "root", "ineslucasalves");
200.
201.                 PreparedStatement aux = con.prepareStatement("SELECT * from Uti
    lizador");
202.                 ResultSet st = aux.executeQuery();
203.                 while (st.next()){
204.                     Utilizador u = new Utilizador();
205.                     u.idUtilizador = st.getInt("idUtilizador");
206.                     u.email = st.getString("email");
207.                     u.nome = st.getString("nome");
208.                     u.data = st.getTimestamp("Data_Nascimento");
209.                     u.password = st.getString("password");
210.
211.                     List<Transferencia> tr = new ArrayList<>();
212.
213.                     PreparedStatement aux2 = con.prepareStatement("SELECT * fro
    m Transferência as t inner join Produto as p on t.Produto_idProduto = p.idProd
    uto where t.Utilizador_idUtilizador = ?");
214.                     aux2.setInt(1,u.idUtilizador);
215.
216.                     ResultSet st2 = aux2.executeQuery();
217.                     while (st2.next()){
218.                         Transferencia t = new Transferencia();
219.                         t.idProduto = st2.getInt("Produto_idProduto");
220.                         t.designacao = st2.getString("Nome");
221.                         t.data = st2.getTimestamp("Data");
222.
223.                         tr.add(t);
224.                     }
225.                     u.trans = tr;
226.
227.                     enviaUtilizadores(u);
228.                 }
229.
230.
```

```
231.            }
232.
233.            public static void main (String args[]) throws SQLException {
234.                //publicador();
235.                //utilizadores();
236.                //produtos();
237.                carregaProdutos();
238.                carregaUtilizadores();
239.            }
240.        }
```