Universidade do Minho

Mestrado Integrado em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2018/2019

Mercado de compra e vendas online

Luís Capa	**A81960**
Moisés Antunes	**A82263**
Pedro Capa	**A83170**
Tiago Pinheiro	**A82491**

novembro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Resumo

No âmbito da disciplina de *BD* foi-nos proposto criar uma base de dados sobre um tema à escolha. O tema escolhido por nós foi uma base de dados sobre um mercado de compra e venda online, uma vez que é algo que cada vez é mais utilizado.

O trabalho está dividido em duas fases. Nesta primeira fase, começamos por conceber um modelo concetual, contendo a base estrutural do nosso trabalho. Traduzimos o modelo concetual para um modelo lógico, respeitando as 3 primeiras formas normais. Para completar a nossa base de dados foi preciso fazer o seu povoamento, que consiste em introduzir dados sobre clientes, produtos e compras, para verificar o seu bom funcionamento. No final foi preciso criar *queries*, *views*, transações e *triggers*, que respondem de forma eficaz aos requerimentos exigidos.

O trabalho realizado foi finalizado com sucesso, conseguindo responder a todos os requisitos, tanto exigidos pelo enunciado como os exigidos pela base de dados.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados

Acrónimos: Unidade Curricular (UC), Base de Dados (BD), Gigabyte (GB), Número de Identificação Fiscal (NIF), Entidade Relacionamento (ER), Sistema de gestão de base de dados (*SGBD*)

Índice

1. Introdução

O aumento do uso da Internet tornou possível realizar algumas atividades sem sair de casa, entre as quais, comprar e vender. Num site de compras online estão disponíveis todo o tipo de produtos, desde material escolar até roupa existindo uma maior variedade de produtos, uma vez que alguns produtos são difíceis de encontrar em lojas físicas. Como não há necessidade de ir à loja o gasto em tempo e em dinheiro diminui, estando esta disponível a qualquer altura. Por estes motivos este tipo de mercado tem tido bastante sucesso.

1.1 Contextualização

Foi-nos proposto criar uma base de dados na UC de BD, com a finalidade de avaliar a nossa capacidade na construção de esquemas, concetual e lógico, *queries* e na gestão de uma base de dados. Cada grupo escolhia o tema do seu trabalho. A escolha foi "Mercado de Compra e Vendas *Online*", pois alguns membros usam com alguma frequência este tipo de mercado.

1.2 Apresentação do caso de Estudo

Como este trabalho era construído do zero, então neste projeto focámo-nos numa aplicação que pudesse servir de modelo. Por esta razão focamo-nos no *site* online *Amazon*, uma vez que é a empresa online com maior sucesso da atualidade. Sendo um dos primeiros *sites* deste género e devido ao sucesso todo que obteve o dono da *Amazon* é neste momento o homem mais rico do mundo e foi a segunda empresa do mundo a atingir o valor de 1 trilião de dólares.

O *site* online *Amazon* é o maior sítio da internet no que se refere a compra e venda de produtos. Este vende artigos em 1ª mão de grandes marcas como a Samsung e a LG e artigos em 2ª mão que qualquer utilizador que esteja registado na *Amazon* pode fazer. O *Amazon*, como qualquer *site online*, podem ser pesquisados artigos em específico, como o *Xiaomi Redmi* 5 *Plus*, ou procurar um tipo de produtos como *smartphones*.

1.3 Motivação e objetivos

No nosso percurso académico não tivemos contacto direto no que se refere a construções de bases de dados. Esta *UC* foi projetada para praticar todos os aspetos relevantes de um sistema de bases de dados quer estes sejam relacionais ou não relacionais. Para isso foi proposto a realização de um trabalho com a finalidade de melhorar as nossas competências na construção e gestão de uma *BD*.

Como em praticamente todas as aplicações informáticas é necessário uma base de dados a aprendizagem da gestão da mesma torna-se importante para o nosso futuro, uma vez que pode vir a ser o que fazemos no futuro.

1.4 Viabilidade e Vantagens do Projeto

A implementação de um sistema de base de dados para esta aplicação trouxe sobretudo vantagens como resposta rápida aos pedidos de informação, múltiplo acesso, flexibilidade e integridade da informação. Neste projeto em particular, se um utilizador efetuar uma operação, esta será rápida, visto que a base de dados contém toda a informação necessária acerca do utilizador para a realizar. Já que o custo elevado da construção de uma base de dados é a sua maior desvantagem, especialmente se esta for mal concetualizada, este projeto torna-se viável, até porque é relativamente simples e foi investido algum tempo na sua concetualização, minimizando a desvantagem mencionada.

2. Levantamento e análise de requisitos

2.1 Métodos de levantamento e análise de requisitos adotados

No processo de levantamento e análise de requisitos foi utilizada uma abordagem centralizada, ou seja, foram seguidos os seguintes passos:

- **Entrevista:** Conversa com cliente sobre os requisitos que ele considera importantes no sistema;
- **Recolha de requisitos**: Recolha de requisitos, sobre o sistema, com potenciais clientes do site;
- **Reunir**: Agrupar os vários requisitos dos diferentes utilizadores numa única lista de requisitos
- **Verificação de requisitos**: Estudo dos requisitos levantados de forma a garantir que não haja inconsistências, ou conflitos, entre diferentes requisitos;
- **Validação de requisitos**: Garantir que todas as principais funcionalidades foram cumpridas com precisão e que o sistema seja implementável;

2.2 Requisitos Levantados

2.2.1 Requisitos de Descrição

Para que um utilizador pudesse usar este *site* de compra e vendas era pedido as seguintes informações sobre este, tais como o *NIF*, que era o seu identificador, a *password*, estas duas eram necessárias se o utilizador quisesse autenticar, a morada, visto que as compras eram entregues ao domicílio, a sua data de nascimento, uma lista de contactos, email e telefone, e ainda uma lista de métodos de pagamento, ou seja, os métodos que dispõe para carregar a conta. Um utilizador também tem um saldo.

Os produtos que são colocados à venda tem um identificador, uma designação, descrição, preço, quantidade e têm de estar associados a um utilizador, o que o colocou para venda.

Uma compra é referente a um só produto, no entanto, e possível escolher a quantidade que se pretende comprar deste produto. Logo, a compra tem uma quantidade, o preço total da compra e está associada a um carrinho.

O carrinho é uma lista de compras, na qual regista-se a data das compras e o método de transporte que o utilizador escolhe para entregar os produtos.

Para efetuar operações sobre o saldo o utilizador tem diversos métodos de pagamento disponíveis. Também é fornecido algumas formas de entrega, ou métodos de transporte, que está associado um custo e um tempo estimado de entrega.

2.2.2 Requisitos de exploração

Após algumas conversas com o cliente, foi decidido as informações que cada um dos tipos de utilizadores tinha acesso.

Requisitos do utilizador

- Criar uma conta na aplicação, colocando alguns dados para se registar;
- Ver produtos que estão disponíveis no mercado;
- Filtrar os produtos que pretende ver/comprar do mercado;
- Aceder a todas as compras realizadas;
- Aceder a informações pessoais;
- Ver informações detalhadas sobre cada forma de pagamento ou transporte;

Requisitos do administrador

- Aceder à lista de todos os produtos no sistema;
- Aceder a informações pessoais de todos os utilizadores;
- Aceder a todas as compras no sistema;
- Ver a faturação total;
- Ver a faturação de cada cliente;
- Ver estatísticas do mercado e dos utilizadores;

2.2.3 Requisitos de Controlo

Para que a dados se mantivessem privados e a base de dados se mantivesse confiável foi decidido que cada um dos tipos de utilizador teria os seguintes privilégios.

Requisitos do utilizador

- Comprar produtos que estão disponíveis no mercado;
- Adicionar/Alterar informações pessoais;
- Colocar um artigo à venda;
- Carregar/Levantar dinheiro para/da conta;
- Criar uma conta na aplicação, colocando alguns dados para se registar;

Requisitos do administrador:

- Adicionar/Remover métodos de pagamento ou transporte;

2.3 Análise geral dos requisitos

Após o levantamento dos requisitos foi organizado uma entrevista com o cliente de forma a perceber se estes estavam de acordo com as suas ideias. Segundo os requisitos levantados, chegou-se à conclusão que atendem às funcionalidades básicas para o utilizador usar a aplicação, ao mesmo tempo restringindo os seus direitos.

3. Modelo Conceptual

3.1 Apresentação da abordagem de modelação realizada

O primeiro passo na construção do modelo concetual foi definir as principais entidades, para isso foram analisados os requisitos do sistema e identificar todos os nomes. No passo seguinte foram identificadas as entidades e associados os diferentes atributos a cada entidade, e o seu tipo, por exemplo se eram identificadores da entidade, atributos multivalorados, atributo derivado ou um atributo composto. Foi necessário encontrar as principais relações entre as várias entidades e o tipo de relações e a multiplicidade das relações. Por fim, foi verificado se havia redundância no modelo e se este estava de acordo com os requisitos.

3.2 Identificação e caracterização das entidades

- Utilizador
- Métodos de Pagamento
- Produto
- Compra
- Carrinho
- Método de Transporte

Como seria de esperar, uma das entidades é o utilizador. Este é o responsável pelas operações de compra ou venda no site. Este dispõe de diferentes métodos de pagamentos (que são por si só uma entidade) como transferência bancária, cartão de crédito, entre outros. Como foi mencionado acima, o utilizador pode vender ou comprar produtos no site, sendo que os produtos, como entidades, devem ser listados e caraterizados pelo seu vendedor. Várias compras individuais de produtos formam um carrinho de compras, ao qual os produtos serão adicionados após as compras, que serão oficializadas dada a confirmação do carrinho final. Os produtos comprados serão então transportados até ao seu comprador pelo método mais conveniente.

3.3 Identificação e caracterização dos relacionamentos

- Utilizador – Métodos de Pagamento
- Utilizador – Produto (Venda Apenas)
- Produto – Compra
- Compra – Carrinho
- Utilizador – Carrinho
- Carrinho – Método de Transporte

O método de pagamento é escolhido pelo utilizador que dispõe de várias opções de como quer pagar o seu carrinho. O utilizador deve efetuar a compra adquirindo os produtos que pretende, sendo que a compra individual de cada produto será automaticamente adicionada ao carrinho de compras do utilizador, no qual estes aguardam confirmação do comprador. O carrinho é depois atribuído a um método de transporte, através do qual chegará ao seu comprador. Note-se que existe uma relação direta entre utilizador e produto, mas neste caso o utilizador é necessariamente o vendedor, já que o comprador se relaciona com o produto através das entidades que lhe permitem fazer a compra.

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

- Utilizador: *NIF*, Data de Nascimento, Morada, Nome, *Password*, Telemóvel, *Email* e Saldo.
- Métodos de Pagamento: *ID* e Designação.
- Produto: *ID*, Categoria, Preço, Quantidade, Nome e Descrição.
- Compra: *ID*, Preço e Quantidade.
- Carrinho: *ID* e Data.
- Método de Transporte: Designação, Custo, Tempo e Descrição.

O utilizador necessita de fornecer dados pessoais para se registar no *site* e o utilizar. O *NIF* é necessário para realizar transferências de dinheiro (que será adicionado ou retirado do saldo), a data de nascimento para determinar a idade do utilizador (por exemplo, para proibir acesso a menores), o nome e a *password* para garantir apenas ao utilizador acesso à sua conta pessoal, morada para assuntos das entregas e contatos caso necessário (sendo que o nº de telemóvel tem obrigatoriamente nove algarismos). Cada método de pagamento tem um nº de identificação, assim como cada produto, compra e carrinho. Os produtos são organizados por categoria e é possível atribuir-lhes, para além do nome e preço base, a quantidade a vender e uma breve descrição. A compra detalha, para além da quantidade de produtos comprados, o preço que se refere ao custo total da compra, que pode ser diferente do preço base do produto no caso de ter havido alteração do preço devido a algum desconto, negociação entre os intervenientes ou outro motivo. O carrinho mostra a data da compra e o método de transporte é identificado pelo nome, tendo também portes de envio, tempo de entrega e descrição do método.

3.5 Detalhe ou generalização de entidades

É possível ver através das alíneas anteriores que houve destas (detalhe e generalização de entidades) situações nas entidades. Um caso de generalização é o fato de o comprador e o vendedor serem ambos instâncias da mesma entidade, neste caso a entidade “Utilizador”. Um caso de detalhe está nas entidades “Produto” e “Compra”, nas quais temos diferentes preços conforme o preço listado e aquele que foi de fato pago.

3.6 Apresentação e explicação do diagrama ER

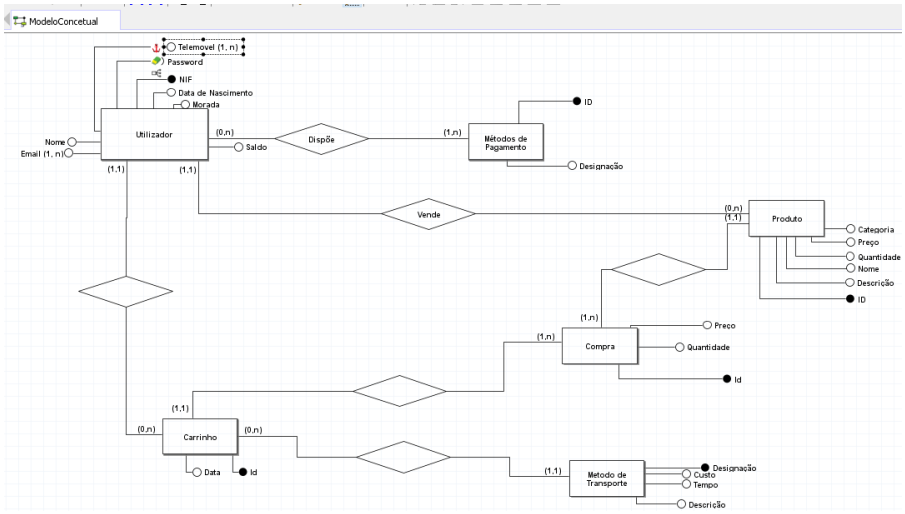


Figura 1: Esquema Concetual

Estas entidades e atributos foram os escolhidos, como referido nas secções anteriores, nomeadamente as secções 3.2 e 3.4, sobretudo tendo em conta que os atributos referentes aos contactos do utilizador são multivalorados e que através do esquema é fácil perceber quais dos atributos são chaves primárias. Os relacionamentos criados e os seus tipos são os explicados na secção 3.3, sendo que o esquema concetual ajuda a perceber que a relação “Utilizador-Produto” é apenas em caso de venda e não de compra, ou seja, um comprador relaciona-se com o produto que compra através das entidades “Compra” e “Carrinho”.

3.7 Validação do modelo de dados com o utilizador

O esquema concetual inclui o diagrama *ER* e a documentação necessária para o descrever. Por vezes, o esquema não corresponde ao pensamento inicial, portanto é essencial rever e alterar o esquema conforme seja preciso. Isso implica por vezes voltar atrás e alterar entidades, relacionamentos e atributos até que o esquema esteja o mais próximo possível do que pretendemos que seja o resultado final.

Um exemplo disso foi a criação da entidade “Fatura”, que agora não se encontra no esquema, que continha informações sobre as compras e o carrinho, tais como o preço pago e a data da compra. Tal entidade foi eventualmente retirada, isto porque se considerou que as restantes entidades poderiam assumir as suas funções, tornando o modelo mais compacto e eficaz.

4. Modelação Lógica

4.1 Construção e validação do modelo de dados lógico

O modelo de dados lógico foi construído a partir do modelo conceptual feito anteriormente, usando-o como um esquema base. Para efetuar a validação foi preciso garantir que o modelo é estruturalmente correto e suporta todos os requerimentos desejados.

Para atingir o sucesso na construção e validação de um modelo de dados lógico, é preciso que todos os seguintes aspetos sejam cumpridos com sucesso:

- Derivar as relações para o modelo lógico – partindo do modelo conceptual tem de ser efetuada uma correta transição das relações, por outras palavras, as relações entre entidades que existam no conceptual têm de ter essa mesma relação no modelo lógico.

- Os restantes aspetos são enunciados e sucintamente explicados nos pontos 4.3 a 4.6 deste relatório.

4.2 Desenho do modelo lógico

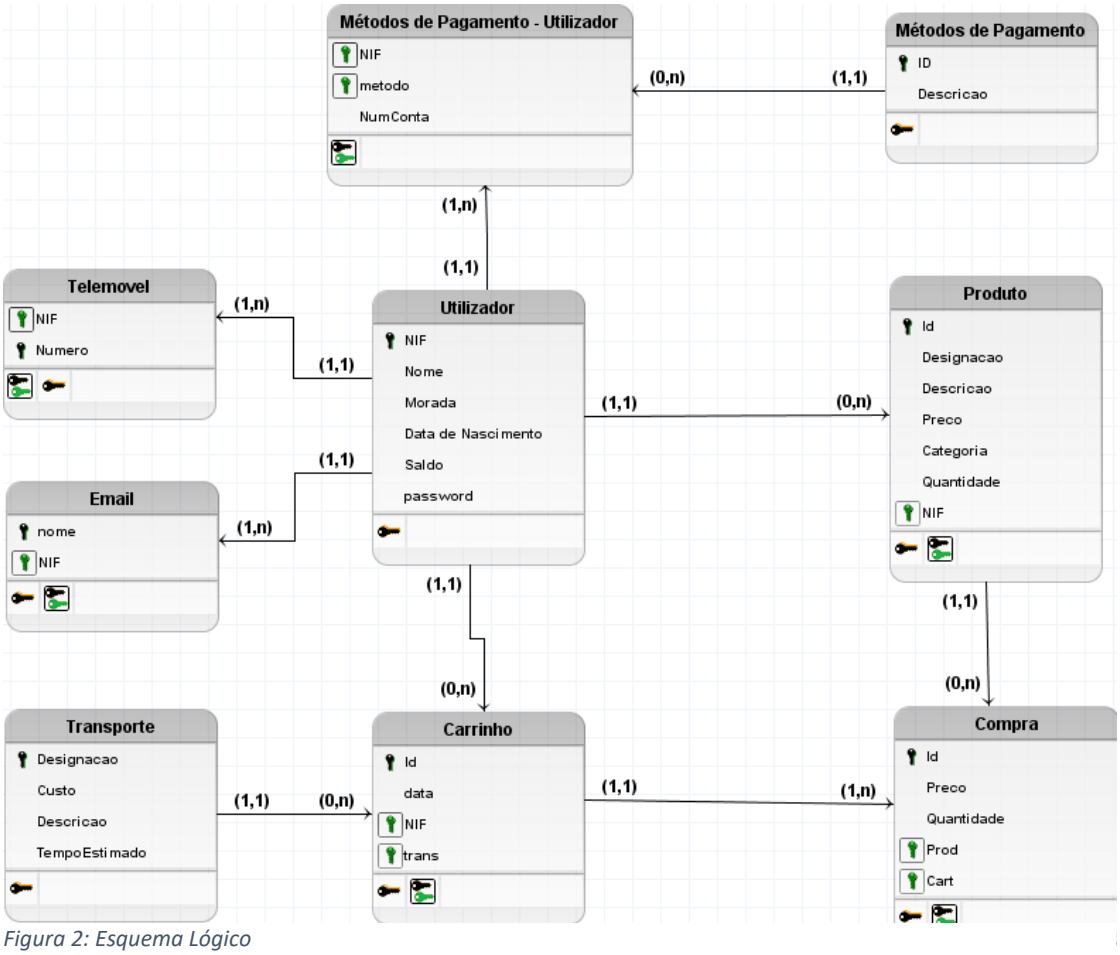


Figura 2: Esquema Lógico

4.3 Validação do modelo através da normalização

O modelo é válido através da normalização se respeitar as três primeiras regras da forma normal.

Analisando o modelo, verificamos que as tabelas existentes não têm identificadores repetidos e os atributos são atómicos possuindo, no máximo, um valor. Assim, concluímos que o modelo respeita a Primeira Forma Normal.

Garantindo a verificação da primeira forma normal, podemos prosseguir para a verificação da segunda. Através da observação das tabelas, conseguimos concluir que todos os atributos normais, os que não possuem chave, são unicamente dependentes da chave primária dessa tabela. Deste modo, verificamos a Segunda Forma Normal.

Por último, depois de garantidas as duas primeiras, falta verificar a terceira forma normal para o modelo ser válido. Como em todas as tabelas do modelo os atributos são independentes entre si e dependentes da respetiva chave primária, podemos afirmar que o modelo está normalizado até à Terceira Forma Normal.

4.4 Validação do modelo com interrogações do utilizador

Através de uma conversa com o utilizador do sistema, obtivemos a validação do nosso modelo lógico.

Durante a conversa, o utilizador fez inúmeras questão para verificar o modelo, como o nosso modelo conseguiu responder as certas interrogações, obtivemos a validação do mesmo.

4.5 Validação do modelo com as transações estabelecidas

Para efetuar a validação do modelo com as transações estabelecidas é necessário tentar resolver as várias operações manualmente.

Depois de resolvidas todas as transações usando este método podemos concluir que as relações presentes no nosso esquema lógico e o nosso modelo suportam as transações pedidas.

Assim, o modelo lógico é válido perante as transações estabelecidas.

4.6 Revisão do modelo lógico com o utilizador

Depois nosso modelo estar acabado e devidamente documento foi exposto a um utilizador que iria avaliar, para garantir que o modelo seria válido e ideal para os requerimentos desejados.

Depois de o modelo lógico ter sido revisto pelo utilizador, ele comunicou que o nosso modelo ia de encontro ao modelo pedido, respondendo muito bem a todos os requerimentos pedidos.

5. Implementação Física

5.1 Seleção do sistema de gestão de bases de dados

Para gerir a base de dados foi usado um sistema relacional que permite manter a segurança, integridade e consistência de dados. Este tipo de modelo suporta relações entre diferentes entidades em tabelas através de chaves estrangeiras. As chaves primárias permitem normalizar os dados evitando redundância de dados das tabelas. Por estes motivos é possível criar, atualizar, consultar e gerir a base de dados, de uma forma mais simples.

O sistema de gestão de base de dados (*SGBD*) utilizado foi o proposto pelos docentes da *UC,* o *MySQL*. Este sistema usa algoritmos complexos que suportam a concorrência no acesso à BD, enquanto mantêm a sua integridade.

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

No desenho de uma base de dados após a criação do modelo lógico do sistema, este é "traduzido" para um esquema físico.

Os esquemas têm "traduções" entre si, por exemplo, as entidades no modelo físico são referidas como tabelas e os atributos de cada entidade são colunas. A maior diferença entre os esquemas é que o modelo físico particulariza o tipo de dados, por exemplo, específica que a chave primária dos utilizadores, o *NIF*, é um inteiro, ou um que a data é do tipo date. Sendo esta a maior diferença a tradução não foi um passo complicado na construção da base de dados.

5.3 Tradução das interrogações do utilizador para SQL

Foram muitas as exigências do utilizador quanto às funcionalidades do programa. Neste subcapítulo iremos mostrar apenas algumas delas.

Uma informação muito requisitada é a verificação de todos os produtos disponíveis para venda, uma vez que esta informação é quase sempre consultada quando um utilizador acede ao *site*. Os produtos estão disponíveis no caso de a quantidade ser maior a 0.

```
-- Ver todos os produtos disponíveis
SELECT * FROM produto p
    WHERE p.quantidade > 0;
```

Figura 3: Interrogação dos produtos disponíveis

Para verificar quanto um utilizador recebeu num intervalo de tempo usa-se um *procedure* que recebe como argumento o *NIF* do utilizador que se pretende verificar, bem como duas datas. No procedimento filtram-se os produtos que pertencem ao utilizador. Verificam-se quais os produtos estão na tabela de compras dentro do intervalo de tempo e soma-se o valor total.

```
-- Quanto um utilizador recebeu num intervalo de tempo
Delimiter //
Create Procedure UtilizadorRecebeuTempo(IN nif INT, In begin DATE, In end DATE)
    Begin
    SELECT SUM(c.preco)
    FROM
    utilizador u,
    produto p,
    compra c,
    carrinho ca
    WHERE
    u.NIF = nif AND nif = p.NIF
        AND p.Id = c.Prod
        AND ca.Id = c.cart
        AND ca.data BETWEEN begin AND end;
    End //
Delimiter //
```

Figura 4: Interrogação do valor recebido

Para verificar quais são os 5 utilizadores que mais faturaram com o sistema agrupa-se os produtos aos utilizadores. Somam-se os preços, ordena-se por ordem decrescente e retira-se os 5 que mais receberam.

```
-- Top 5 clientes que mais receberam com o sistema
SELECT u.NIF, u.nome, SUM(c.Preco) from utilizador u, produto p, compra c
    where u.NIF = p.NIF AND p.Id = c.prod
        GROUP BY u.NIF
            Order by SUM(c.Preco) DESC
                LIMIT 5;
```

Figura 5: 5 utilizadores que mais receberam com o sistema

O objetivo da seguinte *query* era ver todas as compras nas quais dois utilizadores estavam simultaneamente envolvidos. Para começar foram filtrados todos os produtos que o utilizador 2 tinha colocado à venda. O sistema une as tabelas que contêm compras referentes aos produtos do utilizador 2. Também foram filtrados todos os carrinhos no qual o utilizador 1 estava envolvido. Por fim, foram unidas as duas tabelas resultantes pelo atributo “Cart” da compra e o “Id” do carrinho. Foi utilizado o operador *OR*, mas desta vez os dois utilizadores tinham os papéis trocados.

```
-- Todas as compras em que os dois utilizadores estão envolvidos
Delimiter //
Create Procedure comprasUtilizadores(IN user1 INT, IN user2 INT)
BEGIN
    Select Distinct c.Id, c.Preco, c.Quantidade, c.Cart, c.Prod, p.Designacao
        From Utilizador u, Carrinho car, Compra c, Produto p
            Where p.NIF = user2
                AND c.Prod = p.Id
                AND c.Cart = car.Id
                AND car.NIF = user1
                OR
                (p.NIF = user1
                AND c.Prod = p.Id
                AND c.Cart = car.Id
                AND car.NIF = user2);
END //
Delimiter //
```

Figura 6: Interrogação compras entre dois utilizadores

A seguinte interrogação refere-se a um *trigger*. Este *trigger* atualiza o saldo do comprador e do vendedor. Sempre que é inserida uma compra na base de dados, é, inicialmente, adquirido o *Id* do comprador e o *Id* do vendedor. É invocada a função "atualizaSaldoQuantidade", que atualiza o saldo de ambos os utilizadores. Nesta interrogação é assumido que outra camada aplicacional é que controla se a compra é válida ou não, ou seja, assume-se sempre que o comprador tem saldo suficiente para realizar a compra.

```
-- Trigger que atualiza o saldo de um utilizador ao inserir uma compra
Delimiter //
Create Trigger AtualizaSaldo AFTER INSERT ON compra
    For Each Row
    Begin
        Declare Uvende INT;
        Declare Ucompra INT;
        Declare precoProduto Decimal(16,2);

        Select  NIF_Utilizador_Vende(New.Prod) INTO Uvende;
        SELECT  NIF_Utilizador_Compra(New.cart) INTO Ucompra;
        SELECT  getValorProduto(New.Prod) INTO precoProduto;

        Set     precoProduto = New.Quantidade * precoProduto;

        Call    atualizaSaldoQuantidade(New.Prod, Uvende, Ucompra, New.Quantidade, precoProduto);
    End //
Delimiter //

-- Retirar saldo pelo transporte
Delimiter //
Create Procedure retiraCustoTrans(IN nif INT, IN preco Decimal(16,3))
    Begin
```

Figura 7: Interrogação da atualização do saldo após uma compra

5.4 Tradução das transações estabelecidas para SQL

Para garantir maior consistência na base de dados, foram criadas algumas transações. Neste capítulo são dados alguns exemplos de transações, nas quais foi considerado que as interrogações estarem relacionadas.

A transação abaixo é, basicamente, o registo de uma operação de compra. Primeiro, adiciona-se o carrinho e depois, são adicionadas as compras desse carrinho.

```
Start Transaction;

Insert Into carrinho
    Values (curdate(), 'TNT', 1);

Insert Into compra
    Values (2, 5, 4),
           (30, 5, 6),
           (5, 5, 10);
COMMIT;
```

Figura 8: Transação de inserção de carrinho

Nesta transação contêm o processo de inserção de um utilizador para a base de dados. Para começar foi inserido a informação principal do cliente, a seguir, foram adicionados os contactos deste, tal como os métodos de pagamento. Por fim, foi adicionada o saldo do cliente.

```
-- Transação que adiciona cliente e toda a informação sobre ele
Start Transaction;

Insert Into utilizador (NIF, Nome, Morada, DataNascimento, password)
    Values (136390670, 'João Castor', 'Gualtar', '1990-08-23', 'IvanZaytsev');

Insert Into telemovel
    Values ('236558656', 136390670),
           ('753779279', 136390670);

Insert Into email
    Values ('zaytsevIvan@gmail.com', 136390670);

Insert Into utilizador_metodosPagamento
    Values (136390670, 2, '34j5jsi58ugsh4h9ah94ha9f');

Update Utilizador u
Set u.saldo = 250.14
Where u.NIF = 136390670;

COMMIT;
```

Figura 9: Transação da inserção de um utilizador

5.5 Escolha, definição e caracterização de índices em SQL

Os índices são fundamentais em base de dados, visto que permitem uma maior facilidade na gestão da mesma.

Aceder à informação de uma linha em que a tabela tenha índices é mais rápido pois acede-se diretamente à informação, no caso da chave existir na tabela, e sabe-se que se retira sempre o que se pretende pois não existem duas linhas com chaves iguais. Estes podem ser comparados com os *Map,* que por exemplo, são usados em programação orientado aos objetos. As tabelas já vêm por *default* ordenadas pela chave primária e como estas são as mais usadas em *joins* e no acesso ás tabelas, então estas tarefas são mais eficientes.

Na base de dados do projeto são usados índices, por exemplo, para identificar o utilizador. Sabendo que os *NIF* são únicos, é garantido que a informação que é fornecida sobre os seus dados está correta. Para o caso em que os identificadores apenas servem para relações, por exemplo, para identificar que produtos fazem parte de uma compra, é usado *auto incremente.* Este método está presente nos índices do produto, compras e carrinho.

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

Este é um passo importante na criação da base de dados, pois pode ser necessário adquirir novo *hardware* para armazenar a base de dados, no presente bem como no futuro dependendo da taxa de crescimento da mesma. Caso esta etapa não seja bem feita, o espaço ocupado do disco atinja uma percentagem muito elevada e não seja feito nada para ampliar a capacidade de armazenamento da base de dados, este tornar-se há muito lento bem como ser impossível adicionar mais informação na mesma.

Como a base de dados inicialmente apenas tem apenas as tabelas e algumas informações *default*, como por exemplo, os métodos de pagamento ou os formas de transporte disponível, então o tamanho da base de dados será muito pequeno. No entanto é esperado um crescimento exponencial dos utilizadores do *site* nos dois primeiros anos bem como o número de produtos colocados postos à venda, devido à promoção do *site* e a partir desse momento seja linear. Por estes motivos espera-se que o tamanho da base de dados cresça em 50 *GB* por ano nos dois primeiros anos e 25 *GB* por ano nos anos seguintes.

5.7 Definição e caracterização das vistas de utilização em SQL

As vistas em *sql* podem tornar a base de dados mais eficiente. No caso de um acesso a uma tabela com algumas restrições ser pouco eficiente e ser utilizado com alguma frequência podem ser criadas vistas com a finalidade de um acesso à informação mais eficiente. No caso deste projeto foram pedidas informações mensais e anuais acerca dos carrinhos realizados.

```sql
-- Lista de Carrinhos do mês de Novembro de 2018
CREATE VIEW Carrinho_Novembro_2018 AS
    SELECT * FROM carrinho ca
    WHERE ca.data BETWEEN '2018-11-01' AND '2018-11-31';

SELECT * from Faturas_Janeiro_2018;
```

Figura 10: Vista sobre a lista dos carrinhos do mês de novembro

```sql
-- Lista de Carrinhos do ano de 2018
Create View Carrinho_2018 AS
    SELECT * FROM carrinho ca
    WHERE ca.data BETWEEN '2018-01-01' AND '2018-12-31';

SELECT * from Carrinho_2018;
```

Figura 11: Vista sobre a lista dos carrinhos do ano de 2018

Para os produtos como a tabela seria muito extensa então verificar quais são os produtos disponíveis para a venda iria ser uma operação pesada então a *view* dos produtos será atualizada frequentemente.

```sql
-- Lista de produtos disponíveis para venda
Create VIEW ProdutosDisponiveis AS
    SELECT * from produto
        where quantidade > 0;

SELECT * from ProdutosDisponiveis;
```

Figura 12: Vista sobre os produtos disponíveis

5.8 Definição e caracterização dos mecanismos de segurança em SQL

A base de dados representa uma parte essencial de uma aplicação, portanto é fundamental que esta esteja protegida. Falhas de segurança na base de dados podem afetar outras partes do sistema.

Há várias formas de aumentar a segurança como:

- Limitar a informação que cada utilizador possa aceder;
- Backup

As *passwords* e os *NIF*, apenas podem ser acedidas pelos utilizadores das contas. As restantes informações são públicas e podem ser acedidas por todos os utilizadores.

Uma base de dados na qual sejam feitos regularmente *Backups* tem um risco menor de perder informação. Caso haja uma falha no disco a informação poderá nunca mais ser recuperada. Se houver um *Backup* atualizado os dados não serão perdidos.

5.9 Revisão do sistema implementado com o utilizador

Após o sistema ser implementado, é necessário revê-lo para assegurar o seu funcionamento apropriado. Para uma base de dados trabalhar corretamente, o seu objetivo principal é guardar e aceder a dados eficientemente. A sua eficiência pode ser medida ao observar vários fatores, tais como a taxa de transferência, o tempo de resposta e o espaço ocupado na memória. Idealmente, pretende-se que todos esses três valores sejam o mais baixos possível, mas tal é difícil de concretizar. Geralmente, de maneira a minimizar um destes fatores, outros serão sacrificados, mas o ideal será procurar um equilíbrio entre os três. Quando este equilíbrio é encontrado, surgem vantagens tanto a nível da máquina, como melhor desempenho, menos hardware e manutenção deste, como também para as pessoas, já que a eficiência do programa satisfaz tanto o(s) seu(s) criador(es) como o cliente.

6. Migração para Base de Dados NoSQL

6.1 Justificação da utilização de um sistema NoSQL

Sistemas *NoSQL* oferecem maior flexibilidade do que sistemas SQL, visto que os últimos são obrigados a seguir um modelo previamente criado do seu formato, enquanto os primeiros podem ser alterados conforme seja necessário sem obrigação de alterar o esquema inicial. O *NoSQL* também tem a vantagem da escalabilidade, isto é, o crescimento da quantidade de dados no sistema não afeta tanto a quantidade de recursos necessários para a manutenção da base de dados, o que para alguém que trabalha com recursos limitados e procura sobretudo diminuir os custos da base, é uma enorme mais-valia. Neste momento a base de dados é relativamente pequena, mas para o caso de haver um grande crescimento este tipo de sistemas gere bem o armazenamento de grandes quantidades de informação.

6.2 Identificação e descrição dos objetivos da base de dados, em termos de aplicações e de utilizadores

Esta base de dados tem como objetivo possibilitar aos seus utilizadores a compra e venda de produtos *online*, sendo que nesta segunda fase foi criada uma base de dados não relacional, ou melhor dizendo, foi feita a migração da base de dados relacional anterior para a base atual através de um programa em Java. Tendo em conta as vantagens deste tipo de base de dados, a principal finalidade seria fazer procuras em documentos da mesma coleção. Por exemplo, procurar os produtos disponíveis, ou seja, os produtos que tinham quantidade superior a 0, ou visualizar os produtos mais caros do sistema.

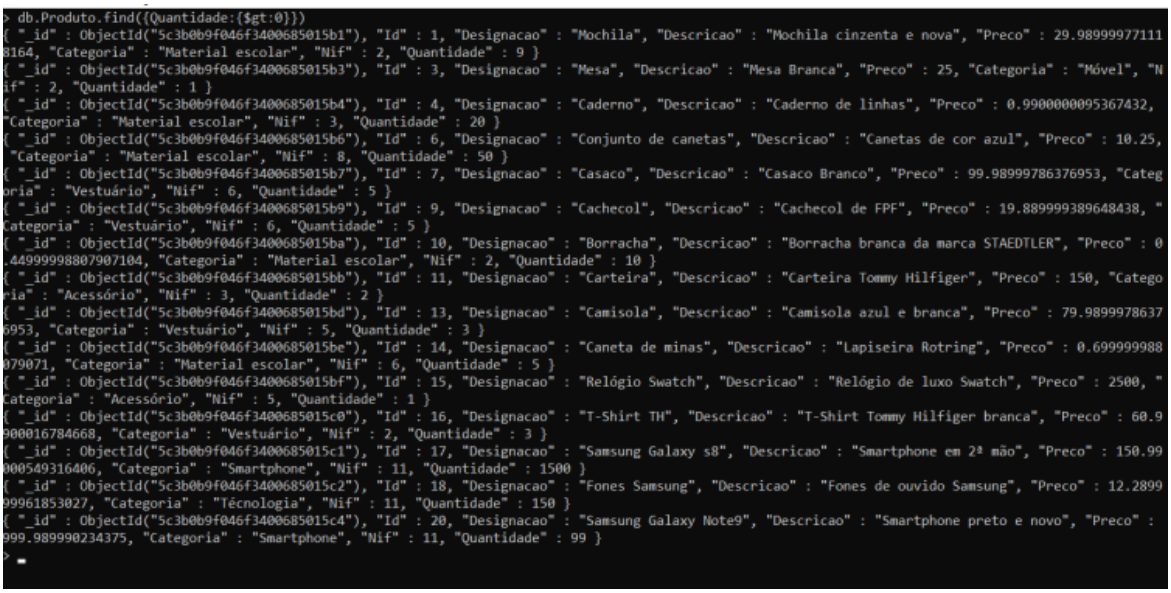
6.3 Identificar e explicar que tipo de questões (necessidades) serão realizadas sobre o sistema de dados NoSQL

Uma das desvantagens do *MongoDB* é o fato dos relacionamentos entre coleções ser limitado, algo que foi utilizado na base de dados relacional. Os campos de um documento no *MongoDB*, no entanto, podem ter listas, algo que o *MySQL* não tinha e que pode ser usado para compensar a falta de relacionamentos, tendo estas sido usadas por esse motivo, no entanto alguns documentos podem continuar a ter identificadores de outros documentos. As questões que realizadas sobre o sistema de dados *NoSQL* são mais simples do que as realizadas na base de dados relacional. As questões foram as seguintes:

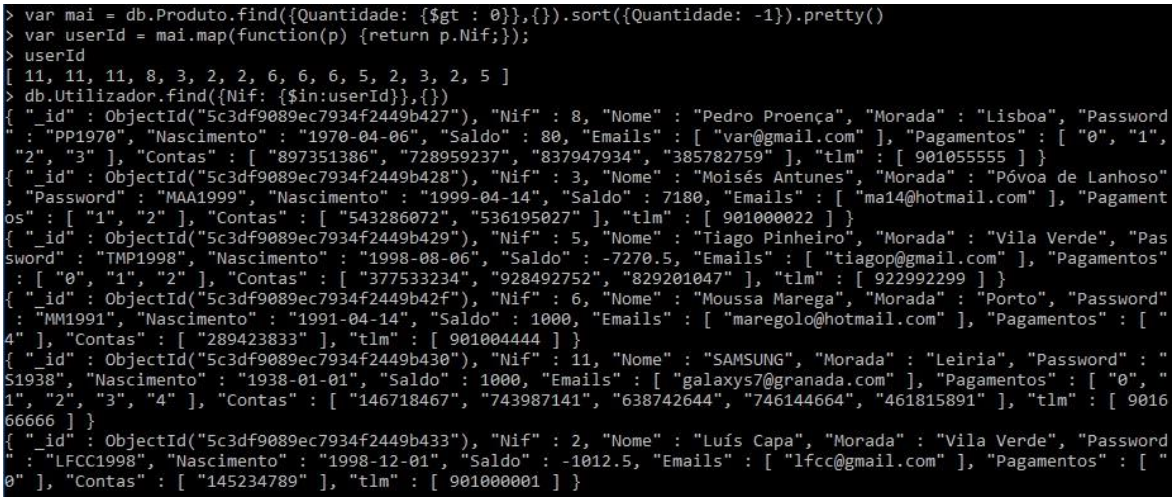
- Encontrar os 4 Produtos mais caros: é preciso aceder à coleção do "Produto", ordenando os seus documentos pelo campo "Preco" e impor o limite de retorno a 4;

```
> db.Produto.find().sort({Preco: -1}).limit(4).pretty()
{
        "_id" : ObjectId("5c3b0b9f046f3400685015c3"),
        "Id" : 19,
        "Designacao" : "Pintura A Última Ceia",
        "Descricao" : "Pintura Leonardo da Vinci",
        "Preco" : 50000,
        "Categoria" : "Arte",
        "Nif" : 1,
        "Quantidade" : 0
}
{
        "_id" : ObjectId("5c3b0b9f046f3400685015b2"),
        "Id" : 2,
        "Designacao" : "Relógio",
        "Descricao" : "Relógio Rolex em ouro",
        "Preco" : 7000,
        "Categoria" : "Acessório",
        "Nif" : 3,
        "Quantidade" : 0
}
{
        "_id" : ObjectId("5c3b0b9f046f3400685015bf"),
        "Id" : 15,
        "Designacao" : "Relógio Swatch",
        "Descricao" : "Relógio de luxo Swatch",
        "Preco" : 2500,
        "Categoria" : "Acessório",
        "Nif" : 5,
        "Quantidade" : 1
}
{
        "_id" : ObjectId("5c3b0b9f046f3400685015b5"),
        "Id" : 5,
        "Designacao" : "Quadro Mona Lisa",
        "Descricao" : "Mona Lisa, de Leonardo da Vinci",
        "Preco" : 1000,
        "Categoria" : "Arte",
        "Nif" : 6,
        "Quantidade" : 0
}
```

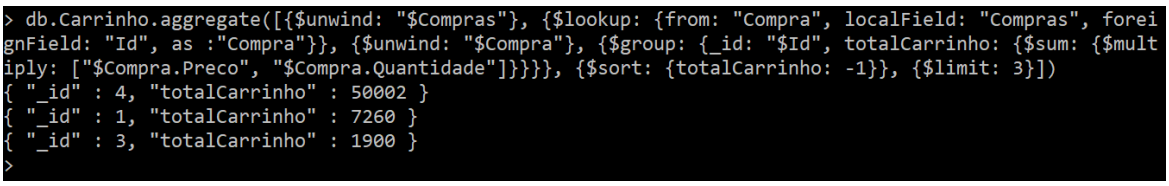
- Verificar os produtos disponíveis no mercado: mais uma vez, é preciso aceder à coleção do "Produto" e depois basta verificar quais produtos tem a "Quantidade" superior a 0;



- Encontrar os utilizadores com produtos à venda: nesta questão vamos recorrer a duas variáveis auxiliar, a primeira (*mai*) vai armazenar o retorno da *query* que verifica os produtos disponíveis, a segunda (*userId*) vai guardar o resultado de um *map* aplicado à "*mai*" que vai retornar os *NIFs* dos vendedores do produto. Depois, a "*userId*" vai ser utilizador para encontrar os Utilizadores através do seu NIF.



- Encontrar os 3 carrinhos mais caros: é necessário aceder ao campo "Compras" do Carrinho e depois fazer o somatório do "Preco" de cada "Compra".



- 5 Utilizadores que mais receberam: Aceder ao campo "Produto" de cada Compra, obtendo o seu id. Com esse id é possível obter o produto, e com esse produto encontramos o Utilizador. O retorno vai conter o NIF do Utilizador, as soma de todas as receitas e o seu nome.

```
> db.Compra.aggregate([{$lookup: {from: "Produto", localField: "idProduto", foreignField: "Id", as: "Produto"}},
{$unwind: "$Produto"}, {$lookup: {from: "Utilizador", localField:"Produto.Nif", foreignField: "Nif", as: "User"}}
, {$unwind: "$User"}, {$group: {_id: "$User.Nif", receitas: {$sum: {$multiply:["$Preco","$Quantidade"]}}, User: {
$mergeObjects: "$User"}}}, {$sort: {receitas:-1}}, {$limit: 5}, {$project: {"User.Nome":1, receitas:1}}])
{ "_id" : 1, "receitas" : 50000, "User" : { "Nome" : "Lucas Silva" } }
{ "_id" : 3, "receitas" : 8050, "User" : { "Nome" : "Moisés Antunes" } }
{ "_id" : 6, "receitas" : 1000, "User" : { "Nome" : "Moussa Marega" } }
{ "_id" : 11, "receitas" : 1000, "User" : { "Nome" : "SAMSUNG" } }
{ "_id" : 8, "receitas" : 80, "User" : { "Nome" : "Pedro Proença" } }
>
```

6.4 Definir a estrutura base para o sistema de dados NoSQL que satisfaça os requisitos e as questões apresentadas anteriormente

Neste caso, como foi decidido migrar para uma Base de Dados *MongoDB*, foram criadas coleções para todas as entidades que tinham sido definidas no modelo concetual. Na *MongoDB* foram criadas no total as seguintes 6 coleções, uma para cada entidade.

- Utilizador - ia conter todos os documentos que caraterizavam os utilizadores do sistema
- Produto - que continha a informação de todos os produtos do sistema
- Compra - os documentos desta coleção identificavam todas as compras do sistema
- Carrinho - os carrinhos feitos pelos utilizadores, que é basicamente, uma lista de compras
- MetodosPagamento - tem todas as formas de pagamento do sistema
- Transporte - coleção que contém os dados de todos os meios de transporte possíveis do mercado

Como numa base de dados baseada em documentos é possível um campo ser uma lista, então os atributos multivalorados podiam ser inseridos num documento, ao contrário da base de dados relacional, por exemplo os contactos de um utilizador. Também foi decidido, que tal como na base de dados relacional, alguns documentos iriam conter um campo com o identificador de um documento de outra coleção.

6.5 Identificar os objetos de dados no sistema SQL que serão utilizados para alimentar o novo sistema

Como o objetivo era migrar para uma nova base de dados, tornando-a equivalente à original, então todos os objetos foram transferidos, apesar de nem todos os objetos de dados no sistema SQL serem usados nas interrogações da nova base de dados. Alguns destes objetos foram alterados devido às diferentes características dos dois sistemas de base de dados. Não foram criadas coleções para todas as tabelas devido às características da base de dados *NoSQL*.

6.6 Mapear o processo de migração de dados, descrevendo o processo de conversão dos vários objetos de dados

Para a realização da migração dos dados para a nova base de dados, foi decidido criar um programa que extraia os dados da base de dados relacional e mais tarde inseria-os na nova base de dados não relacional. Na fase inicial do desenvolvimento do projeto, foram criadas algumas estruturas de dados que iriam guardar temporariamente os dados. Para fazer a ponte entre o programa e a base de dados relacional foi usada uma API do Java, o JDBC. Com o uso de alguns drivers foi possível conectar o programa às bases de dados *MySQL* e *MongoDB*. Alguns objetos de dados não foram alterados em relação à base de dados relacional, no entanto, outros foram alterados. Os objetos que tinham tabelas que representavam atributos *mutlivalorados* foi criado um campo que era uma lista deste tipo de atributos. Em alguns relacionamentos de um para muitos a chave estrangeira estava numa entidade foi removida e a chave primária foi inserida numa lista na entidade que estava relacionada, por exemplo, em vez de a "Compra" ter uma chave estrangeira para o carrinho que pertencia, na nova base de dados o carrinho tem uma lista com os identificadores das respetivas compras.

6.7 Explicar o processo de migração de dados, explicando de forma detalhada as suas principais etapas - extração, transformação e carregamento

A primeira e última etapas do processo de migração consistem essencialmente no que lhes dá os seus nomes - a primeira etapa, extração, é a recolha dos dados da base de dados SQL enquanto a última, carregamento, é a colocação desses dados na base de dados *NoSQL*. Obviamente, os dados da primeira base não podem ser simplesmente inseridos na segunda, sendo para isso que serve a segunda etapa, transformação. Esta trata-se de vários processos menores como a filtragem, ordenação, limpeza e validação de dados.

No programa desenvolvido foram criadas classes com métodos que iam aceder exclusivamente a uma tabela da base de dados e iam guardar os objetos de dados temporariamente, por exemplo foi criada uma classe *ProdutoDAO*, que continha métodos que procurava, inseria ou editava produtos na base de dados relacional.

No início do processo já se sabia que iam ser criadas classes para cada entidade, portanto foram criadas classes baseadas nas entidades identificadas no modelo concetual. Estas classes iam ser baseadas nas tabelas e colunas de cada tabela, mas iam sofrer algumas transformações, visto que já se conseguia criar listas, ao contrário do *MySQL*. Por exemplo, um objeto do tipo "Utilizador" já podia conter listas, para este caso uma lista de contactos e métodos de pagamento.

Para guardar temporariamente esses objetos de dados foram criadas estruturas de dados para todas as classes anteriormente identificadas, que iriam guardar temporariamente esses dados.

Antes de se inserir cada tipo de objeto foi criado uma coleção na nova base de dados, através da utilização da classe *MongoCollection*. Em seguida, para todas os objetos guardados nas estruturas de dados foi criado um documento, que iria conter a informação desses objetos. Usando o método *append* foram inseridos todos os campos nesse documento. Por fim, depois de terem sidos inseridos todos os campos no documento, este foi inserido na coleção respetiva usando o método

insertOne. Este método insere um documento na BD. Como a base de dados não continha muita informação todos os dados foram guardados temporariamente em memória e mais tarde inseridos um a um na base de dados *NoSQL*.

6.8 Apresentar e descrever a implementação do processo de migração de dados

A implementação do processo de migração de dados neste trabalho prático foi, relativamente, simples. A implementação esta dividida em 3 packages: Classes, *ClassesDAO* e *ClassesMongo*. Estas classes vão tratar da transformação, extração e carregamento, respetivamente. A fase da extração resume-se a 3 métodos: "*getConnection*", "*createStatement*" e o "*executeQuery*". Todos estes métodos já estão predefinidos em Java e os seus objetos de retorno também. O *getConnection* cria uma ligação ao *MySQL* (precisa um *username* e uma password). O *createStatement* recebe o *statement* que irá ser executado pelo *MySql* e o *executeQuery* executa esse *statement*.

O processo da transformação passa por converter os dados extraídos do *MySql* para "*Sets*", com o respetivo tratamento de dados. Por exemplo, foi criado um "*Set*" de carrinhos para guardar todos os carrinhos.

Quanto à fase de carregamento, é criado uma ligação ao Mongo através do método *MongoClient*, que recebe um *localhost* e a sua porta. Criada a ligação, é feita a criação de uma *MongoDatabase* e é criada uma coleção para cada objeto que criámos na fase de transformação. Depois disso bastou percorrer os "Sets" dos objetos onde estavam armazenadas as informações e criar um documento para cada objeto, com o auxílio do método: "*document*".

Por fim, também existe a classe *Migracao* que contém o método *main*, que vai interligar todas as fases do trabalho.

6.9 Apresentar a forma como as questões identificadas anteriormente podem ser satisfeitas com o novo sistema, utilizando a linguagem de interrogação do sistema NoSQL

Como as bases de dados *NoSQL* são muito rápidas no que respeita a buscas, as interrogações criadas neste novo sistema foram baseadas nesse aspeto, ou seja, as *queries* criadas foram de busca usando o método *find* quando era uma busca numa coleção e o método *aggregate* para reunir documentos de várias coleções.

Algumas das questões identificadas anteriormente, como as vistas ou os *triggers* não foram implementados na base de dados *MongoDB*, uma vez que o principal objetivo desta fase do trabalho era apenas migrar os dados.

6.10 Apresentar conclusões sobre o trabalho realizado, abordando de forma crítica as diversas ações realizadas

O trabalho realizado, no geral, foi bem conseguido, uma vez que o objetivo principal foi atingido, ou seja, a migração dos dados para uma base de dados não relacional orientada a documentos. Tendo em conta que no futuro é previsto um crescimento da base de dados, seriam melhor em futuras migrações não transferir todos os dados de uma vez só, mas sim guardar temporariamente em memória um número limitado de objetos. Como não foram realizados *triggers* ou vistas, este seria um ponto do projeto a melhorar. Apesar de as bases de dados *NoSQL* serem melhores no que toca a gerir uma grande quantidade de dados, no sistema não deu para constatar tal facto, até porque o tamanho da base de dados era reduzida. Relativamente ao desenvolvimento de interrogações que envolvia a união de várias tabelas foi percetível que o sistema SQL era mais simples do que as interrogações criadas no sistema *MongoDB*. Pelo contrário na altura de inserção de dados nas duas bases de dados como no *MongoDB* as coleções não têm estrutura definida, notou-se uma maior liberdade e facilidade nesta questão comparando com o sistema SQL. Apesar de ambos os sistemas terem várias diferenças, não se pode considerar um sistema melhor do que o outro.

7. Conclusões e Trabalho Futuro

Atualmente, várias coisas que se fazem no dia-a-dia necessitam de bases de dados, como o uso de cartões multibanco, preenchimento de inquéritos ou inscrição em *websites*. Num *site* que pretende atrair várias pessoas, é fundamental a escolha apropriada das caraterísticas de uma base de dados. Qualquer operação nesta loja necessita de aceder à base de dados, desde a compra e venda de produtos até à consulta de operações anteriores, demonstrando assim a sua importância. Isso obriga a ponderar as nossas opções acerca das caraterísticas das bases de dados, como as entidades, relações e atributos, até que estes funcionem juntos de forma coesa e eficiente.

Essa obrigação aumentou os nossos conhecimentos acerca do tema, e deixa-nos com boas expectativas acerca do trabalho que podemos realizar nesta área no futuro.

8. Referências Bibliográficas

Thomas Connolly, Carolyn Begg 2005. DataBase Systems. A pratical approach to Design, Implementation and Management. 4. Available: https://www.dropbox.com/sh/akiqghi79eo8ti8/AAAW5EHvvxwUvAFrgO2DwFZja/3%C2%BA%20Ano/1%C2%BA%20Semestre/Bases%20de%20Dados/Apontamentos%20extras?dl=0&preview=Database_Systems-Thomas_Connolly_4th-Edition.pdf&subfolder_nav_tracking=1 [novembro 2018]