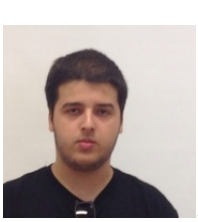
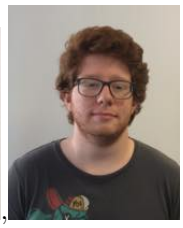
Projeto Java-LI 3-Grupo 24-MIEI

João Bernardo Freitas A74814
Alexandre Martins A77523

Maio 2020

1 Abstract

Esta fase do projeto tem como principal objetivo a criação de um sistema de gestão de vendas, na linguagem **Java**, que permita, através de uma lista de produtos, clientes e vendas, extrair informação útil.
Assim, este relatório pretende sumarizar todos os esforços efetuados para alcançar o objetivo proposto.

Contents

2 Introdução

Este trabalho foi-nos proposto no âmbito da unidade curricular de Laboratórios de Informática III, com o objetivo de, a partir de uns ficheiros de dados, ler estes mesmos e processar a informação neles contida, armazenando-a em estruturas de dados que achamos adequadas de forma a possibilitar uma consulta rápida desta mesma. Sendo assim possível responder de uma forma eficaz a todas as queries apresentadas pelos docentes.

Ao longo deste relatório, vamos descrever o processo de realização do projeto, explicando os passos dados e a as decisões tomadas para alcançar os objetivos, assim como os resultados que obtivemos.

3 Descrição do Problema

A partir de três documentos base, contendo informação sobre os clientes, os produtos e as vendas de uma certa empresa, temos de criar um sistema para gerir as vendas desta mesma empresa. Primeiramente é necessário eliminar qualquer erro relativo a introdução de clientes ou produtos no sistema, verificando se estes são válidos ou não.

De seguida, tendo em conta os clientes e produtos que existem na empresa, é necessário também validar as vendas, verificando se estas foram feitas por clientes existentes, assim como os produtos e se o código da própria venda está correto.

Após todas as validações, partindo de dados corretos temos de criar uma estrutura para guardar os clientes, uma para os produtos, outra para toda a faturação e por fim uma última que faz a gestão separada por filiais da empresa.

Tendo toda a informação guardada resta-nos encontrar respostas eficazes para as queries que nos foram fornecidas, que pretendem obter informação sobre as várias vendas, clientes e produtos da empresa.

4 Estruturas de Dados

As estruturas de dados servem para armazenar e organizar dados e devem ser escolhidas e criadas de forma a serem usadas eficientemente, facilitando a procura e modificação delas mesmas.

Nesse sentido, e tendo em conta os requisitos deste trabalho, utilizamos diversas estruturas oferecidas pelo **Java**. De entre todas as estruturas oferecidas decidimos utilizar as seguintes:

- ArrayList
- TreeSet
- Set

Cada uma destas adaptada ao tipo de informação armazenado e escolhida consoante a nossa necessidade de aceder aos dados e trabalha-los da forma mais rápida e eficaz possível.

4.1 Clientes

Os clientes são definidos por uma letra maiúscula seguido de quatro números, por exemplo **A1111** por isso, para o catálogo de clientes decidimos utilizar um **Set** de clientes, visto que esta estrutura não permite a existência de elementos duplicados.

```
  Cliente
private String cliente;


  CatalogoClientes
private Set<Cliente> catClientes;
```

4.2 Produtos

Os produtos são semelhantes aos clientes, sendo que a única diferença está no facto de ter duas letras maiúsculas em vez de apenas uma, exemplo **AA1111**. Como tal decidimos aplicar uma solução semelhante aos clientes, tendo novamente um **Set** de produtos.

```
  Produto
private String produto;


  CatalogoProduto
private Set<Produto> catProduto;
```

4.3 Vendas

Os ficheiros de input referentes ás vendas incluem um número elevado de informação, sendo que nem toda essa informação poderá ser relevante para todas as queries, como tal decidimos criar duas estruturas distintas, **CatFiliais** e **Faturacao**, que nos irão permitir otimizar a procura de dados dependendo da query.

4.3.1 Faturacao

A estrutura fatura é uma estrutura que está mais focada na venda de um certo produto, como tal, inclui toda a informação relevante a essa venda e um identificador. Essa estrutura é depois guardada num **TreeSet**, onde as operações básicas são em executadas em tempo **O(log n)**. Tal como num **Set**, esta não permite duplicados, impedindo a repetição de uma fatura, e ordena-as de acordo com um comparador que tem como referência o id delas.

```
  Fatura
private int id;
private double preco;
private Cliente Cliente;
private Produto Produto;
private int quantidade;
private char promo;
private int filial;
private int mes;

  Faturacao
private TreeSet<Fatura> faturas;
```

4.3.2 CatFiliais

Esta estrutura foi criada de forma a responder rapidamente a todas as queries que focam nas filiais, por isso decidimos usar um **ArrayList** com 3 filiais, sendo que cada uma contém dois **HashMap**, onde um irá conter todas as informações relativas ás compras de um certo produto numa filial e outro irá conter as informações das compras efetuadas por um cliente numa filial.

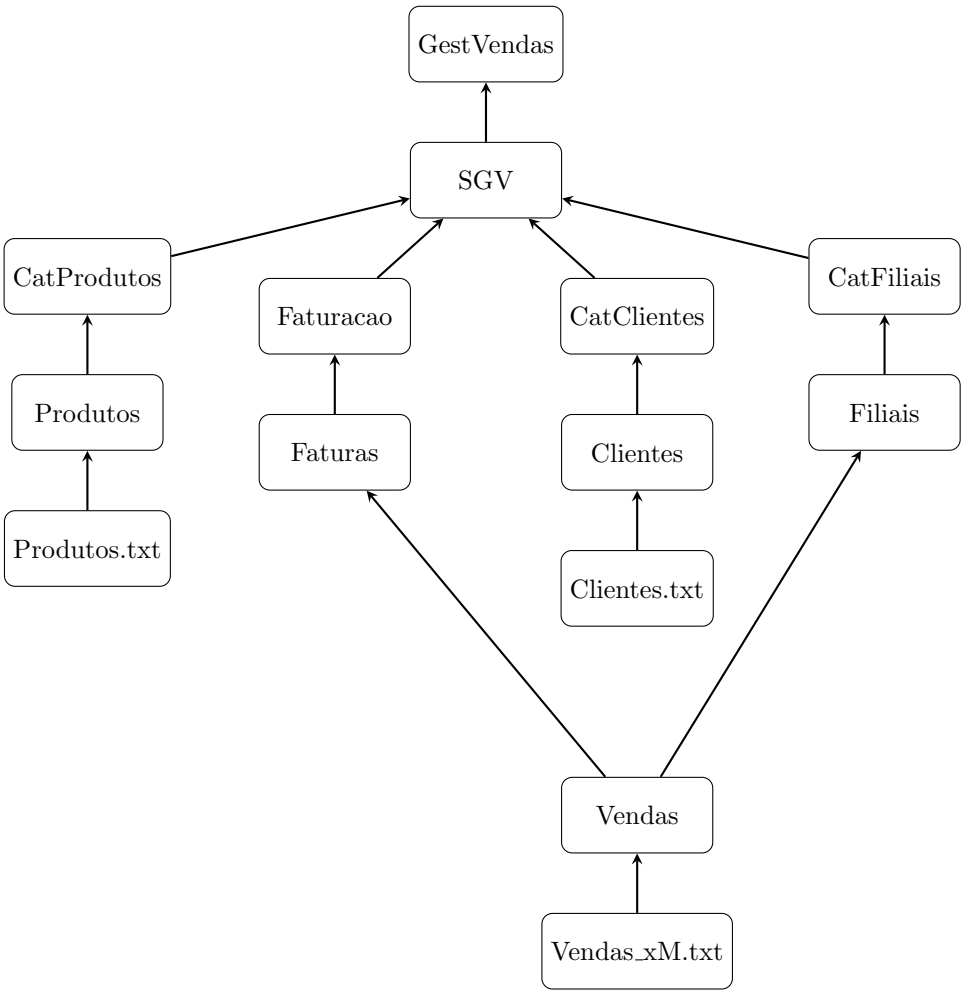
```
 CatFilial
private ArrayList<Filial> filiais;

 Filial
private int id;
private HashMap<Produto, List<Venda>> produtosVendas;
private HashMap<Cliente, List<Venda>> clientesVendas;
```

4.4 SGV

A estrutura SGV, Sistema de Gestão de Vendas, é a estrutura que contém todas as anteriores, permitindo assim cruzar informação entre elas para responder a todas as queries.

```
private CatProdutos catProdutos;
private CatClientes catClientes;
private CatFilial filiais;
private Faturacao faturacao;
```

5 Arquitetura



5.1 Diagrama de classes

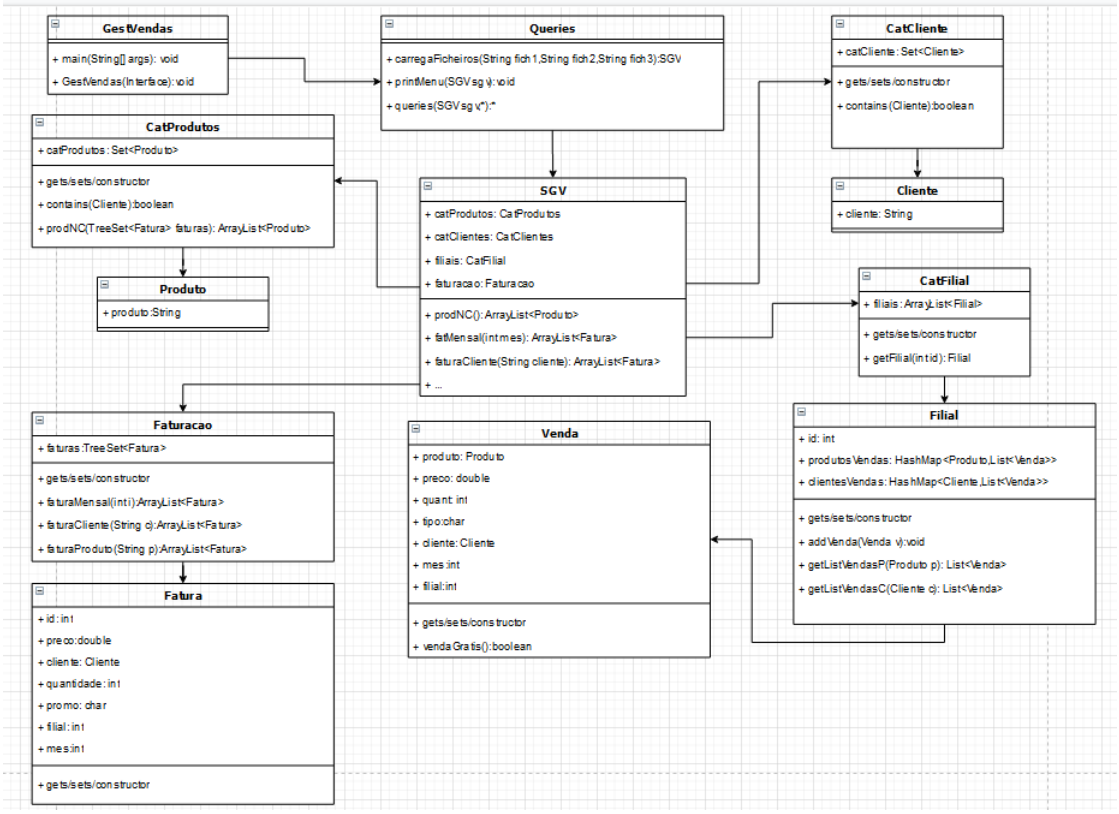


Figure 1: Diagrama de classes

6 Resultados

Foi realizada uma recolha de dados para uma comparação entre dois modos de leitura, uma delas usando um BufferedReader e outra usando a classe Files do Java. Cada teste foi realizada três vezes sendo que o resultado apresentado nas tabelas é a média desses três tempos medidos em segundos.

6.1 Leitura sem parsing

Sem Parsing		
	BufferedReader	Files
1M	0,6493	0,9829
3M	1,0169	5,2364
5M	1,8858	6,7830

Figure 2: Tabela das médias dos tempos de leitura sem parsing

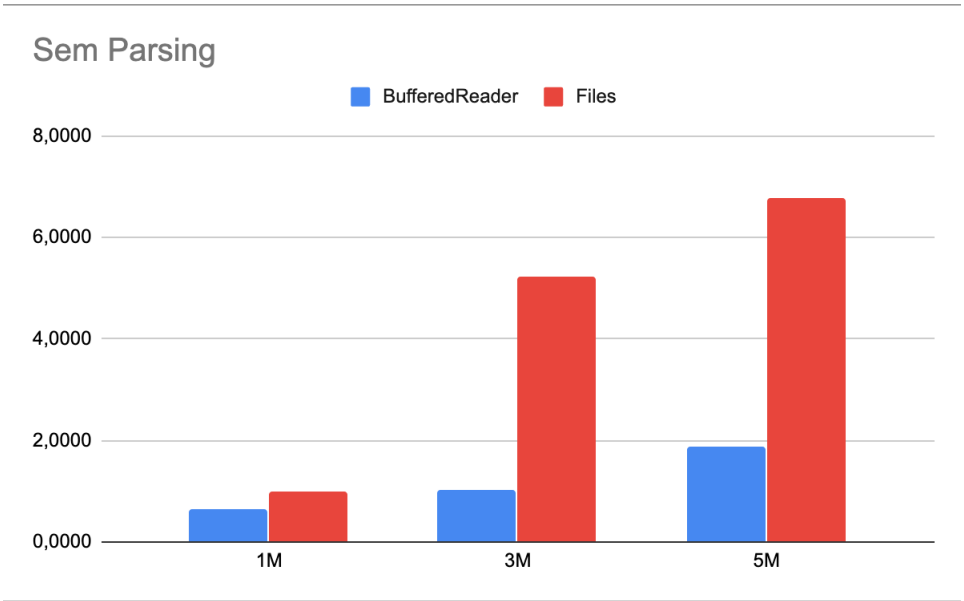


Figure 3: Gráfico dos tempos medidos de leitura sem parsing

6.2 Leitura com parsing e criação de Instâncias

Parsing + Criação de Instâncias		
	BufferedReader	Files
1M	1,8885	2,8985
3M	4,1156	8,2439
5M	5,7566	13,3437

Figure 4: Tabela dos tempos médios de leitura com parsing e criação de instâncias

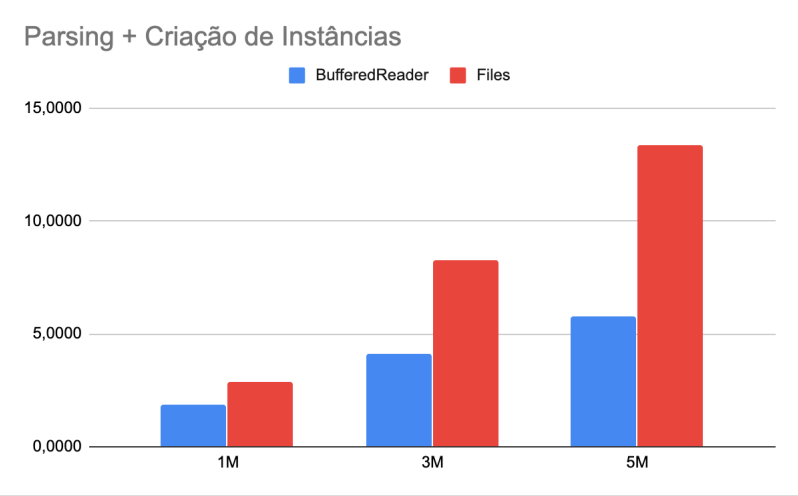


Figure 5: Gráfico dos tempos médios de leitura com parsing e criação de instâncias

6.3 Leitura com parsing, instâncias e validação

Parsing + Instâncias + Validação		
	BufferedReader	Files
1M	2,2961	2,4563
3M	4,1622	8,8502
5M	7,1702	12,6480

Figure 6: Tabela das médias dos tempos de Leitura com parsing criação e validação de vendas

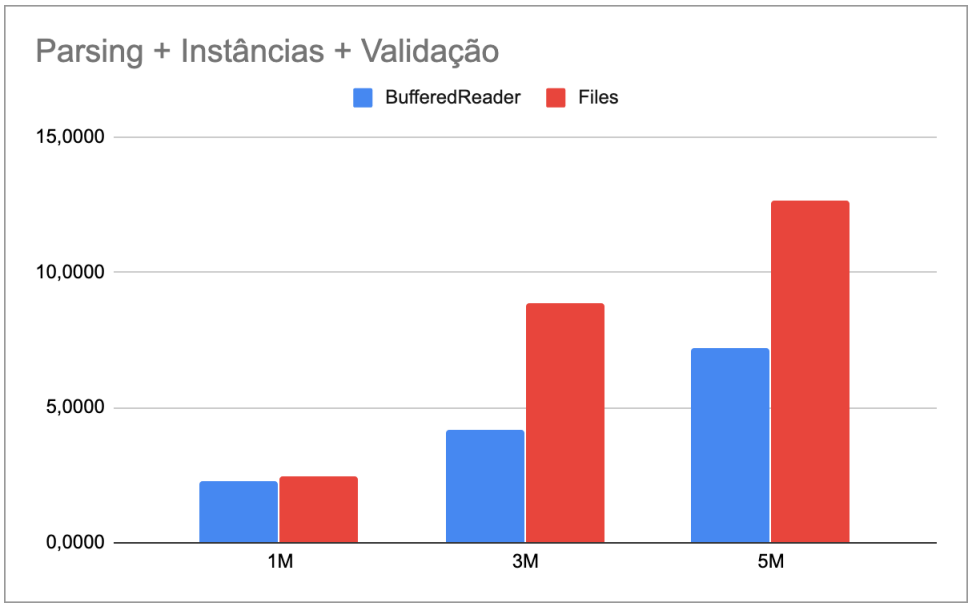


Figure 7: Gráfico das médias dos tempos de Leitura com parsing criação e validação de vendas

Como podemos conferir em todas as tabelas e gráficos, pela informação recolhida podemos afirmar que o método que utiliza um BufferedReader para ler um ficheiro é bastante mais rápido do que a utilização de Files. Esta situação torna-se cada vez mais evidente com o aumento do tamanho do ficheiro.

Assim, a partir desta informação comprova-se que este método é uma boa escolha para a realização do projeto onde é necessária a leitura de vários ficheiros.

6.4 Tempo de execução das queries

Por fim temos os resultados das queries interativas para os 3 ficheiros input.

Resultado das queries			
Query	Vendas_1M	Vendas_3M	Vendas_5M
LoadTimes	5.41	13.70	23.73
Query 1	1.33	4.32	6.88
Query 2	6.95	41.45	67.35
Query 3	0.04	0.096	0.163
Query 4	0.05	0.092	0.142
Query 5	0.05	0.094	0.134
Query 6	0.85	1.27	1.600
Query 7	0.08	0.33	0.377
Query 8	0.18	0.515	0.647
Query 9	0.0011	0.0013	0.0014
Query 10	74.49	− − −	− − −

Table 1: Tempo em segundos das queries para um dado número de vendas

NOTA: Não são apresentados resultados para a query 10 para os ficheiros 3M e 5M porque demoravam demasiado tempo e paramos a execução

7 Conclusão

Após a realização do projeto sentimos que este foi concluído com sucesso. Estruturamos o nosso Sistema com base em estruturas já existentes do **Java**, o que nos deu uma maior flexibilidade para trabalhar os dados eficazmente.

As nossas maiores dificuldades encontraram-se na query 10, onde inicialmente não entendemos bem o que era pretendido e, após consultar com colegas que nos explicaram o que os docentes lhes comunicaram o que pretendiam, tivemos ainda assim alguns problemas na sua resolução, obtendo tempos de execução muito longos. No entanto todas as outras queries foram executadas rapidamente, sendo a maior parte delas praticamente instantâneas.

De um modo geral consideramos que o objetivo foi cumprido, desenvolvemos um sistema capaz de armazenar a informação fornecida e geri-la de acordo com os requisitos dados pelos docentes.

O presente trabalho serviu para um aprimoramento das nossas aptidões relativamente à organização de dados para obter uma maior eficiência tanto a lê-los como a interpretá-los.