Universidade do Minho
Mestrado Integrado em Engenharia Informática

Laboratórios de Informática III

Sistema de Gestão de Vendas

Grupo 34

Ana Afonso A85762

Márcia Teixeira A80943

Pedro Silva A82522

Índice

Introdução

O presente projeto enquadra-se na unidade curricular de Laboratórios de Informática lll, na qual foi proposta a elaboração de um Sistema de Gestão de Vendas.

Recorrendo à aplicação de uma arquitetura baseada em Model, View e Controller, bem como à utilização de determinadas estruturas de dados, o sistema a desenvolver apresenta como principal objetivo o armazenamento de informações de vendas relacionando produtos, clientes e vendas.

Em foco, encontra-se ainda o encapsulamento de dados, que é de considerável importância, uma vez que visa impedir possíveis alterações efetuadas por agentes externos.

A fim de garantir o bom funcionamento do sistema torna-se necessário responder de forma ponderada e consciente às *queries* enunciadas, assegurando a eficiência do mesmo.

Diagrama de Classes

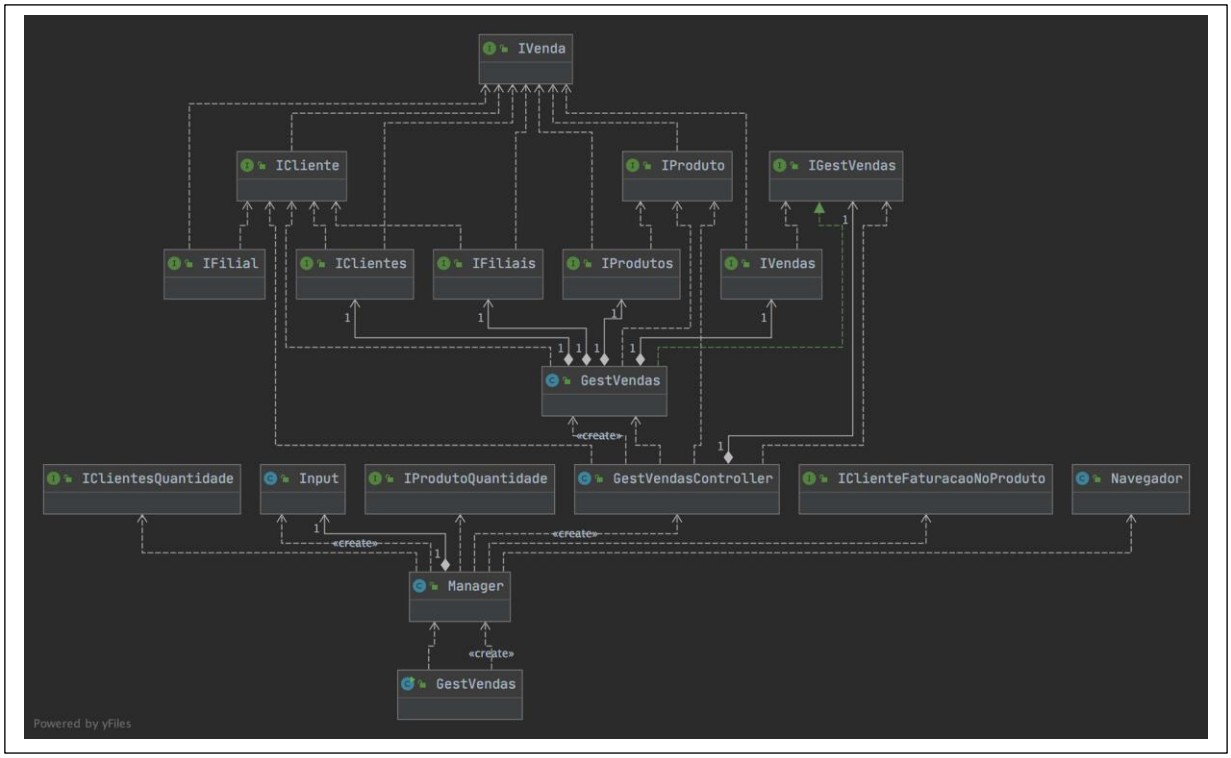

Figura 1 – Diagrama de Classes

Breve Descrição da Estrutura

Na parte C do presente projeto foi necessária a elaboração de uma estrutura de dados pormenorizada e complexa.

Porém, nesta parte, dado o caráter adaptativo da linguagem JAVA foi possível responder a uma grande fração de *queries* sem que, para isso, fosse fundamental uma elaboração tão profunda das estruturas de dados, pelo que a conceção das mesmas tornou-se muito mais simples (por exemplo, devido ao facto de estas *queries* não exigirem distinção entre compras promocionais e normais, o que não foi tido em consideração na criação da estrutura).

Apresenta-se de seguida uma esquematização da estrutura:

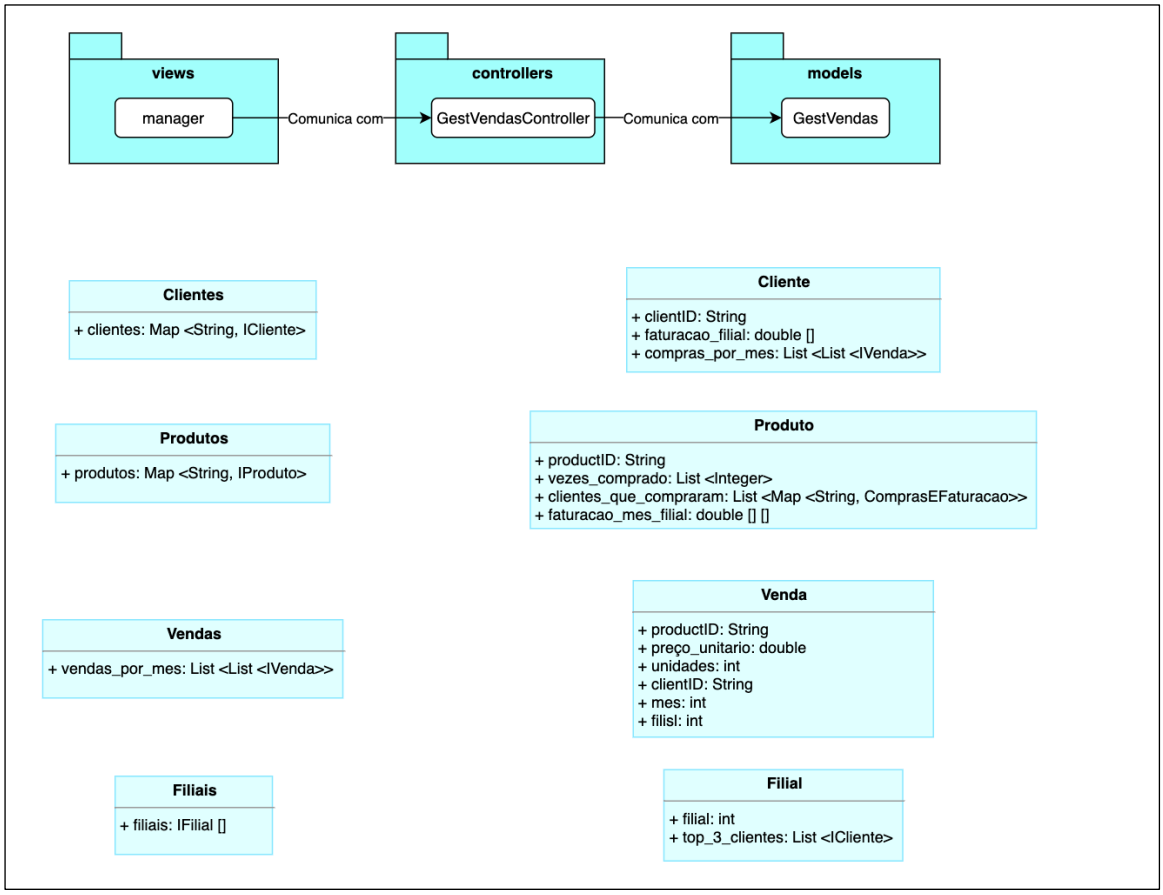

Figura 2 – Estrutura de dados e respetiva comunicação

Views

Este *package* inclui o código da componente que realiza a interação com o utilizador.

Para além das classes que se considera redundante referir (nomeadamente, **Colors**, **Input** e **Meses**), existem duas classes principais, sendo estas, o **Manager** e o **Navegador.**

O **Manager** imprime para o ecrã tanto os resultados como as exigências relativas às *queries* enunciadas.

O **Navegador** trata-se de uma classe auxiliar que permite imprimir grandes massas de resultados sob a forma de um catálogo navegável.

O **NavegadorQuery10** foi gerado com o intuito de dar resposta à *query* 10, que solicitava a apresentação de uma tabela de um produto por página, aquando da procura por informações relativas a este *query* no catálogo.

A ***View*** salienta preocupações, nomeadamente a usabilidade e também ser visualmente agradável, sendo estas tidas em consideração na elaboração da mesma.

Controllers

Dado que o objetivo do *controller* pode ser unificado em apenas uma classe, decidiu-se criar o **GestVendasController**, cuja função é encaminhar os pedidos para o **GestVendas**.

Models

Este *package* é formado pelas classes principais (responsáveis pelas operações matemáticas).

O ***Model***, dado o seu cariz lógico e matemático, é completamente independente da ***View*** e do ***Controller***.

Para que o sistema se tornasse ágil e adaptável a novas eventuais mudanças no futuro foi imprescindível a criação de interfaces associadas a todos os models. Deste modo,

aquando de uma pequena ou grande alteração, não haverá a necessidade de modificar grandes quantidades de código, possibilitando assim uma mais facilitada evolução do sistema. As interfaces são uma forma de descrever a funcionalidade dos métodos da classe respetiva.

Clientes

Esta classe visa responder à necessidade da existência de um espaço que contenha todos os clientes de um sistema. Para esse efeito, foi criada um *HashMap* que permite, de forma eficiente, o armazenamento e eventual pesquisa de cada cliente.

Cliente

Esta classe foi criada devido à inevitabilidade da existência de uma estrutura capaz de incluir as informações de cada cliente. Como mencionado anteriormente, cada cliente encontra-se armazenado num *HashMap* existente na classe Clientes, uma vez que é necessário existir um identificador de cliente (*key*) e o cliente em si (*value*). Para além do identificador, a classe Cliente tem a faturação separada por filiais (*array* de *doubles*) e as compras separadas por mês (lista de doze listas de vendas).

Produtos

Esta classe foi elaborada com o intuito de ser viável o acesso e armazenamento, de forma eficiente, de todos os produtos que coexistem no sistema. A ideia concebida passa também pela utilização das vantagens inerentes aos *HashMaps,* assim foi criado um *map* que em cada posição tem guardado um identificador do produto (*key*) e o produto em si (*value*).

Produto

Esta classe foi gerada para armazenar a informação de cada produto, designadamente o seu identificador, o número de vezes que foi comprado por mês (lista de inteiros), os clientes que compraram esse produto por mês (lista de doze *maps* que contêm o

identificador do cliente e a classe auxiliar **ComprasEFaturacao**) e a faturação por mês e por filial (*array* de *arrays* de *doubles*).

Vendas

Esta classe foi elaborada para guardar as vendas separadas por meses, no sentido de melhor responder a certas *queries*.

Venda

Esta classe foi gerada para armazenar a informação de cada venda, designadamente o identificador do produto, o preço unitário, a quantidade (em unidades), o identificador do cliente, o mês e filial da venda efetuada.

Filiais

Dado que esta classe não foi necessária para melhorar a estrutura a nível de armazenamento da informação foi criada com o intuito de responder diretamente à *query* 7, por isso tem-se apenas um *array* de classes Filial.

Filial

Como foi mencionado anteriormente, a classe Filiais, e consequentemente, a classe Filial foram geradas para responder apenas a uma *query*, pelo que esta classe apenas inclui o número que identifica a filial e uma lista dos top três clientes nessa filial.

BenchMarking e Medidas de Performance

	Vendas_1M.txt	Vendas_3M.txt	Vendas_5M.txt
Leitura	0.117651	0.279174	0.463682
Parsing e Validação	2.02304	5.18071	8.71763
Carregamento Completo	5.829572173	29.098460092	58.327389854
Query 1	0.0190709	0.0124717	0.0115378
Query 2	0.00216625	0.00218109	0.00213755
Query 3	0.000817133	0.000939405	0.000918986
Query 4	0.0000140912	0.0000172709	0.0000122336
Query 5	0.000300096	0.000579303	0.000670082
Query 6	0.43366	1.27041	1.89599
Query 7	0.000178965	0.000159049	0.000159738
Query 8	0.165865	0.495009	0.82299
Query 9	0.000277911	0.000406203	0.000384833
Query 10	0.297347	0.422732	0.297546

Figura 3 - Tempos de execução em segundos

Em termos de estruturas de dados, as decisões tomadas tiveram em consideração não só o tempo de execução, como também a agilidade para a procura e manipulação de dados. Neste sentido, foram implementados *HashMaps* nas estruturas devido à sua rapidez de resposta.

Aquando da resolução das *queries* foi determinado que seria necessária a utilização de *TreeMaps* (com tempos de procura superiores aos HashMaps). Nos casos em questão, dado que é pedido a apresentação de resultados ordenados, a utilização de *TreeMaps* identificou-se como sendo a melhor opção, ainda que se note uma diferença considerável nos tempos de execução das *queries* onde estas estruturas são utilizadas.

Numa generalidade, a utilização de determinados tipos de *maps (TreeMaps* ou *HashMaps)* foi justificada com base nas exigências da estrutura desenvolvida.

Os tempos apresentados traduziram-se no resultado da média de dez medições cada (utilizando a classe **Chrono** fornecida pelos docentes). Isto permitiu que os resultados obtidos fossem mais fidedignos.

Conclusão

Em virtude do que foi mencionado, o grupo considera que o sistema gerado é capaz de responder, de uma forma eficiente, a uma panóplia de *queries* que visam o bom funcionamento deste.

Durante a realização do projeto, a principal inquietação esteve presente na busca incessante pela eficiência, servindo esta de fio condutor para diversas decisões.

Ao longo da implementação de todas as classes foram sendo respeitados os requisitos propostos, a organização e encapsulamento de informação.

Dados os desafios apresentados, foi encontrada uma maior dificuldade na elaboração de uma estrutura de dados que pudesse ser, simultaneamente, versátil e rápida na resposta às *queries*. Optou-se, portanto, pelo meio termo entre estas duas qualidades, permitindo que o tempo de resposta fosse favorável para um maior número de *queries*.

Como aspetos a melhorar, seria principal foco diminuir os tempos de execução, bem como realizar mudanças na estrutura, de modo a torná-la mais flexível. Ainda que, houvesse uma especial atenção debruçada sobre este tópico é garantido que hajam melhorias a efetuar.

Este projeto revelou-se bastante enriquecedor para todos os elementos do grupo, uma vez que incentivou não só a aprendizagem e manuseamento relativo a estruturas de dados, como também mitigou a capacidade de autonomia nas decisões tomadas.

Em suma, compreende-se a necessidade da existência de Sistemas de Gestão de Vendas, nomeadamente para empresas, sendo uma forma extremamente rápida de pesquisar por todo e qualquer género de informação que aí se encontre. Ainda que, o sistema desenvolvido responda a uma pequena quantidade de requisitos possíveis, tentou-se ao máximo que este fosse facilmente estendido.