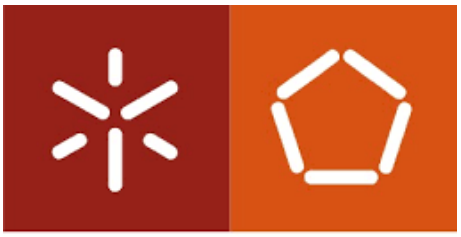
Relatório Projeto Java
Laboratórios de Informática III

Grupo 87:
António Luís de Macedo Fernandes (a93312)
José Diogo Martins Vieira (a93251)
João Silva Torres (a93231)

June 28, 2021

Universidade do Minho

Contents

1 Introdução

No âmbito do desenvolvimento do segundo projeto da unidade curricular Laboratórios de Informática III foi-nos proposto desenvolver, em java, um programa que realize a gestão de reviews. Este tem de ler três ficheiros de grandes dimensões: users.cvs, business.cvs e reviews.cvs.

Dito isto, pretende-se desenvolver uma aplicação que seja capaz de ler e armazenar em estruturas de dados (coleções de Java) adequadas às informações dos vários ficheiros, para que, posteriormente, possam ser realizadas diversas consultas (queries), algumas estatísticas e alguns testes de performance.

Ora, para que tal seja possível tivemos de introduzir e por em prática novos princípios de programação que nos foram ensinados nas aulas ao longo deste semestre como: o encapsulamento; a programação com interfaces, ou seja, criando-se antes das classes as APIs, assim visando tornar a aplicação mais genérica e flexível.

Em seguida abordaremos de forma sintetizada os diferentes módulos do trabalho.

Por fim, na última secção deste relatório iremos falar um pouco sobre os testes de performance.

2 Estratégias Utilizadas

O trabalho proposto foi realizar um Sistema de Gestão de Recomendações, capaz de armazenar e responder a diferente queries sobre a informação.

Sendo que neste semestre, já tinhamos feito um trabalho com o mesmo objetivo mas a linguagem C. Para este projeto foi mais fácil o desenvolvimento da estrutura, pois reutilizamos estratégias usadas no projeto de C, tais como:

- Organização dos Business e Users em HashMap

- Organização da Info sobre os Business e Users em HashMap

As principais diferenças deste trabalho em relação ao projeto anterior, passa pela utilização da data da Review para resolver algumas das queries.

Primeiramente, fizemos associar o id da Review à sua review. Tínhamos depois uma estrutura TimeStruct que guardaria a informação relativamente a:

- todos os meses de cada ano

- todos os anos

- todos os meses independente do ano

Esta estrutura apesar de tornar o tempo de execução das queries associados ao tempo muito rápidas, ocuparia muita memória, e tornaria o processo de leitura do ficheiro de Reviews demorado.

Para combater este uso excessivo de memória e de tempo de leitura demorado, decidimos repensar a forma como estávamos a guardar as Reviews.

Percebemos que não era necessário guardar as Reviews associados ao seu ID. Mudámos assim o Cátalogo das Reviews para associar a cada Ano um Mapa. Esse Mapa associaria cada mês a uma lista de Reviews.

Assim descartamos a nossa ideia original da TimeStruct e substituímos por esta abordagem. Os resultados da comparação destas duas abordagens relativamente ao uso de memória, tempo de leitura e tempo de execução das querys será detalhado mais à frente.

3 Classes do Trabalho

3.1 Model

3.1.1 BusinessInterface

Esta interface representa um business (negócio) e contém os métodos de acesso de informação do negócio.

3.1.2 BusinessCatalogInterface

Esta interface representa um catálogo de businesses que contém métodos de manipulação deste tipo de catálogo e ainda um método que retorna uma lista de ids de negócios não avaliados.

3.1.3 ReviewInterface

Esta interface representa uma review e contém os métodos de acesso de informação da review.

3.1.4 ReviewCatalogInterface

Esta interface representa um catálogo de reviews e contém métodos de inserção, verificação e métodos para a realização de algumas queries.

3.1.5 UserInterface

Esta interface representa um user e contém os métodos de acesso de informação do user .

3.1.6 UserCatalogInterface

Esta interface representa um catálogo de users e contém métodos de manipulação deste tipo de catálogo.

3.1.7 InfoInterface

Esta interface é utilizada para representar informação de um tipo de dados no qual podemos ir buscar o número total de reviews, a nota total, a nota média e o número de ids distintos. Para além disso, contém um método que ordena por ordem descrescente o map com os ids conforme o número de reviews e para tal definimos a classe ParReview. Esta classe (ParReview) é auxilar e serve para guardar o números de reviews e a nota total associada.

3.1.8 InfoStatsInterface

Esta interface representa um catálogo que armazena infos, contendo metódos de manipulção do mesmo e metódos para a realização das queries

3.1.9 StatsInterface

Esta interface representa as diferentes estatísticas pedidas contendo metódos de acesso às mesmas.

3.1.10 StatsStructInterface

Esta interface representa a estrutura responsável por guardar a informação devidamente estruturada para a realização das queries, contendo metódos para a realização das mesmas.

3.1.11 GestReviewsInterface

Esta interface representa a estrutura principal do nosso programa que fica responsável por gerir toda a informação contendo assim os vários métodos de leitura de ficheiros e da realização de todas as queries.

3.2 Queries

3.2.1 Query 1

Para esta query a estratégia utilizada foi percorrer todos os negócios do catálogo, verificar se já foi avaliado, adicionar o respetivo id num TreeSet (ficando assim ordenado alfabeticamente) e devolvendo um arraylist conforme o set anteriormente explicado.

3.2.2 Query 2

Nesta query recebemos como argumento o ano e mês introduzidos pelo utilizador, de seguida, iremos ao catálogo das reviews buscar a lista das Reviews desse respetivo ano e mês. Devolvemos uma lista de inteiros em que o primeiro int será o numero global de reviews feitas(size da lista de reviews) e o outro o número total de users distintos que as realizaram (size do set criado).

3.2.3 Query 3

Nesta query, dado como argumento o user_id, começamos por armazenar num array de Infos a informação das reviews por mês do dado user. Para isso, percorremos cada mês de cada ano do catálogo de reviews e e por cada review que apresentar o mesmo user_id que a revisou, atualizamos o nosso array de infos. De seguida definimos uma classe auxiliar chamada Query3Ouput que armazena o número total de reviews, a nota média e quantos negócios distintos avaliou de cada info do nosso array.

3.2.4 Query 4

A estratégia para esta query é idêntica à query 3, o que difere é que em vez de ser o user_id como argumento é um business_id.

3.2.5 Query 5

Para esta query, fomos buscar a info associada a um user_id e retornamos uma lista de ParIdCount na qual organizámos por ordem decrescente de quantidade e, para quantidades iguais, por ordem alfabética dos nomes dos negócios.

3.2.6 Query 6

Nesta query, dado o número introduzido pelo utilizador, vamos determinar o conjunto dos n negócios mais avaliados (com mais reviews) em cada ano, para isso devolvemos um mapa em que fazemos associar a cada ano uma lista de ParIdCount que contém o id do negócio e o número total de distintos utilizadores que o avaliaram.

3.2.7 Query 7

Para esta query, vamos determinar os top 3 negócios com mais reviews em cada cidade e para tal definimos uma classe auxiliar (Query7Output) onde vamos associar a cada cidade uma lista de ParIdCount que contém os ids dos top 3 negócios e o número de reviews feitas associado.

3.2.8 Query 8

Para esta query, dado o número introduzido pelo utilizador, vamos retornar uma lista de ParIdCount que contém os n user_ids e o número de negócios distintos avaliados associado.

3.2.9 Query 9

Dado um business_id e um número introduzidos pelo utilizador, determinamos o conjunto dos n users que mais o avaliaram e, para cada um, qual o valor médio de classificação. Para tal, definimos uma classe auxilar (TripleIdMediaCount) em que associamos a cada id do user o número de reviews e a média de classificações.

3.2.10 Query 10

Nesta query vamos percorrer todos estados e cidades e retornamos um mapa que associa a cada key (estado) um mapa que faz associar a cada cidade uma lista de TripleIdMediaCount que associa a cada id do negócios a média de classificação e quantos negócios foram feitos.

3.3 Utils

3.3.1 Crono

Esta classe foi dada pelos professores e é utilizada para calcular os vários tempos de execução.

3.3.2 Define

Aqui estão definidas as constantes e Macros utilizadas no nosso programa.

3.3.3 Paginacao

Esta classe é utilizada para fazer a paginação dos outputs das diferentes queries.

3.4 View

Nesta classe encontram-se definidos metódos de impressão no ecrãn, nomeadamente das diferentes queries e dos menus do programa.

3.5 Controller

Neste módulo encontra-se o nosso controlador e este é responsável pela interação do utilizador com o nosso programa.

3.6 GestReviewApp

Nesta classe encontra-se a main do nosso programa, que começa por ler os ficheiros e apresentar o tempo e memória gasta e de seguida manda correr o controlador.

4 Testes De Performance

Para os Testes de Performance, decidimos comparar as nossas duas diferentes abordagens para o guardar de informação associada à data das Reviews.

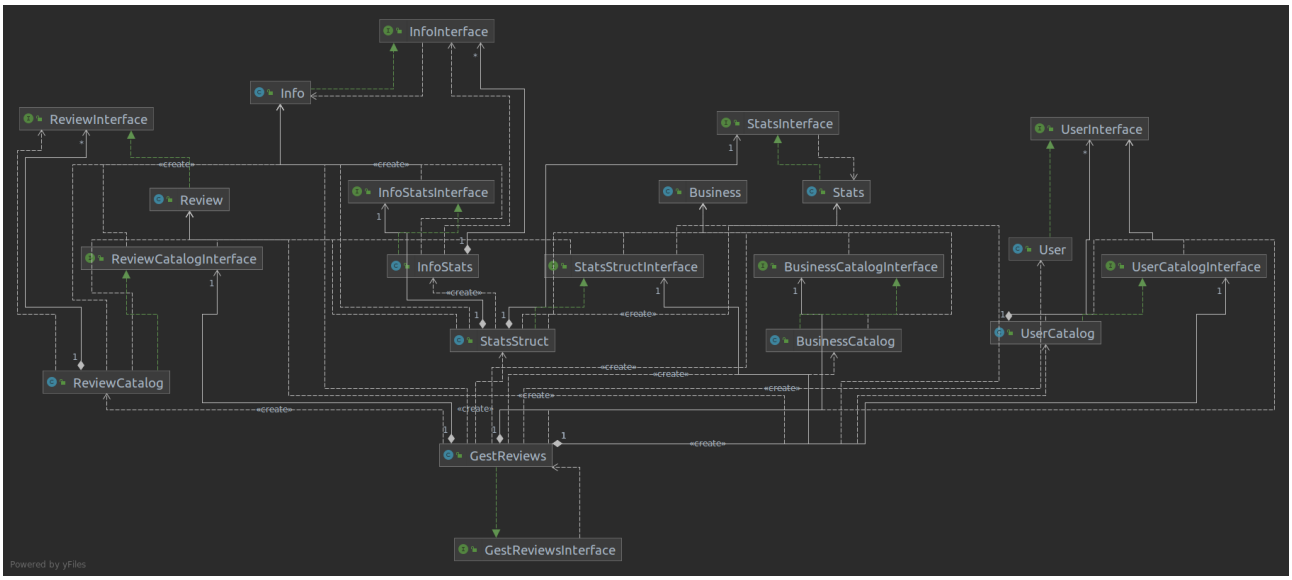
Os testes foram realizados num ambiente Linux, Ryzen 3700U (Processador), 8GB RAM, um disco SSD. Apresentamos o resultado da média de 5 execuções.

-	S/ Time Struct (atual)	C/Time Struct (antiga)
Ler Business:	9.0.670 s	0.690 s
Ler Users:	0. 9.483 s	10.103 s
Ler Reviews:	6.040 s	14.001 s
Memória:	1.532 GB	2.496 GB
Query 2:	6.37×10^{-3} s	8.807×10^{-4} s
Query 3:	5.33×10^{-1} s	9.453×10^{-4} s
Query 4:	8.89×10^{-2} s	1.99×10^{-5} s
Query 6:	6.03×10^{-1} s	6.47×10^{-2} s

Verificamos que na abordagem com a TimeStruct apesar de haver uma melhoria no tempo de realização das querys, a quantidade de memória a mais necessária era demasiada. Tal como tempo de execução na leitura do ficheiro de Reviews que aumenta significativamente.

Com estes resultados, optamos pela abordagem sem a TimeStruct, que apesar de perder um bocado no tempo das querys, ganha muito na memória e na leitura do ficheiro de Reviews.

5 Diagrama de Classes



6 Conclusão

Nesta fase final do projeto constatamos que cumprimos tudo o que nos foi pedido e desenvolvemos este trabalho de forma simples e eficiente, repeitando o encapsulamento.

Tivemos tempos de execução bastante satisfatórios, o que nos leva a crer que, de uma forma geral, tivemos um bom aproveitamento.

Concluindo, este trabalho ajudou-nos a consolidar e desenvolver novas aptidões em programação, nomeadamente em Java.