Laboratórios de Informática III
Trabalho Prático

Sistema de Gestão de Recomendações - Java

Universidade do Minho
Junho 2021

Bernardo Saraiva a93189
José João Gonçalves a93204
Rui Moreira a93232

I. Introdução

Este projeto surgiu no âmbito da cadeira de Laboratórios de Informática III, e tem como objetivo fundamental ajudar na consolidação experimental dos conhecimentos teóricos e práticos adquiridos até ao momento (tanto em Programação Orientada aos Objetos como em LI3) na linguagem de programação Java.

Para tal, foi-nos proposta a criação de uma aplicação em Java baseada na utilização das interfaces e das Collections de Java, cujo objetivo é que seja capaz de ler e armazenar em estruturas de dados a informação, para que posteriormente haja a possibilidade de realização de consultas iterativas, consulta de estatísticas e testes de performance relativas à gestão básica de um Sistema de Gestão de Recomendações.

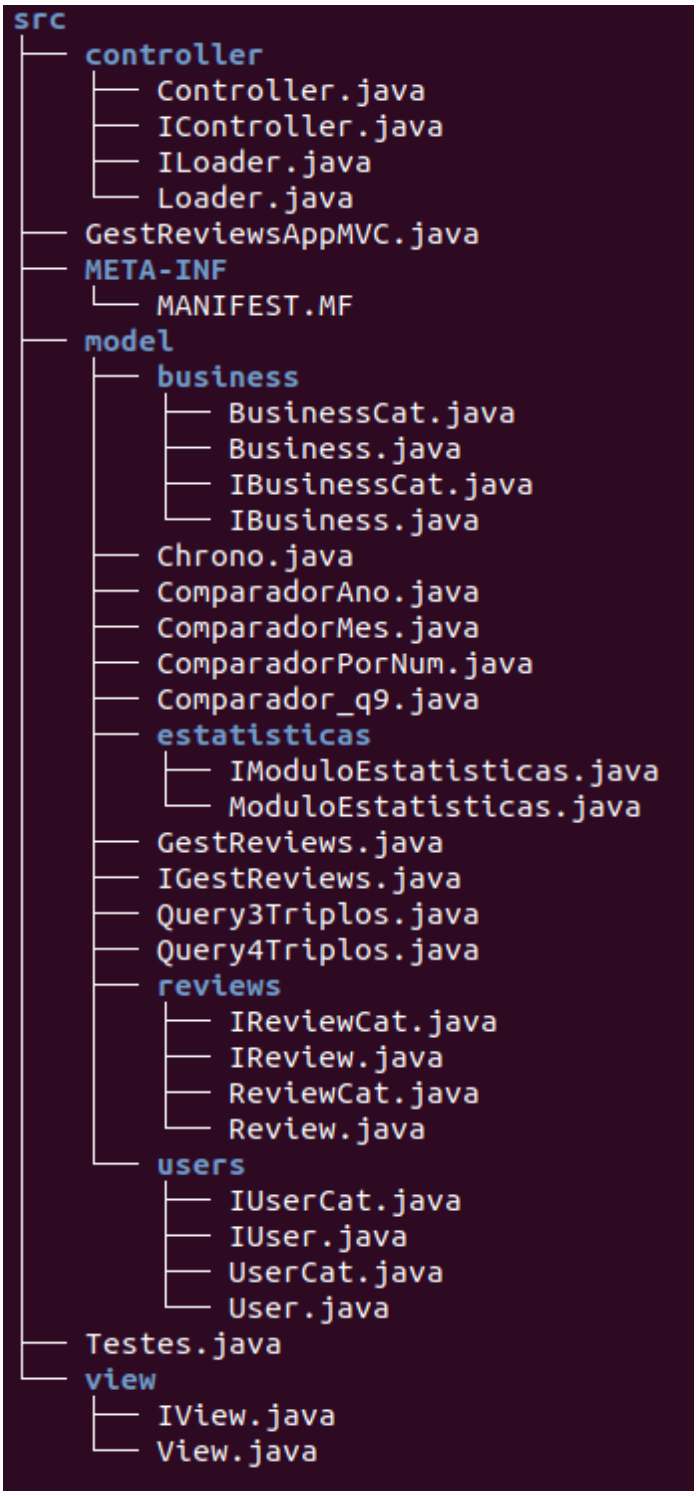
Deste modo, foram utilizados três ficheiros de teste, sendo um relativo aos negócios, o segundo relativo à informação dos utilizadores, e o terceiro às reviews feitas, que incluem toda a informação que é tratada neste projeto.

Para cumprir o objetivo proposto com o maior rigor possível, o grupo tentou ter a máxima atenção relativamente às técnicas de encapsulamento, abstração e capacidade de evolução controlada (expansibilidade), pelo que foram criadas várias classes e passamos a um breve resumo e apresentação das mesmas, assim como as estratégias utilizadas.

Para todas as classes foi criada uma interface, com vista a tornar a aplicação mais genérica e flexível, cujo nome começa por ***I***, seguido do nome da classe correspondente. **Ex: ILoader.**

II. Apresentação e descrição da arquitetura

Na realização deste projeto foi utilizado o modelo de arquitetura MVC, separando em 3 packages: **view**, **controlle**r e **model**, sendo este último dividido pelo package das estatísticas, reviews, users e businesses. Demonstramos em seguida a arquitetura do projeto.



Módulo Controller

1. Controller

O controller conhece o modelo e a view e estabelece a ligação entre eles. Aqui, é tratada a resposta às queries (não é o código das queries), assim como o tempo despendido em cada uma.
Também foram implementados neste módulo os métodos de leitura de ficheiros binários e de texto.

2. Loader

Neste módulo, temos os métodos de carregar e fazer o parsing dos três ficheiros de entrada, que fazem a leitura linha a linha, fazem a validação da informação, separam as informações pelo carácter de separação “;” e chamam o método de adicionar ao catálogo que está no model. Para além disto, ainda calculam as estatisticas à medida que a leitura dos dados é processada. Estes métodos são o **loadBusinesses**, **loadReviews** e **loadUsers**.

Este processo é feito de modo similar para os users, os businesses e para as reviews.

Também está presente o método que carrega os dados através dos ficheiros binários, com o nome de **binaryLoader.**

Módulo View

O módulo da view é a representação das informações e das opções que são mostradas ao utilizador da aplicação, pelo que grande parte do que deixamos neste módulo são métodos que utilizam ‘Sistem.out.println’ ou validações de comandos.

Módulo Model

Este módulo é o principal componente da arquitetura disposta, sendo a estrutura dinâmica da aplicação, independente da interface e manuseia a informação, lógica e regras da aplicação.

Dentro deste package, foram criados quatro packages, sendo um para reviews, outro para users ,outro para reviews e por fim um para as estatísticas.

Dentro de cada um destes, temos uma estrutura semelhante, sendo compostos por uma classe correspondente ao tipo de informação e uma classe que compõe o catálogo desse tipo de informação à exceção das estatísticas que não possui nenhum catalogo.
Para cada uma das classes Users, Review e Business foi redefinido o método de hashCode para que a manipulação do catálogo possa ser mais eficaz e forneça um melhor desempenho.
Passamos a explicar em seguida com mais detalhe.

Business, Review e Users

Business- esta classe tem com variáveis de instância todas as informações que o ficheiro dos businesses fornece, sendo elas o business id, o business name, a cidade, o estado e a lista de categorias (optamos por recorrer a uma lista de strings, pelo que cada categoria é um elemento da lista).

User- esta classe conta com o user id, o user name e a lista dos amigos como variáveis de instância.

Reviews- esta classe tem como variáveis de instância o id da review, o id do user que fez a review, o id do negócio que foi avaliado, o número de estrelas atribuído, a classificação como useful, funny e cool, a data da review e o texto associado à review.

Para cada um destes tipos de informação há um catálogo correspondente que armazena toda a informação.

BusinessCat- neste catálogo decidimos recorrer a um Map para armazenar a informação em que a chave é o business id (diferente de todos os restantes) e o value é o próprio Business. Foram também desenvolvidos método de remover e de adicionar ao catálogo, sendo um deles para adicionar um Business ao catálogo (map) e outra que recebe as informações de um business tal como são lidas pelo loader (em string) e formata para o tipo de dados que o Business aceita. Posteriormente, recorre ao construtor parametrizado de Business para gerar um novo business e adiciona ao catálogo. Este ultimo método permite reforçar o encapsulamento e a abstração.

ReviewCat- tal como catálogo anterior, decidimos recorrer a um Map para armazenar a informação, sendo a chave o id da review (valor único) e o value é a própria review. Tal como no anterior, foram também desenvolvidos os métodos de adicionar uma Review, adicionar através das informações provenientes do loader e remover de um catálogo, para que possamos manter a abstração e o encapsulamento.

UserCat- neste caso, também recorremos a um Map para armazenar a informação dos utilizadores, sendo que a chave é o id do utilizador e o value é o próprio User. Os métodos de remover e adicionar (ambos) também foram implementados.

Estatísticas

Como o próprio nome sugere, este módulo é o responsável por armazenar os dados referentes às estatísticas pedidas no enunciado. Para aumentar a eficiência de toda a aplicação, o cálculo de todos estes dados é realizado aquando do carregamento dos catálogos. Nas funções do módulo Loader, responsável pelo carregamento em memória

central toda a informação são chamadas funções deste módulo de forma a que a atualização das estatísticas seja possível. A maior parte das estatísticas é calculada através do incremento direto das variáveis do módulo por exemplo : o total de users e negócios.
Porém existem estatísticas calculadas de outra forma, como é exemplo da classificação por mês em que o somatório das estrelas é calculado a nível mensal e apenas no fim de todas as reviews serem calculadas é que a média de facto é calculado dividindo a soma obtida pelo número de reviews mensais.

Queries (GestReviews)

As queries são um conjunto de consultas interativas que permitem operar sobre a estrutura de dados. Passamos então a explicar as estratégias que adotamos para cada query:

Query 1- Esta query permite obter a lista dos negócios que nunca obtiveram nenhuma avaliação, ordenada alfabeticamente. Para tal, adicionamos todas as reviews a uma List e utilizamos o método Collections.sort para ordenar essa mesma lista alfabeticamente e de maneira "*case insensitive*".

Query 2- A partir de uma mês e um ano fornecidos pelo cliente, esta query fornece o nº total de reviews realizadas e o nº total de utilizadores distintos que as realizaram, nesse mesmo mês desse ano. O método desenvolvido para esta query recebe como argumentos o mês e o ano como inteiros e, através de um for-each loop, percorre o Catálogo das reviews (Map) verificando as que se enquadram no mês e data pedidos. Nos casos em que se verifica, é incrementado o nº total de reviews e adicionado a um Set (uma vez que não permite elementos repetidos) para verificar o nº de users distintos. Finalmente, é devolvido um par que tem o nº total e o tamanho do Set, que corresponde ao nº total de users distintos a fazer as reviews.

Query 3- Para um dado id de utilizador, esta query propõe que se determine o número de reviews feitas, o nº de negócios distintos avaliados e a nota média atribuída para cada mês.
Começamos por percorrer o entrySet do catálogo de reviews e verificar se foi feita pelo utilizador pretendido ou não (caso não seja não tem interesse). Em seguida, caso seja o utilizador pretendido, é procurado no Map chamado **res** o nº de reviews para aquele mês e a soma da nota, que estão armazenados num **Query3Triplos**. Esta classe foi criada pelos alunos para poder armazenar o trio de informação que é pedida neste exercício, sendo que tem como variáveis de instância estas informações e é apenas necessário fazer o getter para essa informação. Finalmente, esta informação é ordenada para ser apresentada pela ordem dos meses, uma vez que num Map a informação não consegue ser ordenada pela ordem pretendida através de um comparador criado para ordenar por mês - **ComparadorMes**.

Query 4- É-nos pedido nesta query que seja calculado para cada mês o total de avaliações, o número de utilizadores diferentes e a nota média da classificação. Para atingir este objetivo, procedemos de modo similar à query anterior, sendo que utilizamos um Map para armazenar toda a informação e uma classe **Query4Triplos** que atua da mesma maneira que a classe usada para a anterior, mas apenas muda a informação que é guardada. Também foi usado o Set para ordenar a informação por mês.

Query 5- Esta query devolve um conjunto de pares, constituídos pela quantidade de vezes que o utilizador x avaliou o negócio, bem como, o nome do negócio avaliado. Para a organização da estrutura devolvida foi criado um comparador (q5-comparator) que organiza o conjunto como o enunciado sugere. Inicialmente, a estrutura dos reviews é percorrida de forma a obter um HashMap em que a chave contém o businessID e o valor contém o número de vezes que este foi avaliado pelo utilizador. Por fim, percorremos esta estrutura recentemente criada e adicionamos o par chave/valor ao conjunto de retorno, substituindo o businessID pelo business name. Por fim,

Query 6- Para chegarmos ao resultado pedido, nesta query começamos por criar uma estrutura auxiliar que guarda num HashMap todas as reviews, organizando as mesmas por ano e atribuindo a cada negócio a lista de utilizadores que o avaliou. De seguida, a informação sobre cada negócio é convertida num conjunto ordenado de forma decrescente (através do comparador q6) pelo número de utilizadores que fizeram reviews. Cada item desta nova estrutura é um par que contém o nome do negócio e o número de vezes que foi avaliado. Tornando assim possível retirar os x mais avaliados, o que corresponde a retirar os primeiros x elementos. Por fim, basta percorrer estes elementos recentemente selecionados e calcular a quantidade de users distintos que lhe fizeram reviews, para isso basta adicionar a lista de users inicial a um set e no fim calcular o seu tamanho.

Query 7- Inicialmente começamos por construir um HashMap em que a cada cidade corresponde outro HashMap, sendo que a cada negócio dessa cidade corresponde o número de reviews. Depois é só percorrer o HashMap e para cada cidade apenas deixar os três negócios mais famosos em termos de número de reviews.

Query 8- Para esta query começamos por criar um HashMap que a cada user está associado o número de businesses diferentes aos quais deu review (percorrendo o catálogo das reviews) . Posteriormente é convertido para um Set ordenado de Map.Entry e depois é apenas necessário devolver os X (dado pelo utilizador) utilizadores que avaliaram mais negócios diferentes.

Query 9- Nesta query é pedido que para um business id e um número X, seja devolvido o conjunto dos X users que mais avaliaram esse negócio e o valor médio da avaliação desses users. Inicialmente, para cada review do catálogo, é criado um map das reviews do negocio cujo business id é igual ao que é passado, sendo a key é o UserId e o value é um par de total reviews e acumulado dos valores de review. Em seguida, sendo que o map referido anteriormente está completamente preenchido, a informação é passada para um novo map, sendo que neste novo map é feito o cálculo da média (indo ao par que está no value e fazendo a divisão do seu valor pela sua key). Por fim,

selecionamos apenas os X maiores para que a informação seja correspondente ao que nos foi pedido e devolvemos a informação numa List de pares.

Query 10- Para esta query optamos por, percorrendo o catálogo das reviews, criar um HashMap com os businessesID e a cada um estava associado a sua média de classificação. Depois, percorrendo o catálogo dos businesses que contém o state e city, foi só organizar por cidades e estados.

Testes

Tal como proposto, foi criado uma classe de testes, sendo que o seu objetivo passa por fornecer uma noção do tempo de execução de cada um dos pilares deste projeto, assim como a variação do consumo de memória. Apresentam-se em seguida os resultados obtidos.
É de notar que o valores podem diferir conforme os argumentos passados, sendo descritos os argumentos utilizados nos nossos testes abaixo.

Tipo de teste	Argumentos	Tempo necessário	Uso de memória
Load catálogo Business	---	0.470631033 s	107 MB
Load catálogo Reviews	---	9.996164422 s	1165 MB
Load catálogo Users	---	8.13976549 s	420 MB
Query 1	---	0.947363694 s	174 MB
Query 2	03-2009	0.39534398 s	100 MB

Query 3	User id = eLAYHxHUutiXswy-CfeiUw	0.206363114 s	102 MB
Query 4	Business id = jFYIsSb7r1QeESVUnXPHBw	0.206363115 s	103 MB
Query 5	User id = eLAYHxHUutiXswy-CfeiUw	0.367020923 s	258 MB
Query 6	Top 5	0.892436195 s	136 MB
Query 7	---	0.677119562 s	243 MB
Query 8	10 utilizadores	0.851382702 s	46 MB
Query 9	5 utilizadores e business id = jFYIsSb7r1QeESVUnXPHBw	0.197136124 s	103 MB
Query 10	---	0.652342095 s	155 MB

Conclusão

Em suma, pensa-se que este trabalho consegue alcançar todos os objetivos propostos com o devido rigor e correção, complementando os conhecimentos e estratégias adquiridos em outras cadeiras desta linguagem.