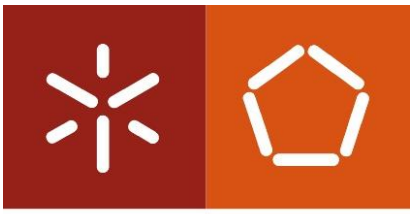


Universidade do Minho

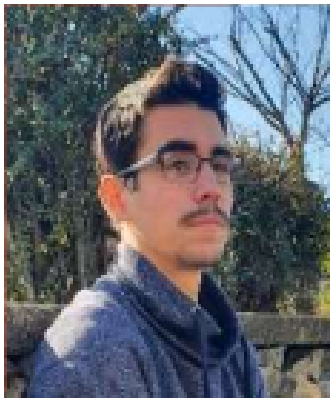
Mestrado Integrado em Engenharia Informática

LI 3 - Trabalho Prático de Java

Grupo 90

Alexandre Silva Martins (A93315)
Luís Francisco Ferreira Faria (A93219)
Tiago André Leça Carneiro (A93207)

Ano Letivo 2020/2021



Conteúdo

Capítulo 1

Introdução

Para este projeto, foi pedida a criação de uma aplicação Desktop em Java, baseada na utilização das interfaces e das coleções de JCF (Java Collections Framework), cujo objetivo é a realização de consultas interativas de informações relativas à gestão básica de um sistema de recomendação/classificação de negócios.

A aplicação deve ser capaz de, antes de mais, 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.

Capítulo 2

Classes

2.1 GestReviewsMVC

Classe responsável por arrancar o programa. Para isto o método *main* inicia a execução do Controlador e Vista;

2.2 Business

```
private final String id;
private final String name;
private final String city;
private final String state;
private Set<String> categories;
```

Business é uma das várias classes Modelo do sistema e é onde é guardada a informacao relativa aos negócios.

2.2.1 BusinessCatalog

```
private final Map<String,Negocio> catalogo;
```

BusinessCatalog é a classe onde é criado um catálogo que usa a Interface Map, onde a chave é o id do negócio e o valor é o própio negócio. Esta classe facilita a gestão dos negócios.

2.3 User

```
private final String id;
private final String name;
private Set<String> friends;
private boolean friendsParsed;
```

User é uma das várias classes Modelo do sistema e é onde é guardada a informacao relativa aos utilizadores.

2.3.1 UserCatalog

```
private final Map<String, Utlizador> catalogo;
```

UserCatalog é a classe onde é criado um catálogo que usa a Interface Map, onde a chave é o id do utilizador e o valor é o própio utlizador. Esta classe facilita a gestão dos utilizadores.

2.4 Review

```
private final String id;
private final String userId;
private final String businessId;
private float stars;
private int useful;
private int funny;
private int cool;
private LocalDateTime date;
private String text;
```

Review é uma das várias classes Modelo do sistema e é onde é guardada a informação relativa às avaliações.

2.4.1 ReviewCatalog

```
private final Map<String, Set<Avaliacao>> mapUserRev;
private final Map<String, Set<Avaliacao>> mapBusRev;
```

Review Catalog é a classe onde é criado dois catálogos que usam a Interface Map. No mapUserRev a chave é o id do utilizador e o valor são as avaliações associadas ao id, que estão ordenadas pela data. No mapBusRev a chave é o id do negócio e o valor é a avaliações associadas ao id, que estão ordenadas pela data. Esta classe facilita a gestão dos avaliações, o que por si, torna a realização das queries mais eficiente.

2.5 GestReviews

```
private final CatalogoUtilizadores uc;
private final CatalogoNegocios bc;
private final CatalogoAvaliacoes rc;
private final Estatisticas stats;
```

A classe GestReviews é a classe onde: são preenchidas as estruturas de dados, recorrendo a ficheiros de texto com utilizadores, negocios e avaliacoes; se lê e escreve os ficheiro objeto e as querries são resolvidas.

2.5.1 Query 1

Ordena e retorna os ids dos negócios não avaliados.

Para isso, obtemos a lista de ids de negócios, recorrendo ao BusinessCatalog e verificamos se é chave no mapBusRev do ReviewCatalog e devolve uma Collection de Strings.

2.5.2 Query 2

Dado um mês e um ano, retorna um par com o número total global de reviews realizadas e o número total de users distintos que as realizaram.

Para isso, recorre às estatísticas e devolve um par (AbstractMap.SimpleEntry¡Integer,Integer¿).

2.5.3 Query 3

Dado um código de utilizador, determina, para cada mês, quantas avaliações fez, quantos negócios distíntos avaliou e que nota média atribuiu.

Para isso, usando o mapUserRev do ReviewCatalog, obtemos um TreeSet com as avaliações feitas pelo utilizador e, tendo em conta que estão ordenadas pela data, percorremos-lo e criamos um QuintuploMesAnoReviewcountBusinesscountRating ao fim de cada conjunto de datas iguais.

No fim, devolve uma lista da classe QuintuploMesAnoReviewcountBusinesscountRating.

2.5.4 Query 4

Dado um código de um negócio, determina, para cada mês, quantas avaliações dele é feita, quantos utilizadores distintos o avaliaram e que nota média foi lhe atribuida.

Para isso, usando o mapBusRev do ReviewCatalog, obtemos um TreeSet com as avaliações feitas sobre esse negócio e, tendo em conta que estão ordenadas pela data, percorremos-lo e criamos um QuintuploMesAnoReviewcountUsercountRating ao fim de cada conjunto de datas iguais.

No fim, devolve uma lista da classe QuintuploMesAnoReviewcountUsercountRating.

2.5.5 Query 5

Dado o código de um utilizador determinar a lista de nomes de negocios que mais avaliou (e quantos), ordenada por ordem decrescente de quantidade e, para quantidades iguais, por ordem alfabetica dos negócios.

Para isso, usando o mapUserRev do ReviewCatalog, obtemos um TreeSet com as avaliações feitas pelo utilizador e percorremos-lo, guardando num Map, o id do negócio como chave e o seu número das suas avaliações como valor, que depois será convertido numa lista da classe PairBusNameNum.

No fim, devolve essa lista de PairBusNameNum.

2.5.6 Query 6

Dado um N (número positivo), determina o conjunto dos N negócios mais avaliados em cada ano, indicando o numero total de utilizadores distintos que o avaliaram.

Para isso, percorremos o mapBusRev do ReviewCatalog, e depois percorremos os TreeSet com as avaliações feitas sobre cada negócio e, tendo em conta que estão ordenadas pela data, criamos os Triple_BusinessId_ReviewCount_ReviewersCount que vão ser acrescentados ao valor de um Map que tem o ano como chave e um Set da classe Triple_BusinessId_ReviewCount_ReviewersCount como valor. De seguida, percorremos esse Map e a cada entry criamos uma nova entry identica mas só com os N negócios mais avaliados em cada ano e adicionamo-la a um TreeMap com um comparator, para haver uma ordenação automática pelo ano quando a inserimos.

No fim, devolve esse TreeMap onde a chave é o ano e o valor é a lista da classe

Triple_BusinessId_ReviewCount_ReviewersCount.

2.5.7 Query 7

Dado um N (numero positivo), determina, para cada cidade, a lista dos N mais famosos negócios em termos de número de avalições.

Para isso, usando o mapBusRev do ReviewCatalog, criamos uma lista de PairBusIdNum (contêm o id de um negócio e o número das suas avaliações) que juntamente com o bc do BusinessCatalog, são usados para criar um Map em que a chave é uma cidade e o valor é uma lista dos PairBusIdNum (contêm o id do business e o número das suas avaliações) que se localizam nessa cidade. De seguida, percorremos esse Map e a cada entry criamos uma nova entry identica mas só com os N negócios mais avaliados em cada cidade e adicionamo-la a um novo Map.

No fim, devolve um Map onde a chave é o nome da cidade e o valor é a lista da classe PairBusIdNum.

2.5.8 Query 8

Dado um N (numero positivo), determina os N utlizadores que mais avaliaram negócios distíntos, juntamente com os seus números de negócios distíntos avaliados.

Para isso, usando o mapUserRev do ReviewCatalog, criamos um TreeSet da classe PairUserIdBusNum (contêm o id de um utilizador e o número de avaliações que fez), que foi inicializado com um comparator que compara o número de avaliações.

No fim, devolve os primeiros N PairUserIdBusNum do TreeSet numa lista da classe PairUserIdBusNum.

2.5.9 Query 9

Dado um N (numero positivo), determina os N utilizadores que mais avaliaram um negócio específico, juntamente com os números de avaliações que fizeram e as notas média que atribuiram.

Para isso, usando o mapBusRev do ReviewCatalog, criamos um TreeSet da classe TripleUse-

rIdRevNumRating (contêm o id de um utlizador, o número de avaliações que fez e a nota média que atribuiu), que foi inicializado com um comparator que compara o número de avaliações.

No fim, devolve os primeiros N TripleUserIdRevNumRating numa lista da classe TripleUserIdRevNumRating.

2.5.10 Query 10

Determina para cada estado, cidade a cidade, os seus negócios e a média de classificação de cada negócio.

Para isso, usando o mapBusRev do ReviewCatalog, e o bc do BusinessCatalog, criamos um Map onde a chave é o nome do estado e o valor é outro Map onde a chave é o nome da cidade e o valor é lista da classe PairBusIdNum

No fim, devolve-se esse Map.

2.6 Stats

```
private final List<String> filePaths;
private int globalReviewCount;
private int wrongReviewCount;
private int nonImpactfulReviewCount;
private float globalAverageRating;
private int businessCount;
private int businessesReviewedCount;
private int businessesNotReviewedCount;
private int userCount;
private int reviewersCount;
private int nonReviewersCount;
private final Map<pairMesAno, MonthInfo> monthInfos;
```

Stats é uma das várias classes Modelo do sistema e é a classe que armazena as estatísticas dos negócios, dos utlizadores e das avaliações.

2.7 Controlador

Classe que gere o fluxo da aplicação. Esta é a classe responsável por gerir o menu do estado e o menu das queries, disponível para todas as entidades e conecta a Vista e o Modelo do sistema.

No *menu do estado* temos as seguintes opções:

1- Ler ficheiros default: Preenche as estruturas de dados recorrendo aos ficheiros de texto com utilizadores, negócios e avaliações, dados pelos professores.

2- Ler ficheiros (modo texto): Preenche as estruturas de dados recorrendo a ficheiros de texto com utilizadores, negocios e avaliacoes.

3- Ler ficheiro objeto: Lê as estruturas de um ficheiro objeto.

4- Guardar em ficheiro objeto: Escreve as estruturas para ficheiro objeto.

O Controlador comunica com o GestReviews.

2.8 Vista

A Vista comunica apenas com o Controlador.

As classes Menu (criada por José Creissac Campos) e CommandLineTable (adquirido no site https://www.logicbig.com/how-to/code-snippets/jcode-java-cmd-command-line-table.html), que é usada para criar as tabelas que são imprimidas no terminal e para fazer a paginação, são as classes usadas na Vista.

Possui vários métodos, nomeadamente para receber algum tipo de informação da entidade que usa a app ou para mostrar algum tipo de informação.

Capítulo 3

Interfaces

As Interfaces: Avaliacao, CatalogoAvaliacoes, CatalogoNegocios, CatalogoUtlizadores, Estatisticas, Negocio e Utilizador foram criadas com o intuito de tornar o código reutilizável e a aplicação mais genérica e flexível.

Capítulo 4

Testes de Performance

De forma a obter a maior consistencia possível, todas as queries foram executadas 5 vezes, utilizando os ficheiros fornecidos pelos professores.

As especificações das máquinas utilizadas foram as seguintes:

Máquina 1

Processador : Intel i5-8400

Ram : 16G DDR4

Disco : SSD

Máquina 2

Processador : Intel i7-6500U

Ram : 8G

Disco : SSD

4.1 Leitura dos ficheiros em modo texto

Nota : Durante a leitura, os amigos dos Utilizadores foram armazenados numa só string, ou seja, não foi feito o parsing dos amigos.

Máquina 1 - 29,72308 segundos

Máquina 2 - 85,89544 segundos

4.2 Leitura dos ficheiros objeto

Nota : Durante a leitura, os amigos dos Utilizadores foram armazenados numa só string, ou seja, não foi feito o parsing dos amigos.

Máquina 1 - 441,33258 segundos

4.3 Guardar em ficheiro objeto

Máquina 1 - 179,20193 segundos

4.4 Query 1

Máquina 1 - 0,19781 segundos

Máquina 2 - 1,00230 segundos

4.5 Query 2

Argumentos - jul/14

Máquina 1 - 0,000541 segundos

Máquina 2 - 0,042392 segundos

4.6 Query 3

Argumentos - RtGqdDBvvBCjcu5dUqwfzA

Máquina 1 - 0,003134 segundos

Máquina 2 - 0,0513397 segundos

4.7 Query 4

Argumentos - bZiIIUcpgxh8mpKMDhdqbA

Máquina 1 - 0,003578 segundos

Máquina 2 - 0,0377803 segundos

4.8 Query 5

Argumentos - RtGqdDBvvBCjcu5dUqwfzA

Máquina 1 - 0,050116 segundos

Máquina 2 - 0,0693387 segundos

4.9 Query 6

Argumentos - 10

Máquina 1 - 0,408508 segundos

Máquina 2 - 0,9055656 segundos

4.10 Query 7

Máquina 1 - 0,033359 segundos

Máquina 2 - 0,0988478 segundos

4.11 Query 8

Argumentos - 10

Máquina 1 - 0,478504 segundos

Máquina 2 - 1,0548869 segundos

4.12 Query 9

Argumentos - 10 — bZiIIUcpgxh8mpKMDhdqbA

Máquina 1 - 0,011066 segundos

Máquina 2 - 0,0501212 segundos

4.13 Query 10

Máquina 1 - 0,150195 segundos

Máquina 2 - 0.4485641 segundos

Capítulo 5

Conclusão

Para concluir, estamos satisfeitos com aquilo que conseguimos alcançar. Embora houvesse arestas que pudessem ser polidas, as nossas soluções para os problemas apresentados foram objetivas e eficazes, criando assim um projeto convincente.

Apêndice A

Diagrama de Classes

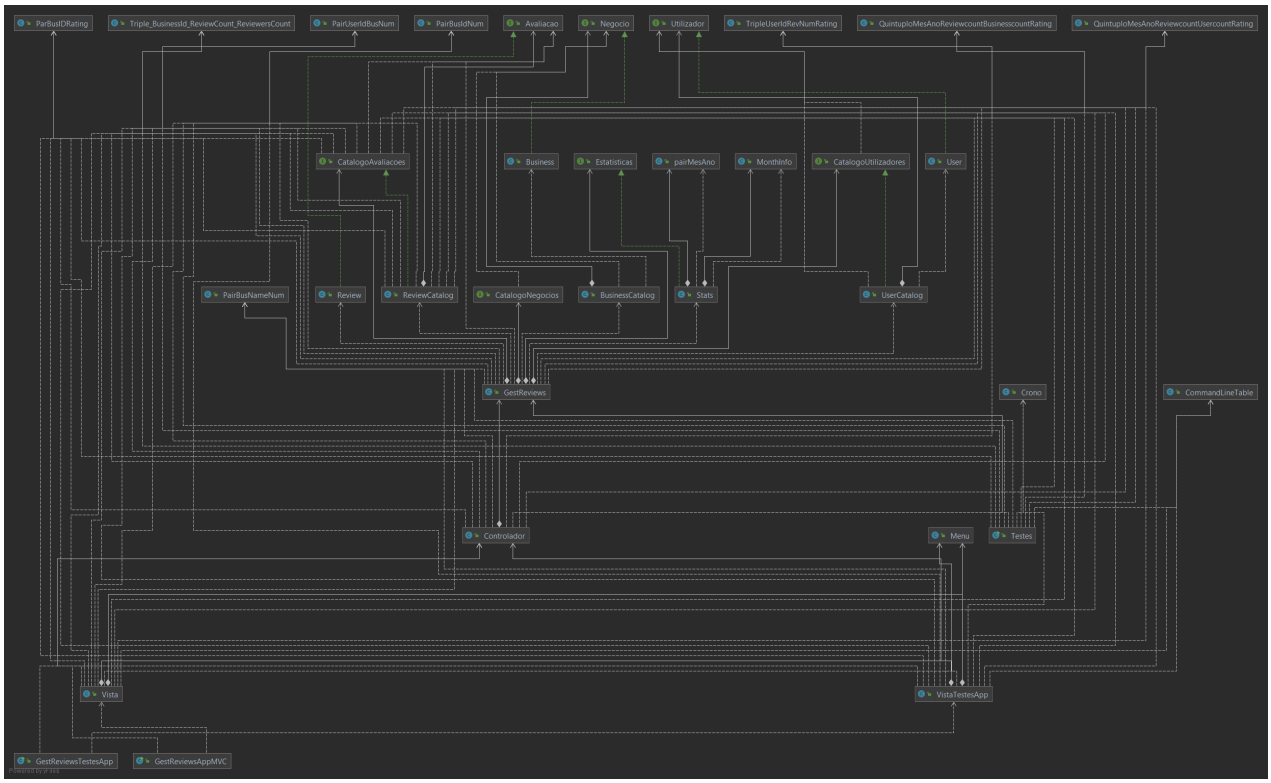


Figura A.1: Diagrama de classes do programa, gerado pelo *IntelliJ*