Laborátorios de Informática III

2º trabalho prático

Src: https://www.yelp.com/dataset

Autores:

Vasco Oliveira
Nº 93208

Diogo Matos,
Nº 93234

Miguel Fernandes,
Nº 94269

10.06.2021

MIEI/LEI

Introdução

Esta segunda fase do projeto de Laboratórios de Informática III tem como objetivo reformular a primeira fase, feita na linguagem de programação C, para Java.

Para tal foram usados os conhecimentos adquiridos ao longo do semestre na cadeira de Programação Orientada a Objetos, bem como os que foram obtidos enquanto fomos programando com a linguagem Java.

Mais uma vez, os dados utilizados referem-se à plataforma "Yelp" e podem ser obtidos através do seguinte link.

Processo

APIs presentes e sua estruturação

Descrição das diferentes consultas iterativas

1. A consulta utiliza um Set<String> que contém todos os business_id e à medida que percorre as reviews, retira deste Set os id que aparecem. No final este set é ordenado. A escolha do set foi devido a procura e remoção mais rápidas do que usar por exemplo uma lista.
2. Nesta query também foi utilizado um Set<String> pela mesma razão de ao bocado e também pelo facto de o Set não permitir repetições. Mais uma vez foram percorridas todas as reviews e sempre que estas correspondiam ao mês e ano procurados era adicionados ao Set.
3. Para a realização desta query optamos por criar uma classe auxiliar onde guardava o Set dos ids, número de Reviews e o número de estrelas. Desta forma, bastou-nos ter uma map com a key do mês e o valor desta classe, à medida que íamos iterando as reviews, atualizamos os respetivos values. O Mapa permite um acesso mais rápido e mais organizado.
4. A consulta 4 é em todo idêntica com a 3, apenas o que muda é o filtro que aplicamos em cada iteração das reviews, neste caso é business_id passado como argumento à função.
5. Para a resolução desta query optamos por usar um Map em que a chave era o business_id e a o valor era o número de vezes que este id já avaliado. Ao fim de todos os valores serem avaliados, este keyset é ordenado tendo em conta os

valores deles e também a ordem alfabética. A razão de usar o Map foi a mesma da query 3.

6. De maneira a organizar melhor os dados utilizamos um Mapa em que a chave era o ano e o value era outro Mapa em que a chave era o business_id e no value tinha o set de user_id correspondentes. A partir deste mapa derivamos uma lista de String ordenada pelo site do Set e que contém o business id e o respetivo size. Mais uma vez, este uso de Map dentro de um Mapa facilita muito os acessos e dá-nos a liberdade de organizar os dados em "gavetas", optimizando muito o tempo da query.
7. A query 7 acaba por ser muito parecida com a 6, só que temos um mapa organizado pelas cidades e nos values, mais uma vez temos um mapa em que a chave é o business_id e nos values temos o número de reviews.
8. Na query 8, repetindo o que foi feito nas querys anteriores, utiliza um mapa que depois o transforma numa lista dos keyset organizados pelos respectivos values;
9. A resolução da 9 é idêntica à 8, reforçando a ideia de que quando precisamos de organizar valores em que temos de ter em conta mais do que 1 parâmetro o mapa é a melhor solução, pelas razões já faladas anteriormente.
10. Sendo esta a query mais arrojada, acaba por ser a que também tem o método mais "complicado" porém seguindo o raciocínio adotado nas anteriores. Para a resolução da mesma dividimos o problema por partes, fazendo Maps dentro de Maps. Ficando assim com um primeiro Mapa organizado pelos estados, outro Map organizado pelas cidades em que depois temos todos os business_id e as respectivas estrelas.

Descrição das diferentes estatísticas

Para efetuar as estatísticas dos dados recebidos decidiu-se que uma estrutura de dados, a qual teria a informação mais relevante e pertinente de uma forma organizada, seria adequada.

Assim optou-se por receber o nome do ficheiro e o número de reviews erradas(embora indiretamente através do método getNumeroDeReviewsErradas da ReviewsCat) à quando da invocação de uma nova instância de método.

Para além disso, receberia-se as reviews, businesses e users na forma de ReviewCat, BusinessCat e UsersCat respetivamente e estes seriam transformados de volta num mapa com chave de id e valor da estrutura correspondente.

Esta forma de estruturamento torna o acesso aos dados bastante mais fácil.

De seguida analisam-se as diferentes estratégias para as estatísticas.

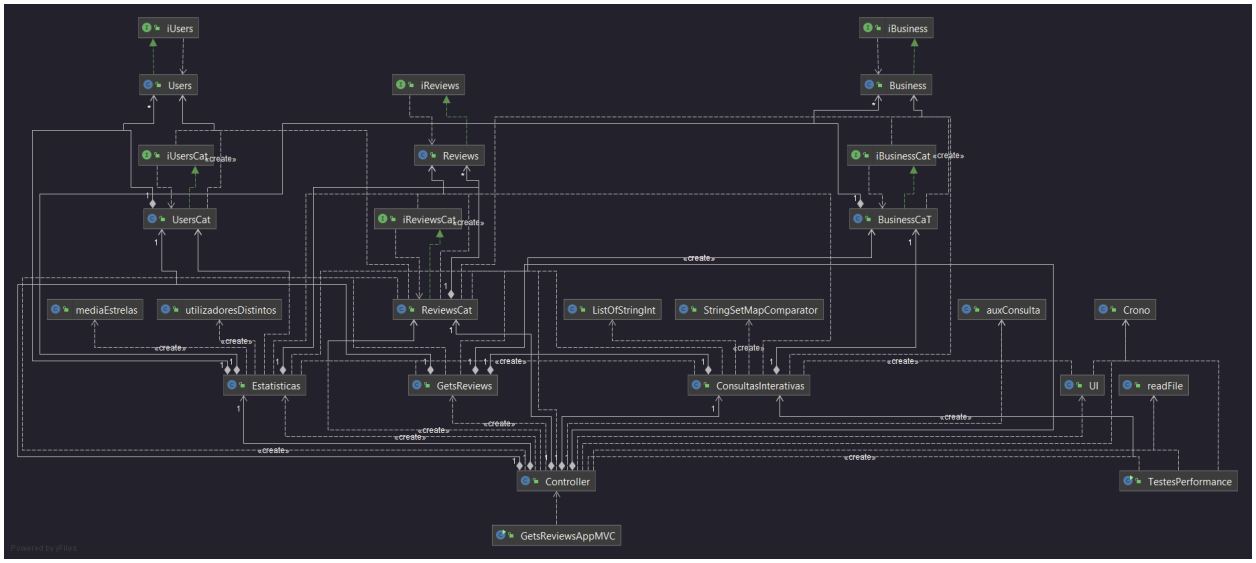
1. Como já dito, para obter o nome do ficheiro e o número de reviews erradas basta aceder aos dados do próprio método os quais foram atribuídos a quando da invocação. Para o número total de negócios e de utilizadores bastou ver o tamanho dos valores dos mapas correspondentes. Para o resto das estatísticas, estas seguem um padrão muito semelhante, o qual não vale a pena entrar muito em pormenor, mas passa em criar um mapa e no caso da existência de um id selecionado, adicionar o elemento ao mapa e incrementar um contador, este padrão foi utilizado para todas as estatísticas do ponto 1.
2. Nas estatísticas do ponto 2, optou-se por devolver um mapa com a chave com o valor do ano da Review correspondente, exceto para a média global de Reviews.
    a. Para este primeiro ponto, escolheu-se percorrer o mapa das reviews , obtendo para cada um o ano e mês de publicação, e após, caso o ano não existisse, criar um novo par chave/valor, com o ano na chave e um array de 12 posições no valor(meses do ano). Após, para cada reviews, incrementar de um valor, o ano e mês correspondente.Finalmente devolveu-se o mapa.
    b. De forma semelhante ao ponto a, criou-se um mapa contendo desta vez no valor 12 instâncias da estrutura auxiliar mediaEstrelas para cada novo ano apresentado. Por cada review percorrido, alterou-se o número de estrelas, número de reviews e média do mês correspondente. Para a média global de reviews, usou-se o mapa obtido anteriormente e simplesmente foi-se somando o número de reviews e estrelas para cada instância no mapa e finalmente dividiu-se obtendo-se a média global.
    c. De forma similar ao ponto a e b, criou-se um mapa com chave do ano da review, mas desta vez com um array de 12 utilizadoresDistintos. No qual para cada mês verificava-se se o utilizador já tinha publicado algum review percorrendo os ids já presentes, e caso não, adicionava-se ao array de ids e incrementava-se um valor ao número de utilizadores distintos, isto para cada review.

Arquitetura final da aplicação e razão para tal modularidade

Para desenvolver este projeto decidimos utilizar a estratégia MVC(Model, View, Controller) aprendida nas aulas de programação orientada a objetos. Assim, apresenta-se um diagrama da estrutura utilizada.

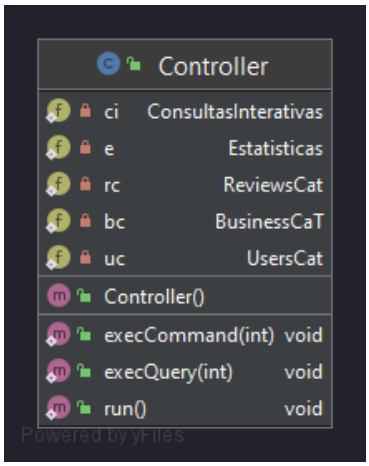
Diagrama de classes

src



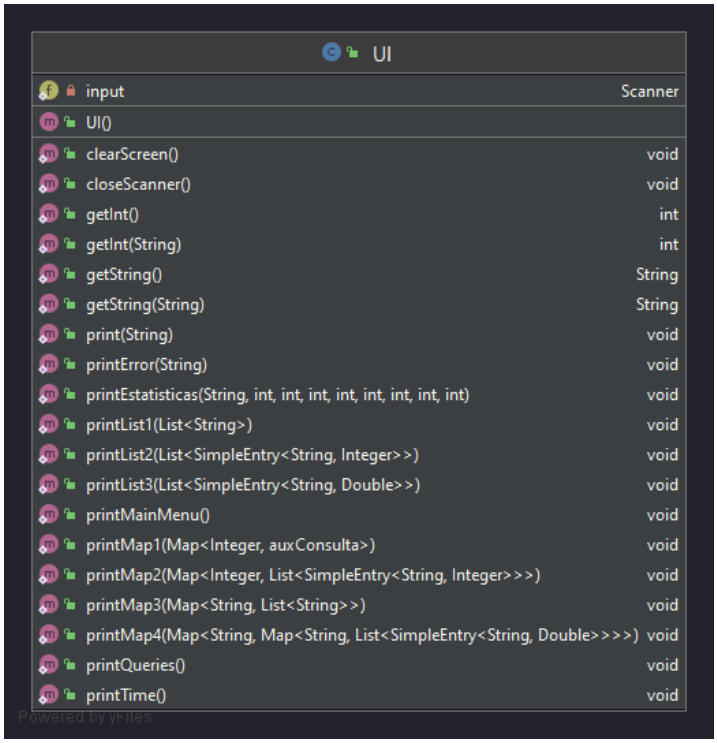
Controller

Controller



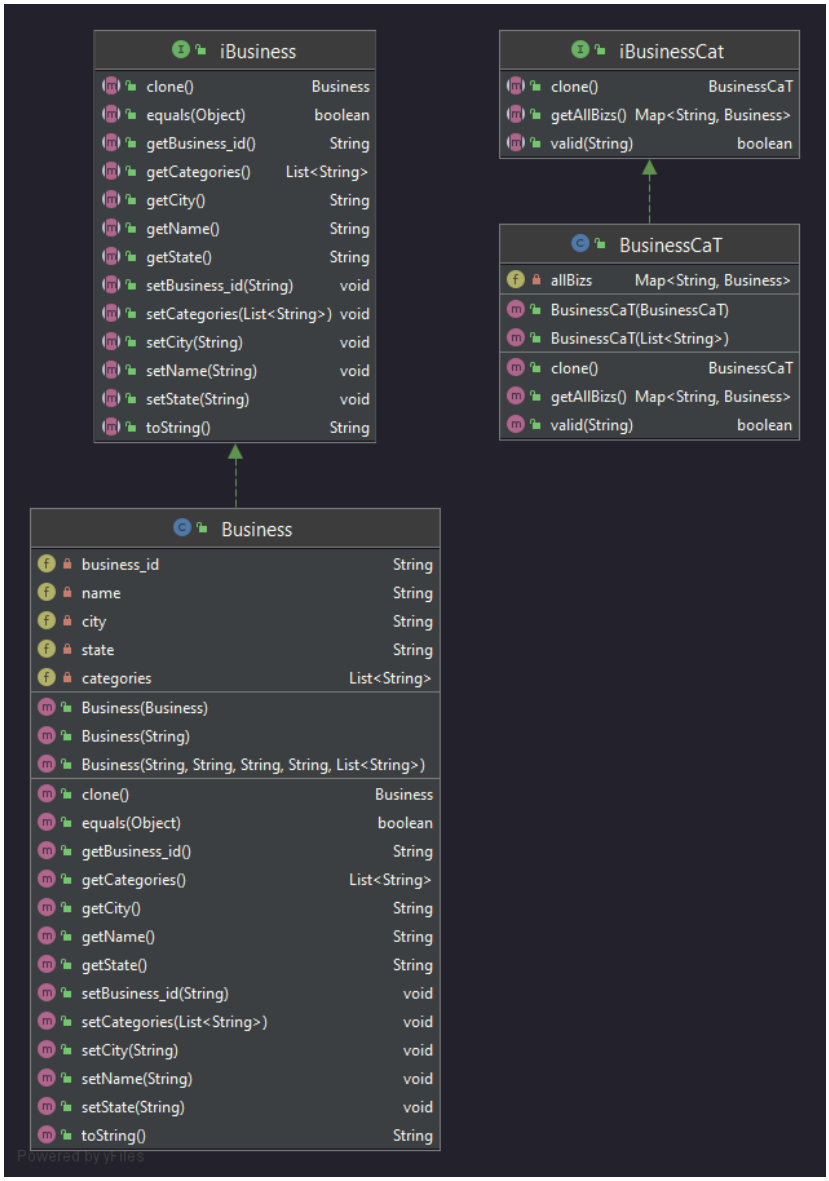
View

UI


UI
input Scanner
UI()
clearScreen() void
closeScanner() void
getInt() int
getInt(String) int
getString() String
getString(String) String
print(String) void
printError(String) void
printEstatisticas(String, int, int, int, int, int, int, int, int) void
printList1(List<String>) void
printList2(List<SimpleEntry<String, Integer>>) void
printList3(List<SimpleEntry<String, Double>>) void
printMainMenu() void
printMap1(Map<Integer, auxConsulta>) void
printMap2(Map<Integer, List<SimpleEntry<String, Integer>>>) void
printMap3(Map<String, List<String>>) void
printMap4(Map<String, Map<String, List<SimpleEntry<String, Double>>>>) void
printQueries() void
printTime() void
Powered by yFiles

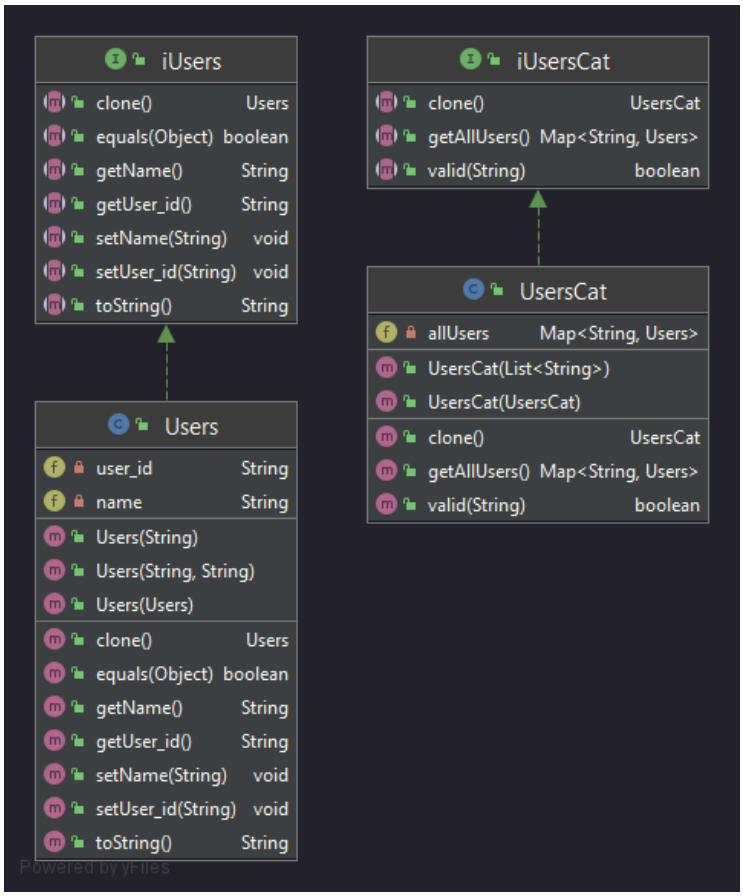
Model

Package BusinessF

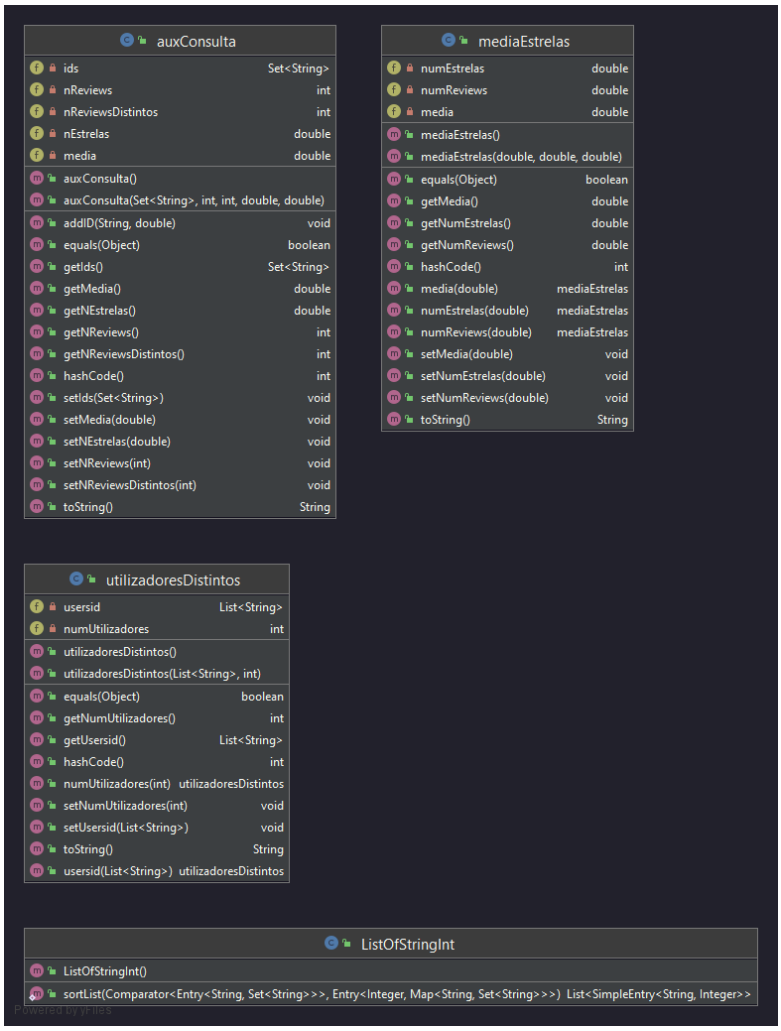


Package ReviewsF

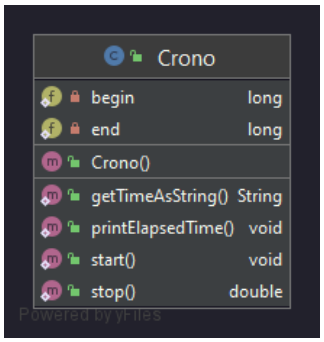
Package UsersF



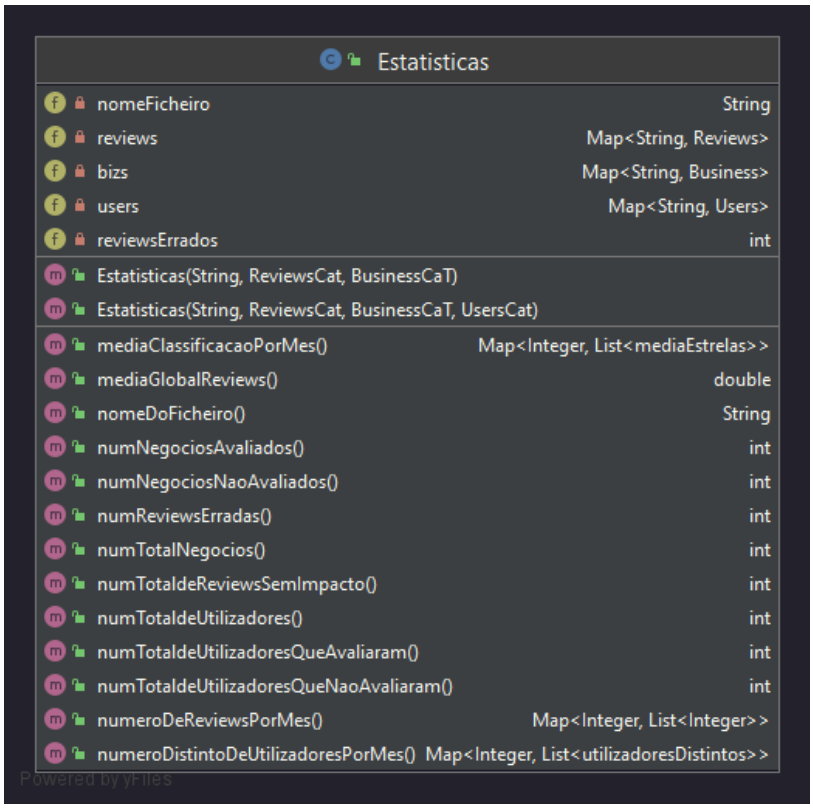
Package EstuturasAuxiliares



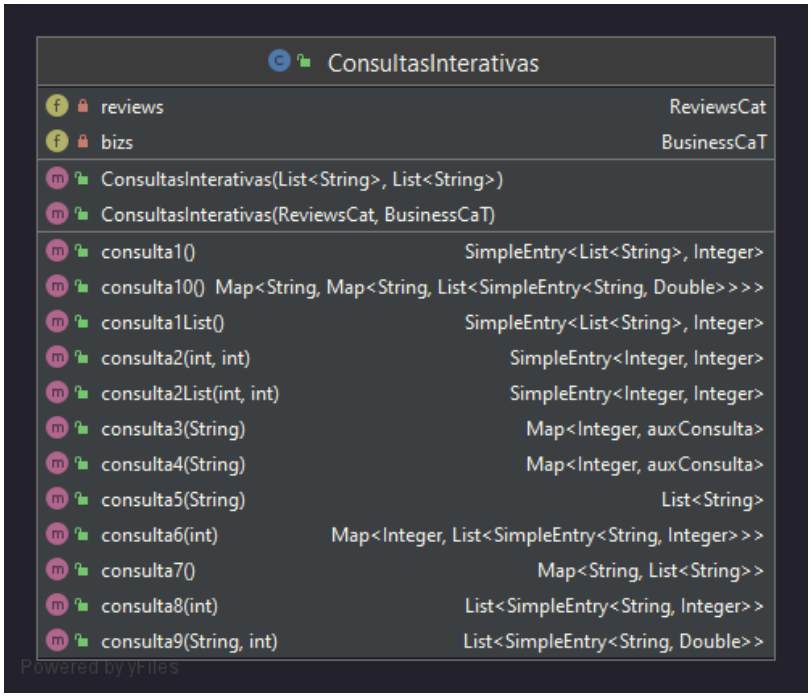
Crono



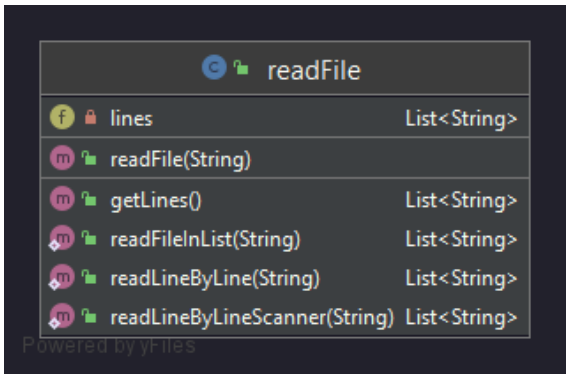
Estatísticas



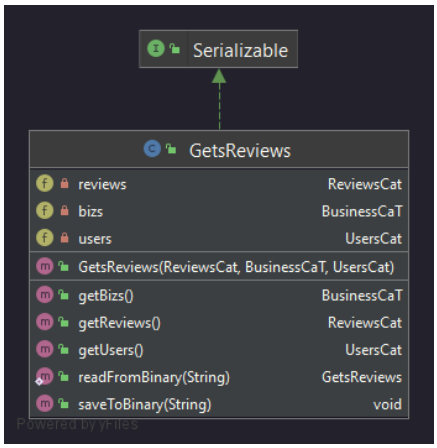
ConsultasInterativas



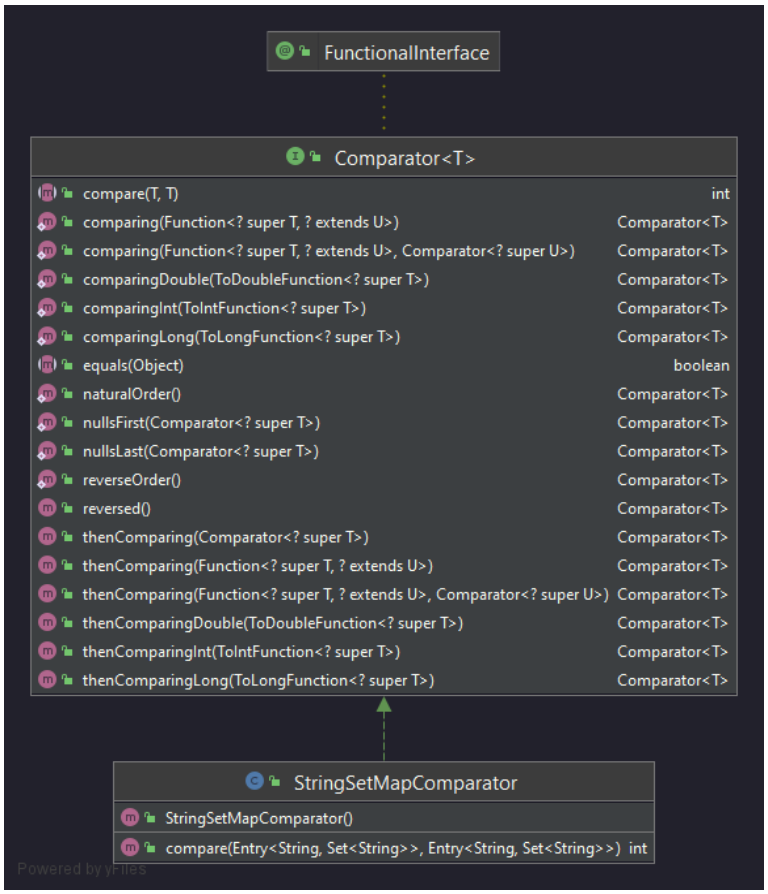
readFile



GetsReviews

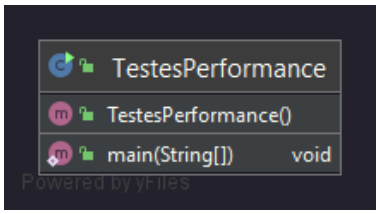


StringSetMapComparator

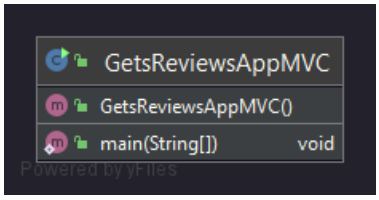


Fora do MVC

TestesPerformance



GetsReviewsAppMVC



Especificações de computador de teste

Memória: 15,5 GiB

Processador: Intel Core i7-8750H CPU @ 2.20GHz * 12

Placa gráfica: Intel UHD Graphics 630 (CFL GT2)

GNOME: 3.28.2

Sistema Operativo: 64-bits (UBUNTU 18.04.5 LTS)

Disco: 86,2 GB

Testes

Consultas Iterativas

1.

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.226	226.0
2	0.176	176.5
Média	0.201	201.3

2. Para o mês 3 e o ano 2018

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.087	87.3
2	0.086	86.1
Média	0.087	86.7

3.Para o user_id: “7MTNM_Rwlc4up5PF180rDQ”

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.102	102.8
2	0.079	79.0
Média	0.091	90.9

4.Para o business_id : “EXOsmAB1s71WePlQk0WZrA”

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.123	123.0
2	0.116	116.5
Média	0.120	119.8

5. Para o user_id : “YoVfDbnISIW0f7abNQACIg”

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.083	83.5
2	0.084	84.2
Média	0.084	83.9

6. Para x=15

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	1.165	1165
2	1.095	1095
Média	1.130	1130

7.

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.831	831.0
2	0.825	825.1
Média	0.828	828.1

8. Para x=15

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.820	820.8
2	0.618	618.3
Média	0.719	719.6

9. Para o business_id “A-xLw4u4wYusT1A2N-1Vxw” e x=10

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.094	94.96
2	0.094	94.55
Média	0.094	94.76

10.

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	0.577	577.6
2	0.589	589.1
Média	0.583	583.4

Tempo de leitura

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	16.99	16997
2	17.21	17215
Média	17.10	17106

Estatísticas

Tentativa	Tempo execução segundos	Tempo execução milisegundos
1	2.308	2308
2	2.419	2419
Média	2.364	2364

Estatísticas para os ficheiros fornecidos[1]

Nome do ficheiro:

Teste.txt

Numero de reviews erradas:

43513

Número total de negocios:

160582

Número de negócios avaliados:

27962

Número de negócios não avaliados:

132620

Número total de utilizadores:

2189457

Número de utilizadores que avaliaram:

536875

[1] As estatísticas correspondentes à segunda parte são bastantes extensas, para ocupar menos espaço decidiu-se que, caso seja necessário a sua apresentação, aquando da demonstração do projeto, serão mostradas.

Número de utilizadores que não avaliaram:

1652582

Número de reviews sem impacto:

956487

Resultados dos testes de performance(em segundos)

Tempo para ler todos os ficheiros a ler o ficheiro todo de uma vez:

10.408

Tempo para ler todos os ficheiros com Scanner:

251.996

Tempo para ler todos os ficheiros com BufferedReader:

9.654

Tempo para consulta 1 usando Set<>:

0.295

Tempo para consulta 1 usando List<>:

3080.825

Tempo para consulta 2 usando Set<> para o ano 2012 e mês 5:

0.079

Tempo para consulta 2 usando List<> para o ano 2012 e mês 5:

0.076

Tempo para consulta 3 com user_id 7MTNM_Rwlc4up5PF180rDQ :

0.742

Tempo para consulta 4 com business_id EXOsmAB1s71WePlQk0WZrA :

0.468

Tempo para consulta 5 com user_id YoVfDbnISIWOf7abNQACIg :

0.425

Tempo para consulta 6 com x=15 :

2.047

Tempo para consulta 7:

1.546

Tempo para consulta 8 com x=15:

0.993

Tempo para consulta 9 com business:id = A-xLw4u4wYusT1A2N-1Vxw e x=15:

0.252

Tempo para consulta 10:

1.015

Conclusão

Neste trabalho, similar ao passado de C, tivemos muita dificuldade com a gestão de memória, infelizmente, o nosso trabalho tem algumas “memory leaks” que não conseguimos detetar e corrigir, o que acaba por influenciar ligeiramente a performance deste, nomeadamente na leitura e escrita em binário de ficheiros.

Outro aspeto negativo e a melhorar é a repetição de código, quer nas querys como nas estatísticas, assumimos que era possível melhorar, adotando as recomendações do enunciado, justificando este problema uma vez mais, devido à falta de coordenação na divisão dos trabalhos.

Porém, ficamos contentes com os resultados obtidos nas querys e estatísticas, achamos que os tempos obtidos estão razoavelmente bons e em suma foram bem concebidas.

Em termos de conclusão, achamos que o trabalho se enquadra perfeitamente nos conceitos lecionados e aprendidos nas aulas de Java e permitiu-nos pôr em prática o nosso estudo e desenvolver novas capacidades de programação orientada a objetos.