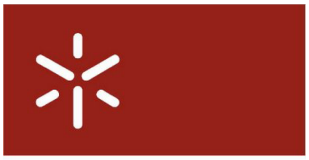
Universidade do Minho

Departamento de Informática



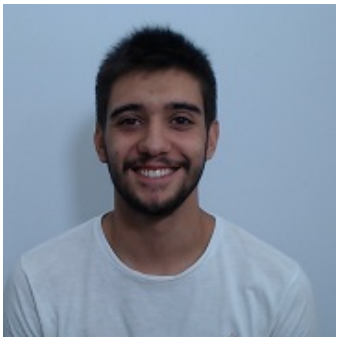
Universidade do Minho

Laboratórios de Informática III

Sistema de Gestão e Consulta de Recomendações de Negócios na Plataforma Yelp

---

Grupo 19- Parte Java



André Vieira

Francisco Andrade

Joana Sousa

André Gonçalves Vieira A90166
Francisco Alves Andrade A89513
Joana Castro e Sousa A83614

Junho de 2021

Conteúdo

1 Introdução

Este relatório é referente à segunda parte do projeto iniciado no presente semestre no âmbito da unidade curricular de Laboratórios de Informática III.

Na segunda fase deste projeto é proposta a realização de uma aplicação de *Desktop*, sendo *java* a linguagem utilizada para a realização da mesma, que permita a execução de consultas interativas de informações relativas a uma gestão básica de um sistema de recomendação e classificação de negócios.

Aquando a realização do projeto proposto, a maior dificuldade do grupo foi a escolha da estrutura de dados a utilizar de modo a escolher a estrutura mais eficiente de modo a diminuir o tempo de procura de informações e o armazenamento destas.

Ao longo deste documento é descrita toda a fase de planeamento e desenvolvimento do projeto, bem como a clarificação das soluções implementadas pelo grupo.

2 Solução Implementada

2.1 Planeamento Inicial

Numa aproximação inicial a este problema, o grupo começou por utilizar uma estrutura de armazenamento para a classe **Catalogo**. Esta classe continha todos os 3 Catálogos necessários para o desenvolvimento deste projeto. Porém, esta abordagem mostrou-se ineficiente, conduzindo à necessidade de ser criada uma classe para cada catálogo, de modo a tornar a procura mais eficaz. É possível observar pelas imagens ilustradas abaixo a forma como o processo se desenvolve: podemos concluir que cada catalogo terá a sua própria classe, sendo esta estrutura diferente num deles, o que nos facilita o processo de procura.

Relativamente ao Catalogo dos Users e dos Business, a estrutura implementada foi relativamente igual. Sabendo a primeira letra do id de um determinado user/business é possível aceder diretamente a um **TreeMap** que, por sua vez, nos dá acesso ao id do business/user. A partir deste ponto, chegamos automaticamente a toda a informação relativa a esse id. Inicialmente, o *load* dos ficheiros demorava cerca de 1 minuto e 30 segundos e cada query rondava os 15 minutos. Logo, foi necessário pensar numa alternativa que tornasse todo o processo mais rápido. Visto que este trabalho requer uma pesquisa de informação quase instantânea, apesar da capacidade que cada ficheiro conter à volta dos 3 GB, entendemos que o planeamento inicial não seria a melhor abordagem.

2.2 Solução Final

Em relação ao catalogo dos Businesses e User a estrutura implementada foi a seguinte :
HashMap<Character, TreeMap<String,Business>> catalogo;
HashMap<Character, TreeMap<String,User>> catalogo;
Esta procura é feita através do primeiro character do id de cada um : ao aceder ao mesmo é possível ter acesso aos vários id's e cada um contém a informação relativa ao seu Busines/User.

No Catalogo das Reviews decidimos implementar algo diferente, uma vez que é necessária uma pesquisa mais aprofundada. O mesmo pode fazer travessias pelos outros dois catalogos, através do id de cada um.

HashMap<String,HashMap<SimpleEntry<Integer,Integer>, TreeMap<String,List<Review>>>> catalogo; é a estrutura implementada para o **catalogo das Reviews**.

Primeiramente, é utilizado um HashMap que através do businessID, dá acesso a um novo HashMap. Este segundo HashMap tem como KEY um par "(ano, mês)", de forma a conseguirmos filtrar pelo ano e mês de cada business. O value deste último HashMap é um TreeMap que a cada userId nos devolve todas as reviews efetuadas pelo user, sendo esta a key do TreeMap.
Esta estrutura foi implementada para que, caso fosse fornecido o mês, o ano e o businessId, nos levasse a todas as reviews efetuadas por todos os users, filtradas pelos parâmetros fornecidos.

3 MVC

MVC é o acrônimo de Model-View-Controller que foi pensada desde os primórdios deste projeto. O Model tem contacto com o Controller para este posteriormente executar o que lhe é pretendido, pois consiste na parte lógica da aplicação, designado por "GuestReviewsApp". A View é apenas a visão geral de todos os menus e o Controller é onde se executa a mediação entre a entrada e a saída, entitulado por "GuestReviews". Pensa-se ter conseguido uma boa abordagem deste padrão de projeto e a implementação desta visão em trabalhos futuros.

4 Estruturas de Dados

4.1 Classes Business, User e Review

A classe *Business* tem as segiuntes variáveis de instância:

```
private String id;
private String name;
private String city;
private String state;
private List<String> categories;
private int ncategories;
```

Estas variáveis correspondem a todas as características apresentadas por um negócio devidamente identificado.

A classe *User* tem as segiuntes variáveis de instância:

```
private String id;
private String name;
private List<String> friends;
private int nfriends;
```

Apesar de serem declaradas, não são utilizadas as variáveis *friends* e *nfriends*. A classe *Review* tem as segiuntes variáveis de instância:

```
private String id;
private String user_id;
private String business_id;
private float stars;
private int useful;
private int funny;
private int cool;
private Date data;
private String text;
```

Estas variáveis correspondem a todas as características apresentadas por uma *Review* devidamente identificada, com um id de um *User* e de um *Business* existentes.

4.2 CatBusiness e CatUsers

A classe *CatBusiness* tem as segiuntes variáveis de instância:

```
private HashMap<Character,  TreeMap<String,Business>>  catalogo;
private String ficheiro;
private int numeroTotalBusinesses;
private int businessesAvaliados;
```

A classe *CatUsers* tem as segiuntes variáveis de instância:

```
private HashMap<Character,  TreeMap<String,User>> catalogo;
private String ficheiro;
private int numeroTotalUsers;
private int usersAtivos;
```

Os diagramas abaixo apresentados demonstram a estrutura representada pela variável *catalogo* destas duas classes.

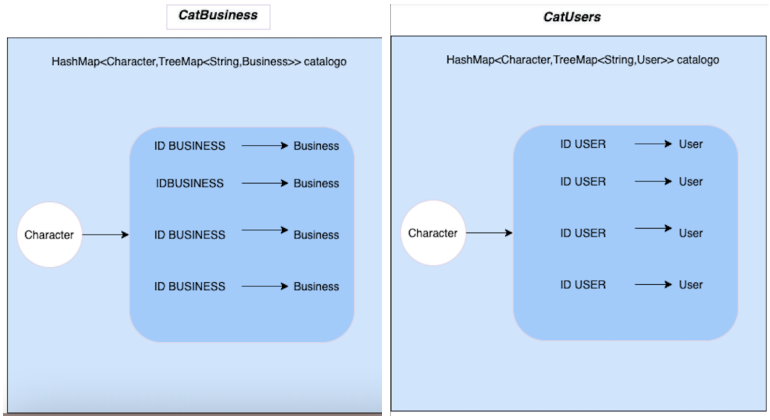


Figura 1: Estrutura das variáveis catalogo da classe *CatBusiness* e *CatUsers*, respetivamente.

4.3 CatReviews

A classe *CatReviews* tem as segiuntes variáveis de instância:

```
private HashMap<String,HashMap<SimpleEntry<Integer,Integer>,
        TreeMap<String,List<Review>>>> catalogo;
private String ficheiro;
private int numeroTotalReviews;
private int numeroReviewsErradas;
private int reviewsSemImpacto;
```

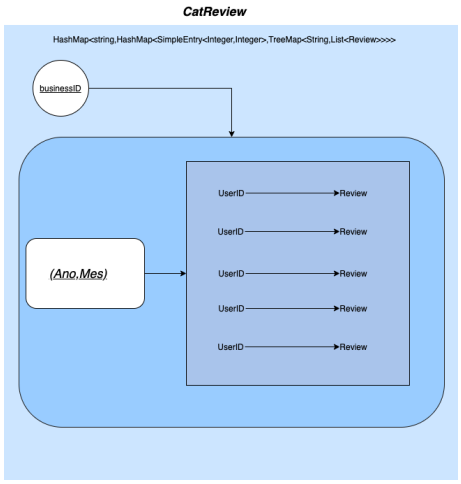


Figura 2: Estrutura da variável catalogo da classe *CatReviews*

5 Interação com o Utilizador

Quando interagimos primeiramente com o programa é nos apresentado o Menu Inicial. Podemos escolher entre as diferentes 4 opções e em cada uma teremos opções mais específicas do que pretendemos executar.

Figura 3: Início do programa

Numa primeira fase, escolheu-se a opção 1 de forma a ser efetuada a leitura dos ficheiros pretendidos para que à posteriori se consiga consultar as várias tarefas que se deseja. Nesta primeira funcionalidade foram utilizadas *threads*, dada a sua eficiência fazendo isto de forma concorrente(isto apenas para os catálogos de businesses e users,uma vez que o parsing do catalogo das reviews depende destes). Existe ainda a possibilidade de serem consultadas as estatísticas mensais e anuais (opção 2), tal como as várias consultas interativas (opção 3). Caso seja objetivo procurar informação após o término do programa mas sem voltar a carregar todos os ficheiros, é dada a opção de gravar o estado do programa e carregá-lo novamente numa altura futura. Por último, é fornecida a hipótese, de no caso de uma escolha acidental, retroceder para o menu inicial. De notar que encontra-se disponível no ecrã o tempo de leitura de cada opção escolhida.

Figura 4: Carregamento de Ficheiro

Ao utilizar a opção *2* do menu inicial o utilizador entrará na parte de consulta de estatísticas, podendo estas ser gerais ou mensais. Nas figuras abaixo demonstradas temos exemplos da consulta

de Estatísticas Gerais e Estatísticas Mensais relativas ao ano de 2020 por parte do utilizador do programa.

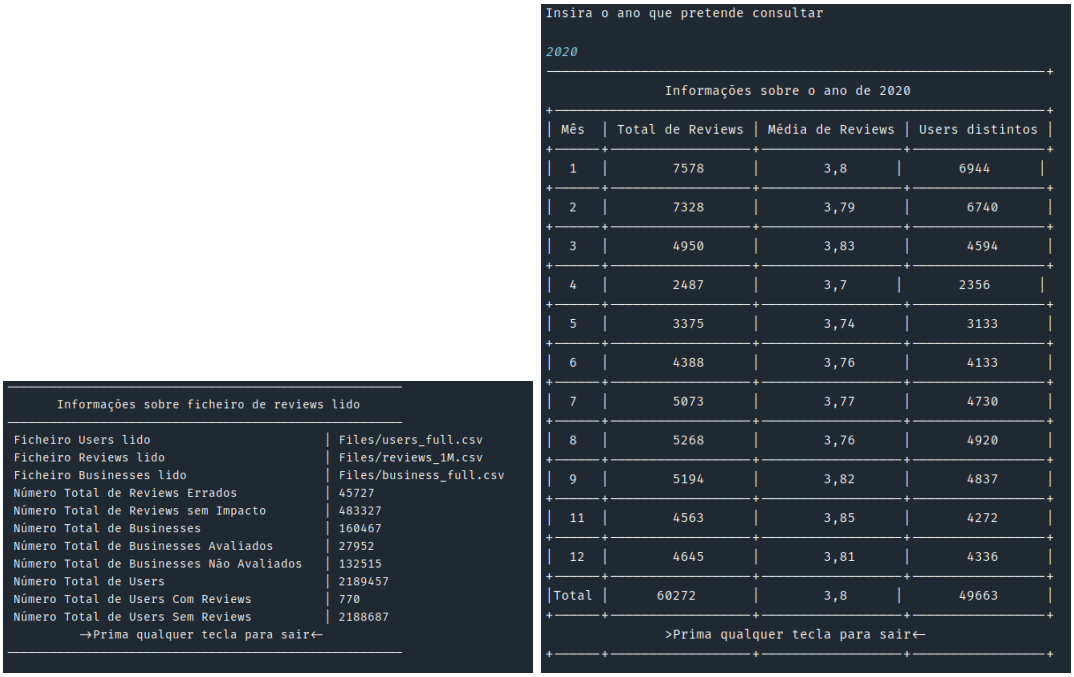


Figura 5: Exemplo de Estatísticas Globais e exemplo de Estatísticas Mensais relativas ao ano de 2020

Nas imagens apresentadas são ilustradas as diversas opções do menu inicial. Na primeira imagem encontra-se uma amostra de uma querie aleatoriamente selecionada e o seu resultado. Na imagem seguinte é visível a opção de gravar o ficheiro e a prova como esse ficheiro é criado.

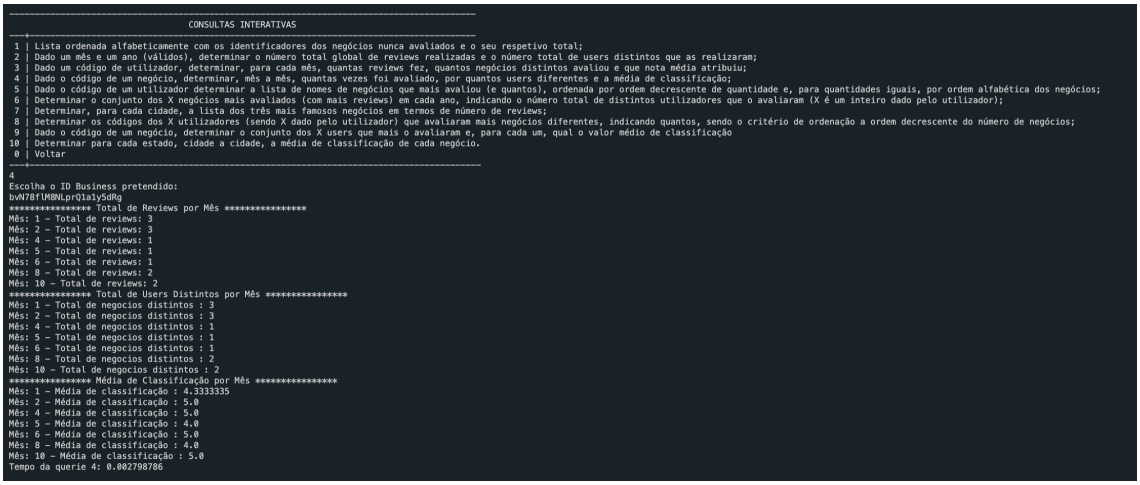


Figura 6: Exemplo da execução da querie 4

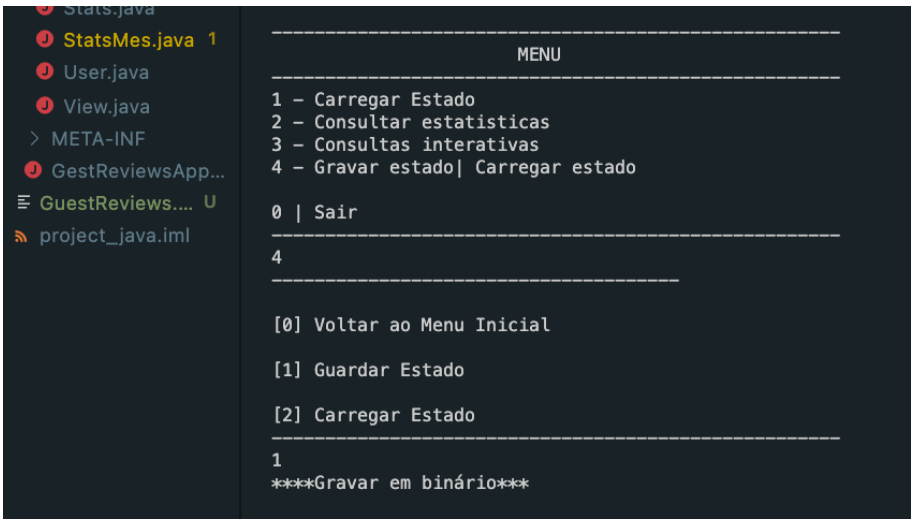


Figura 7: Exemplo de armazenamento de estado

6 Queries

A aplicação foi desenvolvida de forma a conseguir fornecer ao utilizador um conjunto de informações sobre o Sistema de Gestão e Consulta de Recomendações de Negócios na Plataforma Yelp de forma simples e legível. Para isso, o projeto foi baseado nas queries expostas no enunciado do mesmo. Estas queries dividem-se em dois tipo: Estatísticas (2) e Interativas (10).

6.1 Queries Estatísticas

1. **Estatísticas Gerais**

   Nas estatísticas globais o objetivo é informar o utilizador sobre a generalidade das *Reviews*, *Users* e *Businesses* dos quais temos dados, fornecendo informações tais como os últimos ficheiros lidos, número de *Reviews* erradas e com 0 impacto, número total de negócios e número de negócios avaliados e não avaliados, número total de *Users*, número de *Users* Ativos e Inativos. Para tal, ao realizar o *parsing* de todos os ficheiros, vamos recolhendo informações gerais, armazenando-as em variáveis de instância.

   Posteriormente, caso o utilizador consulte as Estatísticas Globais, iremos percorrer o catalogo de *Businesses* de forma a saber quantos deles contêm avaliações e após isto, faremos o mesmo para o catalogo de *Users* com o intuito de saber quantos destes realizaram pelo menos uma *Review*.

2. **Estatísticas Mensais**

   As Estatísticas Mensais informam o utilizador sobre os dados de um ano, escolhidos por este, fornecendo dados como o número total de *Reviews* realizadas nesse mês, a média de classificação de *Reviews* nesse determinado mês e o número de *Users* distintos que realizaram *Reviews* nesse mês. Para obter estas informações, acedemos ao catalogo de *Reviews* para a data pretendida e recolhemos as informações acima apresentadas e insere-as numa matriz para armazenar estas estatísticas por mês.

6.2 Queries Interativas

1. Nesta primeira query foi necessário percorrer todo os characters do catalogo dos businesses.De seguida selecionamos cada id de cada business, pois é a key do TreeMap. Ao termos acesso a esta key, percorremos o catalogo das reviews e caso este catalogo nao contiver esse id, significa que o mesmo não foi avaliado e, desta forma será adicionado a um TreeSet.

2. A execução desta querie foi pensada consoante a estrutura de dados implementada. Primeiramente vamos percorrer todos os idbusinesses.Como recebemos um mês e um ano e temos na nossa estrutura uma key que corresponde a um par de mês e ano, vamos apenas buscar os values dos valores que recebemos. Posto isto, sabemos que o tamanho da lista das reviews vai incrementando por cada iduser e os users já são distintos e serão adicionados a um TreeSet.

3. Para descobrir o numero total de reviews, quantos negócios foram avaliados e que nota media foi atribuida por um user definido pelo utilizador, em cada mês, existe a query 3. Esta query percorre todos os businesses e todos os pares com ano e mês e vai percorrer todas as reviews do utilizador fornecido.

4. Esta Tarefa é bastante semelhante à tarefa anterior porém, nesta fase é fornecido o businessid o que nos dá direto aceso a todas as suas informações, tornando o processo de procura das reviews que contém, dos users que o avaliaram e a sua média de classificação mais eficiente.

5. Também era necessário determinar todos os nomes de negócios que mais avaliou, esta querie foi verificar se existia o código do user e se isto acontecesse, o id do business seria adicionado à key do TreeMap e o seu value seria o valor que já se encontrava acrescentando o size da lista. De seguida com todos os id's pretendidos, foi necessária uma procura pelo catalogo dos businesses de forma a ter acesso ao nome de cada um. Foi possível a ordem alfabética dos nomes pela mesma quantidade.

6. Para determinar os X negócios mais avaliados, inicializamos uma *HashMap* onde serão colocados o número de *Reviews*, sendo esta a *key*, desse negócio e o seu id, sendo este o *value*. Após isto, iremos buscar à *HashMap* os X negócios com mais *Reviews*.

7. Esta *Querie* tem como finalidade determinar, para cada cidade, a lista dos três negócios mais famosos, ou seja, com mais *Reviews*. Para tal, inicializamos uma *HashMap* onde iremos colocar a cidade e todos os negócios que a ela pertencem. Posteriormente, acedemos ao catalogo de *Reviews* e colocamos, numa nova *HashMap*, o número total de *Reviews* e o negócio a que estas *Reviews* pertencem.

8. Esta querie não foi finalizada, apesar de várias tentativas exaustivas, não conseguindo obter os resultados pretendidos.

9. De acordo com um código de negócio fornecido conseguimos determinar os users que mais o avaliaram e a sua médias seguimos um pouco o pensamentos para alíneas anteriores, e conseguimos ainda implementar o limite de utilizadores.

7 Testes e Performance

Para uma melhores perceção dos tempos que são apresentados fizemos uma compilação destas duas imagens que nos mostram a mesma querie mas para businesses diferentes. Desta forma, conseguimos perceber que os tempos são bastante bons e para diferentes inputs.

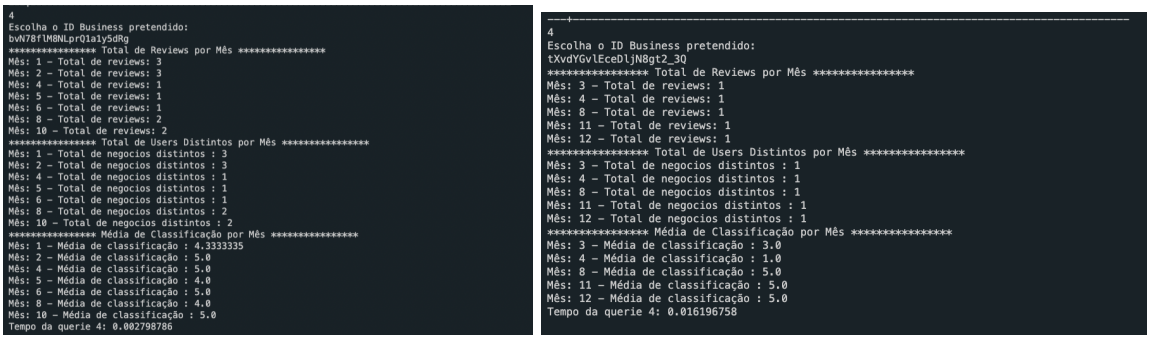


Figura 8: Observação do tempo de execução da mesma querie para businessesid diferentes

Para a obtenção dos resultados apresentados abaixo, foi utilizado um computador com um processador i5 de 8-cores com 1,4 GHz e 16GB de memória RAM.

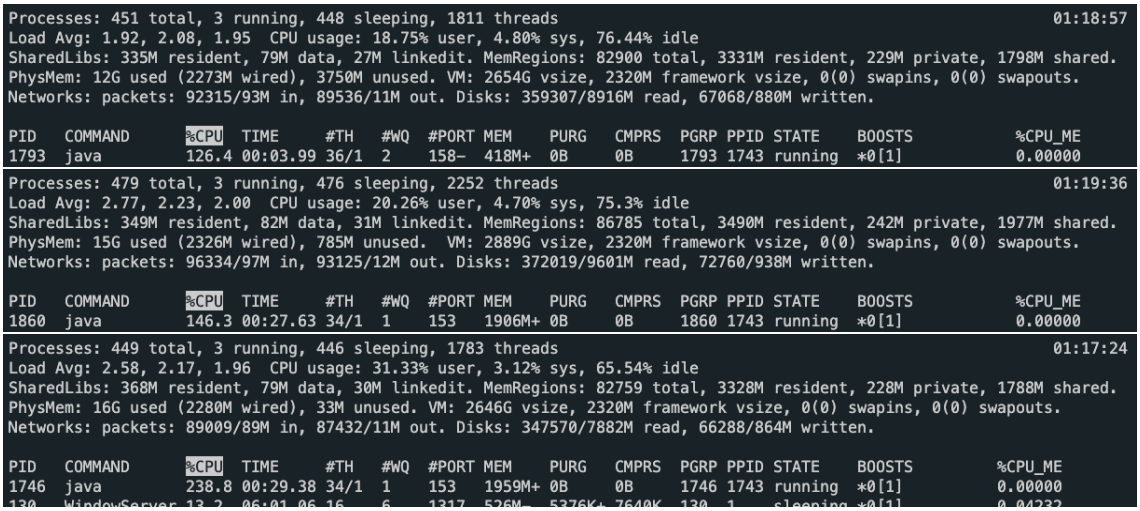


Figura 9: CPU ao longo do carregamento dos ficheiros

Na tabela abaixo temos vários tempos de execução e a respetiva *querie* a que estes pretencem, sendo este o tempo médio de 20 medições.

	Parsing	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q9**
Tempo(s)	20.023	1.274	0.02025	0.13235	0.002613	0.125965	0.167725	0.49792	0.004143

Tabela 1: Tempos de execução em segundos de diferentes *queries* e do *parsing*.

8 Conclusão

Finalizado o segundo e último projeto da Unidade Curricular de Laboratórios de Informática III, exigiu um pouco mais de planeamento, pois foi necessário uma estrutura bem implementada de acordo com as funcionalidades propostas. Durante a realização deste, foram várias as etapas em que encontrámos dificuldades, que tiveram de ser ultrapassadas para chegar ao resultado final pretendido.

A elaboração deste trabalho exigiu a aplicação de todos os conhecimentos consolidados anteriormente tanto no trabalho prático anterior, como ainda conhecimentos de outras unidades curriculares.

Apesar do reduzido tempo, tendo em conta a fase do semestre em que nos encontramos, o grupo deu o seu melhor e apresenta um produto final do qual se consegue orgulhar, não descartando a possibilidade de, com mais tempo de trabalho, realizar algumas melhorias e implementar ainda mais e melhores funcionalidades. Visto que gostaríamos de ter implementado as funcionalidades todas sugeridas no enunciado, infelizmente, não nos foi possível.

Porém, de acordo com os dados obtidos, relativos à performance da aplicação, é verificado que o tempo que as queries requerem é bastante bom e apenas o tempo necessário para guardar e carregar o estado é que se torna mais extenso. Inicialmente percebemos que estávamos perante um problema de performance mas depois conseguimos superar e tornar a estrutura um pouco mais complexa, porém mais eficiente.

Acredita-se que o objetivo principal do trabalho, de desenvolver uma aplicação que seguisse o modelo MVC, capaz de realizar as tarefas apresentadas no enunciado com os resultados devidamente apresentados, mantendo o código simples e flexível, foi concretizado com sucesso.

A Diagrama de Classes

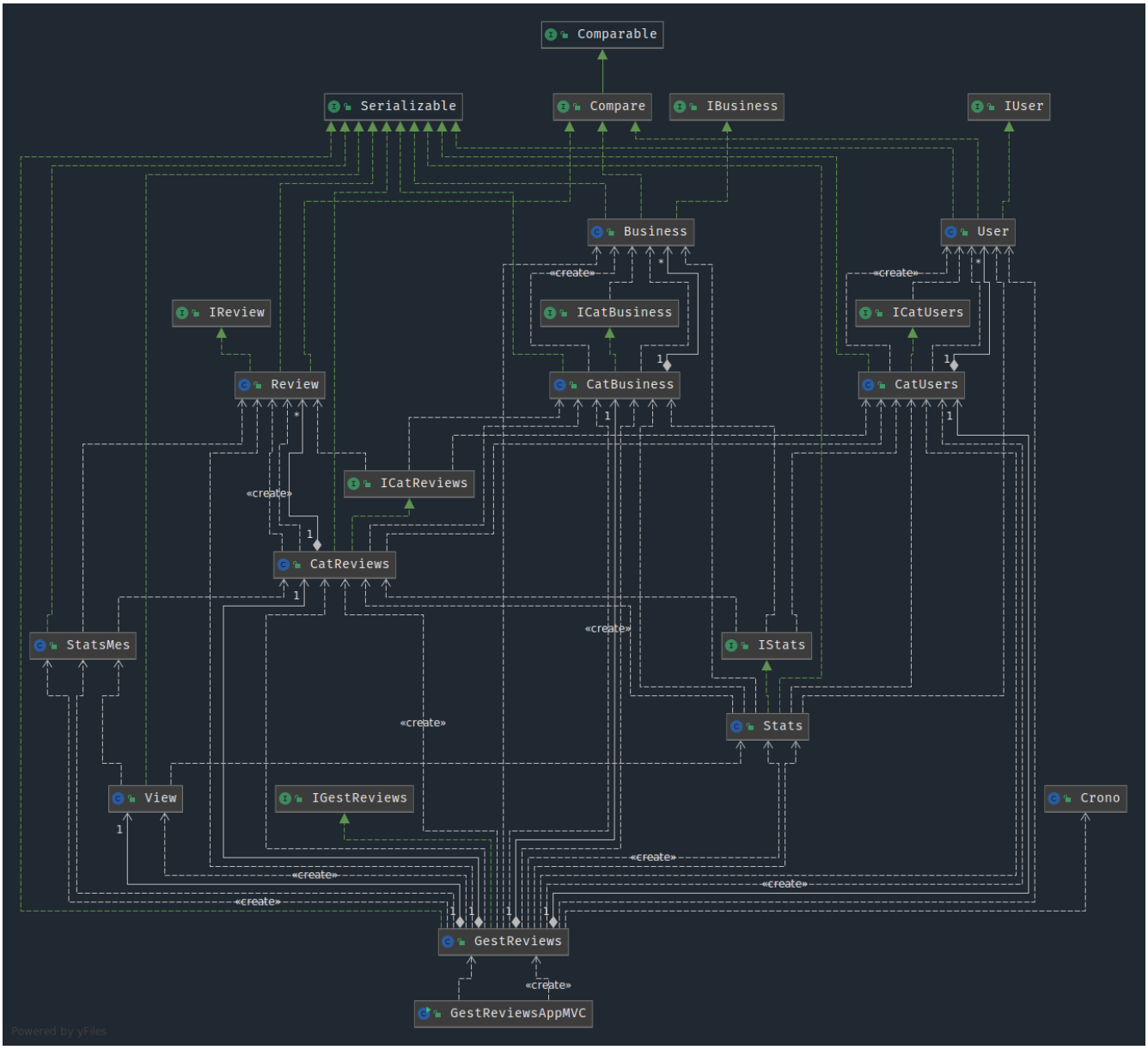


Figura 10: Diagrama de Classes da Aplicação