Laboratórios de Informática 3

MIEI - 2º ano - 2º semestre
Universidade do Minho

Projeto em Java - Grupo 11


Inês Sampaio
A72626


Frederico Pinto
A73639


André Sousa
A74813

12 de Junho de 2018

Conteúdo

1 Introdução

No âmbito da unidade curricular de Laboratório de Informática III, havia sido proposto o desenvolvimento, em C, de um sistema capaz de processar ficheiros XML (onde se armazenam os dados utilizados pelo Stack Overflow), e, uma vez processada essa informação, deveria também ser possível executar um conjunto de interrogações específico, de forma eficiente, como por exemplo saber em que perguntas participou determinado utilizador, ou quantos posts foram feitos num determinado intervalo de tempo.

Nesta fase, porém, pretende-se que as interrogações e todo o trabalho seja desenvolvido em JAVA. Neste sentido, o presente relatório apresenta e justifica todas as decisões tomadas ao longo do projeto bem como as escolhas das estruturas de dados utilizadas.

O Stack Overflow é um site de "pergunta e resposta"(Q&A) destinado a profissionais ou entusiastas da programação. Atualmente, é uma das maiores comunidades online de developers a nível mundial, permitindo a qualquer utilizador colocar questões, que serão depois respondidas por outros utilizadores.

Com cerca de 8.4 milhões de utilizadores, 15 milhões de questões já colocadas, às quais se responderam 24 milhões de vezes, o volume de informação gerada na plataforma é enorme, além de muito útil e valiosa. Mas analisar uma quantidade assim de dados pode tornar-se um processo bastante demorado e complexo, dadas as operações necessárias para cruzar os diferentes tipos de informação disponíveis.

2 Abordagem ao projeto

Perante a proposta de projeto da disciplina, e tendo por base todo o trabalho desenvolvido em C, decidimos implementar como estrutura de dados quatro *HashMaps*, estruturas que nos permitem guardar toda a informação necessária à resolução das queries de forma eficiente. Comparativamente, julgamos que a estrutura que implementamos na primeira fase do trabalho prático desenvolvido em **C**, seria demasiado para implementar em *Java*, já que nesta linguagem dispomos de várias *APIs* e mecanismos que que nos possibilitam executar a mesma tarefa de uma forma muito mais acessível.

Ao longo da próxima secção iremos enumerar as principais classes definidas e as suas características.

3 Classes

3.1 Person

Tendo o Stack Overflow uma quantidade muito grande de utilizadores, é necessária a criação de uma classe que descreva o utilizador e a sua informação. Desta forma, foi criada a classe **Person**.

Esta classe possui diversas variáveis de instância que a identificam:
- uma identificação única, o seu **id**;
- um **nome**;
- uma breve descrição sobre o utilizador (**aboutme**);
- a sua **reputação**;
- o **número de posts** efetuados;

Com estas variáveis somos capazes de representar toda a informação acerca dos utilizadores, de maneira a responder a todas as queries que os envolvem.

```
1 public class Person {
2     private long id;
3     private String nome;
4     private String aboutme;
5     private long reputacao;
6     private int nposts;
```

3.2 Question

Em relação às perguntas, foi criada uma classe **Question** que guarda a informação pertinente sobre uma pergunta. Esta classe tem como variáveis de instância:
- o **id** da pergunta, que é único;
- a **data** da pergunta;
- o seu **título**;
- o **número de respostas**;

- o **identificador do autor** da pergunta;
- o **identificador da melhor resposta**;
- a **pontuação da melhor resposta**;
- a lista das **tags** que a pergunta possui;

Desta maneira, e sendo esta uma classe fundamental, conseguimos obter informações essenciais sobre certos parâmetros que nos permitem responder com mais eficiência a determinadas queries.

```
1 public class Question {
2     private long id;
3     private LocalDate date;
4     private String titulo;
5     private int nRespostas;
6     private long autor;
7     private long bestAnswer;
8     private float pontuacaoBestA;
9     private ArrayList<String> tags;
```

3.3 Answer

Um pouco semelhante ao que acontece com as perguntas, foi criada também uma classe **Answer**, que guarda as informação relativas a uma resposta. Tem como variáveis de instância:

- o **id** da resposta, que tal como nas perguntas, é único.
- a **data** da resposta;
- o **identificador da pergunta** à qual a resposta se refere;
- o **identificador do autor**;
- o **número de votos** que a resposta contém;

Com estas informações, tal como nas perguntas, conseguimos obter um acesso mais fácil às informações para responder a certas queries.

```
1 public class Answer {
2     private long id;
3     private LocalDate date;
4     private long idpai;
5     private long autor;
6     private int votes;
```

3.4 Tag

A criação da classe **Tag** foi necessária para a resolução de uma query. Esta classe é definida pelas variáveis de instância:

- o **id** da tag;
- o **nome** da tag;
- um **contador**, que é sempre inicializado a 0;

```
1 public class Tag {
2     private long id;
3     private String nome;
4     private int contador;
```

3.5 TCDExample

A TCDExample, é a classe concreta que mexe diretamente na informação das classes, e analisa a informação dos ficheiros que queremos passar. Neste caso, possui como variáveis de instância os mapas dos utilizadores, dos perguntas, das respostas, das tags e o parser, que processa as querys, com a informação necessária para as resolver:

```
1 public class TCDExample implements TADCommunity {
2     private MyLog qelog;
3     private CatUsers users;
4     private CatAnswers answers;
5     private CatQuestions questions;
6     private CatTags tags;
7     private Parser parse;
```

3.6 CatUsers

A classe CatUsers tem como função armazenar os utilizadores de maneira a poder-se realizar as queries. Implementa um *Map* criado para guardar os utilizadores da plataforma, de forma simples e clara. Nela, a cada chave - ID do utilizador, é associado um **Person**(utilizador).

```
1 public class CatUsers {
2     private Map<Long, Person> users;
```

3.7 CatTags

A classe CatTags tem como função armazenar as tags de maneira a poder-se realizar queries .Para tal, tem como variáveis de instância uma *HashMap*, utilizando como chave o nome da tag, e tendo associado como valor um apontador para uma **Tag** que tem informações da tag.

```
1 public class CatTags {
2     private HashMap<String, Tag> tags;
```

3.8 CatQuestions

A classe CatQuestions cria um HashMap para as questões para que vão ser usadas nas queries. Para tal utiliza como chave o identificador da pergunta e como valor a questão.

```
1 public class CatQuestions {
2     private HashMap<Long, Question> questions;
```

3.9 CatAnswers

A classe CatAnswers cria um HashMap criada para conter todas respostas(Answer) que vão ser processadas nas queries. Para tal, como nas perguntas, utiliza o id da resposta e a resposta como constituintes do mapa.

```
1 public class CatAnswers {
2     private HashMap<Long, Answer> answers;
```

3.10 Comparator

Por forma a facilitar a resposta às interrogações, definimos uma serie de classes, como a *ComparatorDatesQuestion*, *ComparatorVotes*, entre outras, que nos permitem fazer comparações entre os dados fornecidos. Abaixo da-mos o exemplo de uma delas, a classe *ComparatorMostActive* que compara dois utilizadores(Person) e verifica qual deles é o mais ativo (pelo número de posts).

```
1  class ComparatorMostActive implements Comparator<Person>{
2          public int compare(Person a, Person b){
3                  int r = a.getNposts() - b.getNposts();
4                  if (r == 0)
5                          return 0;
6                  else if (r < 0)
7                          return 1;
8                  else
9                          return -1;
10         }
11 }
```

4 Parser

Nesta secção iremos falar sobre um dos passos mais importantes na concretização deste projeto, o parser. Para a realização do mesmo, usamos funções já fornecidas pela biblioteca libxml2.

É nesta fase que toda a informação contida nos dumps é lida, analisada, agrupada e posteriormente carregada em diferentes estruturas de dados, consoante o tipo de informação lida. Desta maneira, estamos perante a query que mais tempo demora a ser efetuada, por razões óbvias e totalmente compreensíveis. Desenvolvemos três funções que, juntas, fazem o parse total dos dumps, ou seja, captam todos os dados relevantes para respondermos com sucesso a todas as questões propostas no enunciado. De todo o dump fornecido, concluímos que apenas seria necessário parsar os ficheiros de input Users.xml, Posts.xml e Tags.xml.

5 Queries

Após termos efetuado com sucesso o parse e o carregamento dos dados para todas as estruturas que definimos, estamos agora em condições de avançar para a resolução das interrogações propostas.

Para realizar todas as queries propostas peloo enunciado, utilizamos a classe *java.util.stream*, que permite efetuar várias tipos de operações com a informação armazenada nas estruturas de dados. Esse tipo de operações permitem obter os resultados desejados com maior eficiência.

Comparando esta classe com as iterações em ciclo for, podemos verificar que esta apresenta uma maior rapidez na resolução das queries. Logo optámos por resolver todas as perguntas propostas no enunciado, utilizando *Streams*.

6 Conclusão

Com a realização da primeira fase do projeto (desenvolvimento em C) fomos-nos apercebendo das dificuldades associadas à gestão e análise de grandes quantidades de dados, e o esforço necessário para que essa gestão seja o mais eficiente possível. Desde arranjar estruturas não muito complexas para cada tipo de informação, formular um *parse* que permitisse mostrar e gerir os dados de forma aceitável, até à criação de uma interface que pudesse resolver as interrogações propostas, cada passo foi um novo desafio. Ao concluir o desenvolvimento deste trabalho, nesta fase já em Java, não podemos deixar de referir as diferenças entre as duas linguagens. Se, por um lado, o processo de desenvolvimento, implementação de estruturas e responder às queries nos tenha parecido mais acessivel em Java, sendo também mais fácil interpretar o código, trabalhar em C foi, definitivamente, mais aliciante.

Não podemos deixar de notar que, todo projeto nos ajudou a evoluir enquanto programadores, forçando-nos a ser versáteis e críticos, quando estamos perante um trabalho assim.