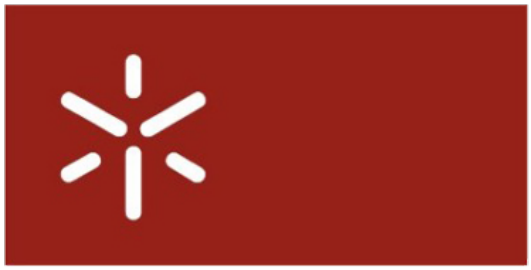
Projeto de Laboratórios de Informática 3
Grupo 46

Carlos Miguel Lopes Sá Barbosa - A82324
José Pedro Saraiva de Carvalho - A80424
Ricardo Rodrigues Martins - A78914

10 de Junho de 2018



Universidade do Minho

Conteúdo

1 Introdução

Este projeto, realizado no âmbito da unidade curricular Laboratórios de Informática 3, tem como objetivo o parsing e carregamento de informações pertinentes de ficheiros .xml provenientes do site *StackOverflow* para uma estrutura de dados à escolha dos alunos, de modo a que seja possível responder a 11 queries dadas pelos docentes. Ao contrário do que foi realizado na primeira parte deste semestre, este projeto deve ser realizado em Java e espera-se que seja uma evolução do conceito anterior, tendo maior atenção à estruturação do código e ao modo como se efetua o parsing, o que requer uma abordagem mais cuidada e ligeiramente diferente ao problema proposto.

2 Classes

O grupo criou, com o intuito de resolução deste trabalho, 9 classes, sendo que estas possuem os seguintes nomes e objetivos sintetizados:

- Answers: Classe responsável pela definição de uma pequena estrutura com os dados de uma resposta
- Comunidade: Classe que representa a comunidade inteira contendo as estruturas de dados mais significativas desta
- TCDExample: Classe onde se encontram os principais requesitos do projeto (load, queries e clean)
- ParserPosts: Parse do ficheiro que contém os Posts de xml para java
- ParserTags: Parse do ficheiro que contém as Tags de xml para java
- ParserUsers: Parse do ficheiro que contém os Users de xml para java
- Posts: Classe abstrata que contem parâmetros comuns às questions e answers
- Answers: Classe responsável pela definição de uma pequena estrutura com os dados de uma resposta
- Questions: Classe responsável pela definição de uma pequena estrutura com os dados de uma pergunta
- Users: Classe responsável pela definição de uma pequena estrutura com os dados de um utilizador

Nos seguintes tópicos o grupo irá discutir estas classes de forma mais detalhada.

2.1 Parsers: ParserPosts, ParserTags, ParserUsers

Esta parte do relatório descreve a criação dos vários parsers de xml para java existentes neste projeto. Para o parsing, o grupo utilizou o método de parsing em blocos como forma de otimizar o tempo de execução do programa e para evitar o carregamento do projeto todo para a memória. Para tal foi utilizado o SAX (Simple API for Xml) visto que este faz uma gestão muito inteligente da memória disponível.

Todos os parsers tiveram obrigatoriamente de usar os seguintes imports:

- java.io.File;
- javax.xml.parsers.SAXParser;
- javax.xml.parsers.SAXParserFactory;
- org.xml.sax.Attributes;

- org.xml.sax.SAXException;
- org.xml.sax.helpers.DefaultHandler;

Estes imports possuem o propósito de fornecer as ferramentas do SAX a aplicar no trabalho.

Em todos os parsers também teve de ser criada uma classe extra para aplicar funções existentes no DefaultHandler do SAX. Dentro desta classe foi criado um construtor. Dentro deste utilizou-se o super como forma de poder ter acesso a toda a informação definida no DefaultHandler. A seguir definiu-se a função startElement. A razão pela qual esta possui 4 parâmetros em vez de dois deve-se somente à aplicação do override a esta função. No interior desta função é feita a conversão de xml para java utilizando as funçoes getIndex (para verificar se a caraterística se encontra definida na row) e a getValue (se se encontrar definida, extrai o valor). No fim acrescenta-se, com base nas caraterísticas retiradas do xml a entidade em questão (user, question, answer ou tag) ao respetivo hashmap definido na Comunidade, sendo que as funções que efetuam tais adições também se encontram definidas na classe Comunidade.

2.2 Estruturas de Dados utilizadas: Comunidade

A classe Comunidade contém as maiores estruturas de Dados do projeto responsáveis pelo armazenamento da informação toda contida nos ficheiros xml utilizados para a resolução do trabalho. Esta classe possui as seguintes estruturas de dados principais:

- HTU: HashMap que possui como value uma estrutura Users (que ira ser explicitada mais à frente neste relatório) e como key o id desse user em particular
- HTQ: HashMap que possui como value uma estrutura Questions (que ira ser explicitada mais à frente neste relatório) e como key o id dessa question em particular
- HTA: HashMap que possui como value uma estrutura Answers (que ira ser explicitada mais à frente neste relatório) e como key o id dessa answer em particular
- HTT: HashMap que possui como key uma String (que ira corresponder ao valor da tagname de cada tag) e como value o id dessa tag em particular

Como foi passível de ser observado anteriormente, o grupo deu bastante ênfase ao uso de HashMaps como estruturas principais do trabalho, devido principalmente à procura de elementos ser efetuada de forma bastante eficiente e poder encontrar facilmente entidades com base em algum seu parâmetro, sendo que neste caso foi utilizado o parâmetro id para esse efeito. Para além disto nesta classe estão incluídas funções que adicionam a estes HashMaps Users, Questions e Answers.

2.2.1 Estruturas de Dados Pequenas: Answers, Questions, Users

As classes Users, Questions e Answers servem para, com base nos parâmetros extraídos dos respetivos ficheiros xml, extrair a informação de uma entidade em específico, quer esta seja um utilizador quer um post.

Para a classe Users os parâmetros são os seguintes:

- id: o Id do User;
- name: o DisplayName do User;
- rep: a Reputation do User;
- aboutme: o AboutMe do User;
- numques: o Número de Questions do User;
- numans: o Número de Answers do User;

- ques: as Questions feitas pelo User;
- ans: as Answers feitas pelo User

Para a classe Questions foram utilizados 3 parâmetros (os primeiros) que também são comuns às Answers, e por esse motivo encontram-se na classe abstrata Posts. Os parâmetros são os seguintes:

- id: Id da Question;
- data: Data de criação da Question;
- owneruserid: Id do criador da Question;
- tags: Lista com as tags toda contidas na Question;
- anscount: Número de Answers que respondem à Question;
- ans: Lista com as ans que respondem a Question;
- numtag: Número de tags contidas na Question;
- title: Título da Question;

Para a classe Answers foram utilizados os mesmos 3 parâmetros da classe abstrata Posts. Os parâmetros são os seguintes:

- id: Id da Answer;
- data: Data de criação da Answer;
- owneruserid: Id do criador da Answer;
- parentid: Id da Question à qual a Answer responde;
- score: Score da Answer;
- comcount: Número de comentários da Answer;

2.3 Resolução das queries propostas: TCDExample

Quer o load, quer as queries, quer o clear estão contidas na TCDExample. Após o parse dos dados, as queries foram resolvidas da seguinte forma:

2.3.1 Load

Recebe uma string que contém o path para a pasta que contém os ficheiros xml dos Users, Posts e Tags. Após isto inicializa uma Comunidade, cria 3 stringbuilders com o path da pasta e faz o parser dos users, posts e tags. Depois a cada um dos stringbuilders faz o append da parte do path que falta para chegar ao respetivo ficheiro xml. Utiliza as funções definidas para esse efeito que se encontram nas classes ParserUsers, ParserPosts e ParserTags.

2.3.2 Query 1

Recebe como input um id dum post. Poderá acontecer uma de 3 coisas: -O id não existir, logo devolve em ambos os campos; -O id ser de uma pergunta, onde irá devolver o título da pergunta e o nome do utilizador que fez essa pergunta; -O id ser de uma resposta, onde irá ser devolvido o título e o nome do autor da pergunta correspondente.

2.3.3 Query 2

Dado como input um certo número N. Cria dois arraylists um com long ints e outro com Users e cria também um hashmap com a informação toda do hashmap de Users da Comunidade. Copia para um arraylist todos os Users do hashmap de Users e após isto cria um comparator que é utilizado para ordenar os Users por quantidade de Questions e Answers que possuem. Após isto os Ids dos N Users com mais Questions e Answers sao passados para o ArrayList de longs que é retornado.

2.3.4 Query 3

Recebe como inputs duas LocalDates, uma usada como data de início e outra usada como data de fim. Após isto são criados dois hashmaps um com a informação toda do hashmap das Questions e outro com a informação do Hashmap das Answers. Após isto percorrem-se todas as Questions e Answers, verificando quais têm data de criação dentro do intervalo de tempo. Se estiver dentro do intervalo de tempo então o contador usado para as Questions ou Answers, dependendo do que se tratar, será incrementado. No final, os dois contadores são retornados.

2.3.5 Query 4

Recebe uma String e duas LocalDates, uma usada como data de início e outra usada como data de fim. Após isto são usados dois arraylists, um com longs e outro com questions, e um HashMap que recebe a informação toda do HashMap das Questions. A seguir verifica se possui a string parâmetro como tag e se está contido dentro do intervalo de tempo definido. Se tudo isto se verificar, então acrescenta a Question ao arraylist de Questions. Após isto, é feito um comparator que é depois usado para organizar as Questions por data de criação. Após isto, selecionam-se os Ids das Questions organizadas por data e passam-se esses valores para o arraylist de longs que será retornado.

2.3.6 Query 5

Recebe como input um long, que é o id de um utilizador. Primeiro verifica-se se existe esse utilizador na comunidade: se não existir, devolve um par vazio, se existir, vai buscar o "aboutme"do utilizador e as perguntas e respostas todas, colocando-as em 2 listas separadas, organizadas por datas (das mais recentes às mais antigas). Após tal procedimento, aplica-se o algoritmo de organização de arrays merge, organizando por datas todos os posts, e concluindo com a criação da lista com os id's das perguntas e respostas mais recentes.

2.3.7 Query 6

Recebe como inputs um int N, e duas LocalDates, uma usada como data de início e outra usada como data de fim. São criados dois arraylists, um de longs e outro de answers. Depois é criado um HashMap de answers que recebe os valores da hashtable de answers. Após isto verifica-se se as answers estão dentro do intervalo de tempo. Depois é criado um comparator para ordenar por score as respostas. Após isto acrescenta-se os ids das answers do arraylist ordenado ao arraylist de longs, que depois é retornado.

2.3.8 Query 7

Recebe como inputs um int N e duas LocalDates, uma usada como data de início e outra usada como data de fim. São criados dois arraylists, um de longs e outro de questions. Após isto verifica se as questions estão dentro do intervalo de tempo, e se estiver, passa o clone da question para o arraylist de questions. Após isto, verifica-se se as respostas dessa question estão dentro do intervalo de tempo e se não estiverem, decrementamos o anscount. A seguir é utilizado um comparator que organiza o arraylist por anscount. Depois acrescenta-se o id das questions à lista de longs que será retornada.

2.3.9 Query 8

Recebe como inputs uma String e um int N. São criados dois arraylists, um de longs e outro de questions, e um HashMap de questions que recebe os valores do hashmap de questions. Após isto são acrescentadas N Questions que tenham a String dada como parâmetro no título ao arraylist de questions. Após isto é criado um comparator que organiza as questions por data e que é aplicado ao arraylist. Depois os ids das questions ordenadas são acrescentadas ao arraylist de longs que é retornado.

2.3.10 Query 9

Recebe como inputs dois ids de Users e um int N. São criados dois arraylists, um de longs e outro de questions, e um HashMap de users que recebe os valores do hashmap de users. Verifica se ambos os Users existem, e se existe relação nos posts entre os users das seguintes formas:

- O user com Id id1 fez a question e o user com Id id2 responde;
- O user com Id id2 fez a question e o user com Id id1 responde;
- Ambos os users respondem.

A seguir foi feito um comparator que organiza as Questions por data e é aplicado ao arraylist de Questions. A seguir copia-se o ids das Questions ordenadas para o arraylist de longs que é retornado.

2.3.11 Query 10

Recebe como input o id de uma question. Vai-se buscar ao HashMap de Questions a Question com esse id e após isto são percorridas as respostas desta e calcula-se a melhor resposta com base na fórmula existente no enunciado. Retorna-se o id desta.

2.3.12 Query 11

Recebe um int N e dois LocalDates, uma usada como data de início e outra usada como data de fim. São criados dois arraylists, ambos de Users. Foram criados também dois hashmaps, um que contém a HTU e outro a HTT. É adicionado a um dos arraylists de Users os Users do respetivo HashMap, e é feito um comparator para de seguida se organizar os Users por reputação. Após isto são adicionados ao outro arraylist de Users os N Users com melhor reputação. Depois para cada User vê-se as Tags das Questions deles e vão-se acrescentando as suas tagnames com os respetivos contadores(quantas vezes a tag foi usada naquele intervalo de tempo) ao hashmap que tem uma string como key e um int como value. Após este passo são definidos dois arraylists um de String e outro de longs. Adiciona-se ao de String o keySet do HashMap, que tem uma string como key e um int como value. Usando as Strings, são procurados os seus ids no hashmap de tags e são acrescentados por ordem ao arraylist de longs que é posteriormente retornado.

2.3.13 Clean

Aplica a todas as estruturas grandes da Comunidade a função clean definida na classe com o mesmo nome.

3 Conclusão

Depois de realizar o projeto, concluiu-se que este foi de grande importância no desenvolvimento de capacidades de trabalho em grupo e programação em Java, que foi uma linguagem que teve de ser aprendida à medida que o grupo desenvolveu o projeto, o que obrigou ao domínio das diferentes caraterísticas e estruturação específicas da programação orientada aos objetos num espaço de tempo bastante reduzido. Apesar de alguns desafios, particularmente a utilização de um parser baseado em SAX, o projeto tem a capacidade de responder ao que foi pedido com eficiência aceitável graças a soluções inteligentes como a simplificação de datas e boa utilização das estruturas fornecidas nas bibliotecas do Java.