# Laboratórios de Informática III - Projecto em Java

Adriana Meireles (a82582), Eduardo Jorge Barbosa (a83344), Filipe Monteiro (a80229)

***Resumo*—Este trabalho prático teve como objectivo replicar o anterior projeto (aplicado em C) mas agora em Java - a organização de grandes volumes de dados de forma a responder a questões, pré definidas, em tempo útil. Os objectos de estudo foram vários dumps do *Stack Exchange*.**

## I. INTRODUÇÃO

O fundamental deste trabalho prático consistiu em converter as existente estruturas e querys construidas no anterior projeto, em estruturas mais dinâmicas, de mais flexibilidade e portabilidade, assim como melhoras de desempenho por parte de *features* existentes na linguagem de programação Java.

## II. ESTRUTURAS DE DADOS

### A. MAIN STRUCT

```
public class Main_Struct {

private HashMap<Long,Profile> profiles;
private HashMap<Long, Post> posts;
private HashMap<String,Long> tags;
private Tardis tardis64;

}
```

Tal como no projeto em C, a estrutura principal é constituida por 3 HashMaps, uma para os perfis, outra para posts e outra para tags, e ainda a reformulado Tardis que contem os posts todos organizados pelo critério *x*.

### B. USER

Na Class de um Utilizador existem, entre os dados pessoais (nome, bio, ID, etc) uma TreeSet de Posts pertencentes ao utilizador (ordenados naturalmente do mais recente - mais antigo). Sendo uma Class tão básica, não foi necessário acrescentar extras para possivelmente melhorar desempenhos.

```
public class Profile{
    private TreeSet<Post> posts;
    private String about_me;
    private String name;
    private long n_posts;
    private long id;
    private int reputation;
}
```

Nesta foram implementados apenas os métodos básicos (construtores, gets, sets, clone, toString).

### C. POST, QUESTION, ANSWER

A Class Post é uma Class Abstracta de onde duas classes extendem: Question e Answer. Foi utilizada uma class Abstrata para podermos tratar destas duas subclasses como se fossem "do mesmo tipo", facilitando assim o uso destas.

Class Post:

```
public abstract class Post {

    private long type;
}

public class Question extends Post {

    private LocalDateTime creation_date;
    private HashMap<Long,Answer> answers;
    private String title;
    private List<String> tags; //TODO Maybe we can
    private long id;
    private long owner_id;
    private long n_answers;
    private long comments;
    private long score;
}

public class Answer extends Post {

    private LocalDateTime creation_date;
    private long id;
    private long owner_id;
    private long parent_id;
    private long comments;
    private long score;

}
```

Esta possui também um método compARE FALTA VER ISTO.

### D. TARDIS

//eduardo

### E. ESTRUTURAS AUXILIARES EXCEPÇÕES

Para facilitar o manuseamento de algumas querys, decidiu-se criar e/ou recriar algumas classes já implementadas pelo Java, mas com pequenos acréscimos. Nomeadamente:

```
public class BoundedTreeSet<E> extends TreeSet<E>
    private int limit;

    @Override
    public boolean add(E e){
        super.add(e);
        if(this.size() > this.limit){
        this.remove(this.last());
        }
        return true;
```

```
        }
}
```

Esta TreeSet, funciona exatamente como uma TreeSet habitual, perfeita para inserções ordenadas, mas tem uma informação e cuidado extra: tem um limite de tamanho. Sempre que estiver no tamanho máximo insere ordenadamente e retira o último, mantendo sempre o seu tamanho controlado. Isto foi nos útil particularmente na Query 5;

Em relação a comparadores, temos as seguintes:

- PostCreationDateComparator –> Compara dois Posts em relação á sua data de criação, de forma **Descendente**.
- ProfileNPostsComparator –> Compara dois Utilizadores em relacão ao número de posts, de forma **Descendente**.
- QuestionCreationDateComparator –> Compara duas Question em relação á sua data de criação, de forma **Descendente**.

Quanto a excepções, criamos as seguintes:

- NoPostFoundException –> Caso um ID nao exista entre os Posts.
- NoProfileFoundException –> Caso um ID nao exista entre os Perfis.
- NoTagFoundException –> Caso um ID nao exista entre as Tags. SUJEITO A TROCA

### *F. BENCHMARKS*