Instituto Politécnico de Setúbal – Escola Superior de Tecnologia de Setúbal

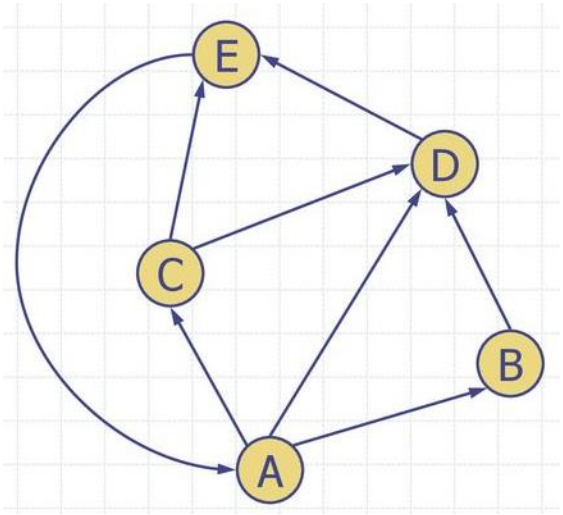
Licenciatura em Engenharia Informática

Unidade Curricular de Programação Avançada

Docente: André Sabino

Ano Letivo 2020/2021

SOCIAL NETWORK



Ana Rita Leal – 180221025

Francisco Moura – 180221015

Miguel Rosa – 180221023

Tiago Farinha – 180221011

Turmas SW-04 e SW-05

ÍNDICE

TADs IMPLEMENTADAS

Neste projeto, foram implementadas as TADs Stack, Queue, List e Map. A TAD Stack foi utilizada de acordo com o padrão de Software Memento. (ex: classe Caretaker)

```
public class Caretaker {
    private final SocialNetwork socialNetwork;
    private final Stack<Memento> undo;
```

A TAD Queue foi utilizada no método que permite percorrer o dígrafo em largura. (ex: classe DirectGraph)

```
private ArrayList<Vertex<V>> BFS(Vertex<V> v) {
    ArrayList<Vertex<V>> path = new ArrayList<>();
    Set<Vertex<V>> visited = new HashSet<>();
    Queue<Vertex<V>> queue = new LinkedList<>();
    visited.add(v);
    queue.add(v);
    while (!queue.isEmpty()) {
        Vertex<V> vLook = queue.remove();
        path.add(vLook);
        for (Edge<E, V> edge : outboundEdges(vLook)) {
            if (!visited.contains(edge.vertices()[1])) {
                visited.add(edge.vertices()[1]);
                queue.add(edge.vertices()[1]);
            }
        }
    }
    return path;
}
```

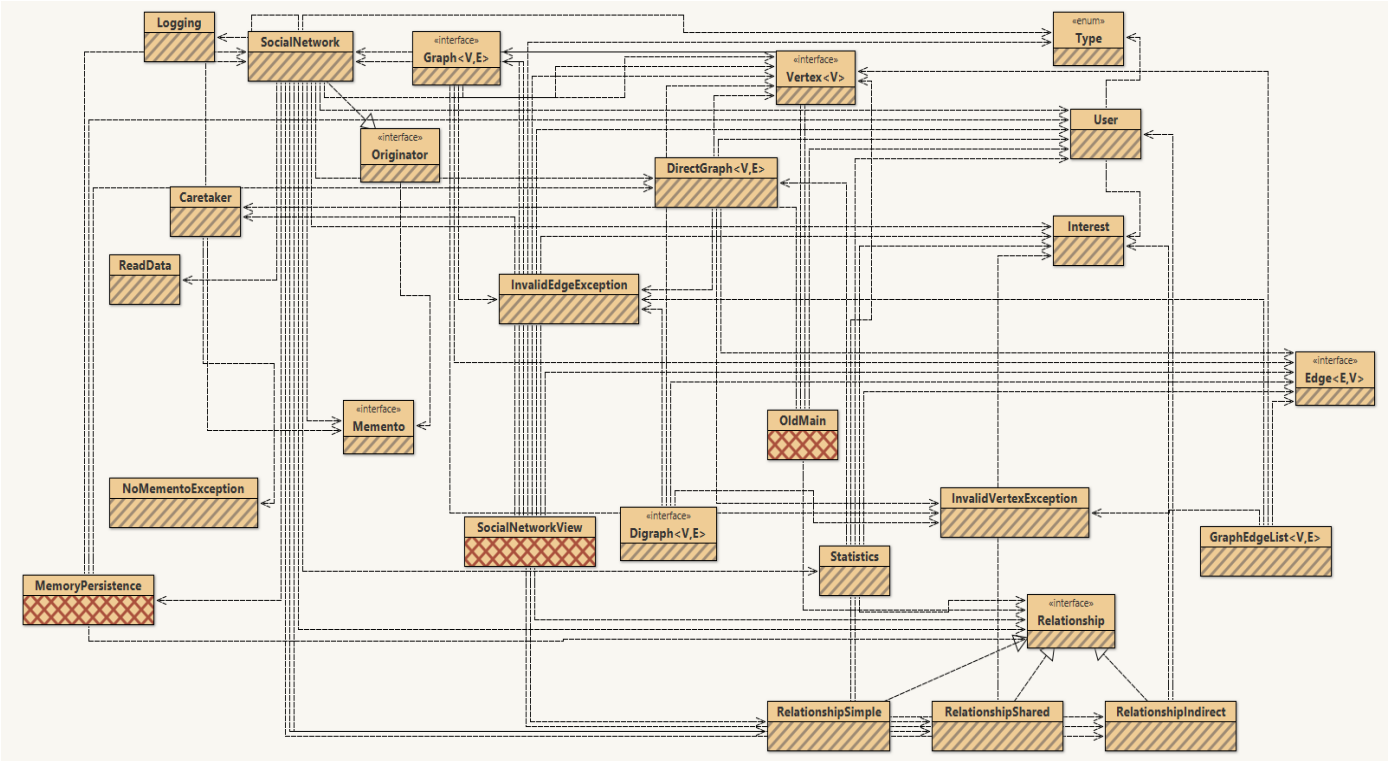
As TADs List e Map foram utilizadas várias vezes ao longo do projeto. (ex: classes Interest e SocialNetwork)

```
public class Interest implements Serializable {
    private final int id;
    private final String name;
    private final ArrayList<String> idsOfUsers;
```

```
public class SocialNetwork implements Originator, Serializable {

    private final HashMap<Integer, User> users;
    private final HashMap<Integer, ArrayList<String>> relationships;
    private final HashMap<Integer, Interest> interests;
    private final Logging log = Logging.getInstance();
    private final Statistics statistics;
    private DirectGraph<User, Relationship> graph;
    private String userNamesFile, relationshipsFile, interestNamesFile, interestsFile;
    MemoryPersistence memoryPersistence;
```

DIAGRAMA DE CLASSES



PADRÕES DE SOFTWARE

Foram utilizados os padrões Strategy, Singleton, Memento, DAO e uma variante do MVC abordada nas aulas teóricas denominada de ModelView na realização deste projeto. Contudo, consideramos que poderíamos ter implementado os padrões Observer e a totalidade do MVC.

O padrão Memento permitiu que guardássemos o estado do dígrafo em utilização no SocialNetworkView à medida que o fossemos percorrendo e realizando as ligações entre os vários Users.

O padrão Data Access Object (DAO) permitiu que guardássemos objetos, instanciando-os com o modo de persistência pretendidos e utilizando métodos para aceder ao objeto pretendido. Neste caso, os objetos encontram-se guardados em ficheiros no formato Json e foram criados métodos para salvar e atualizar um dado ficheiro.

O padrão Strategy foi utilizado para criar uma interface com um template genérico e facilmente adaptável, como forma de divisão dos vários relacionamentos em diretos simples, diretos com partilha de interesses e indiretos, facilitando as suas interligações e o funcionamento com a interface Serializable.

Relativamente ao padrão Observer, consideramos que o mesmo poderia ser relevante no que toca a notificar o utilizador quando é feita uma atualização a um modelo guardado em disco. Desta forma, sempre que o utilizador atualizasse um modelo anteriormente guardado, seria notificado sempre que fossem feitas alterações ao estado atual do mesmo.
Por fim, teríamos aplicado a totalidade do padrão MVC para dividir responsabilidades da classe SocialNetworkView que trata da visualização da classe SocialNetwork., Dessa forma, estaria implementando uma classe Model, uma classe View e outra Controller.

POSSÍVEIS BAD SMELLS

Ao longo do nosso projeto, deparámo-nos principalmente com três possíveis Bad Smells:

- Duplicated Code (código duplicado)
- Message Chains (chamada sucessiva de métodos)
- Long Methods (métodos muito extensos)

Apesar disso, consideramos que, com a implementação dos padrões de software indicados que não foram implementados, estes bad smells possam ser evitados tornando o código mais eficiente e melhor estruturado.