

**Algoritmos e Estruturas de Dados – 2021/2022**

**1º Semestre**

**Relatório Projeto BdFi**

**(Base de Dados de Filmes na Internet)**

**Realizado por:**

**60236, Guilherme Poças**

**61052, João Oliveira**

Índice

[1 Introdução 2](#_Toc90049453)

[2 Estrutura do projeto 3](#_Toc90049454)

[2.1 Diagrama de classes 3](#_Toc90049455)

[2.2 Tipos Abstratos de Dados 4](#_Toc90049456)

[2.2.1 Database 4](#_Toc90049457)

[2.2.2 Show 4](#_Toc90049458)

[2.2.3 Person 4](#_Toc90049459)

[2.2.4 Participation 5](#_Toc90049460)

# Introdução

Durante o 1º semestre do 2ª ano, na disciplina de Algoritmos e Estruturas de Dados, abordamos complexidades temporais e estruturas de dados. A fim de colocarmos em prática os temas aprendidos, foi-nos pedido para fazer um projeto em duas fases. Uma primeira fase em que o programa teria muitos poucos dados e numa segunda fase que tivesse preparado para ter milhares de dados e funcionar de forma eficiente.

Neste relatório vamos abordar as estruturas de dados utilizadas e a justificação de a termos usado. Iremos também considerar as complexidades temporais das várias operações que o nosso programa suporta bem como a complexidade espacial do programa.

# Estrutura do projeto

Neste capítulo está o diagrama de classes do nosso projeto e também uma explicação das as estruturas de dados usadas.

## Diagrama de classes

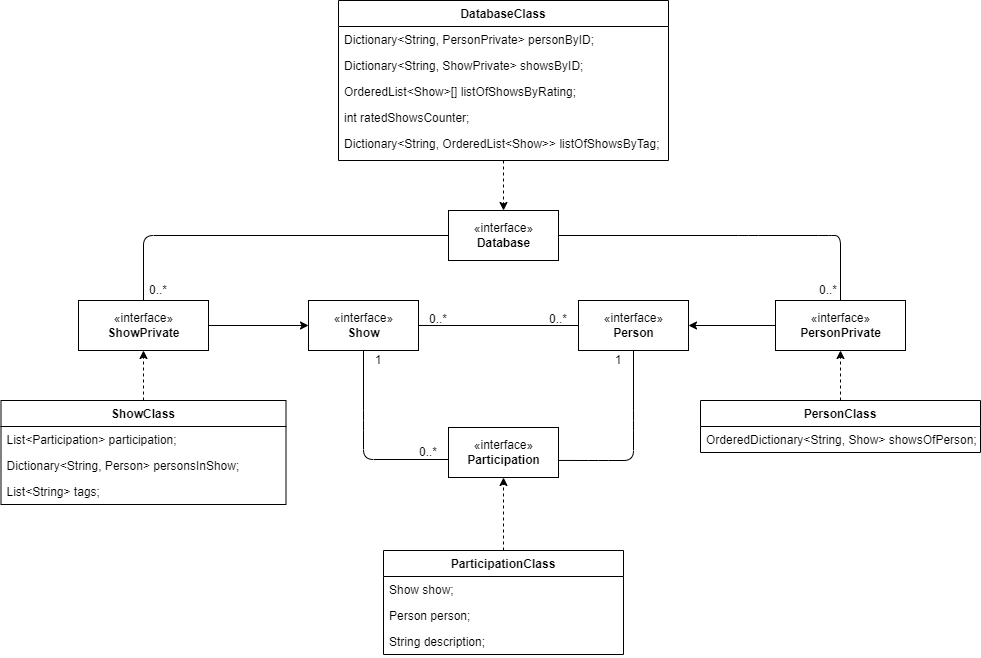


Figura - Diagrama de Classes

Na imagem acima encontra-se um diagrama com as classes e interfaces do nosso programa, com as estruturas de dados presentes em cada uma.

## Tipos Abstratos de Dados

### Database

A Database é a interface que é implementada pela DatabaseClass. Lá definimos 4 estruturas de dados:

1. Dictionary<String, PersonPrivate> personByID;
2. Dictionary<String, ShowPrivate> showsByID;
3. OrderedList<Show>[] listOfShowsByRating;
4. Dictionary<String, OrderedList<Show>> listOfShowsByTag;

Tanto na 1ª como 2ª, as implementações são SepChainHashTable. Optámos por ela pois não precisamos de manter a ordem de inserção dos elementos. E tanto a pesquisa como a remoção têm complexidade constante. A chave é o id da person e do show, respetivamente, e o valor é a personPrivate e o showPrivate, respetivamente.

A 3ª é um array normal de OrderedList que implementam uma lista ligada ordenada. É um array com tamanho de 11, sendo que em cada posição estão todos os shows com o rating correspondente ao índice do array. Os shows em cada posição estão ordenados por título.

Na 4ª, à semelhança da 1ª e 2ª, a implementação é um SepChainHashTable. A chave é uma tag. O valor é uma lista ordenada, implementada por uma lista ligada ordenada, que contém os shows com a tag da chave. Os shows estão ordenados por titulo.

### Show

O Show representa um programa da Base de Dados, no qual podem participar Pessoas e ao qual pode ser atribuído uma avaliação ou uma tag.

Define 3 estruturas de dados:

1. List<Participation> participation;
2. OrderedDictionary<String, Person> personsInShow;
3. List<String> tags;

Na 1ª e 3ª, as implementações serão DoubleList, pois na 1ª estrutura é necessário manter os dados por ordem de inserção, para uma futura listagem, e na 3ª apenas utilizaremos o seu iterador, que tem complexidade constante, sendo a ordem indiferente.

A 2ª estrutura tem uma implementação SepChainHashTable, que será bastante eficiente, visto que não queremos adicionar Persons iguais (com o mesmo id) e esta estrutura tem complexidade constante na inserção, e porque apenas utilizaremos o seu iterador, que tem complexidade constante.

#### ShowPrivate

O ShowPrivate será um interface que estende o Show com os métodos que o modificam de alguma forma, visto

### Person

#### PersonPrivate

### Participation