

Relatório do trabalho prático – parte I

MovieDatabase

Sara lamela moreira

Aluno nº 8544

emanuel de jesus correia carvalho

Aluno nº 8820

Trabalho realizado sob a orientação de:

Luís Ferreira

**Linguagens de Programação II**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, 26 de Abril de 2020

Índice

[1 Introdução 3](#_Toc38798766)

[2 Classes e herança 4](#_Toc38798767)

[3 Estrutura de Dados e Diagrama de Classes 5](#_Toc38798768)

[4 Funcionalidades 7](#_Toc38798769)

[5 Conclusão 9](#_Toc38798770)

[Bibliografia 11](#_Toc38798771)

Lista de Figuras

Figura 1: Diagrama de Classes 6

# Introdução

No âmbito da disciplina de Linguagem de Programação, desenvolvemos um programa de gestão de Filmes e utilizadores, sendo que este ainda está na sua construção. Nesta primeira parte do projeto desenvolvemos o esqueleto do programa, a sua estrutura e definimos as suas Classes principais, assim como as suas heranças base. Ao longo do segundo projeto estes podem sofrer alterações.

# Classes e herança

Algo que é definido como *Class* é do tipo de referência. Estas classes contêm variáveis com conteúdo nulo que após serem instanciadas pelo operador *new* podem ser atribuídos dados manualmente, podem ser atribuídos dados que definimos nos construtores dessa classe como dados base, ou que pode ser atribuído um objeto de um tipo compatível criado noutra parte do código. Estas classes possuem blocos de código caracteristicamente atribuídos sendo que este pode ser herdado por outra classe. Esta herança permite que uma segunda classe (classe “filha”) consiga aceder aos dados da classe “pai” sem que haja repetição de código, completando-o com as suas próprias características.

No nosso projeto foram criadas três classes “pai” – Movie, Movies e Person – temos também três classes “filhas” – MovieDescription, MovieDisplayInfo e User. Estas são as classes que temos na totalidade sendo que poderemos ver a relação entre elas no diagrama de classes deste relatório.

# Estrutura de Dados e Diagrama de Classes

Como referido no capítulo anterior, o código está estruturado pelas seguintes classes: Movie, Movies, Person, MovieDescription, MovieDisplayInfo e User.

As ligações entre elas são as seguintes:

**Movie** (classe “pai”) **-> MovieDescription, MovieDisplayInfo** (classes “filhas”).

**Movies** (classe “pai” sem “filhos”).

**Person** (classe “pai”) -> **User** (classe “filha”).

**Movie** tem como objetivo conter a informação principal de cada filme. É obrigatório para a criação do mesmo, mas como apenas contém dados básicos é possível adicionar um filme que ainda não tenha estreado.

**MovieDescription** acrescenta dados à classe pai podendo conter detalhes mais específicos de cada filme, sendo que é possível o utilizador votar no filme atribuindo uma pontuação.

**MovieDisplayInfo** acrescenta dados à classe pai podendo conter o cartaz do filme e o seu trailer com uma função de play e stop que faz a contagem de reprodução do vídeo e nos informa do momento de paragem.

**Movies** interligado à classe **Movie** e **MovieDescription** através de dois campos: movieList e movieRatingList. Esta classe vai ter o objetivo de guardar a informação que está em cada classe no seu respetivo array para poder aceder à totalidade de dados que o repositório tem e permite através das suas funções alterar certos dados.

**Person** tem como objetivo conter a informação principal de quem publica o filme, este contém os dados mais pessoais.

**User** acrescenta dados à classe que vão ser visíveis, mas que são menos pessoais como o User ID e o User Name.

Existem dois enumerados fora das classes, mas que estão ligados às respetivas, sendo que são dados alteráveis através da classe **Movies**.

**Program** é onde está contido o *main* do nosso projeto.

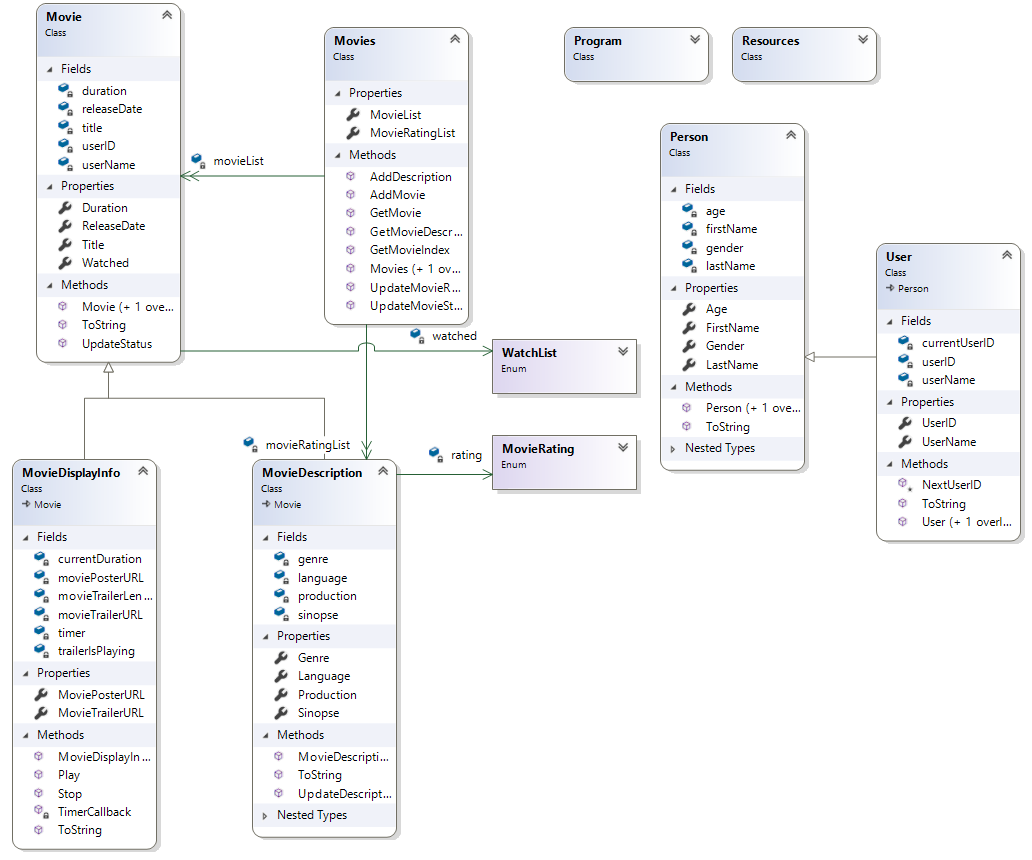


Figura 1: Diagrama de Classes

# Funcionalidades

Neste projeto foram implementadas de momento dois tipos de funcionalidades principais.

A primeira funcionalidade principal é a de registar novos filmes, as suas descrições e as suas informações de Display (poster e trailer) mostrando quem o registou. Para esta funcionar utilizamos as classes e as suas heranças e funções *ToString* para transmitir os dados que são colocados manualmente no *main* ao ser instanciadas.

A segunda funcionalidade principal é a de poder alterar se o filme já foi visto pelo utilizador ou não e a sua votação do filme. Para esta funcionar criamos uma classe **Movies** que reúne os registos de outra classe e através de *arrays* guarda esses registos.

Esta procura, através do título do filme, se este existe, e caso sim, devolve a informação contida no array desse filme ou, numa diferente função, devolve o índice desse array sendo que com estes dados conseguimos invocar funções para atualizar os dados mencionados acima.

# Conclusão

De uma vista geral, o projeto em desenvolvimento está a ter um bom rumo. Com o que lecionamos até ao momento conseguimos estruturar e desenvolver um esqueleto para este projeto que pode prosperar bastante, aplicando a matéria e desenvolvendo os conhecimentos adquiridos. Esperamos durante o resto do semestre conseguir a capacidade de desenvolver muito mais o mesmo para trabalhar na segunda parte do projeto, como desenvolver as funcionalidades da aplicação, fazer um interface e usar uma base de dados.

# Bibliografia

Stack Overflow - *Where Developers Learn, Share, & Build Careers*

Disponível em: <https://stackoverflow.com/>

Panjuta, Denis (último update 2019) Complete C# Masterclass: *Learn C# Programming - WPF, Databases, Linq, Collections, Game Development with Unity. More than just the C# basics!*

Disponível em: <https://www.udemy.com/course/complete-csharp-masterclass/>