

**Instituto Superior de Engenharia de Lisboa**

Engenharia Informática e de Computadores

*Programação na internet*

*Trabalho Prático*

**Autores:**

Ricardo Gonçalves, nº 37323

Hugo Barrocas, nº 37330

Hugo Reis,nº38652

**Lisboa, 2 de novembro de 2017**

**Índice**

Índice

[Introdução 3](#_Toc497418369)

[Modelo de Dados (DTO) 3](#_Toc497418370)

[Movie 3](#_Toc497418371)

[MovieDetails 3](#_Toc497418372)

[Ator 3](#_Toc497418373)

[Descrição funcional 4](#_Toc497418374)

[Informação complementar Fornecida 4](#_Toc497418375)

[Implementação 4](#_Toc497418376)

[ANEXOS 4](#_Toc497418377)

# Introdução

Foi solicitado a criação de uma aplicação web para consulta de filmes e o seu cast Atores e diretores , e também a consulta de Atores com biografia e filmes em que participaram

Para execução do trabalho foi utilizado a linguagem Javascript suportada em Node.js

# Modelo de Dados (DTO)

De acordo com a solicitação implementamos o Modelo de dados ilustrado abaixo.

Onde temos 3 objetos principais

### Movie

Objeto que é usado quando são feitas as pesquisas por um nome de um filme e que contém informação simples sobre o filme

### MovieDetails

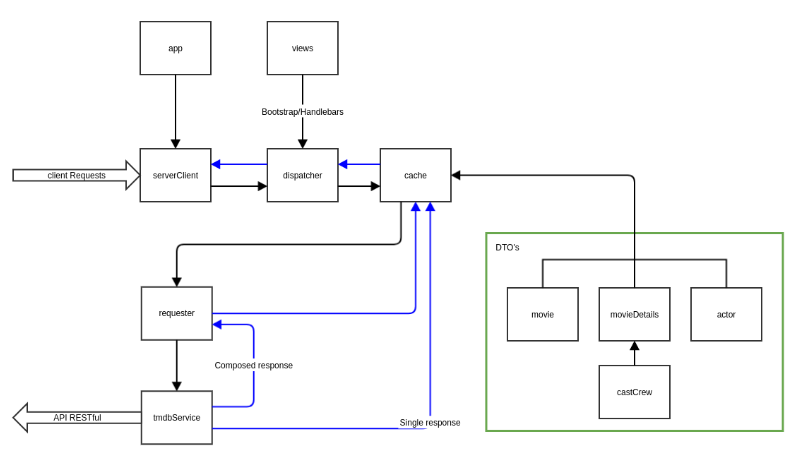
Este objeto contem informação detalhada sobre um filme em concreto e contem também os atores que nele participam e os seus diretores.

### Ator

Este objeto é utilizado para guardar a informação detalhada de um ator, a sua biografia e também em que filmes esteve presente

### Cast crew

Este objecto é utilizado internamente pelo moviedetails para criar os seus actores e os directores, para poderem ser solicitados pela camada de apresentação.



# Descrição funcional

Na nossa aplicação tentamos aplicar o modelo abaixo assinalado, onde tentamos estabelecer um caminho de comunicação por camadas, em que cada camada comunica com um módulo e não existem saltos de comunicação entre módulos de forma a torna-la o mais modular e flexível possível.

Para iniciar a aplicação é necessário correr o comando “Node App.js” que inicia o servidor a escuta na porta 8080

O ponto de entrada para o pedido do cliente é o modulo *serverclient* que recebe valida e trata as url e caso sejam válidas para as nossas pesquisas, inicia a comunicação para o *dispatcher*.

O *dispatcher* têm o papel de efetuar o roteamento do pedido recebido do *serverclient* e dar inicio a execução dos módulos necessários para a construção do mesmo.

No módulo *dispatcher* é também onde são construídas as páginas Html recorrendo as frameworks *handlebars* e *bootstrap*, sendo que nesta fase estão a ser usadas de uma forma simples para o display dos resultados.

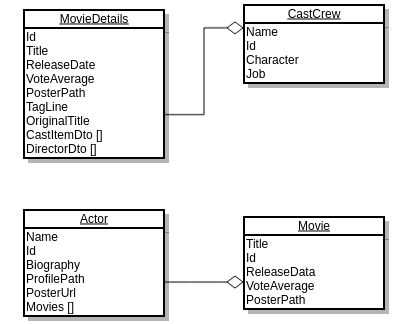
O Modulo cache é onde são construídos os nossos “objetos” de acordo com a especificação, para poderem ser armazenados para não serem repetidos os pedidos que já foram feitos anteriormente, seja de filmes , seja de atores.

A cache é também responsável por manter a cache dentro dos limites de forma a não criar problemas de *resource starvation* , neste caso memória, por omissão o valor é de 1000 entradas para atores e filmes.

O modulo *requester* , é responsável e fazer os pedidos a API externa RESTfull *api.themoviedb.org.*  quando os objetos não estão presentes na cache.

Para alem dos pedidos a API o *requester* também compõe os pedidos necessários para a composição de objectos internos na nossa aplicação quando os mesmos necessitam de informação de mais que uma fonte.

O modulo *tmdbservice* é quem executa os pedidos ah aplicação externa e compõe os Url’s com base nos dados fornecidos pelo requester



# Optimizações e problemas conhecidos

1-Melhorar o tratamento de exceções

2-Foram detetadas algumas in direções no código que serão ser minimizadas

3- melhoramento do tratamento da cache.

4-parametrizações da aplicação através de consola ou ficheiro ini

5-Tornar os módulos agnósticos de forma a que não tenham dependências explicitas de outros módulos de forma a facilitar a criação de testes