Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Modelação e Padrões de Desenho– Semestre de Verão – 2017/2018

Trabalho 3

Objectivos: Prática com `CompletableFuture<T>` e API assíncrona.
Data limite de entrega: 18 de Junho de 2017

NOTA: Implemente **os testes unitários necessários** e altere os existentes, de modo a validar o funcionamento das funcionalidades pedidas em todas as partes do trabalho.

No âmbito da biblioteca `movapi` pretende-se tornar a sua API assíncrona. Deverá criar um novo projeto `movasync` com base no anterior `movlazy` e adaptá-lo de acordo com os requisitos deste enunciado.

Parte 1

De modo a que biblioteca `movlazy` passe a usar IO não-bloqueante, altere a interface `IRequest` para que o seu método `getBody()` passe a retornar `CompletableFuture<String>`, devendo as implementações desta interface ser actualizadas em conformidade.

Da mesma forma devem ser actualizados o modelo de domínio, `MovWebApi` e `MovService` para que passem a oferecer uma API assíncrona. Ou seja, métodos e *getters* que retornavam `Supplier<T>` passam a retornar `CompletableFuture<T>` e os que retornavam `Supplier<Stream<T>>` passam a retornar `CompletableFuture<Stream<T>>`.

~~Excepção ao método `search()` de `MovService` que continua a retornar `Supplier<Stream<SearchItem>>`. Neste caso~~ Para o método `search()` de `MovService` terá que desenvolver uma solução que desencadeie os pedidos HTTP de todas as páginas na chamada ao `search()` e cujo resultado é agregado num **`CompletableFuture<Stream<SearchItem>>`** ~~*lazy* produzida por um `Supplier`~~.

As estruturas de dados para *cache* `movies`, `cast` e `actors` passam a ser do tipo `ConcurrentHashMap` e a armazenar como valor um `CompletableFuture`.

Devido à natureza reactiva do modelo de domínio o número de pedidos HTTP realizado na obtenção de um objecto poderá ser **maior** que o realizado com `movlazy` do Trabalho 2, pelo que terá que ajustar alguns dos resultados dos `asserts` dos testes unitários. No caso do teste unitário `testSearchMovieWithManyPages` poderá substituir a query `movapi.search("candle")` por outra que retorne apenas 2 ou 3 páginas como por exemplo `search("galo")`.

No caso de testes sobre dados em ficheiro terá que criar/gerar os respectivos ficheiros em falta. No relatório deve explicar como resolveu a criação dos ficheiros de *mock* sobre os dados obtidos de https://api.themoviedb.org.

Pode ainda existir a necessidade de operar mais que uma vez sobre o `Stream` resultante de um `CompletableFuture<Stream<...>>` pelo que deve ter em conta essa situação nos testes unitários.

NOTA 1: **não** poderá criar ou usar explicitamente fios de execução (i.e. `Thread`), **nem** por diferimento de tarefas (i.e. `CompletableFuture.supplyAsync(...)`) nem através de qualquer outro meio.

NOTA 2: **não** poderá bloquear sobre o resultado das computações assíncronas (i.e. `.join()` ou `.get()`) com excepção aos testes unitários e nas continuações que constroem o `Stream<SearchItem>` retornado pelo método `search()`.

Parte 2

Implemente uma aplicação Web usando a tecnologia VertX com *handlers* assíncronos. A aplicação deve disponibilizar as seguintes páginas:

1. Listagem de filmes com um determinado nome recebido por *query-string*. Cada filme tem um *link* para a sua página de detalhes.
2. Detalhes de um filme incluindo um *link* para a página de listagem de créditos desse filme.
3. Listagem dos créditos de um filme. Cada crédito tem um *link* para os detalhes dessa pessoa.
4. Detalhes de uma pessoa incluindo um link para a página de listagem de filmes dessa pessoa.
5. Listagem de filmes de uma pessoa.

As páginas anteriores são acessíveis através dos seguintes caminhos (*paths*):

1. `/movies?name=...`
2. `/movies/:id`
3. `/movies/:id/credits`
4. `/person/id`
5. `/person/id/movies`

A aplicação web nunca poderá bloquear (não fazer `join()` e nem `get()`) na obtenção de um resultado de um `CompletableFuture`.

As páginas de listagens devem ser retornadas no corpo da resposta HTTP em modo *chunked* (`response.setChunked(true)`).