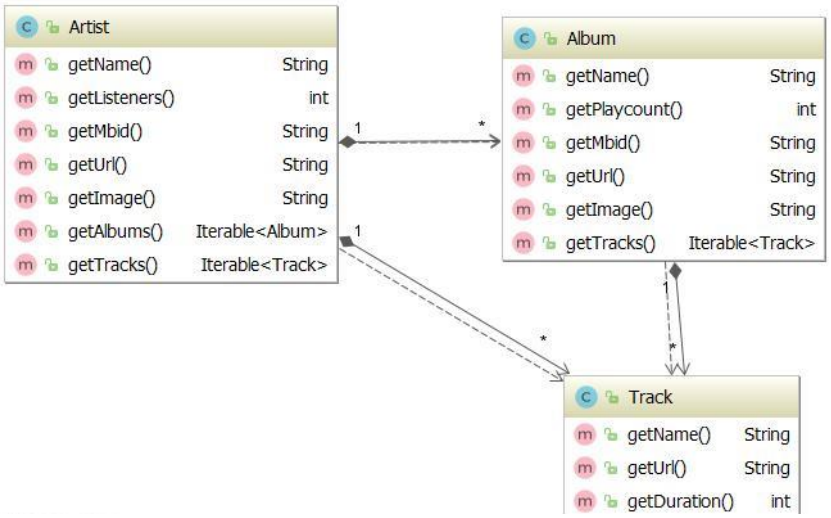
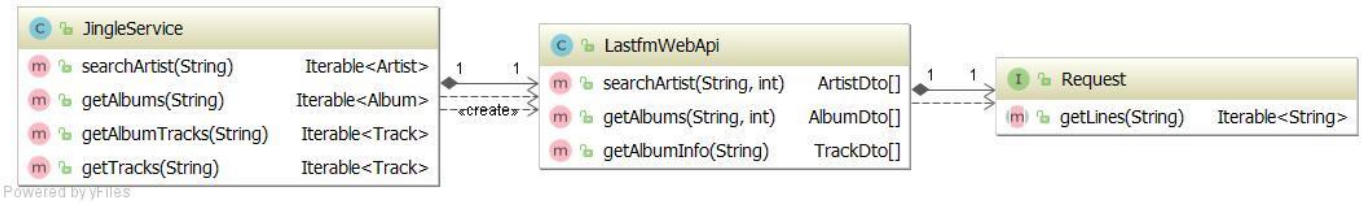
Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Modelação e Padrões de Desenho
2019

A biblioteca `jingle-lazy` disponibiliza informação detalhada sobre artistas, álbuns e músicas. Os dados são obtidos a partir de uma API RESTful: https://www.last.fm/api. O modelo de domínio é formado pelas entidades: `Artist`, `Album` e Track e obedece à especificação apresentada no diagrama de classes seguinte:



As classes do modelo de domínio estão implementadas no módulo `jingle-lazy`. Todas as relações entre as entidades de domínio são mantidas de forma lazy.

A instanciação e navegação dos objectos de domínio é feita por JingleService que recorre à classe LastfmWebapi para realizar os pedidos à Web Api de last.fm.



A biblioteca jingle-lazy recorre à biblioteca jingle-util para executar tarefas auxiliares tais como operações sobre sequências implementadas pela classe LazyQueries com os seguintes operações:

* `last(Iterable<T> src)` - retorna o último elemento da sequência `src`;
* `from(T[] items)` - retorna uma nova sequência lazy com os elementos de `items`.
* `takeWhile(Iterable<T> src, Predicate<T> pred)` - retorna uma nova sequência lazy com os primeiros elementos de `src` que verificam o predicado `pred`
* `flatMap(Iterable<T> src, Function<T, Iterable<R>> mapper)` - retorna uma nova sequência lazy com o resultado das transformações de cada elemento de `src` pela função `mapper` num `Iterable<R>` combinados numa única sequência. Exemplo, dada uma sequência de palavras: “isel”, “super”, “ola”, então uma operação de `flatMap` sobre esta sequência com uma transformação de `String -> Iterable<char>`, deve resultar num `Iterable<char>` com: `‘i’, ‘s’, ‘e’, ‘l’, ‘s’, ‘u’, ‘p’, ‘e’, ‘r’, ‘o’, ‘l’, ‘a’`.

`LastFmWebAPI` usa dados das seguintes rotas da Web API last.fm:

* https://www.last.fm/api/show/artist.search -- pesquisa artistas com um dado nome.
* https://www.last.fm/api/show/artist.getTopAlbums -- obter os albums de um artista identificado pelo seu mbid.
* https://www.last.fm/api/show/artist.getInfo -- obter as músicas (_tracks_) de um álbum identificado pelo seu mbid.

Os resultados da API RESTFul podem ser convertidos através da biblioteca Gson para instâncias de classes pré-definidas (DTOs).

Os métodos de `LastfmWebapi` recebem um segundo parâmetro inteiro correspondente ao número da página. Nestes casos o método correspondente de JingleServive retorna um iterável que percorre os elementos de todas as páginas disponíveis até ser obtida uma página sem elementos. Para tal executa um encadeamento de operações semelhante ao seguinte:

iterate(…) -> 1,2,3,.. -> map(invoque LastfmWebApi) -> arr1, arr2, … -> takeWhile(arr.length != 0) -> flatMap(from(arr)) -> map(dto -> model)

No caso do método `getTracks(String artistMbid)`, repare que este recorre aos 2 métodos `getAlbums` e `getAlbumsTracks` para obter todas as músicas do artista identificado por `artgistMbid`.

Na classe `Artist` o método `getTracksRank(String country)` retorna uma sequência de instâncias de `TrackRank` com as propriedades: `name`, `url`, `duration` e `rank`, esta última de tipo `int`. A propriedade `rank` tem o ranking dessa música (_track_) para o país passado em `country`. Caso aquela música não conste do ranking do país então `rank` tem o seu valor por omissão 0 (zero).

O método `getTracksRank(String country)` segue a mesma abordagem dos métodos `getAlbums` e `getTracks`, isto é com recurso a um método `getTracksRank(String artistMbId, String country)` de `JingleService` que por sua vez recorre à classe `LastfmWebApi`.

Abordagem:

1. `StreamUtils` o método `merge` junta duas sequências com os critérios do exemplo seguinte. Dado `seq1 = {“isel”, “ola”, “dup”, “super”, “jingle” }` e `seq2 = {4, 5, 6, 7}` então fazendo o `merge` que tenha como critério de junção `(str, nr) -> str.length == nr` e como transformação `(str, nr) -> str + nr` resulta numa sequência `{“isel4”, “ola0”, “dup0”, “super5”, “jingle6”}`

2. `LastfmWebApi` o método `getTopTracks(String country, int page)` obtém uma página do ranking de musicas num país através do método geo.getTopTracks da Restful API de Lastfm:
https://www.last.fm/api/show/geo.getTopTracks.

3. `JingleService` o método `getTopTracks(String country)` retorna o ranking completo de todas as músicas num país.

4. `JingleService` o método `getTracksRank(String artistMbId, String country)` faz o _merge_ das sequências dadas por ` getTracks(String artistMbid)` e `getTopTracks(String country)`. Caso uma música de `artistMbid` não conste no ranking dado por `getTopTracks` então essa música fica com o ranking 0.

O módulo jingle-async usa IO não-bloqueante, com uma nova interface `AsyncRequest` com um método getLines()` com API assíncrona. A implementação desta interface para pedidos HTTP GET recorre a uma biblioteca para realização de pedidos HTTP não bloqueantes, i.e. https://github.com/AsyncHttpClient/async-http-client.

Os tipos do modelo de domínio oferecem uma API assíncrona baseada no tipo Observable.

A aplicação Web usa a tecnologia VertX com *handlers* assíncronos. A aplicação disponibiliza as seguintes páginas:

1. Listagem de artistas com um determinado nome recebido por *query-string*. Cada artista tem 2 *links*: para uma listagem dos seus álbuns e outro para uma listagem das suas músicas.
2. Listagem de álbuns de um artista. Cada álbum tem um link para a listagem de músicas desse álbum.
3. Listagem de todas as músicas de um artista.
4. Listagem de músicas de um álbum.

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

1. /artists?name=…
2. /artists/:id/albums
3. /artists/:id/tracks
4. /albums/:id/tracks

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

As listagens são retornadas no corpo da reposta HTTP em modo *chunked* (response.setChunked(true)) sendo a user-interface construída de forma progressiva à medida que a resposta é recebida no browser.