ISEL

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura Informática e de Computadores

Modelação e Padrões de Desenho

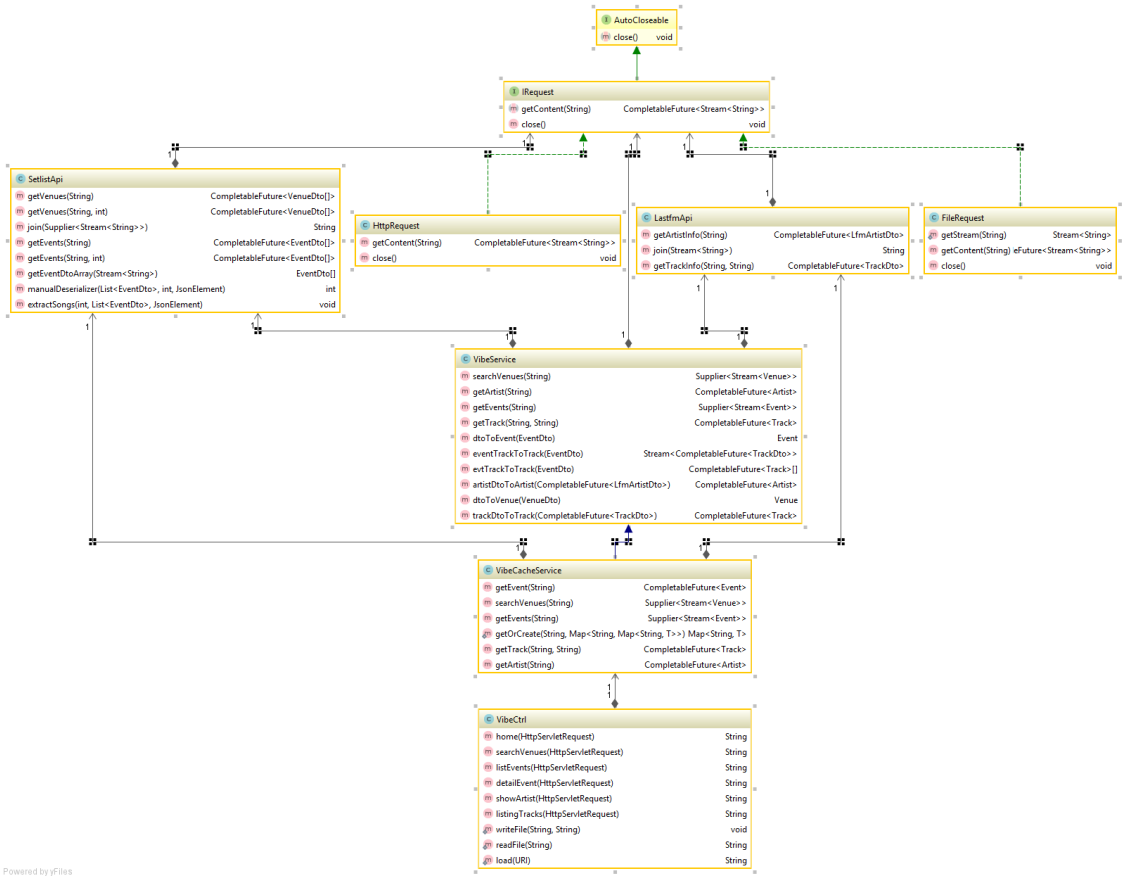
Relatorio do 3º Trabalho

Semestre de Verão

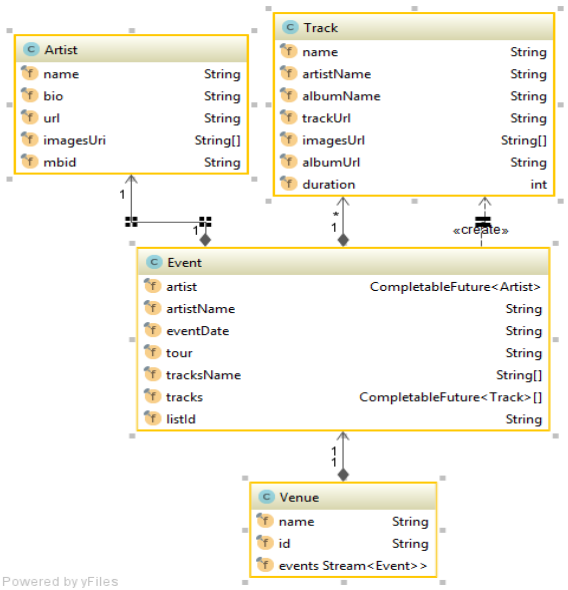
2016/2017

Sara Sobral	40602
Rafael Pereira	40680
Bruno Filipe	41484

Arquitetura:



Modelo de domínio:



Sara Sobral	40602
Rafael Pereira	40680
Bruno Filipe	41484

Endpoints da Aplicação web:

A nossa aplicação web disponibliza os seguintes URIs:

- **/home** - Página principal da aplicação que permite procurar por venues numa cidade.
- **/search/venues** – Retorna listagem de venues para uma dada location.
- **/events/{id}** – Sendo {id} o identificador da venue devolve os eventos dessa venue.
- **/events/details/{id}** – Sendo {id} o identificador do evento retorna informação detalhada do evento.
- **/artist/{name}** – Sendo {name} nome do artista devolve informação detalhada de um artista.
- **/events/tracks/{id}** - Sendo {id] o identificador do evento, retorna a lista de tracks(musicas) tocadas pelo artista nesse evento.

O nosso trabalho segue o modelo MVC (Model View Controller) sendo que o controller é responsável por preencher as views com os dados do modelo.

VibeCtrl (Controller):

Ele apresenta as Caches de páginas html e tem os métodos a ser chamado para cada mapeamento além disso compila as views a serem utilizadas.

String home (HttpServletRequest req) – Apenas retorna o partial com o layout da página principal.

String *searchVenues* (HttpServletRequest req) - Recebendo a localização (cidade) pela query string do caminho, executa o método searchVenues do serviço para aquela cidade, construindo uma página html com uma lista das venues pelo handlebar com o template específico.

String *listEvents* (HttpServletRequest req) - Recebendo o identificador dos recintos a partir do caminho do pedido web, executa o método getEvents do serviço para aquele recinto, construindo uma página html com uma lista das venues pelo handlebar com o template específico, sendo que cada evento tem um link para a página dos seus detalhes, a partir do identificador uníco de eventos.

String *detailEvent* (HttpServletRequest req)- Recebendo o identificador do evento a partir do caminho do pedido web , executa o método getEvent do serviço para aquele evento especifico, construindo uma página html com uma lista das venues pelo handlebar com o template específico, sendo que tem um link para a página do artista e dos detalhes das músicas tocadas, estas duas páginas são criadas por continuações e armazenadas em cache e em disco, por isso é impedida a espera pela conclusão da

construção destas paginas. A página dos detalhes do evento também é guardada em cache e em disco.

String showArtist (HttpServletRequest req) - Vai buscar a página do artista, gerada previamente aquando da construção da página dos detalhes do evento, recorrendo ao artista especifico de cada evento, criado a partir do serviço com o método *getArtistInfo()*, usando o template criado.

String *listingTracks* (HttpServletRequest req) - Vai buscar a página da listagem de músicas, gerada previamente aquando da construção da página dos detalhes do evento, recorrendo ao conjunto de músicas especificas de cada evento, criado a partir do serviço com o método *getTrackInfo()* para cada música desse mesmo conjunto, usando o template criado.

Views:

Realizamos as nossas views com recurso a biblioteca Handlebars defindo templates com codigo html.

Implementação das Caches:

Ao nivel da aplicação web as caches (Event, Artist, Track) de páginas estão implementadas à custa de um mapa sendo o valor delas uma String com o conteúdo html dos ficheiros handlebars. Apenas diferem nas suas chaves, a cache de evento tem como chave o id do evento, a cache de artist tem o nome do artista e a track o id do evento.

Sara Sobral 40602
Rafael Pereira 40680
Bruno Filipe 41484