Relatório

1ºParte



Relatório elaborado por:

36893 – Gonçalo Domingues

36919-André Luz

Introdução

Esta primeira parte projeto visa a construção de uma aplicação *web* na tecnologia *Node.*js que utiliza a *Web API*  da aplicação *Spotify*.

Objectivos

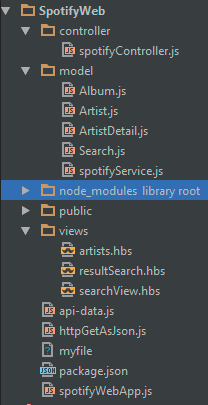
Nesta secção serão descritos os objetivos da primeira fase da aplicação.

Esta aplicação tem como principal objetivo a pesquisa de informação sobre artistas/bandas e assim obter os álbuns dos mesmos.

Descrição Funcional

A nossa aplicação foi elaborada segundo o padrão *MVC* e apresenta a seguinte estrutura:

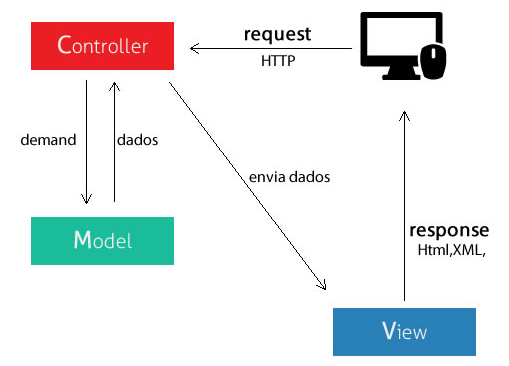
Para realizarmos esta fase foram usados os seguintes *endpoints* disponibilizados pela a API:



O padrão MCV é uma arquitetura que separa as várias camadas da aplicação: camada de interação com o utilizador (*view)*, a camada de manipulação de dados (*model)* e a camada de controlo (*controller)*.

* A camada *Model* é responsável pela obtenção dos dados provenientes da API;
* A camada de *View* é responsável pela exibição dos dados pelo meio de um ficheiro *html*.
* A camada de *Controller* é responsável por receber as requisições do usuário. Esta camada tem um conjunto de funções que definem que determinados *models* são necessários usar e qual a *view* a ser mostrada ao utilizador.

A imagem seguinte simplifica como funciona o padrão utilizado:



Detalhes Técnicos

Devido ao impedimento da utilização da *framework* *express* sentimos a necessidade de criarmos o nosso próprio servidor.

Para tal foi usada uma função do módulo *npm http­* chamada *createServer* que retorna uma instancia de *http.Server* e que recebe como parâmetro um *listener* que responde a eventos *“request”*.

Ao ser realizado um determinado *request* ao servidor, este transforma o url num *endopoint* através do seguinte trecho de código:

**const** server = http.createServer((req, resp) => {  
 **const** urlInfo = url.parse(req.url, **true**)  
 **const** parts = urlInfo.pathname.split('/')  
 **let** endPoint="";  
 **if**(parts.length == 3)  
 endPoint = parts[parts.length -2]  
 **else** endPoint = parts[parts.length-1]

Após ser obtido o *enpoint* a ser utilizado, vai ser verificado se o mesmo existe no controlador *SpotifyController.js* através da função *hasOwnProperty*. Se existir então é executada a função correspondente passando uma *callback* para a mesma.

Após ser mapeada e chamada a função correspondente, o controlador acede ao modelo para que este execute um pedido à API do *Spotify,* através do modulo *npm request*. Após os dados serem processado, o modelo retorna um objeto que irá ser mapeado com o ficheiro *handlebars* da respetiva *View* para que estes sejam mapeados com as tags existentes nas mesmas.

Para a realização desta aplicação foram usados os seguintes *endpoints*:

|  |  |
| --- | --- |
| /v1/search?type=artist | Este *endpoint* permite obter todos os artistas existentes no *spotify* que correspondem a uma determinada *query string* |
| /v1/artists/{id} | Obtém as informações para um único artista identificado pelo seu *Spotify ID* |
| /v1/artists/{id}/albums | Obtém informações sobre os álbuns de um determinado artista. |

O *endpoint* **/v1/search?type=artist** é transformado pelo *model* para a realização do pedido correcto, ou seja, as informações de offset (valor 0 como defeito) para a paginação e o nome do artista a pesquisa. Assim, após transformado, o enpoint fica com o seguinte formato:

/v1/search?query =nameArtist&type=artista&offset=offset

Introdução