

TRABALHO

Projeto – Parte 1

Nome do Aluno

Leonardo Oliveira 2019019

Nome do CTESP

Tecnologias e Programação de Sistemas Informáticos

UNIDADE CURRICULAR:

Aplicações centradas em Redes

DOCENTES:

Hugo Perdigão

DATA:

04/01/2021

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Cofinanciado por:



| | |
|-------------------------------------|----|
| 1. DESCRIÇÃO DO TEMA | 3 |
| 2. ESTRUTURA DO PROJETO | 4 |
| 3. BASE DE DADOS | 5 |
| 4. <i>PACKAGES</i> ADICIONAIS | 6 |
| 5. FUNCIONALIDADES | 7 |
| 6. SERVIÇOS | 14 |
| 7. FRONTEND | 15 |
| 8. REGISTO DE TAREFAS | 16 |
| 9. CONCLUSÃO | 17 |

1. DESCRIÇÃO DO TEMA

Foi proposto pelo professor Hugo Perdição da disciplina Aplicações centradas em Redes, a realização de um relatório relativo ao projeto I.

O tema escolhido foi sobre uma plataforma de música (com base no spotify), onde utilizadores (neste caso artista) poderão se registar e assim fazer upload de suas músicas como importa-las do Youtube (funcionalidade quase 100%)

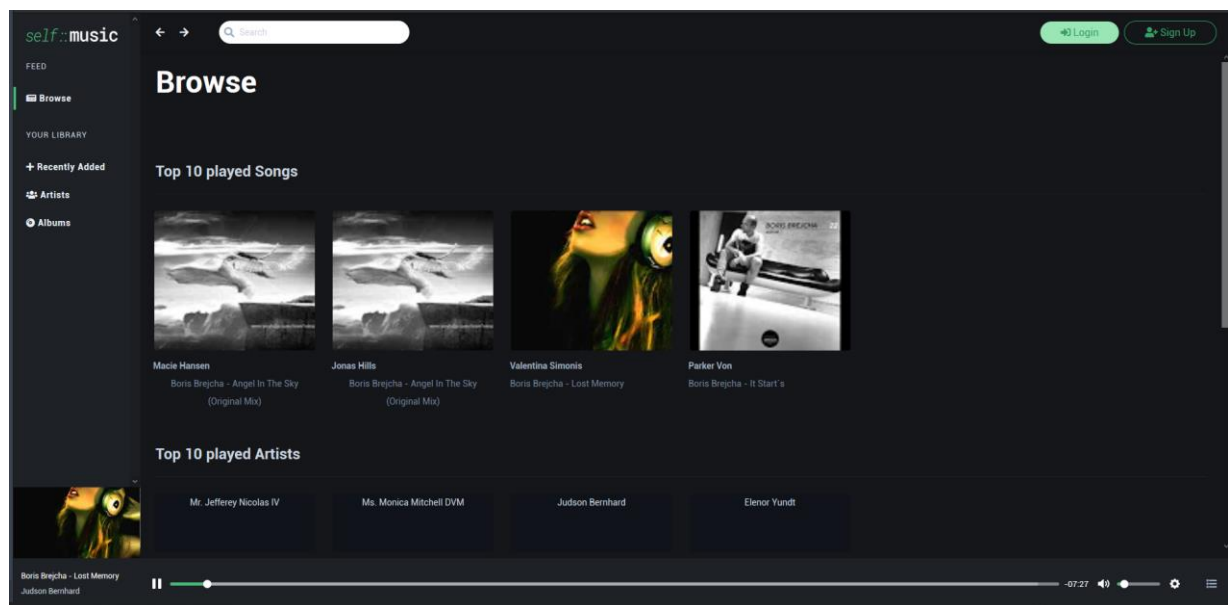


Figura 1 Página Inicial

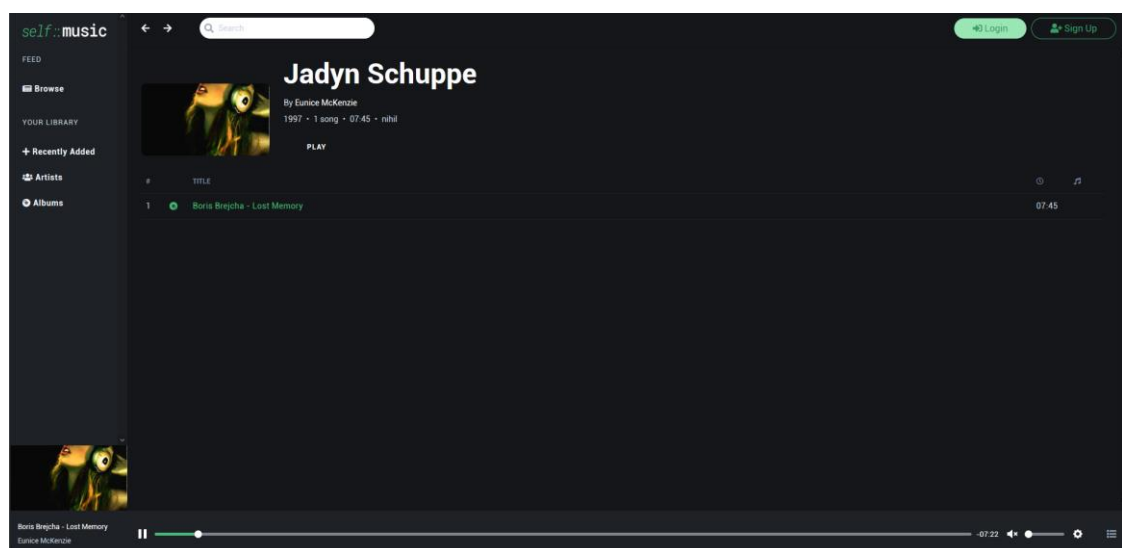
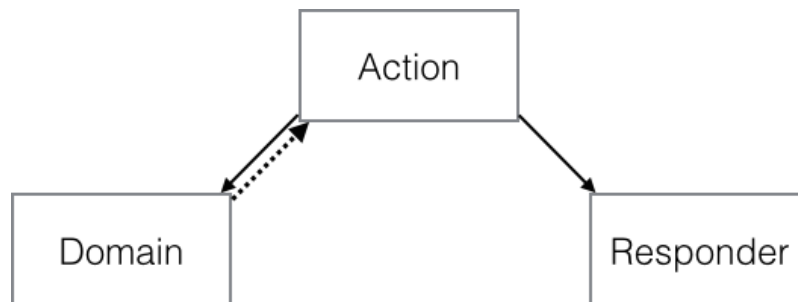


Figura 2 Visualizar albums

2. ESTRUTURA DO PROJETO

O Projeto corrente segue uma arquitetura conhecida como [ADR \(Action-Domain-Responder\)](#), proposta por **pmjones**, com grandes influências da estrutura DDD;

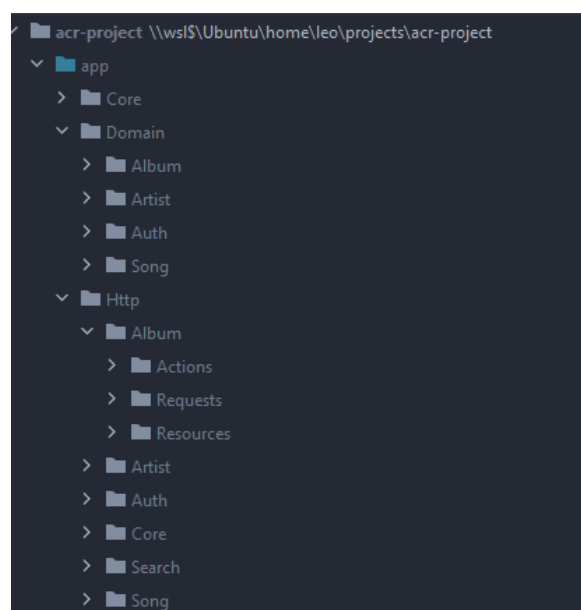


Escolhi esta Arquitetura, porque na minha humilde opinião, a arquitetura MVC já está ultrapassada e contém algumas falhas, por exemplo:

- é bastante comum principiantes fazerem a regra de negócio toda no “**controller**” o que resulta em código esparguete e um “**controller**” gigante igualando a um labirinto (claro que a culpa é do programador iniciante, mas um *controller* é algo muito genérico);
- MVC em projetos enormes torna-se um pesadelo, um caminho sem fim, difícil de ser escalável e mantida sem contar que foi desenhada inicialmente para aplicações gráficas;

Esta Arquitetura “obriga-nos” a separar nosso código por características, por isso, foi escolhida e adaptada conforme as necessidades deste projeto.

Tendo em conta a falta de tempo não foi possível implementar a camada “Responder”, responsável por entregar os dados ao cliente, desta forma os dados são entregados pela camada “Action”.



3. BASE DE DADOS

A arquitetura da base de dados é no geral simples. Um Artista tem vários álbuns, cada álbum tem várias músicas.

A tabela ***played_songs*** irá manter atualizado as “visualizações” de cada música já a tabela ***jobs*** irá guardar a fila de trabalhos a fazer enquanto que ***trakable_jobs*** vai dar-nos a possibilidade de rastrear cada trabalho como seu progresso e estado.

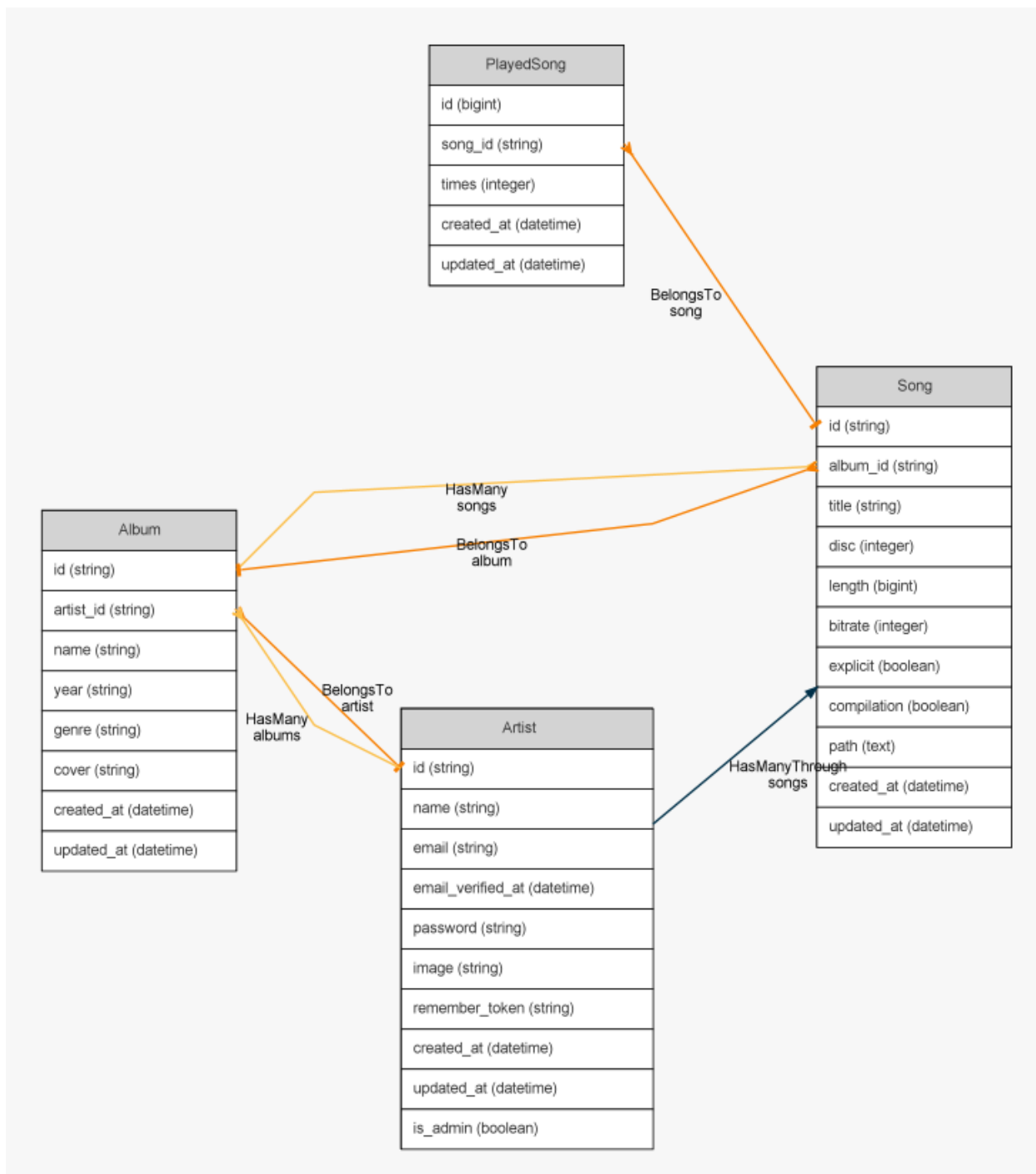


Figura 3 Gráfico de Entidade Relacionamento

4. PACKAGES ADICIONAIS

Ao longo deste projeto foi visto a necessidade de incluir alguns packages adicionais para que o desenvolvimento fosse mais rápido e produtivo.

- **goldspecdigital/laravel-eloquent-uuid:**
 - Este package irá ajudar a integrar uuid ao models do projeto, foi usado de forma a manter coexistência de dados e evitar expor id's ao publico.
- **imtigger/laravel-job-status:**
 - Dá-nos a possibilidade de rastrear os trabalhos a fazer na aplicação, indicar percentagem de progresso e etc... Foi utilizado para verificar o processo de download de conversão de um vídeo do Youtube.
- **league/flysystem-aws-s3-v3:**
 - Este pacote dá a possibilidade de integrar serviços da amazon (S3) e DigitalOcean (spaces) no projeto.
- **spatie/laravel-searchable:**
 - Facilita a vida ao realizar uma busca, onde podemos fazer os nossos models pesquisáveis!
- **titasgailius/terminal:**
 - Realiza uma abstração ao **symfony process component**, tornando mais fácil, rápido e elegante controlar e executar processos.
- **Laravel/telescope:**
 - Debugger desenvolvido pela equipa do laravel, ferramenta essencial para desenvolvimento, ajuda na correção de bugs, visualização de requests, logs, cache, queries, etc...

5. FUNCIONALIDADES

Este projeto irá ser integrado futuramente no meu portfolio (com muitas mudanças claro), por esta razão há funcionalidades complexas, mas bastante interessantes tais como:

- *Http Byte Serving:*

```
Request URL: http://localhost/api/stream/e9d0bbaf-e1d5-40af-b9bd-0255ea6eab23
Request Method: GET
Status Code: 206 Partial Content
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin

Response Headers view source
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Cache-Control: no-cache, private
Connection: keep-alive
Content-Length: 3288674
Content-Range: bytes 2424832-5713505/5713506
Content-Type: audio/mp3
Date: Mon, 04 Jan 2021 20:52:19 GMT
Server: nginx
X-Powered-By: PHP/7.4.13
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 45
```

Como podemos ver pela resposta do servidor (cabeçalho Content-Range) é possível o cliente obter um intervalo de bytes evitando fazer download inteiro e sobrecarregar o servidor (no caso de milhares de clientes simultâneos).

PS: Com melhorias a fazer.

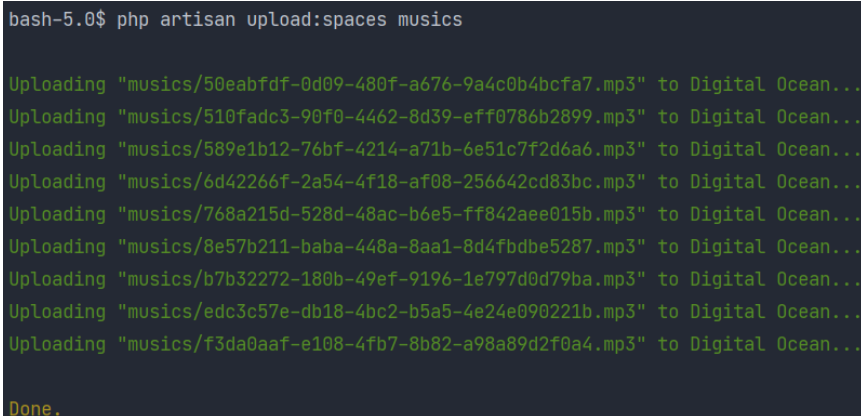
- Upload das músicas locais para nuvem (Digital Ocean ou S3):

Ao executar o seguinte comando todos os ficheiros da pasta **musics** irão ser transferidos para a **cloud** poupando assim espaço.

O processo será transparente e feito em background já que é executado uma tarefa para cada upload (*Queue of jobs*). Tudo isto será processado pelo container **queue** (Figura 3) que no fundo é vários processos em background a consumir as tarefas disponíveis (configuração disponível na pasta Docker).



```
1 php artisan upload:spaces musics
```

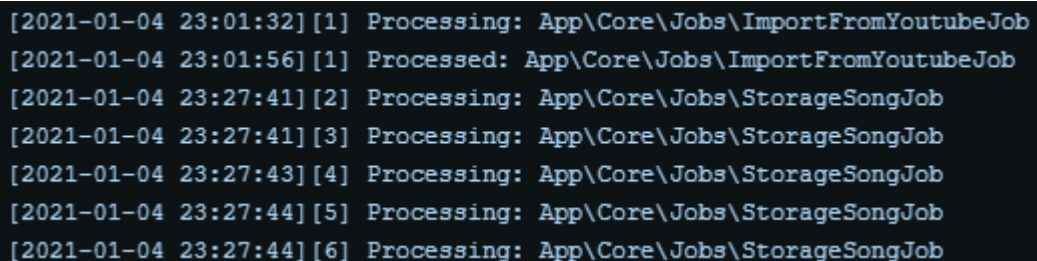


```
bash-5.0$ php artisan upload:spaces musics

Uploading "musics/50eabfdf-0d09-480f-a676-9a4c0b4bcfa7.mp3" to Digital Ocean...
Uploading "musics/510fadc3-90f0-4462-8d39-ef0786b2899.mp3" to Digital Ocean...
Uploading "musics/589e1b12-76bf-4214-a71b-6e51c7f2d6a6.mp3" to Digital Ocean...
Uploading "musics/6d42266f-2a54-4f18-af08-256642cd83bc.mp3" to Digital Ocean...
Uploading "musics/768a215d-528d-48ac-b6e5-ff842aee015b.mp3" to Digital Ocean...
Uploading "musics/8e57b211-baba-448a-8aa1-8d4fbdb5287.mp3" to Digital Ocean...
Uploading "musics/b7b32272-180b-49ef-9196-1e797d0d79ba.mp3" to Digital Ocean...
Uploading "musics/edc3c57e-db18-4bc2-b5a5-4e24e090221b.mp3" to Digital Ocean...
Uploading "musics/f3da0aaf-e108-4fb7-8b82-a98a89d2f0a4.mp3" to Digital Ocean...

Done.
```

Figura 4 Output do comando.



```
[2021-01-04 23:01:32] [1] Processing: App\Core\Jobs\ImportFromYoutubeJob
[2021-01-04 23:01:56] [1] Processed: App\Core\Jobs\ImportFromYoutubeJob
[2021-01-04 23:27:41] [2] Processing: App\Core\Jobs\StorageSongJob
[2021-01-04 23:27:41] [3] Processing: App\Core\Jobs\StorageSongJob
[2021-01-04 23:27:43] [4] Processing: App\Core\Jobs\StorageSongJob
[2021-01-04 23:27:44] [5] Processing: App\Core\Jobs\StorageSongJob
[2021-01-04 23:27:44] [6] Processing: App\Core\Jobs\StorageSongJob
```

Figura 5 Container "queue" a consumir as tarefas

- Importar Músicas do Youtube:
 - Converter para mp3;
 - Download da *thumbnail*;

```
bash-5.0$ php artisan db:seed
Downloading [znQsetWNfLc] - 5.19%
Downloading [znQsetWNfLc] - 11.18%
Downloading [znQsetWNfLc] - 19.17%
Downloading [znQsetWNfLc] - 25.75%
Downloading [znQsetWNfLc] - 33.14%
Downloading [znQsetWNfLc] - 41.13%
Downloading [znQsetWNfLc] - 49.11%
Downloading [znQsetWNfLc] - 55.10%
Downloading [znQsetWNfLc] - 63.09%
Downloading [znQsetWNfLc] - 71.07%
```

Figura 6 Populando a base de dados com músicas baixadas do Youtube

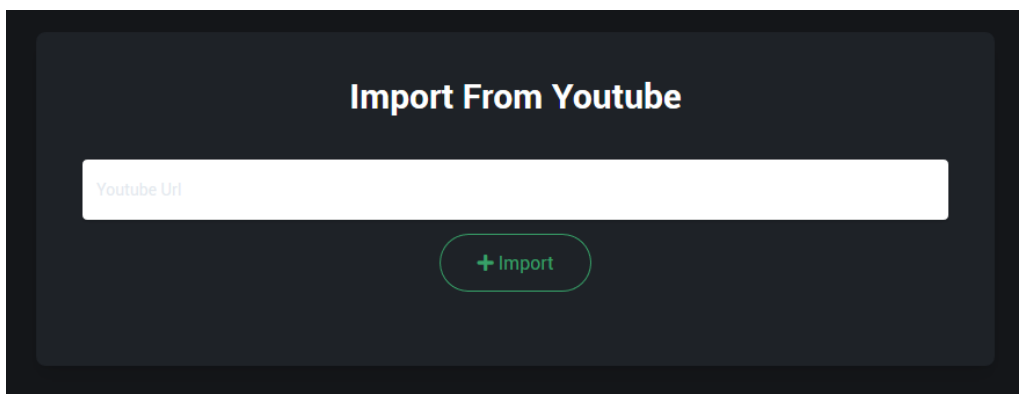


Figura 7 Interface gráfica para Importa músicas do youtube.

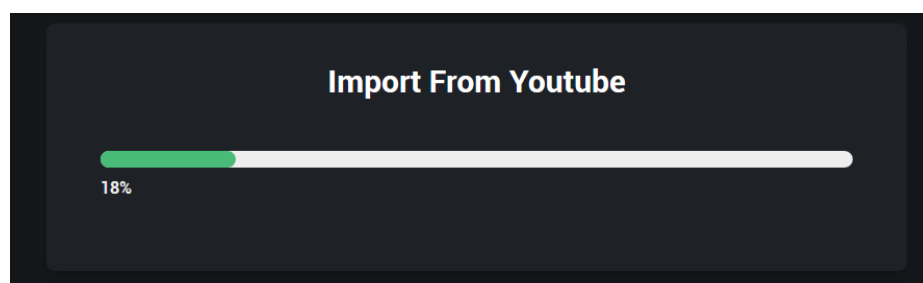
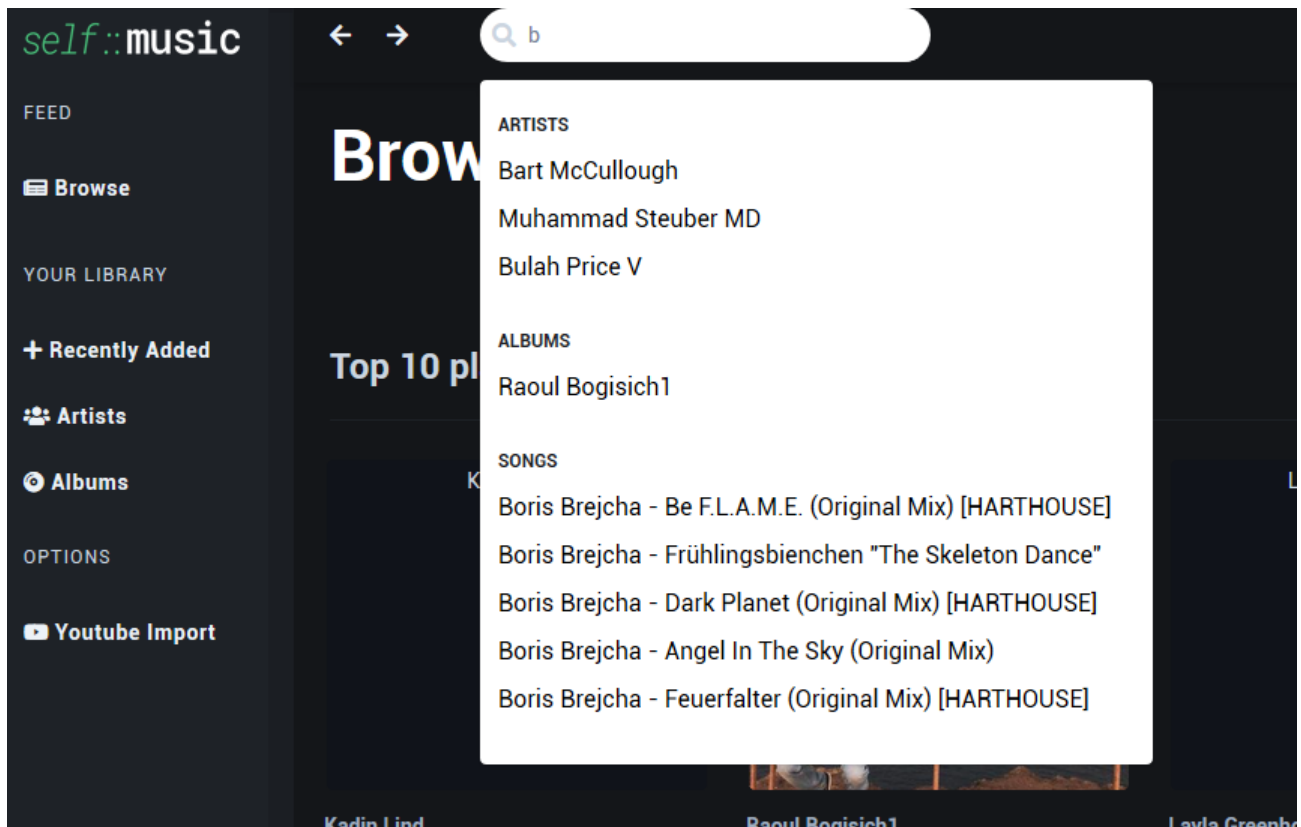


Figura 8 Progresso de importação da música

Ps: A música não é guardada na base de dados ao importar via interface visto que ainda falta a funcionalidade de inserir músicas.

- Procurar músicas, álbuns e artistas:



- Auto Resolução de *Streamers* (Fallback streamers);

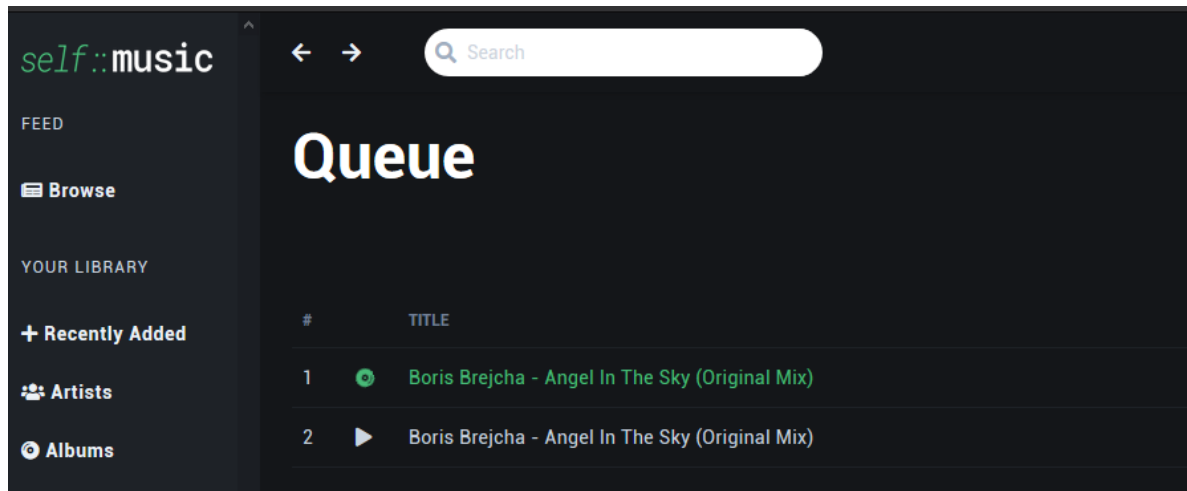
Com este sistema é possível ter um *fallback* de músicas, ou seja, se uma música não existir localmente irá obtê-la da cloud (com melhorias a realizar).

```

1 return StreamerResolver::resolve([
2     SpacesStreamer::class,
3     LaravelStreamer::class,
4 ], $song);

```

- Fila de músicas:



- Top 10 de músicas, álbuns e artistas:

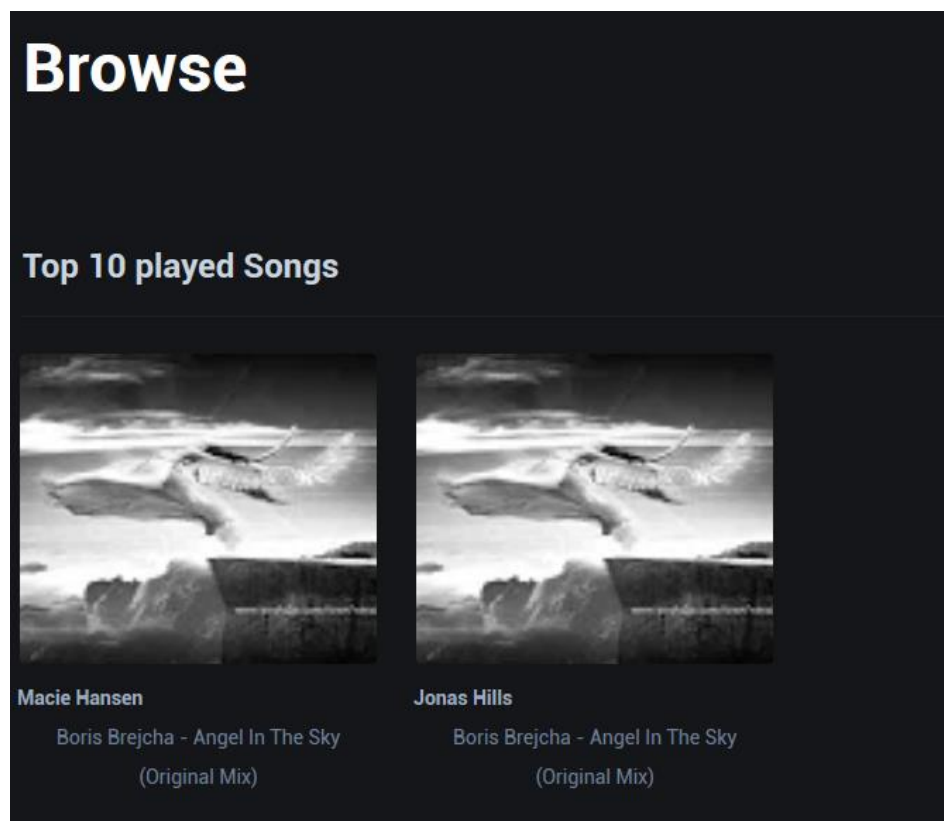
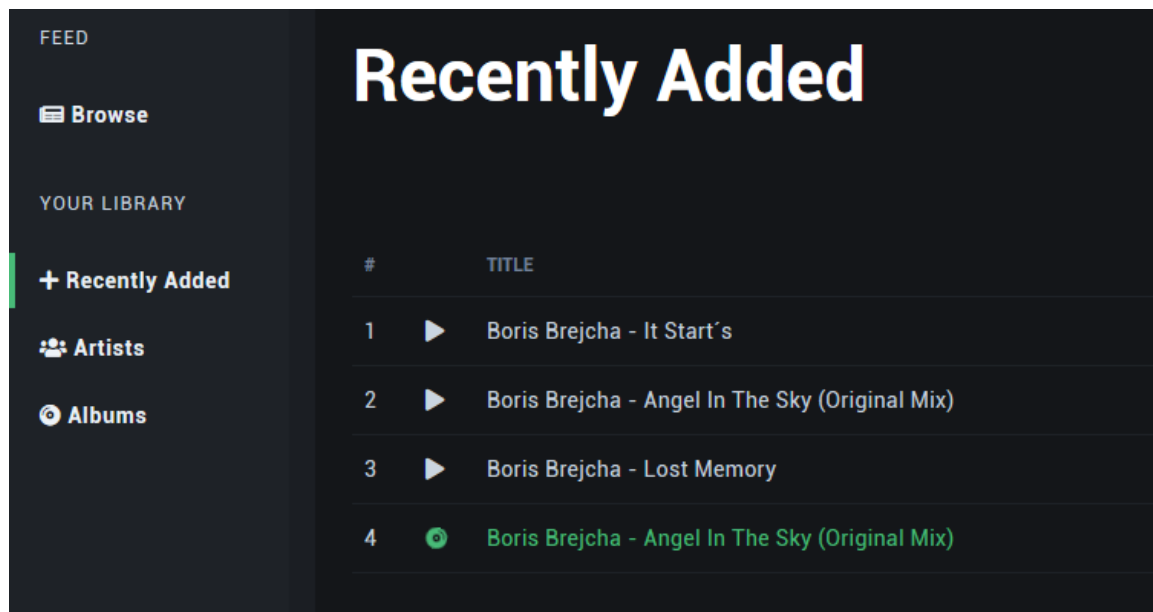


Figura 9 Página principal com top música, álbuns e artistas (com melhorias a fazer)

- últimas músicas adicionadas:



- Login, Registo, e Painel de Administrador:

Foi implementado apenas CRUD de albums por questão de tempo.

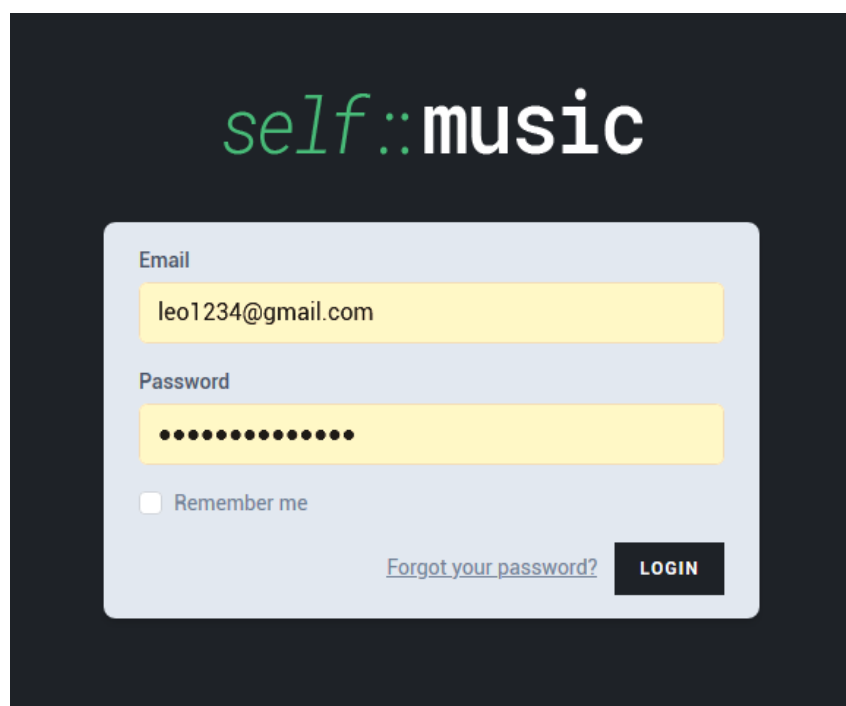
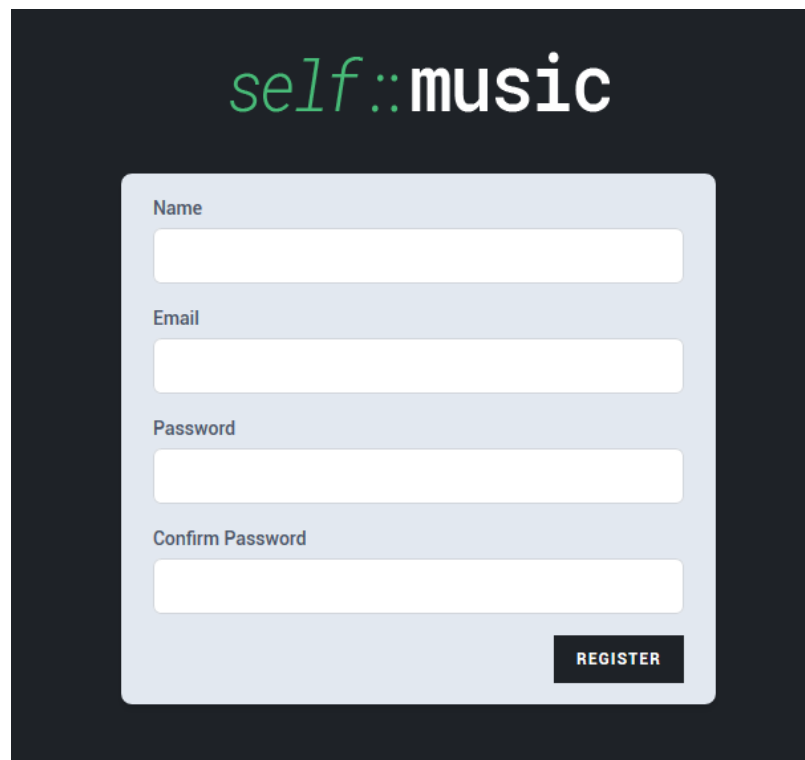
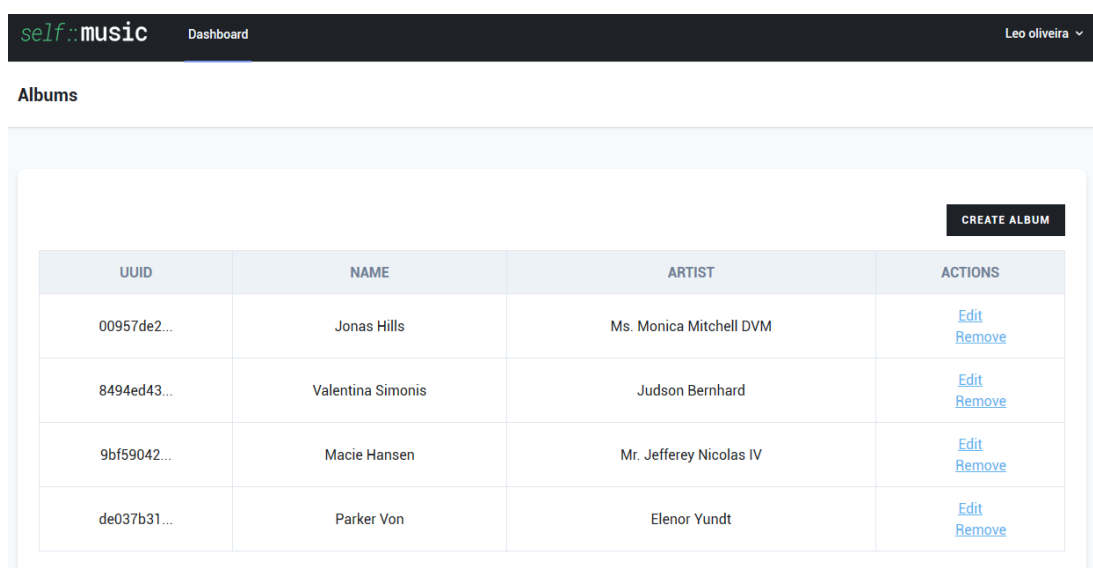


Figura 10 Login



The registration form for self::music is displayed on a dark background. It features a light blue rounded rectangle containing four input fields: Name, Email, Password, and Confirm Password. A black REGISTER button is located at the bottom right of the form.

Figura 11 Registo



The administrator panel for self::music shows a dashboard with the user name 'Leo oliveira'. The 'Albums' section contains a table with four columns: UUID, NAME, ARTIST, and ACTIONS. A 'CREATE ALBUM' button is located at the top right of the table.

| UUID | NAME | ARTIST | ACTIONS |
|-------------|-------------------|-------------------------|--|
| 00957de2... | Jonas Hills | Ms. Monica Mitchell DVM | Edit Remove |
| 8494ed43... | Valentina Simonis | Judson Bernhard | Edit Remove |
| 9bf59042... | Macie Hansen | Mr. Jefferey Nicolas IV | Edit Remove |
| de037b31... | Parker Von | Elenor Yundt | Edit Remove |

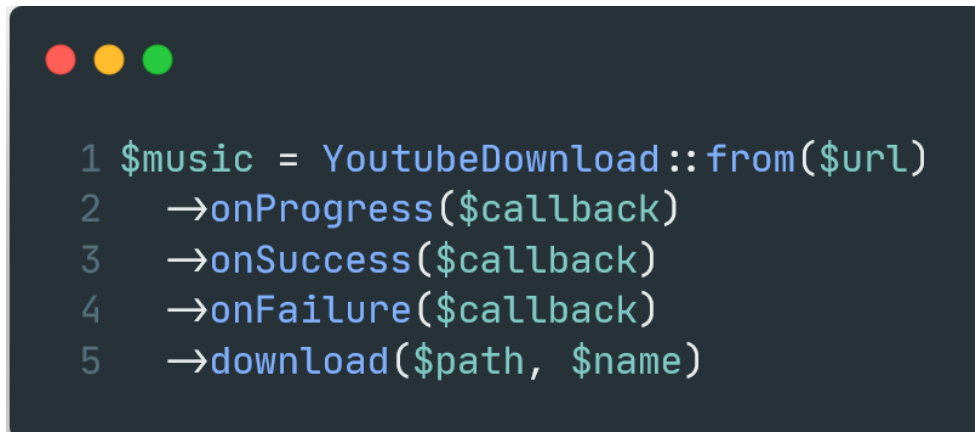
Figura 12 Painel de Administrador

- Permissões:

Foi implementado um sistema de permissões mais basico possivel visto que o tempo estava a apertar, mas futuramente pretendo integrar um sistema de permissões e niveis de acesso rubusto e promissor!

6. SERVIÇOS

- Decidi criar um serviço de download de músicas do Youtube, já que os packages disponíveis não satisfaziam as minhas necessidades. Este serviço está incompleto, mas futuramente pretendo desacoplar da aplicação e torna-lo um package disponível para a comunidade.



```
1 $music = YoutubeDownload::from($url)
2   →onProgress($callback)
3   →onSuccess($callback)
4   →onFailure($callback)
5   →download($path, $name)
```

Figura 13 Futura interface do servido de download.

- Criei também um serviço de *stream* de música para que a manipulação de dados fosse transparente ao utilizador.

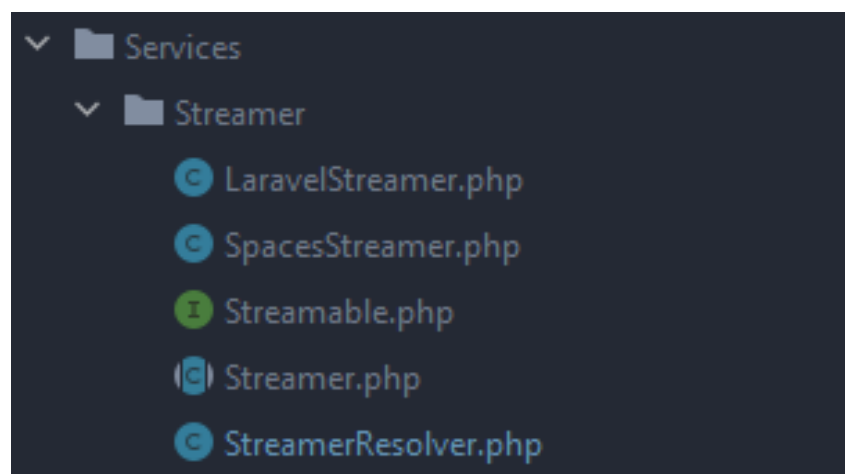


Figura 14 Serviço de Stream de músicas

7. FRONTEND

Todo o frontend foi desenvolvido usando o vue.js e vuex (retirando a autenticação e administrador).
Num futuro próximo pretendo reorganizar esta estrutura e reavaliar certas decisões.

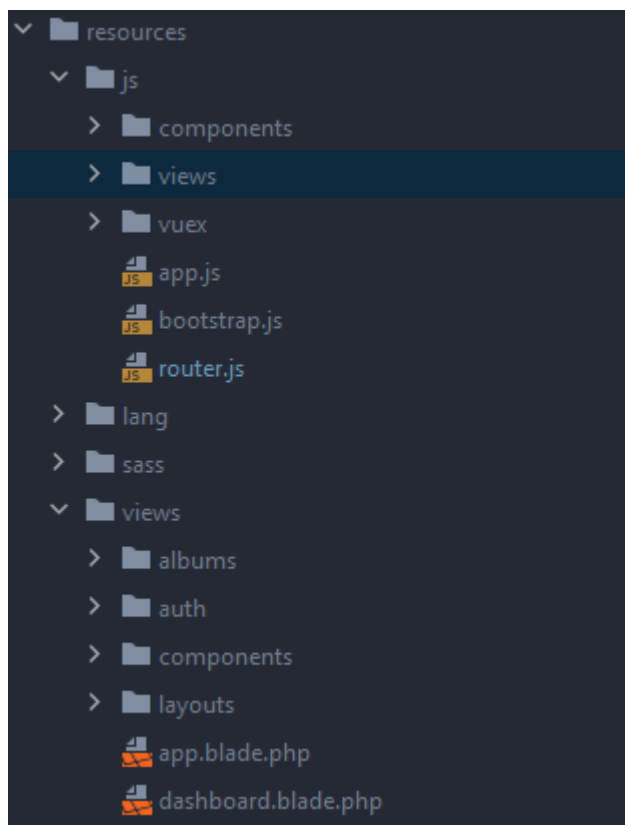


Figura 15 Estrutura de pastas frontend

8. REGISTO DE TAREFAS

!

0 Open

✓

17 Closed

🕒

Make stream resolver

#32 by 13dev was closed 4 days ago

🕒

Make view to import music from youtube

#31 by 13dev was closed yesterday

🕒

Improve login and make it work with artist

#30 by 13dev was closed yesterday

🕒

Store files in Digital Ocean

#24 by 13dev was closed 5 days ago

🕒

Fix youtube-dl is not cleaning the .webm files

#23 by 13dev was closed 5 days ago

🕒

Best way to make top artists and albums

#22 by 13dev was closed yesterday

🕒

Finish Browser Page

#19 by 13dev was closed 8 days ago

🕒

User to Artist

#18 by 13dev was closed 8 days ago

🕒

Get / Download videos from Youtube

#13 by 13dev was closed 8 days ago

Merge User with artist

#12 by 13dev was closed 8 days ago

Fix auto register routes

#10 by 13dev was closed 8 days ago

Change id to uuid

#7 by 13dev was closed 8 days ago

Restruct backend to ADR pattern

#5 by 13dev was closed 9 days ago

Add Google cloud storage to stream music

#4 by 13dev was closed 10 days ago

Change tailwind.config.js

#3 by 13dev was closed yesterday

Style the plyr

#2 by 13dev was closed 19 days ago

Change controllers to Actions

#1 by 13dev was closed 8 days ago

9. CONCLUSÃO

Com isto concluo que este projeto me fez por em praticar as novas novidades do laravel e motivou-me a acabar projetos em mente.

Foi dada a prioridade a funcionalidades não pedidas no enunciado, isto porque há 5 anos já estava realizando um simples crud em laravel lá na versão 5.1 mas ao mesmo tempo tentei completar todas as tarefas pedidas.

Decidi fazer um projeto fora das linhas para que mais tarde seja aproveitado para o meu portfolio, aprender coisas novas e não desperdiçar meu tempo em coisas banais e repetitivas.

Para trás ficaram algumas funcionalidades (ex agrupar álbum por disco) porque achei desnecessário comentar sobre.

Tarefas futuras, roadmap:

- Implementar Responders e reestruturar projeto;
- Acabar o serviço de Download;
- Implementar Unit Tests e Feature tests;
- Code coverage e CI / CD;
- Acabar importar do Youtube (editar thumbnail, name);
- Deploy para servidor SO;
- Inserção de músicas;
- Permissões e Funções;
- Painel de Admin completo;
- Nova Logica de Top musics, artists and albums;
- Transformar backend em uma api, com JWT;
- Frontend à parte e responsivo;
- Múltiplos proxies e user-agents para o Youtube-dl (Pode resultar em ban por IP).
- Realtime chat, Amigos;