

Arthur Costa Serra

Matrícula: 2012382

**Projeto Final de Programação: Uma ferramenta  
para produção de bases de dados para tarefas  
de reconstrução de áudio**

Rio de Janeiro, Brasil

2021

Arthur Costa Serra  
Matrícula: 2012382

# **Projeto Final de Programação: Uma ferramenta para produção de bases de dados para tarefas de reconstrução de áudio**

Trabalho apresentado ao coordenador do programa de pós-graduação em informática da PUC-Rio como requisito para obtenção de nota na disciplina INF2102-Projeto Final de Programação

Pontifícia Universidade Católica do Rio de Janeiro  
Departamento de Informática  
Programa de Pós-Graduação em Informática

Orientador: Sérgio Colcher

Rio de Janeiro, Brasil  
2021

# Sumário

|                                      |           |
|--------------------------------------|-----------|
| <b>Sumário</b>                       | <b>2</b>  |
| <b>1 ESPECIFICAÇÃO DO PROGRAMA</b>   | <b>3</b>  |
| 1.1 <b>Objetivo</b>                  | <b>3</b>  |
| 1.2 <b>Escopo</b>                    | <b>3</b>  |
| 1.3 <b>Requisitos</b>                | <b>3</b>  |
| 1.3.1 Requisitos Funcionais          | 3         |
| 1.3.2 Requisitos Não-Funcionais      | 4         |
| <b>2 ARQUITETURA</b>                 | <b>5</b>  |
| 2.0.1 Estrutura                      | 6         |
| <b>3 TESTES</b>                      | <b>10</b> |
| <b>4 DOCUMENTAÇÃO PARA O USUÁRIO</b> | <b>12</b> |
| 4.1 Telas de navegação               | 13        |
| <b>5 CÓDIGO FONTE</b>                | <b>18</b> |
| <b>5.1 Python</b>                    | <b>18</b> |
| 5.1.1 ./py/fragment_audio.py         | 18        |
| <b>5.2 JavaScript</b>                | <b>25</b> |
| 5.2.1 ./js/main.js                   | 25        |
| 5.2.2 ./js/home.js                   | 27        |
| 5.2.3 ./js/infopage.js               | 28        |
| 5.2.4 ./js/fragpage.js               | 37        |
| 5.2.5 ./js/loadpage.js               | 46        |
| 5.2.6 ./js/playlistpage.js           | 49        |

# 1 Especificação do Programa

## 1.1 Objetivo

Existe um campo no processamento de sinais de áudio, chamado *Audio Inpainting*, que é tão antigo quanto a tarefa análoga no campo das imagens. Contudo, a proporção de pesquisas envolvendo a tarefa de reconstrução em imagens é extremamente discrepante as de reconstrução de áudio. Isso se dá, também, pela quantidade e acessibilidade de bases disponíveis para a pesquisa. A quantidade bases de dados para reconstrução de áudios disponíveis é muito pequena, além disso, frequentemente possuem complexidade e fragmentação baixa, de tal forma que a tarefa seja muito fácil de resolver. Em vista disso, o objetivo deste projeto é desenvolver uma ferramenta de geração de bases de dados para tarefa de *Audio Inpainting*. Nesta ferramenta, a partir de uma coleção padronizada de áudios o usuário poderá gerar uma coleção correspondente temporalmente fragmentada. Por fim, visualizar e avaliar o processo de fragmentação conforme os parâmetros escolhidos.

## 1.2 Escopo

O escopo deste projeto é a produção de uma interface gráfica para produção de bases de dados para reconstrução de áudio. Dispondo de uma visualização dos dados, parâmetros e avaliação dinâmica dos resultados. Otimizando o tempo do processo de extração de características realizando-os em GPU e reduzindo ao máximo o contato direto do usuário com código.

## 1.3 Requisitos

### 1.3.1 Requisitos Funcionais

Foram definidos três Requisitos Funcionais (RF):

- **RF-01 Operar em larga escala:** a ferramenta deve ser capaz de gerenciar diretórios de áudios densos.
- **RF-02 Interface gráfica:** construir um meio de interação gráfico com a ferramenta.
- **RF-03 Liberdade de parametrização:** A ferramenta deve permitir ao usuário liberdade de escolha de parâmetros para codificação dos áudios.

- **RF-04 Aleatoriedade de fragmentação:** a *seed* de fragmentação dos áudios não deve ser estática.

### 1.3.2 Requisitos Não-Funcionais

Os Requisitos Não Funcionais (RNF) desse projeto foram identificados como:

- **RNF-01 Consumo de memória:** processando áudio a áudio de modo a não carregar todos em memória em simultâneo.
- **RNF-02 Usabilidade:** a ferramenta deve minimizar o contato do usuário com código e métricas de qualidade, de modo a possibilitar a avaliação visual e sonora dos resultados. Toda e qualquer ação do usuário deve haver *feedback* de possibilidade da ação. Assim como informação que algo está sendo processado.
- **RNF-03 Estabilidade:** considerando bases de áudio muito extensas, a ferramenta deve operar como baixo custo de memória por grandes períodos.
- **RNF-04 Confiabilidade:** mesmo que a haja repetição de aplicação de base e parâmetros, o retorno de fragmentação deve diferir sempre.

## 2 Arquitetura

A ferramenta foi desenvolvida utilizando o *framework* Electron.js (Versão 13.0.1), para produção da aplicação desktop com integração aos *frameworks* NodeJS (Versão 6.14.12), para controle das telas, e Bootstrap (Versão 5.0), para estrutura gráfica responsiva. Contudo, o processo principal de fragmentação dos áudios foi implementado utilizando a linguagem Python (Versão 3.6.10) com a biblioteca Tensortflow (Versão 2.3.1) que fornece integração com GPUs CUDA.

O projeto está dividido em dois diretórios: *Documentacao* que contem um arquivo PDF com a descrição de todo projeto, e *PFP\_Gen\_Audio\_Dataset* que contem todo o código-fonte do projeto, no que lhe concerne dividido em cinco subdiretórios e um arquivo de configuração como mostra a Figura 1.

```
PFP_Gen_Audio_Dataset/
|----PFP_Gen_Audio_Dataset/
|      |----assets/
|      |      |----arrow-left.svg
|      |      |----arrow-right.svg
|      |      |----copy.svg
|      |
|      |----css/
|      |      |----fragpage.css
|      |      |----global.css
|      |      |----home.css
|      |      |----infopage.css
|      |      |----loadpage.css
|      |      |----playlistpage.css
|      |
|      |----html/
|      |      |----fragpage.html
|      |      |----home.html
|      |      |----infopage.html
|      |      |----loadpage.html
|      |      |----playlistpage.html
|      |
|      |----js/
|      |      |----fragpage.js
|      |      |----home.js
|      |      |----infopage.js
|      |      |----loadpage.js
|      |      |----main.js
|      |      |----playlistpage.js
|      |
|      |----py/
|      |      |----fragment_audio.py
|      |      |----test.py
|      |      |----log.txt
|      |      |----requirements.txt
|      |
|      |----package.json
|
|----Documentacao/
|      |----ProjetoFinalDeProgramacao.pdf
|
|----.gitignore
|----README.md
```

Figura 1 – Arvore de arquivos do projeto.

- *assets*: contém imagens de ícones utilizados nas telas e também funciona como

diretório de arquivos de áudio temporário gerados.

- *css*: contém arquivos .css que definem o estilo gráfico de cada tela.
- *html*: contém arquivos .html que definem todas as telas da aplicação.
- *js*: contém arquivos .js que gerenciam todo fluxo e controle de telas.
- *py*: contém um arquivo .py com as funções principais de fragmentação e um arquivo .txt com as bibliotecas necessárias para execução do script .py.
- *package.json*: É o arquivo de configuração do framework Electron.js com informações de autor, repositório e pacote necessários para execução dos arquivos .js.

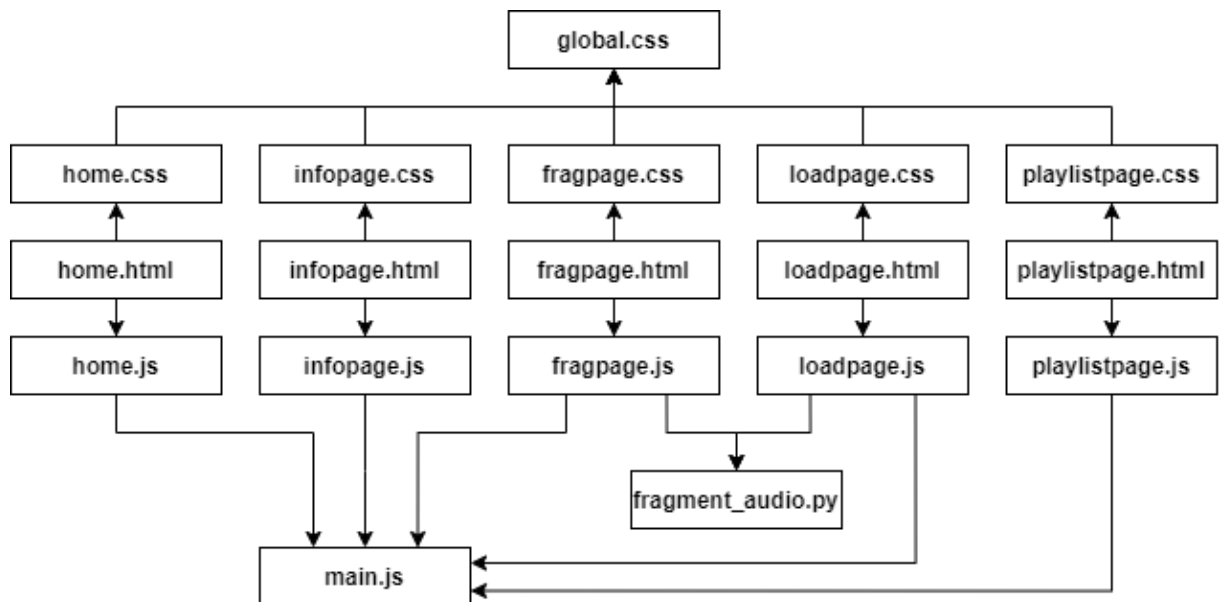


Figura 2 – Diagrama de dependência entre os arquivos.

### 2.0.1 Estrutura

Seguindo o padrão de desenvolvimento *Model-View-Controller* cada framework desempenha um papel no projeto. Como demonstra a Figura 3, o *Model* corresponde ao Electron.js junto ao NodeJS, o *View* corresponde ao *Bootstrap* e o *Controller* ao script Python.

No Electron.js todos os scripts dependem do *main.js* que é configurado como script principal. Nele é definida uma janela derivada do navegador Chromium, o ciclo de vida da aplicação e as funções de comunicação entre os processos. O script principal é a única forma dos processos pertencentes ao Electron.js se comunicarem. Sendo assim, esta é a única maneira dos demais scripts compartilharem variáveis. Ainda no contexto principal é definido a primeira tela a ser executada, no caso *home.html*.

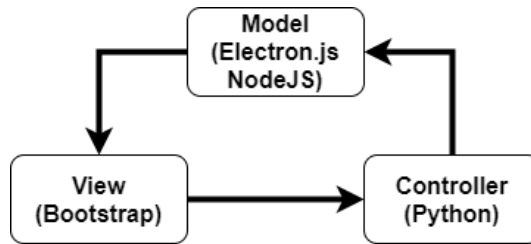


Figura 3 – Correspondência de frameworks aos módulos do padrão MVC.

Cada arquivo *.html* define uma tela da aplicação, os arquivos correspondentes *.css* configuração do estilo e os *.js* (com exceção do *main.js*) definem os eventos dinâmicos de alteração dos formulários *.html*.

O contexto *home*, demonstrado na Figura 4, consiste em informar a aplicação um diretório com os áudios a serem processados.

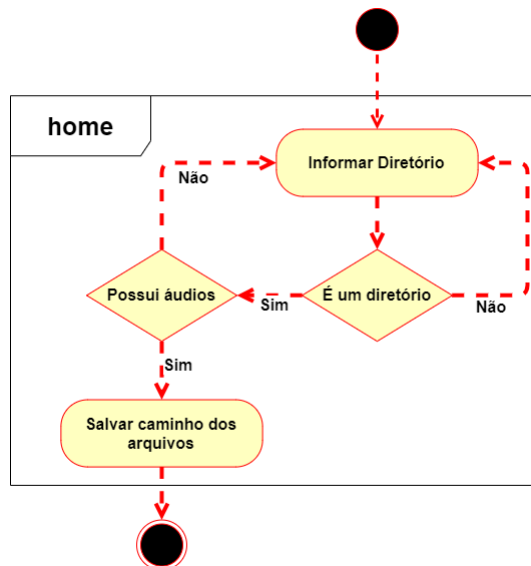


Figura 4 – Diagrama de atividade da tela *home*.

A Figura 5, demonstra o contexto *infopage* que permite ao usuário visualizar os áudios e as informações do diretório informado.

A Figura 6, demonstra o contexto *fragpage*. Nesta etapa o usuário define os parâmetros de fragmentação, pode pré-visualizar o resultado dos parâmetros no áudio de referência carregado.

O contexto *loadpage*, como mostra a Figura 7, contém o processo principal de fragmentação da base fornecida. Visualmente, este contexto é apenas uma barra de progresso.

Por fim, a Figura 8 apresenta o contexto final *playlistpage*, em que é apresentado o resultado da fragmentação lado a lado com a base de referência.



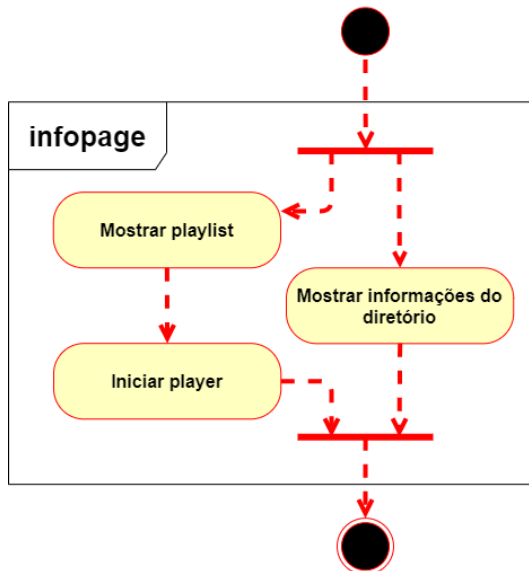


Figura 5 – Diagrama de atividade da tela *infopage*.

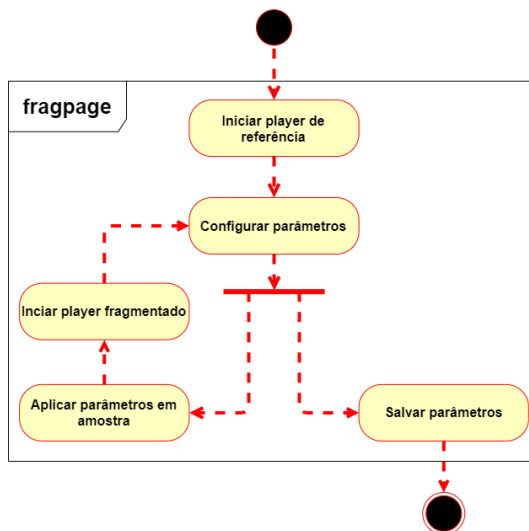


Figura 6 – Diagrama de atividade da tela *fragpage*.

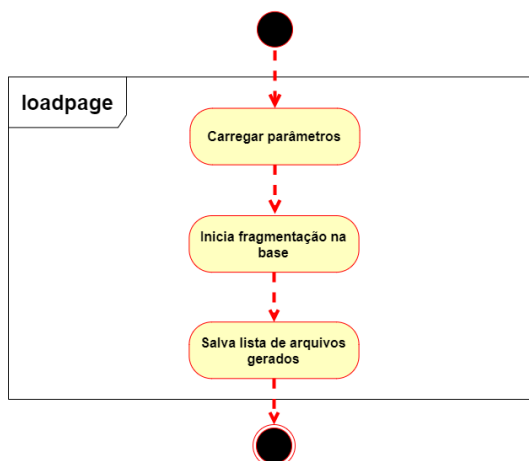


Figura 7 – Diagrama de atividade da tela *loadpage*.

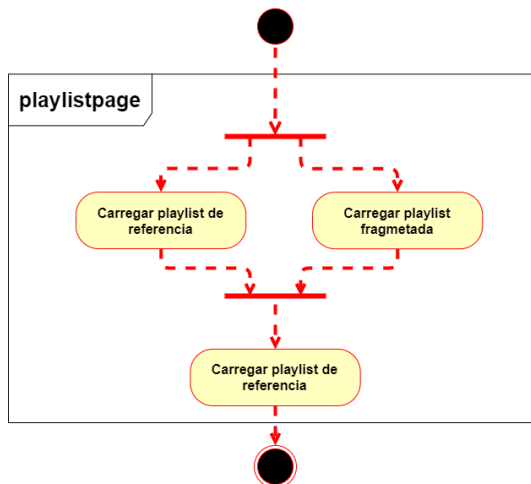


Figura 8 – Diagrama de atividade da tela *playlistpage*.

No entanto, é importante considerar o processo de fragmentação mais detalhadamente. Inicialmente, os parâmetros informados no contexto *fragpage* são compostos pelos parâmetros do algoritmo o algoritmo STFT (Short-Time Fast Fourier) da biblioteca Tensorflow,<sup>1</sup> e pelo valores mínimos e máximos do percentual de oclusão de uma janela do espectro. A Figura 9 demonstra o processo de fragmentação em 3 etapas: (1) transformando a onda em espectrograma; (2) particionar o espectrograma em  $n$  partes com mesma quantidade de unidades de tempo e frequência; (3) aplicar aleatoriamente os percentuais de fragmentação nas partes.

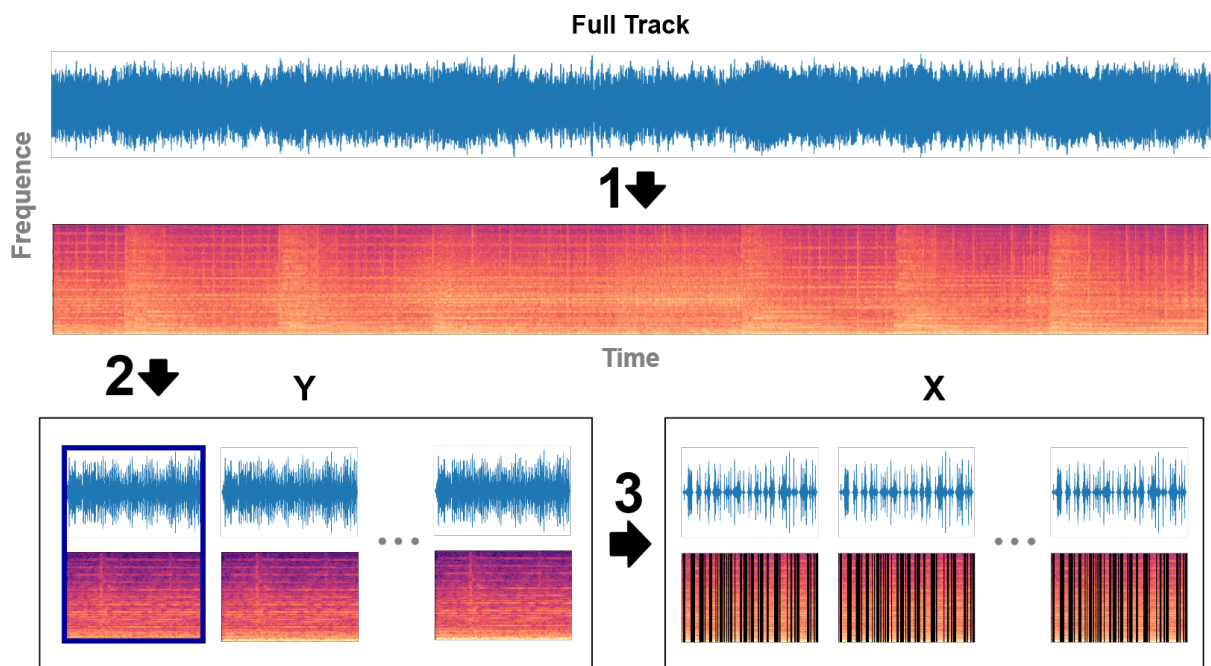


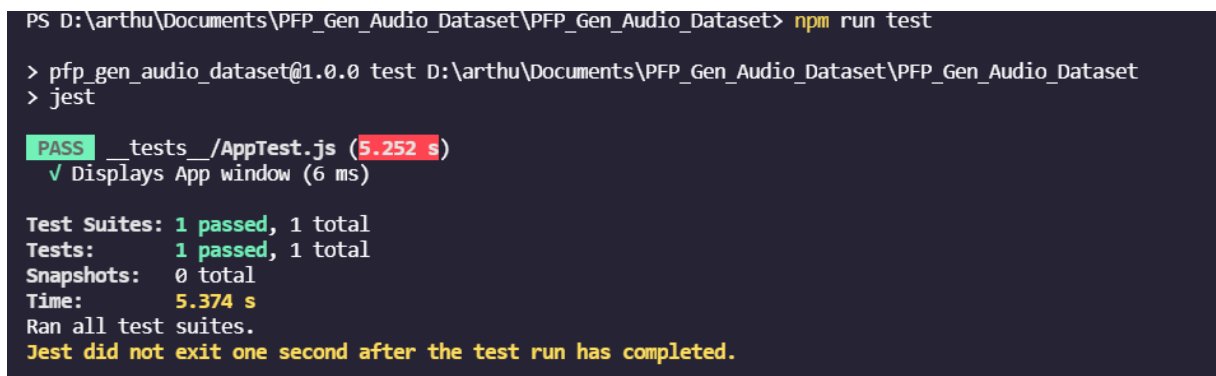
Figura 9 – Método de fragmentação.

<sup>1</sup> [https://www.tensorflow.org/api\\_docs/python/tf/signal/stft](https://www.tensorflow.org/api_docs/python/tf/signal/stft)

### 3 Testes

Dois testes foram realizados para comprovar a viabilidade da aplicação, um para o carregamento das janelas, através do ambiente de teste padrão para o *framework* Electron.js e um teste de unidade para as funções de fragmentação do Python através da função nativa *unittest*.

Através do *framework* de teste Spectron,<sup>1</sup> provou-se o sucesso ao carregamento da janela como demonstra o log da Figura 10.

A terminal window with a dark background showing the execution of a Spectron test. The command 'npm run test' is entered at the prompt. The output shows the test 'Displays App window' passing in 6 ms. Summary statistics show 1 passed test out of 1 total, with a total time of 5.374 s. A warning message at the bottom states 'Jest did not exit one second after the test run has completed.'

```
PS D:\arthu\Documents\PFP_Gen_Audio_Dataset\PFP_Gen_Audio_Dataset> npm run test

> pfp_gen_audio_dataset@1.0.0 test D:\arthu\Documents\PFP_Gen_Audio_Dataset\PFP_Gen_Audio_Dataset
> jest

PASS __tests__/AppTest.js (5.252 s)
  ✓ Displays App window (6 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.374 s
Ran all test suites.
Jest did not exit one second after the test run has completed.
```

Figura 10 – Spectron log de criação de janela.

Contudo, o teste mais importante consiste nas funções do Python, pois nelas que ocorrem o principal objetivo deste trabalho, a fragmentação dos áudios. No total somam quatro funções Python: (1) transforma áudio em espectrograma; (2) transforma espectrograma em áudio; (3) fragmenta e salva um conjunto de áudios; (4) fragmenta e salva uma amostra temporária. A Figura 11, demonstra o log completo de execução do teste o qual está presente as confirmações de uso de GPU e de retorno esperado das funções.

<sup>1</sup> <https://github.com/electron-userland/spectron>

```

PS D:\arthur\Documents\VPFP_Gen_Audio_Dataset> python ./VPFP_Gen_Audio_Dataset/py/test.py
2021-07-01 15:01:16.886292: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart64_101.dll
2021-07-01 15:01:19.266985: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library nvcuda.dll
2021-07-01 15:01:20.344118: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce GTX 1650 computeCapability: 7.5
coreClock: 1.56GHz coreCount: 16 deviceMemorySize: 4.00GiB deviceMemoryBandwidth: 119.24GiB/s
2021-07-01 15:01:20.344290: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart64_101.dll
2021-07-01 15:01:20.348965: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cublas64_10.dll
2021-07-01 15:01:20.352489: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cufft64_10.dll
2021-07-01 15:01:20.354380: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library curand64_10.dll
2021-07-01 15:01:20.359425: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusolver64_10.dll
2021-07-01 15:01:20.363158: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusparse64_10.dll
2021-07-01 15:01:20.373951: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudnn64_7.dll
2021-07-01 15:01:20.374229: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1858] Adding visible gpu devices: 0
2021-07-01 15:01:20.374635: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in
performance-critical operations: AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-07-01 15:01:20.383852: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2b70ad53e30 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2021-07-01 15:01:20.383978: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device 0: Host, Default Version
2021-07-01 15:01:20.384217: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce GTX 1650 computeCapability: 7.5
coreClock: 1.56GHz coreCount: 16 deviceMemorySize: 4.00GiB deviceMemoryBandwidth: 119.24GiB/s
2021-07-01 15:01:20.384318: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart64_101.dll
2021-07-01 15:01:20.384367: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cublas64_10.dll
2021-07-01 15:01:20.384421: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cufft64_10.dll
2021-07-01 15:01:20.384474: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library curand64_10.dll
2021-07-01 15:01:20.384523: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusolver64_10.dll
2021-07-01 15:01:20.384570: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusparse64_10.dll
2021-07-01 15:01:20.384617: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudnn64_7.dll
2021-07-01 15:01:20.384713: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1858] Adding visible gpu devices: 0
2021-07-01 15:01:20.899017: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1257] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-07-01 15:01:20.899150: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1263] 0
2021-07-01 15:01:20.899296: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1276] 0: N
2021-07-01 15:01:20.899546: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1402] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 2907 MB memory) -> physical GPU (device: 0,
name: NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00.0, compute capability: 7.5)
2021-07-01 15:01:20.903289: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2b7349d8270 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
2021-07-01 15:01:20.903475: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device 0: NVIDIA GeForce GTX 1650, Compute Capability 7.5
2021-07-01 15:01:21.022437: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cufft64_10.dll
2021-07-01 15:01:21.190808: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cublas64_10.dll
....
-----
Ran 4 tests in 8.035s
OK

```

Figura 11 – Teste de unidade python.

## 4 Documentação para o Usuário

As orientações de instalação e uso da aplicação consideram a prévia instalação, por parte do usuário, do Python e do NodeJS.<sup>1,2</sup> Inicialmente, o usuário deve realizar o download do código-fonte do projeto, contido em um repositório GitHub.<sup>3</sup> Caso a ferramenta GitHub já esteja instalada no sistema operacional, o mesmo processo pode ser efetuado através do comando:

```
$ git clone https://github.com/arthursrr/PFP_Gen_Audio_Dataset.git
```

Esse processo de clonagem produz um arquivo .zip que posteriormente precisa ser descompactado.

Para instalar os pacotes do NodeJS necessários o usuário deve entrar no contexto do projeto através do comando:

```
$ cd ./PFP_Gen_Audio_Dataset/PFP_Gen_Audio_Dataset
```

Em seguida instalar todos os pacotes necessários através do comando:

```
$ npm install
```

Este comando, automaticamente buscará o arquivo *package.json* que contém todas as dependências necessárias.

Por fim, e não menos importante, os pacotes Python necessários são instalados automaticamente através do comando:

```
$ pip install -r ./py/requirements.txt
```

Todavia, é importante considerar a prévia instalação das ferramentas CUDA (Versão 10.1) e cuDNN (Versão 7.5). Essas duas ferramentas garantem o acesso correto da GPU pela aplicação Python, com processo de instalação varia dependendo do sistema operacional.<sup>4</sup>

Ao fim da preparação do ambiente basta usar o comando de execução:

```
$ npm start
```

---

<sup>1</sup> <https://www.python.org/downloads/release/python-3610/>

<sup>2</sup> <https://nodejs.org/en/download/current/>

<sup>3</sup> [https://github.com/arthursrr/PFP\\_Gen\\_Audio\\_Dataset.git](https://github.com/arthursrr/PFP_Gen_Audio_Dataset.git)

<sup>4</sup> <https://www.tensorflow.org/install/gpu>

## 4.1 Telas de navegação

Na tela inicial da aplicação (Figura 12) o usuário, através da ação *drag&drop* fornecerá o endereço do diretório contendo os áudios a serem fragmentados. Caso o endereço informado não seja diretório ou não contenha arquivos *.wav*, é gerado um aviso via modal, respectivamente representados nas Figuras 13 e 14.

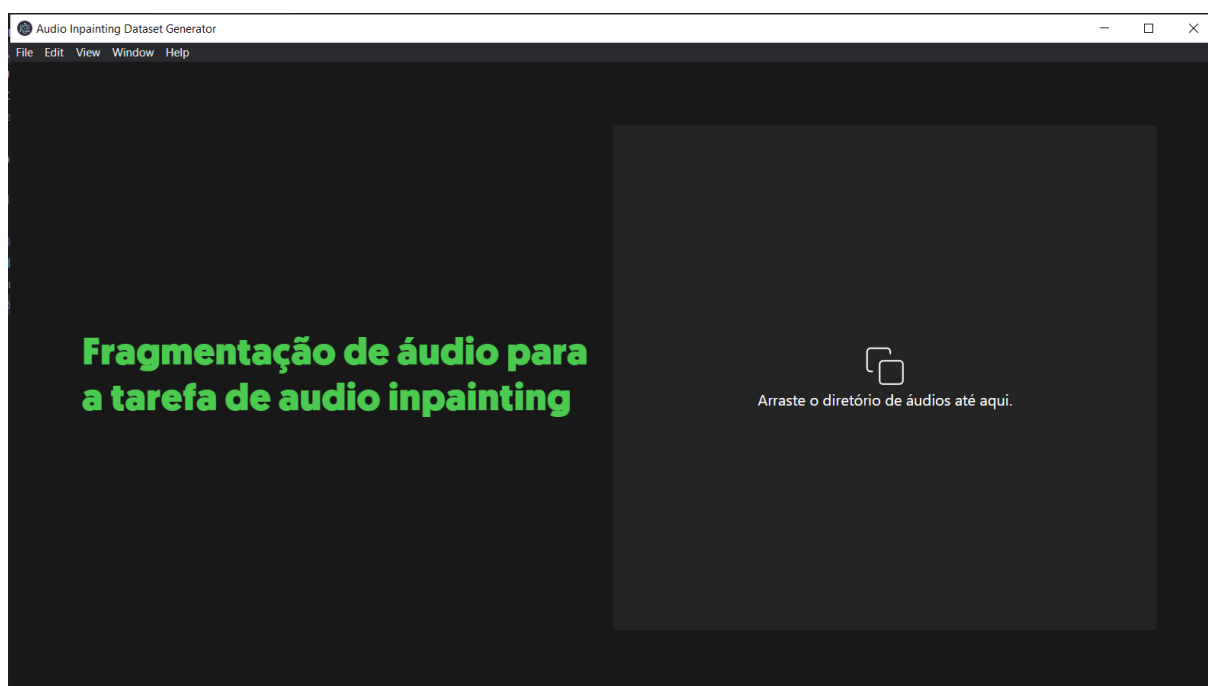


Figura 12 – Tela inicial.

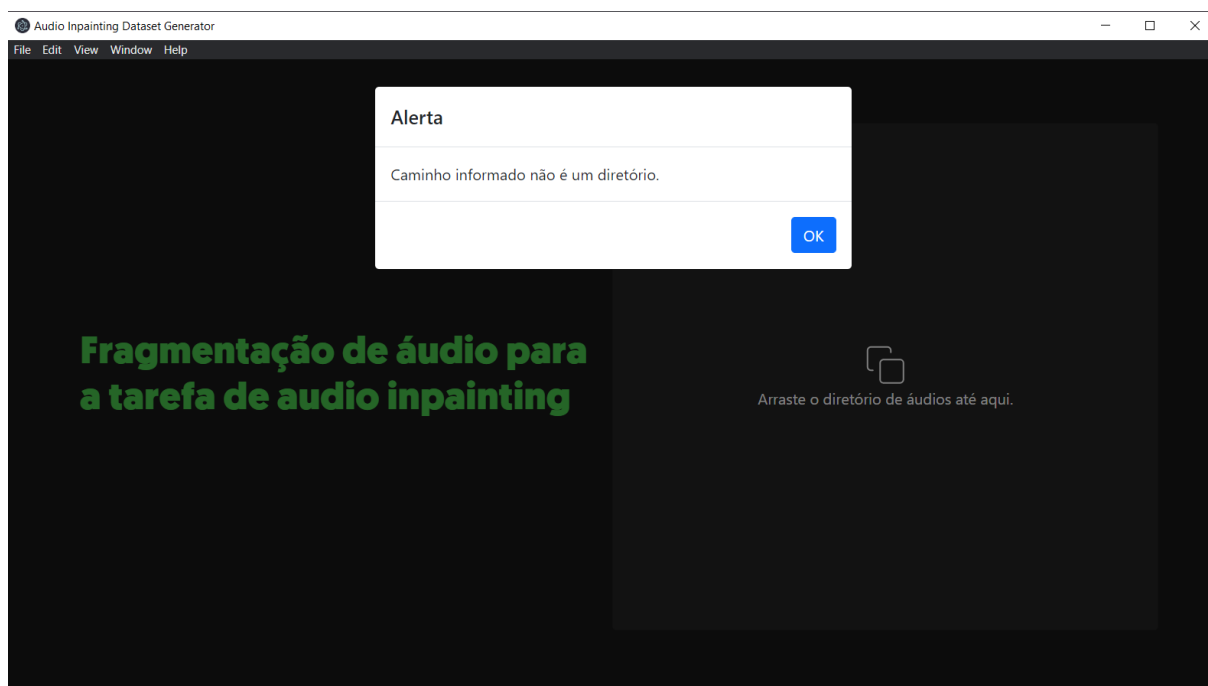


Figura 13 – Modal de aviso para caminho de diretório errado.

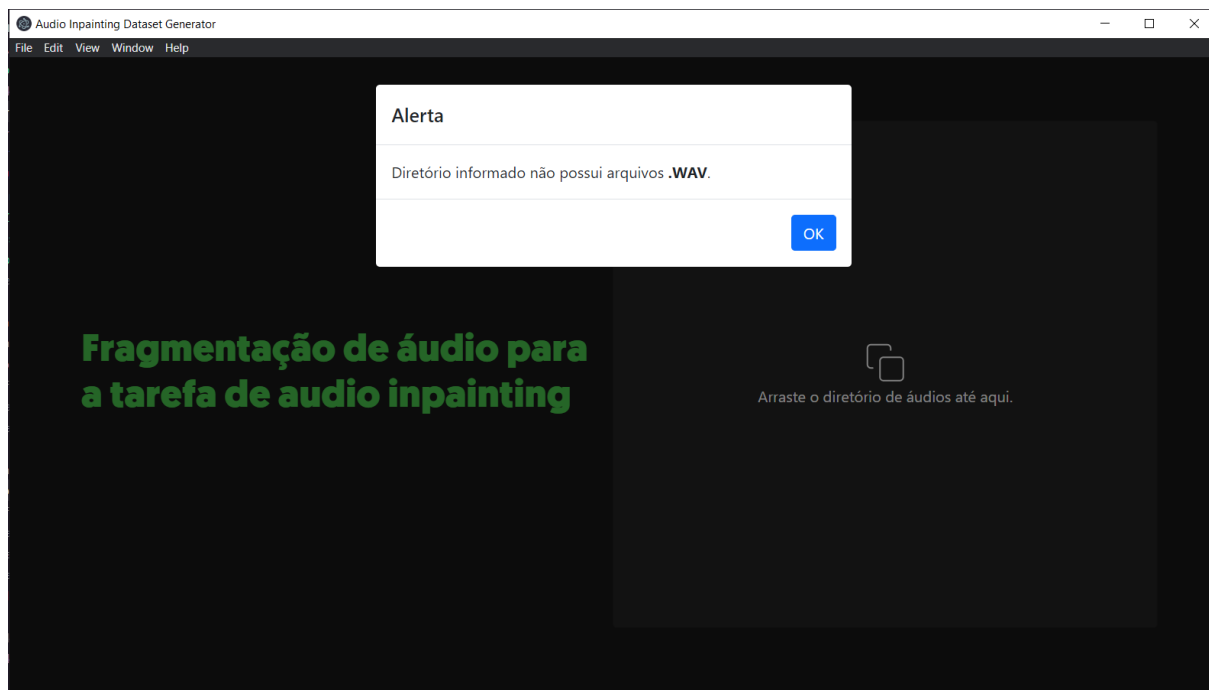


Figura 14 – Modal de aviso para inexistência de arquivos de áudio.

Quando um diretório é corretamente informado, automaticamente a aplicação alterna para a tela de informações do diretório (Figura 15). Nesta tela as informações são assincronamente carregadas, possibilitando o usuário utilizar o player dos arquivos até que as informações sejam finalizadas. Caso o usuário não deseje aguardar o cálculo das informações é possível avançar para a próxima tela ou voltar a tela inicial.

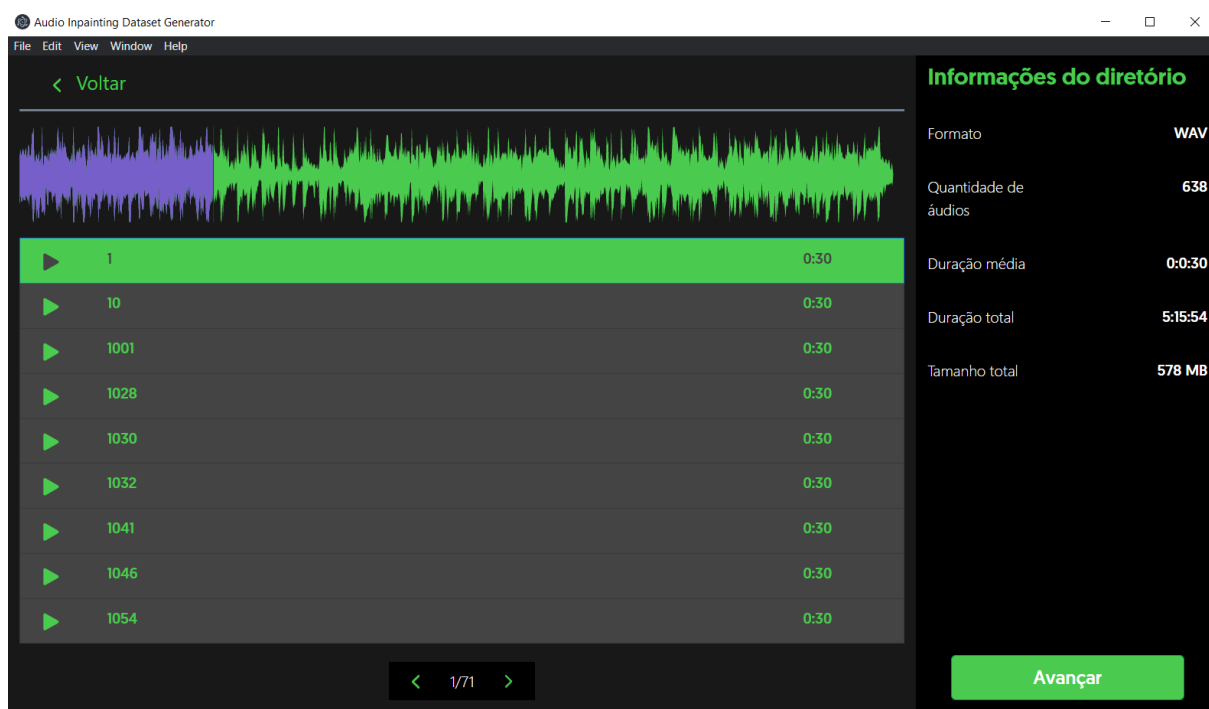


Figura 15 – Tela de informações.

A tela de parâmetros (Figura 16) pode ser considerada a mais importante da aplicação. Nela é possível ter acesso a um player dos áudios de referência, configurar os parâmetros de fragmentação, pré-visualizar o resultado dos parâmetros em um segundo player, configurar o diretório de armazenamento dos áudios fragmentados e escolher se deseja salvar os espectrogramas dos áudios gerados. Ao avançar, os parâmetros de fragmentação são salvos e a aplicação avança para a próxima janela.

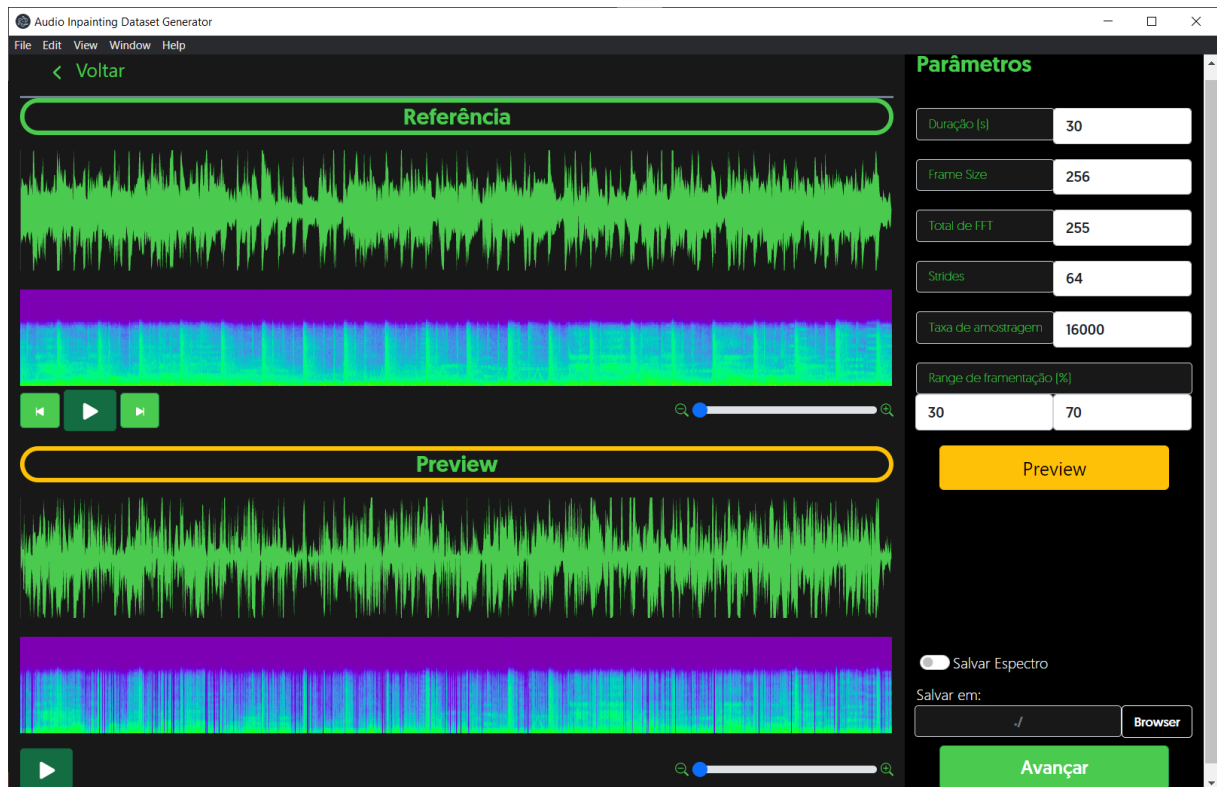


Figura 16 – Tela de parâmetros.

A tela de carregamento (Figura 17) nada mais é que uma barra de progresso do processo sobre o diretório geral. A princípio, o botão avançar dessa tela fica desabilitado até o fim do processo, caso haja alguma inconsistência lógica nos parâmetros informados um aviso é gerado (Figura 18) e o usuário é redirecionado a tela de parâmetros novamente.

Por fim, a tela final (Figura 19) mostra lado a lado duas playlists, uma de referência e a fragmentada gerada pelo processo. Assim o usuário pode auditivamente avaliar como os áudios eram e como ficam. Dessa forma, caso satisfeito a aplicação pode ser fechada ou o usuário pode começar todo o processo do início.



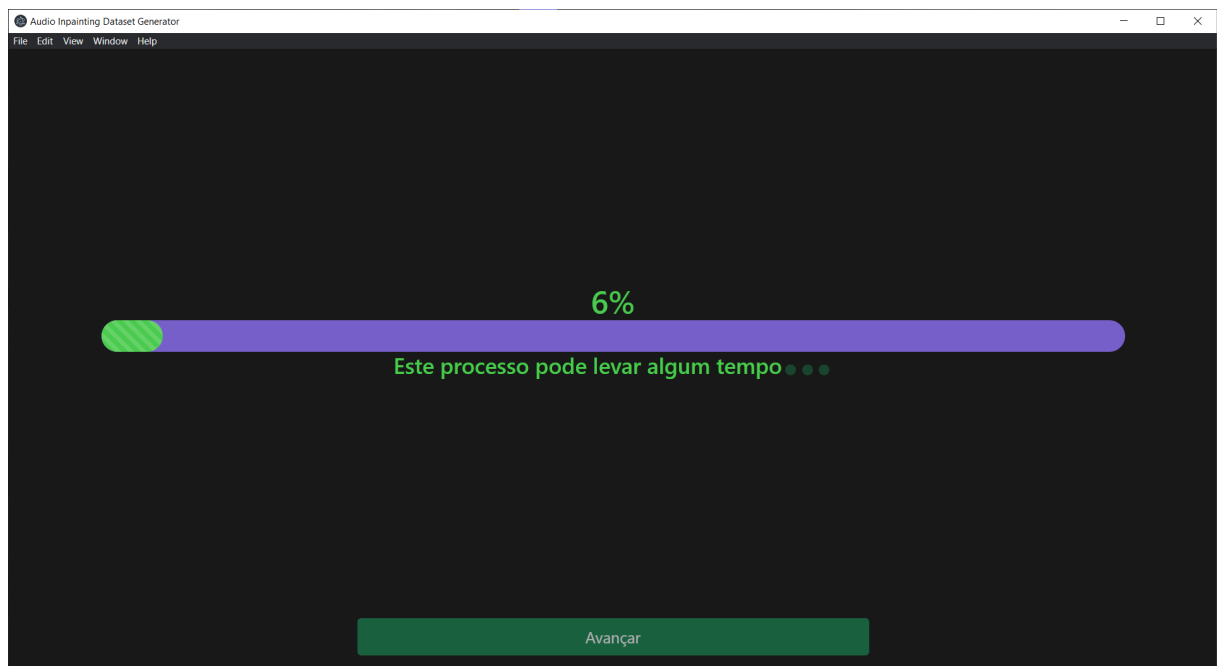


Figura 17 – Tela de carregamento.

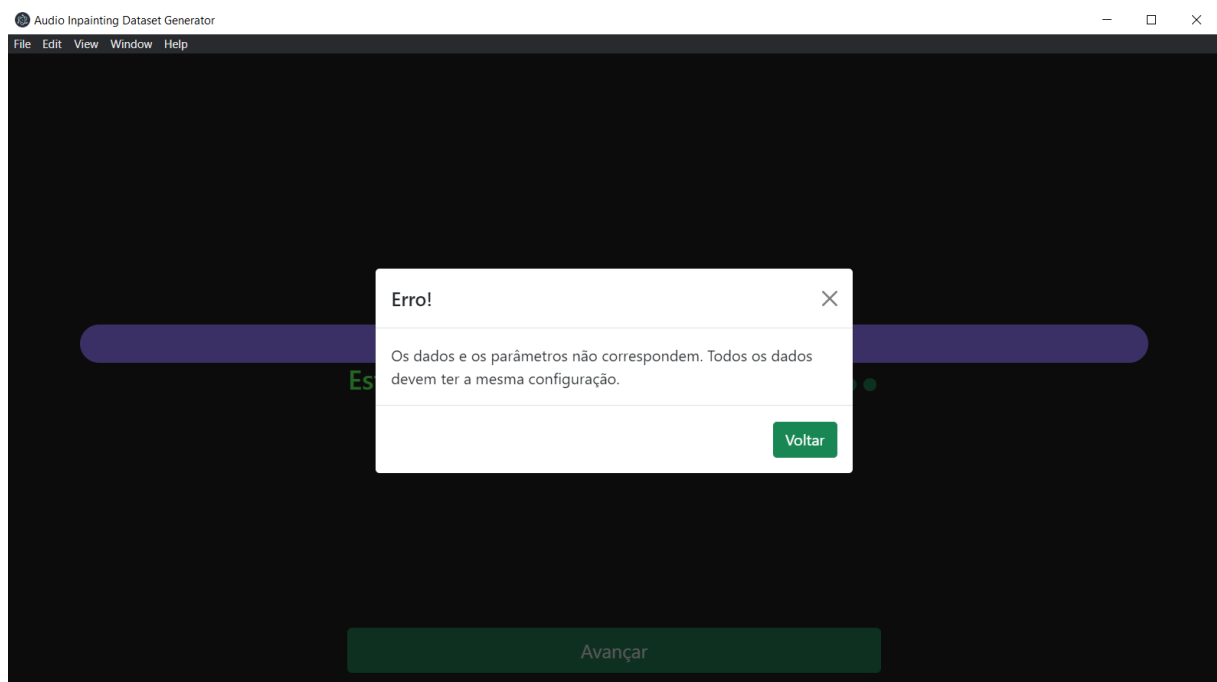


Figura 18 – Aviso de parâmetros equivocados.

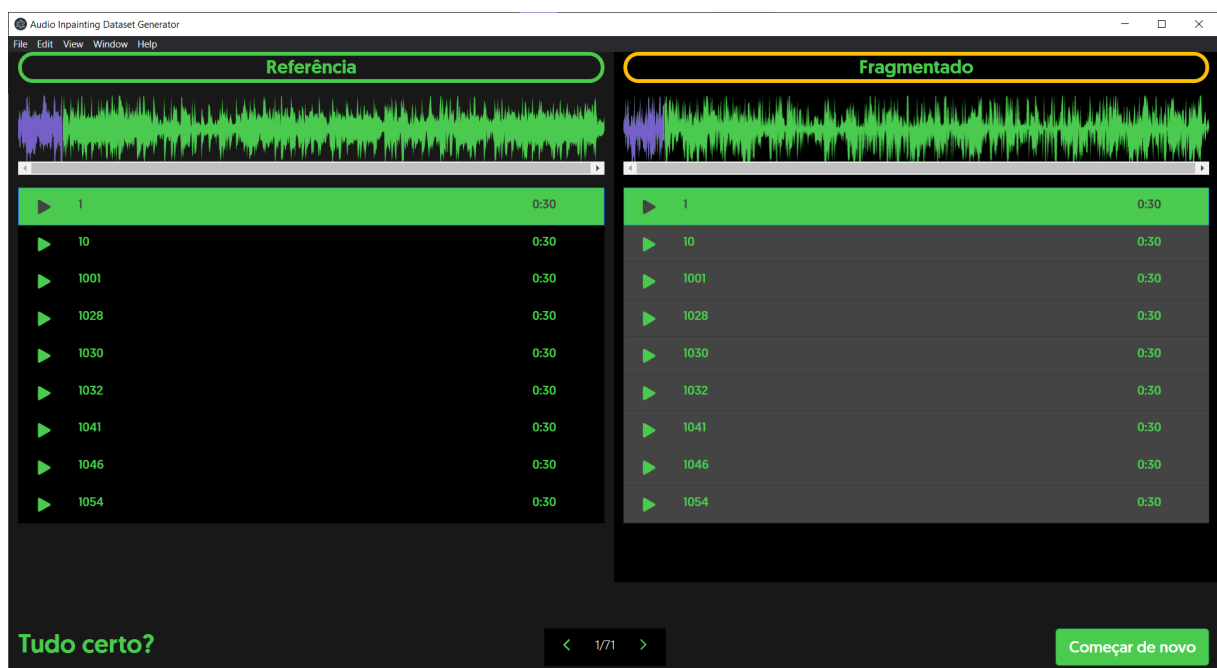


Figura 19 – Tela de Final.

## 5 Código Fonte

Todo o código-fonte deste projeto está disponível no GitHub.<sup>1</sup> Ainda assim, todas as funções documentadas são descritas nas seguintes seções:

### 5.1 Python

#### 5.1.1 ./py/fragment\_audio.py

```
1  '''Autor: Arthur Serra'''
2  ''' Este script descreve funcoes de transformacao e fragmentacao de
    audios      atraves da biblioteca TensorFlow, com suporte para GPUs. A
    execucao desse script depende de uma sequencia de argumentos
    passado ao executa-lo.
3  '''
4  from glob import glob          #Sistema de arquivos
5  import os, sys                #Chamadas de sistema
6  import numpy as np            #Biblioteca numerica
7  import tensorflow as tf        #Biblioteca de tensores
8  import soundfile as sf        #Biblioteca de gerenciamento de arquivos de
    audio
9
10 def audio_to_spectrogram(path, rate=16000, duration = 30,
    frame_length=256, fft_length=255, stride=64, fmin=0, fmax=8000):
11     """
12     Esta funcao recebe um audio como entrada e retorna o espectrograma
    equivalente do mesmo em um tensor.
13     Os valores adotados como padrao nos argumentos resultado em um
    tensor com dimensao de frequencia igual a 128.
14     Recomenda-se utilizar frame_lenght uma potencia de 2, assim para o
    stride quando para o FFT_length.
15     Para esses valores default o fft_length = frame_length - 1 para
    manter o numero de frequencias pares.
16
17     [ARGS]
18         rate: Numero de quadros por segundo do audio
19         duration: Duracao do audio em segundos
20         frame_length: Largura da janela que percorrera o audio
21         fft_length: tamanho do FFT para cada janela
22         stride: tamanho dos saltos
23         fmin: frequencia minima
24         fmax: frequencia maxima
```

<sup>1</sup> [https://github.com/arthursrr/PFP\\_Gen\\_Audio\\_Dataset.git](https://github.com/arthursrr/PFP_Gen_Audio_Dataset.git)

```

25     [RETUNR]
26         tensor format [Time, Frequence]
27     """
28     if os.path.isfile(path):
29         raw_audio = tf.io.read_file(path)
30         audio_tensor = tf.audio.decode_wav(raw_audio,
desired_channels=1, desired_samples=rate*duration)
31         audio_tensor = tf.squeeze(audio_tensor.audio.numpy(), axis=[-1])
32         spectrogram = tf.math.abs(tf.signal.stft(audio_tensor,
33                                                     frame_length=frame_length,
34                                                     frame_step=stride,
35                                                     fft_length=fft_length,
36                                                     window_fn=tf.signal.hann_window,
37                                                     pad_end=True))
38     else:
39         spectrogram = None
40
41     return spectrogram
42
43 def griffin_lim(S, frame_length=256, fft_length=255, stride=64):
44     '''
45     Esta funcao recebe tensor contendo um espectrograma e efetua a
46     tranformada inversa atraves do algoritmo Griffin-Lim.
47     Extremamente importante que os argumentos frame_length, fft_length
48     e stride sejam extamente iguais aos mesmos no processo de
49     codificacao do espetrograma
50     TensorFlow implementation of Griffin-Lim Based on
51     https://github.com/Kyubyong/tensorflow-exercises/blob/
52     master/Audio_Processing.ipynb
53
54     [ARGS]
55         frame_length: Largura da janela que percorrera o audio
56         fft_length: tamanho do FFT para cada janela
57         stride: tamanho dos saltos
58     [RETUNR]
59         tensor format: waveform
60     '''
61     S = tf.expand_dims(S, 0)
62     S_complex = tf.identity(tf.cast(S, dtype=tf.complex64))
63     y = tf.signal.inverse_stft(S_complex, frame_length, stride,
64                               fft_length=fft_length)
65     for i in range(100):
66         est = tf.signal.stft(y, frame_length, stride,
67                               fft_length=fft_length)
68         angles = est / tf.cast(tf.maximum(1e-16, tf.abs(est)),
69                                tf.complex64)

```

```

63     y = tf.signal.inverse_stft(S_complex * angles, frame_length,
64                                stride, fft_length=fft_length)
65     waveform = tf.squeeze(y, 0)
66     return waveform
67
68 def fragment_spectrogram(audio_dir,
69                           save_dir,
70                           frag_min=30,
71                           frag_max=70,
72                           rate=16000,
73                           duration=30,
74                           frame_length=256,
75                           fft_length=255,
76                           stride=64,
77                           subtype='PCM_16',
78                           save_spec=False):
79     '''
80     Esta funcao recebe o path da base de audio WAV produz a
81     fragmentacao e salva os audio novos num path tambem informado.
82     Esse processo sem auxilio de GPU pode levar algum tempo.
83     Recomendo presevar os demais paramentos exceto rate e duracao
84     [ARGS]
85         audio_dir: diretorio onde os audio WAV estao armazenados
86         save_dir: Diretorio onde serao salvos os novos audios
87         frag_min: Porcentagem minima de cortes
88         frag_max: Porcentagem maxima de cortes
89         rate: Numero de quadros por segundo do audio
90         duration: Duracao do audio em segundos
91         frame_length: Largura da janela que percorrera o audio
92         fft_length: Tamanho do FFT para cada janela
93         stride: Tamanho dos saltos
94         spec: Dados serao salvos na forma de espetrongrama (.npy)
95     [RETUNR]
96         [Boolean]: Sucesso do processo
97     '''
98     cortes = [] #lista de proporcao de cortes
99     data_spec = None
100     i_patch = 0
101
102     audio_dir = audio_dir.replace('\\', '/')
103     save_dir = save_dir.replace('\\', '/')
104
105     audio_paths = glob(audio_dir+"/*.wav")
106     if len(glob(audio_dir+"/*.wav"))==0:

```

```

107     audio_paths = glob(audio_dir+"/**/*.wav") #lista de paths de
audios
108
109
110     spec_dest = save_dir+"/Espectrogramas/"
111     audio_dest = save_dir+"/Audios/"
112
113     if save_spec:
114         if not os.path.exists(spec_dest):
115             os.mkdir(spec_dest)
116
117     if not os.path.exists(audio_dest):
118         os.mkdir(audio_dest)
119
120     try:
121         for i in audio_paths:
122             #transforma o audio em um espectrograma
123             spec = audio_to_spectrogram(i, rate=rate,
duration=duration, frame_length=frame_length, fft_length=fft_length,
stride=stride)
124
125             #expande a dimensao do tensor
126             spec = tf.expand_dims(spec, axis=0)
127
128             #salva valores das dimensoes iniciais do spectrograma
129             n_times = spec.get_shape().as_list()[1]
130             n_freq = spec.get_shape().as_list()[2]
131
132             #variavel de armazenamento auxiliar
133             data_spec = None
134
135             if len(cortes) == 0:
136                 fr_min = int((frag_min/100) * n_freq)
137                 fr_max = int((frag_max/100) * n_freq)
138                 n_patch = len(audio_paths) * (n_times//n_freq)
139
140                 cortes = np.random.randint(fr_min, fr_max, size=n_patch)
141
142             ini = 0
143             end = ini+n_freq
144
145             #processo de janelamento
146             while end < n_times:
147                 if data_spec == None:
148                     data_spec = spec[:,ini:end,:]
149                 else:

```

```

150         data_spec = tf.concat([data_spec,
spec[:,ini:end,:]], 0)
151         ini = end
152         end = ini+n_freq
153
154         #aplicando fragmentacao
155         data_spec = data_spec.numpy()
156         for k in range(data_spec.shape[0]):
157             time_cortes =
np.random.permutation(np.arange(n_freq))[:cortes[i_patch]]
158             data_spec[k, time_cortes, :] = 0.0
159             i_patch += 1
160             data_spec = np.reshape(data_spec,
(n_times-(n_times%n_freq), n_freq))
161
162             if save_spec:
163
np.save(spec_dest+os.path.basename(i).split('.')[0]+'..npy',
data_spec)
164
165             #transformando spectrograma para onda
166             wave = griffin_lim(data_spec, frame_length=frame_length,
fft_length=fft_length, stride=stride)
167
168             #salva audio fragmentado
169             sf.write(audio_dest+os.path.basename(i), wave, rate,
subtype=subtype)
170             return True
171         except:
172             return False
173
174 def Preview_fragment_spectrogram(audio_path,
175                                 temp_dir,
176                                 frag_min=30,
177                                 frag_max = 70,
178                                 rate=16000,
179                                 duration=30,
180                                 frame_length=256,
181                                 fft_length=255,
182                                 stride=64,
183                                 subtype='PCM_16'):
184     '''
185         Esta funcao recebe o path de um audio WAV produz a fragmentacao
em um arquivo temporario.
186         Esse processo sem auxilio de GPU pode levar algum tempo.
187         Recomendo presevar os demais paramentros exceto rate e duracao
188         [ARGS]

```

```

189         audio_path: Caminho da amostra a ser fragmetnada
190         temp_dir: Diretorio de salvamento do audio temporario
191         frag_min: Porcentagem minima de cortes
192         frag_max: Porcentagem maxima de cortes
193         rate: Numero de quadros por segundo do audio
194         duration: Duracao do audio em segundos
195         frame_length: Largura da janela que percorrera o audio
196         fft_length: Tamanho do FFT para cada janela
197         stride: Tamanho dos saltos
198     [RETUNR]
199         [Boolean]: Sucesso do processo
200     '''
201     try:
202         cortes = [] #lista de proporcao de cortes
203         data_spec = None
204         i_patch = 0
205
206         #transforma o audio em um espectrograma
207         spec = audio_to_spectrogram(audio_path, rate=rate,
duration=duration, frame_length=frame_length, fft_length=fft_length,
stride=stride)
208
209         #expande a dimensao do tensor
210         spec = tf.expand_dims(spec, axis=0)
211
212         #salva valores das dimensoes iniciais do spectrograma
213         n_times = spec.get_shape().as_list()[1]
214         n_freq = spec.get_shape().as_list()[2]
215
216         #variavel de armazenamento auxiliar
217         data_spec = None
218
219         if len(cortes) == 0:
220             fr_min = int((frag_min/100) * n_freq)
221             fr_max = int((frag_max/100) * n_freq)
222             n_patch = (n_times//n_freq)
223
224             cortes = np.random.randint(fr_min, fr_max, size=n_patch)
225
226         ini = 0
227         end = ini+n_freq
228
229         #processo de janelamento
230         while end < n_times:
231             if data_spec == None:
232                 data_spec = spec[:,ini:end,: ]
233             else:

```



```

234         data_spec = tf.concat([data_spec, spec[:,ini:end,:]], 0)
235         ini = end
236         end = ini+n_freq
237
238         #aplicando fragmentacao
239         data_spec = data_spec.numpy()
240         for k in range(data_spec.shape[0]):
241             time_cortes =
np.random.permutation(np.arange(n_freq))[:cortes[i_patch]]
242             data_spec[k, time_cortes, :] = 0.0
243             i_patch += 1
244             data_spec = np.reshape(data_spec, (n_times-(n_times%n_freq),
n_freq))
245
246             #transformando spectrograma para onda
247             wave = griffin_lim(data_spec, frame_length=frame_length,
fft_length=fft_length, stride=stride)
248
249             #salva audio fragmentado
250             sf.write(temp_dir+"frag_temp.wav", wave, rate, subtype=subtype)
251             return True
252         except:
253             return False
254
255 if __name__ == '__main__':
256     if sys.argv[1] == 'true':
257         print(Preview_fragment_spectrogram(sys.argv[2],
258                                             sys.argv[3],
259                                             int(sys.argv[4]),
260                                             int(sys.argv[5]),
261                                             int(sys.argv[6]),
262                                             int(sys.argv[7]),
263                                             int(sys.argv[8]),
264                                             int(sys.argv[9]),
265                                             int(sys.argv[10]),
266                                             sys.argv[11]
267                                             ))
268     else:
269         if sys.argv[12] == 'true':
270             spec = True
271         else:
272             spec = False
273         print(fragment_spectrogram(sys.argv[2],
274                                    sys.argv[3],
275                                    int(sys.argv[4]),
276                                    int(sys.argv[5]),
277                                    int(sys.argv[6]),

```

```

278         int(sys.argv[7]),
279         int(sys.argv[8]),
280         int(sys.argv[9]),
281         int(sys.argv[10]),
282         sys.argv[11],
283         spec))

```

## 5.2 JavaScript

### 5.2.1 ./js/main.js

```

1  //Autor: Arthur Serra
2  /* ===== main.js =====
3   * Modulos para controlar a vida util da aplicacao e criar uma janela
4   * de navegador nativo (Chromium).
5   * Este script tambem funciona como intermediario de comunicacao entre
6   * os processos.
7   * =====
8  */
9
10 const { app, BrowserWindow, ipcMain, dialog } = require('electron');
    /*Importa funcoes do framework electron.js*/
11
12 var path_dir = null;           /*[String] caminho do diretorio dos
    arquivos de referencia*/
13 var save_dir = null;           /*[String] caminho do diretorio de
    armazenamento dos arquivos produzidos*/
14 var list_files = null;         /*[Array] caminhos de todos os
    arquivos do diretorio de referencia*/
15 var list_generated_files = null; /*[Array] caminhos de todos os
    arquivos do diretorio produzido pela aplicacao*/
16 var frag_args = null;          /*[Array] lista de variaveis do
    processo de producao Python*/
17
18 function createWindow () {
19     // Cria a janela de navegador.
20     const mainWindow = new BrowserWindow({
21         width: 1280,           //largura inicial da janela
22         height: 720,           //altura inicial da janela
23         webPreferences: {
24             nodeIntegration: true,           //permite a importacao de bibliotecas
do Node.js
25             contextIsolation: false,         //permite que outros Scripts alem
deste possa fazer importacoes
26             enableRemoteModule: true         //Permite chamadas de dialogo remoto
com o sistema operacional

```

```

27     }
28 })
29 // carregar o home.html do aplicativo.
30 mainWindow.loadFile('./html/home.html')
31 // mainWindow.webContents.openDevTools() //Habilita o console de
    depuracao durante a execucao da aplicacao
32 }
33
34 /* Este metodo sera chamado quando o Electron tiver terminado
35  * inicializacao e esta pronto para criar janelas de navegador.
36  * Algumas APIs podem ser usadas somente depois que este evento ocorre.
37  */
38 app.whenReady().then(() => {
39     createWindow()
40     app.on('activate', function () {
41         // Em macOS e comum recriar uma janela no aplicativo quando o icone
42         da doca e clicado e nao ha outras janelas abertas.
43         if (BrowserWindow.getAllWindows().length === 0) createWindow()
44     })
45
46 // Saia quando todas as janelas estiverem fechadas, exceto em macOS.
47 // E comum que as aplicacoes e sua barra de menu permanecam ativas ate
48   que o usuario saia explicitamente com Cmd + Q.
49 app.on('window-all-closed', function () {
50     if (process.platform !== 'darwin') app.quit()
51 })
52
53 /*EVENTOS DE COMUNICACAO ENTRE OS PROCESSOS*/
54 //inicia varios evento de escuta para definir os valores das variaveis
55 ipcMain.on("toMain", (event, args) => {
56     path_dir = args[0]
57     list_files = args[1]
58 });
59
60 ipcMain.on("destToMain", (event, args) => {
61     list_generated_files = args;
62 });
63
64 ipcMain.on("argsToMain", (event, args) => {
65     frag_args = args
66 });
67
68 ipcMain.on("argsfromMain", (event, args) => {
69     event.returnValue = frag_args
70 });

```

```

71
72 ipcMain.on("fromMain", (event, args) => {
73     event.returnValue = [path_dir, list_files]
74 });
75
76 ipcMain.on("destTofromMain", (event, args) => {
77     event.returnValue = list_generated_files
78 });
79
80 /*Este evento e especial, pois faz uma chamada de dialogo com o
81 * explorador de arquivos do sistema operacional nativo
82 */
83 ipcMain.on('show-open-dialog', (event, arg)=> {
84     save_dir = dialog.showOpenDialogSync({
85         properties: ['openDirectory'] //Apenas diretorios podem ser
            enxergados pela janela de dialogo
86     });
87     event.returnValue = save_dir
88 })

```

### 5.2.2 ./js/home.js

```

1 //Autor: Arthur Serra
2 /* ===== home.js =====
3 * Este script produz todos os eventos de controle da pagina home.html
4 * =====
5 */
6
7 const{ ipcRenderer } = require('electron') //Importa funcao de
    comunicacao com o processo principal
8 const fs = require('fs') //Importa sistema de
    arquivos do javaScript
9 const glob = require('glob').Glob //Importa sistema de
    arquivos recursivo para iterar sobre um diretorio
10 let $ = jQuery = require('jquery') //Importa comandos JQuery
11
12 //Controle de drag&drop
13 document.addEventListener('drop', (event) => {
14     event.preventDefault();
15     event.stopPropagation();
16
17     for (const f of event.dataTransfer.files) {
18         //confirma se o endereco passado e um diretorio
19         if (fs.lstatSync(f.path).isDirectory())
20         {
21             //itera sobre todos os arquivos .wav do diretorio obtido
22             glob(f.path + '/*/*.wav', {}, (err, files)=>{

```

```

23         //Confirma se ha arquivos desse formato
24         if (files.length > 0) {
25             //Envia a lista dos arquivos .wav para o processo
                principal
26                 ipcRenderer.send('toMain', [f.path, files]);
27                 //Vai para a pagina de informacoes do diretorio
28                 window.location.replace("../html/infopage.html")
29             } else {
30                 //Abre um modal informando que nao ha arquivos .wav
                no diretorio
31                 $("#notWAV").modal('show');
32             }
33         })
34     }else{
35         //Abre um modal informando ao usuario que o caminho
        informado nao corresponde a um diretorio
36         $("#notDir").modal('show');
37     }
38 }
39 });
40
41 //Monitora eventos de Drag&Drop
42 document.addEventListener('dragover', (e) => {
43     e.preventDefault();
44     e.stopPropagation();
45 });
46
47 document.addEventListener('dragenter', (event) => {
48     console.log('File is in the Drop Space');
49 });
50
51 document.addEventListener('dragleave', (event) => {
52     console.log('File has left the Drop Space');
53 });

```

### 5.2.3 ./js/infopage.js

```

1 //Autor: Arthur Serra
2 /* ===== infopage.js
   =====
3  * Este script produz todos os eventos de controle da pagina
   infopage.html
4  * Tendo como principal evento gerar uma playlist dos dados presentes
   no
5  * diretorio informado e carregar algumas informacoes gerais dos dados.
6  * Tais como tamanho, duracao media e total.
7  * =====

```

```

8  */
9
10 const{ ipcRenderer } = require('electron'); //Importa funcao de
    comunicacao com o processo principal
11 const mm = require('music-metadata');      //Importa biblioteca
    obtencao de metadados de audios
12 var fs = require('fs');                    //Importa sistema de
    arquivos
13 var $ = jQuery = require('jquery');        //Importa comandos JQuery
14 var WaveSurfer = require('wavesurfer.js'); //Importa biblioteca de
    visualizacao de ondas e espectro
15 var path = require('path');                //Importa sistema de
    arquivos
16
17 var args = ipcRenderer.sendSync('fromMain', ""); // [Array] faz uma
    chamada ao evento do processo principal para obter o diretorio geral
    e sua lista de arquivos
18 var path_dir = args[0];                    // [String] caminho do
    diretorio geral
19 var list_files = args[1];                  // [Array] caminho de
    todos os arquivos do diretorio principal
20 delete args
21
22 function unitAdjustSize(fileSizeInBytes){
23     /* Esta funcao recebe um numero inteiro que representa uma
24     * escala de bytes e transforma em escalas maiores caso necessario
25     * <ATRIBUTOS>
26     *     fileSizeInBytes: [inteiro] Tamnho dos arquivos em bytes
27     * <RETORNO>
28     *     [inteiro] Tamnho dos arquivos em escala maior
29     */
30     if (fileSizeInBytes < 1048576) return (fileSizeInBytes /
1024).toFixed(0) + " KB"; //Kilobytes
31     else if (fileSizeInBytes < 1073741824) return (fileSizeInBytes /
1048576).toFixed(0) + " MB"; //Megabytes
32     else return (fileSizeInBytes / 1073741824).toFixed(0) + " GB";
        //Gigabytes
33 }
34
35 function unitAdjustTemp(tempTotal){
36     /* Esta funcao recebe um numero inteiro que representa a duracao
37     * de tempo em segundos e transforma para o formato HH:MM:SS
38     * <ATRIBUTOS>
39     *     tempTotal: [inteiro] Duracao em segundos
40     * <RETORNO>
41     *     [String] Horas:Minutos:Segundos
42     */

```

```

43     var hours = Math.floor(tempTotal / (60 * 60));           //quantas horas
    inteira ha na quantidade de segundos total
44
45     var divisor_for_minutes = tempTotal % (60 * 60);
46     var minutes = Math.floor(divisor_for_minutes / 60); //quantos
    minutos inteiro ha na quantidade de segundos fora as horas
47
48     var divisor_for_seconds = divisor_for_minutes % 60;
49     var seconds = Math.ceil(divisor_for_seconds);           //quantos
    segundos restantes fora as horas e os minutos
50
51     return hours + ":" + minutes + ":" + seconds
52 }
53
54 async function getDurationTrack(track, id){
55     /* Esta funcao recebe um numero inteiro que representa a duracao
56     * de tempo em segundos de uma determinada faixa e transforma
57     * para o formato mm:ss
58     * <ATRIBUTOS>
59     *     track: [String] caminho do arquivo
60     *     id:     [String] identificador no formulario infopage.html
61     * <RETORNO>
62     *     [null]
63     */
64     let metadata = await mm.parseFile(track);                 //obtem os
    metadados do uma arquivo de audio
65     let secs = metadata.format.duration;                       //extrai a
    duracao total em segundos dos metadados
66
67     var divisor_for_minutes = secs % (60 * 60);
68     var minutes = Math.floor(divisor_for_minutes / 60); //obtem os
    minutos inteiros
69
70     var divisor_for_seconds = divisor_for_minutes % 60;
71     var seconds = Math.ceil(divisor_for_seconds);           //obtem os
    segundos inteiro fora os minutos
72
73     $(id).text(minutes + ":" + seconds)                       //Atualiza o
    texto no fomulario html via JQuery
74 }
75
76 async function getMetadata(list_files){
77     /* Esta funcao recebe um array de caminhos e retorna os valores
78     * totais de duracao, armazenamentos e taxa de atualizacao
79     * <ATRIBUTOS>
80     *     list_files: [Array] caminho do arquivo
81     * <RETORNO>

```

```

82      *      [Array]
83      */
84      //variaveis auxiliares
85      var count = 0;
86      var perc = 0;
87      //armazena os valores totais
88      var tempTotal = 0;
89      var fileSizeInBytes = 0
90      var samplerate = 0
91
92      for (const element of list_files) {
93          var stats = fs.statSync(element)
94          fileSizeInBytes += stats["size"]           //obtem o tamanho
do arquivo
95          let metadata = await mm.parseFile(element);
96          samplerate += metadata.format.sampleRate   //obtem a taxa de
atualizacao do audio
97          tempTotal += metadata.format.duration     //obtem a duracao
do audio em segundos
98
99          count += 1;
100
101          perc = parseInt(count/list_files.length*100)
          //determina a porcentagem de arquivos processados
102          $(''.progress-bar').css('width', perc+'%').attr('aria-valuenow',
perc); //incrementa a barra de progresso no formulario html
103      }
104
105      return [tempTotal, fileSizeInBytes, samplerate]
106  }
107
108  async function chanegValues(){
109      /* Esta funcao atualiza os valores no formulario HTML
110      * <ATRIBUTOS>
111      *      [null]
112      * <RETORNO>
113      *      [null]
114      */
115      $('#Progress').modal('show');
          //Mostra um modal com a barra de progresso dos dados via
jQuery
116      let total_args = await getMetadata(list_files);
          //[Array] Tempo, armazenamento e taxa de atualizacao total
117      let formato = "WAV";
          //[String] Formato dos arquivos (Atualmente apenas arquivos
WAV sao aceitos)
118      let quantidade = list_files.length.toString();

```



```

119         //[String] Quantidade de audios
120         let tempTotal = unitAjustTemp(total_args[0]).toString();
121         //[String] Tempo total ajustado para HH:mm:ss
122         let tamTotal = unitAjustSize(total_args[1]).toString();
123         //[String] Tamanho total da base
124         let tempMedio =
125         unitAjustTemp(total_args[0]/list_files.length).toString()
126         //[String] Duracao media dos audios
127
128         //COMANDOS JQuery PARA ATUALIZACAO DO FORMULARIO HTML
129         $("#formato").text(formato);
130         $("#quantidade").text(quantidade);
131         $("#tamTotal").text(tamTotal);
132         $("#tempMedio").text(tempMedio);
133         $("#tempTotal").text(tempTotal);
134
135         $("#Progress").modal('hide'); //Ao fim do processo modal de
136         progresso e ocultado
137     }
138
139     //ICONES DE PLAY E PAUSE
140     var pause_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
141         height="30" fill="currentColor" class="bi bi-pause-fill" viewBox="0
142         0 15 15"><path d="M5.5 3.5A1.5 1.5 0 0 1 7 5v6a1.5 1.5 0 0 1-3
143         0V5a1.5 1.5 0 0 1 1.5-1.5zm5 0A1.5 1.5 0 0 1 12 5v6a1.5 1.5 0 0 1-3
144         0V5a1.5 1.5 0 0 1 1.5-1.5z"/></svg>';
145     var play_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
146         height="30" fill="currentColor" class="bi bi-play-fill" viewBox="0 0
147         15 15"><path d="m11.596 8.697-6.363
148         3.692c-.54.313-1.233-.066-1.233-.697V4.308c0-.63.692-1.01
149         1.233-.696l6.363 3.692a.802.802 0 0 1 0 1.393z"/></svg>';
150
151     var currentTrack = 0; // [Inteiro] Indice da faixa a ser carregada
152
153     function player(links){
154         /* Esta funcao produz um player de audio dado um array de caminhos
155         * <ATRIBUTOS>
156         *     links: [Array] caminhos dos arquivo
157         * <RETORNO>
158         *     [null]
159         */
160         currentTrack = 0
161         // CARREGA UMA FAIXA DADOS UM INDICE ATUAL
162         let setCurrentSong = function(index) {
163             links[currentTrack].classList.remove('active');
164             links[currentTrack].childNodes[1].innerHTML = play_icon;
165             currentTrack = index;
166         }
167     }

```

```

152     links[currentTrack].classList.add('active');
153     links[currentTrack].childNodes[1].innerHTML = pause_icon;
154     wavesurfer.load(links[currentTrack].attributes.href.nodeValue);
155 };
156
157 // CARREGA UM AUDIO DADO EVENTO DE CLICK
158 Array.prototype.forEach.call(links, function(link, index) {
159     link.addEventListener('click', function(e) {
160         e.preventDefault();
161         if (links[index].classList.contains('active')) {
162             //CASO O AUDIO DE ESTEJA CARREGADO E FEITO APENAS UM
GERENCIAMENTO DE PLAY E PAUSE
163             if (links[index].classList.contains('pause')) {
164                 wavesurfer.play();
165                 links[index].childNodes[1].innerHTML = pause_icon;
166                 links[index].classList.remove('pause');
167             } else {
168                 wavesurfer.pause();
169                 links[index].childNodes[1].innerHTML = play_icon;
170                 links[index].classList.add('pause');
171             }
172         } else {
173             //CASO O AUDIO AINDA NAO ESTEJA CARREGADO
174             setCurrentSong(index);
175         }
176     });
177 });
178 }
179
180 var wavesurfer // [Object] Instancia gerenciamento de audio
181
182 window.onload = function () {
183     chanegValues(); //executa ao iniciar a pagina
184 }
185
186 document.getElementById("btnVoltar").addEventListener("click",
    function(){
187     window.location.replace("../html/home.html"); //volta para a pagina
    inicial ao clicar no botao
188 });
189
190 document.addEventListener('DOMContentLoaded', function() {
191     //instacia o objeto de visualizacao ao carregar a pagina
192     wavesurfer = WaveSurfer.create({
193         container: '#waveform',
194         waveColor: '#4ACA4E',
195         progressColor: '#765FC9',

```

```

196         height: 100
197     });
198 });
199
200 document.addEventListener('DOMContentLoaded', function() {
201     let naudio = 9; //[[Inteiro]
202     Define a quantidade de audio aparecerar por vez em uma pagina
203
204     let beg_pos = 0; //[[Inteiro]
205     Indice do primeiro audio na pagina em relacao a lista de todos oa
206     audios
207
208     let end_pos; //[[Inteiro]
209     Indice do ultimo audio na pagina em relacao a lista de todos oa
210     audios
211
212     let playlist = document.getElementById('playlist'); //[[Objeto]
213     Carrega o elemento de visualizacao da playlist
214
215     let npage = document.getElementsByClassName('npage'); //[[Objeto]
216     Carrega o elemento de posicao na paginacao
217
218     let currentPage = 1; //[[Inteiro]
219     pagina atual
220
221     let lastPage = Math.ceil(list_files.length/naudio); //[[Inteiro]
222     numero total de paginas
223
224     npage[0].innerHTML = currentPage+"/"+lastPage; //Define o
225     valor da pagina atual no formulario
226
227     if (list_files.length > naudio) {
228         //Checa de o total de arquivos e inferior ao limite de exibicao
229         por vez
230
231         end_pos = naudio;
232         for (let index = 0; index < naudio; index++) {
233             //Produz os containers de exibicao da playlist
234             $('#playlist').append('<div class="list-group-item
235             list-group-item-action audio d-flex" id=audio'+
236             index+' href='+list_files[index]+'
237             '> <div class="col-1" id="icone">'+
238             play_icon+
239             '</div> <div class="col-9" id="nome">'+
240             path.parse(list_files[index]).name+
241             '</div><div id="duracao" class="col-2
242             duracao'+index+'"></div></div>');
243
244             getDurationTrack(list_files[index], ".duracao"+index);
245             //obtem duracao de cada faixa
246         }
247     }
248 }

```

```

229     } else {
230         end_pos = list_files.length;
231         for (let index = 0; index < list_files.length; index++) {
232             //Produz os containers de exibicao da playlist
233             $('#playlist').append('<div class="list-group-item
list-group-item-action audio d-flex" id=audio'+
234             index+' href='+list_files[index]+'
235             '> <div class="col-1" id="icone">'+
236             play_icon+
237             ' </div> <div class="col-9" id="nome">'+
238             +path.parse(list_files[index]).name+
239             '</div><div id="duracao" class="col-2
duracao'+index+'"></div></div>');
240
241             getDurationTrack(list_files[index], ".duracao"+index);
242             //obtem duracao de cada faixa
243         }
244     }
245
246     let links = document.querySelectorAll('.audio'); //[Array] obtem os
cominhos de cada container de audio em exibicao
247
248     player(links); //inicia o player
249
250     wavesurfer.on('ready', function(e) {
251         //play caso carregado
252         wavesurfer.play();
253     });
254
255     wavesurfer.on('error', function(e) {
256         //mostra warnings no console
257         console.warn(e);
258     });
259
260     wavesurfer.on('finish', function() {
261         //Pausa a faixa caso acabada
262         links[currentTrack].classList.remove('active');
263         links[currentTrack].classList.remove('pause')
264         links[currentTrack].childNodes[1].innerHTML = play_icon;
265     });
266
267     let prev = document.getElementById("prev") //Elemento de
paginacao voltar
268
269     prev.addEventListener('click', function(e) {
270         if (currentPage > 1) {
271             //Caso nao seja a pagina inicial
272             currentPage -= 1;

```

```

271         npage[0].innerHTML = currentPage+"/"+lastPage;
272         playlist.innerHTML = '';
273
274         //Define o range de exibicao
275         end_pos = beg_pos;
276         if ((beg_pos - naudio) >= 0) {
277             beg_pos -= naudio;
278         } else {
279             beg_pos = 0;
280         }
281
282         let currentAudio = list_files.slice(beg_pos, end_pos)
283         // [Array] Lista de caminhos dos audios de exibicao
284
285         for (let index = 0; index < currentAudio.length; index++) {
286             //Carrega os elementos dos novos audios
287             $('#playlist').append('<div class="list-group-item
288 list-group-item-action audio d-flex" id=audio'+
289 index+' href='+currentAudio[index]+'
290 '> <div class="col-1" id="icone">'+
291 play_icon+
292 '</div> <div class="col-9" id="nome">'+
293 path.parse(currentAudio[index]).name+
294 '</div><div id="duracao" class="col-2
295 duracao'+index+'"></div></div>');
296
297             getDurationTrack(currentAudio[index],
298 ".duracao"+index); //obtem duracao de cada faixa
299         }
300     }
301     links = document.querySelectorAll('.audio'); // [Array] Lista
302     de audio em exibicao
303     player(links); //inicia o
304     player
305     });
306
307     let next = document.getElementById("next") //Elemento de
308     paginacao avancar
309     next.addEventListener('click', function(e) {
310         if (currentPage < lastPage) {
311             //Caso nao seja a pagina Final
312             currentPage += 1;
313             npage[0].innerHTML = currentPage+"/"+lastPage;
314             playlist.innerHTML = '';
315
316             //Define o range de exibicao
317             beg_pos = end_pos;

```

```

311         if (end_pos+naudio < list_files.length) {
312             end_pos += naudio;
313         } else {
314             end_pos = list_files.length
315         }
316
317         let currentAudio = list_files.slice(beg_pos, end_pos)
318         //[Array] Lista de caminhos dos audios de exibicao
319
320         for (let index = 0; index < currentAudio.length; index++) {
321             //[Carrega os elementos dos novos audios
322             $('#playlist').append('<div class="list-group-item
323             list-group-item-action audio d-flex" id=audio'+
324             index+' href='+currentAudio[index]+
325             '> <div class="col-1" id="icone">'+
326             play_icon+
327             '</div> <div class="col-9" id="nome">'+
328             path.parse(currentAudio[index]).name+
329             '</div><div id="duracao" class="col-2
330             duracao'+index+'"></div></div>');
331
332             getDurationTrack(currentAudio[index],
333             ".duracao"+index); //[obtem duracao de cada faixa
334         }
335         links = document.querySelectorAll('.audio');    //[Array]
336         Lista de audio em exibicao
337         player(links);    //[inicia o
338         player
339     }
340
341     });
342 });

```

#### 5.2.4 ./js/fragpage.js

```

1  //Autor: Arthur Serra
2  /* ===== fragpage.js
   =====
3  * Este script produz todos os eventos de controle da pagina
   fragpage.html

```

```

4  * Tendo como principal evento definir e pre-visualizar paramentros de
    fragmentacao
5  * =====
6  */
7
8  const { ipcRenderer } = require('electron')           //Importa funcao de
    comunicacao com o processo principal
9  const $ = jQuery = require('jquery');               //Importa comandos
    JQuery
10 var WaveSurfer = require('wavesurfer.js');           //Importa biblioteca de
    visualizacao de ondas e espectro
11 const colormap = require('colormap');               //Importa biblioteca de
    mapa de cores para o espectrograma
12 const {PythonShell} = require('python-shell');      //Importa biblioteca de
    integracao com script Python
13 var path = require('path');                          //Importa sistema de
    arquivos
14
15 var args = ipcRenderer.sendSync('fromMain', "");    //[Array] faz uma
    chamada ao evento do processo principal para obter o diretorio geral
    e sua lista de arquivos
16 var path_dir = args[0];                             //[String] caminho do
    diretorio geral
17 var list_files = args[1];                           //[Array] caminho de
    todos os arquivos do diretorio princial
18 delete args
19
20 //ICONES DE PLAY E PAUSE
21 var pause_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
    height="30" fill="currentColor" class="bi bi-pause-fill" viewBox="0
    0 15 15"><path d="M5.5 3.5A1.5 1.5 0 0 1 7 5v6a1.5 1.5 0 0 1-3
    0V5a1.5 1.5 0 0 1 1.5-1.5zm5 0A1.5 1.5 0 0 1 12 5v6a1.5 1.5 0 0 1-3
    0V5a1.5 1.5 0 0 1 1.5-1.5z"/></svg>';
22 var play_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
    height="30" fill="currentColor" class="bi bi-play-fill" viewBox="0 0
    15 15"><path d="m11.596 8.697-6.363
    3.692c-.54.313-1.233-.066-1.233-.697V4.308c0-.63.692-1.01
    1.233-.696l6.363 3.692a.802.802 0 0 1 0 1.393z"/></svg>';
23
24 var currentTrack = 0;    //[Inteiro] Indice da faixa a ser carregada
25
26 var wavesurfer;          //[Object] Objeto gerenciamento de audio de
    referencia
27 var preview_wavesurfer;  //[Object] Objeto gerenciamento de audio de
    cortado
28
29 document.getElementById("btnVoltar").addEventListener("click",

```

```

function(){
30     window.location.replace("../html/infopage.html"); //volta para a
        pagina anterior ao clicar no botao
31 });
32
33
34 document.addEventListener('DOMContentLoaded', function() {
35     //Carrega um mapa de cores
36     var cmap = colormap({
37         colormap: 'cool',
38         nshades: 256,
39         format: 'float'
40     });;
41
42     //instacia o objeto de vizualizacao de referencia
43     wavesurfer = new WaveSurfer.create({
44         container: '#waveform',
45         waveColor: '#4ACA4E',
46         progressColor: '#765FC9',
47         plugins: [
48             WaveSurfer.spectrogram.create({
49                 container: '#wave-spectrogram',
50                 fftSamples: 256,
51                 colorMap: cmap
52             })
53         ]
54     });;
55
56     //instacia o objeto de vizualizacao cortado
57     preview_wavesurfer = new WaveSurfer.create({
58         container: '#preview-waveform',
59         waveColor: '#4ACA4E',
60         progressColor: '#765FC9',
61         plugins: [
62             WaveSurfer.spectrogram.create({
63                 container: '#preview-spectrogram',
64                 fftSamples: 256,
65                 colorMap: cmap
66             })
67         ]
68     });;
69 });;
70
71 document.addEventListener('DOMContentLoaded', function() {
72     // Zoom slider
73     let slider = document.getElementById("zoom");
74     let slider_preview = document.getElementById("zoom_preview");

```



```

75
76     slider.value = wavesurfer.params.minPxPerSec;
77     slider.min = wavesurfer.params.minPxPerSec;
78     // Zoom maximo
79     slider.max = 1000;
80
81     slider.addEventListener('input', function() {
82         wavesurfer.zoom(Number(this.value));
83     });
84
85     //Zoom slider
86     slider_preview.value = preview_wavesurfer.params.minPxPerSec;
87     slider_preview.min = preview_wavesurfer.params.minPxPerSec;
88     // Zoom maximo
89     slider_preview.max = 1000;
90
91     slider_preview.addEventListener('input', function() {
92         preview_wavesurfer.zoom(Number(this.value));
93     });
94
95     // Configura valores iniciais do slider
96     wavesurfer.zoom(slider.value);
97     preview_wavesurfer.zoom(slider_preview.value);
98
99     var playpause = document.getElementById("playpause"); //Elemendo
do comando play/pause para audio de referencia
100     playpause.addEventListener('click', function(e){
101         //Ao evento de click
102         if (playpause.classList.contains("active")) {
103             //Caso ja tenha audio carregado inicia gerenciador de
play/pause
104             if (playpause.classList.contains("pause")) {
105                 playpause.classList.remove("pause");
106                 wavesurfer.play();
107                 playpause.innerHTML = pause_icon;
108
109             } else{
110                 playpause.classList.add("pause");
111                 wavesurfer.pause();
112                 playpause.innerHTML = play_icon;
113             }
114         } else {
115             //Caso nao haja audio carregado
116             playpause.classList.add('active');
117             playpause.classList.add("pause");
118             playpause.innerHTML = pause_icon;
119             wavesurfer.load(list_files[currentTrack]);

```

```

120     }
121   });
122
123   var playpause_preview =
document.getElementById("playpause_preview"); //Elemendo do comando
play/pause para audio cortado
124   playpause_preview.addEventListener('click', function(e){
125     //Ao evento de click
126     if (playpause_preview.classList.contains("active")) {
127       //Caso ja tenha audio carregado inicia gerenciador de
play/pause
128       if (playpause_preview.classList.contains("pause")) {
129         playpause_preview.classList.remove("pause");
130         preview_wavesurfer.play();
131         playpause_preview.innerHTML = pause_icon;
132
133       } else{
134         playpause_preview.classList.add("pause");
135         preview_wavesurfer.pause();
136         playpause_preview.innerHTML = play_icon;
137       }
138     } else {
139       //Caso nao haja audio carregado
140       playpause_preview.classList.add('active');
141       playpause_preview.classList.add("pause");
142       playpause_preview.innerHTML = pause_icon;
143       preview_wavesurfer.load(list_files[currentTrack]);
144     }
145   });
146
147   //Carrega o primeiro audio
148   playpause.classList.add('active');
149   playpause.classList.add("pause");
150   wavesurfer.load(list_files[currentTrack]);
151
152   wavesurfer.on('error', function(e) {
153     //mostra warnings no console
154     console.warn(e);
155   });
156
157   wavesurfer.on('finish', function() {
158     //Pausa a faixa caso acabada
159     playpause.classList.remove('active');
160     playpause.classList.remove('pause')
161     playpause.innerHTML = play_icon;
162   });
163

```

```

164     preview_wavesurfer.on('error', function(e) {
165         //mostra warnings no console
166         console.warn(e);
167     });
168
169     preview_wavesurfer.on('finish', function() {
170         //Pausa a faixa caso acabada
171         playpause_preview.classList.remove('active');
172         playpause_preview.classList.remove('pause')
173         playpause_preview.innerHTML = play_icon;
174     });
175
176
177     let skip_prev = document.getElementById("skip_prev") //Elemento
para avançar faixa de referencia
178     skip_prev.addEventListener('click', function(e) {
179         if (currentTrack > 0 ) {
180             //Caso nao seja a primeira faixa
181             currentTrack -= 1;
182             if (playpause.classList.contains('active')) {
183                 playpause.innerHTML = play_icon;
184                 playpause.classList.add("pause");
185             }
186             wavesurfer.load(list_files[currentTrack]);
187
188             //Reseta o status e desabilita o preview ate q seja
carregado novamente
189             playpause_preview.classList.remove('active');
190             playpause_preview.classList.remove("pause");
191             preview_wavesurfer.pause();
192             playpause_preview.innerHTML = play_icon;
193             playpause_preview.disabled = true
194             slider_preview.disabled = true
195         }
196
197     });
198
199     let skip_next = document.getElementById("skip_next")
200     skip_next.addEventListener('click', function(e) {
201         if (currentTrack < list_files.length ) {
202             //Caso nao seja a ultima faixa
203             currentTrack += 1;
204             if (playpause.classList.contains('active')) {
205                 playpause.innerHTML = play_icon;
206                 playpause.classList.add("pause");
207             }
208             wavesurfer.load(list_files[currentTrack]);

```

```

209
210         //Reseta o status e desabilita o preview ate q seja
carregado novamente
211         playpause_preview.classList.remove('active');
212         playpause_preview.classList.remove("pause");
213         preview_wavesurfer.pause();
214         playpause_preview.innerHTML = play_icon;
215         playpause_preview.disabled = true
216         slider_preview.disabled = true
217     }
218 });
219
220     let save_dir = document.getElementById("saveDir");
221     let label_dir = document.getElementById("path");
222     save_dir.addEventListener('click', function(e){
223         let value = ipcRenderer.sendSync('show-open-dialog', "");
224         //[[String] Faz uma requisicao ao script principal
225         if (value != undefined) {
226             //Caso algum diretorio tenha sido corretamente selecionado
227             label_dir.value = value;
228         }
229     });
230
231     let avancar = document.getElementById("avancar");           //Elemento
para avancar para a proxima pagina
232     let preview = document.getElementById("preview");           //Elemento
para carregar uma amostra de fragmentacao
233     preview.addEventListener('click', function(e){
234         //Ao clicar no elemento preview
235         preview.disabled = true
236         //Desabilita o botao de preview
237         avancar.disabled = true
238         //Desabilita o botao de de avancar
239         //Adiciona icone de carregamento
240         preview.innerHTML = '<div class="spinner-grow text-dark"
role="status"><span class="visually-hidden">Loading...</span></div>';
241         let pvw = true;
242         //[[Boolean] Flag de preview
243         let audio_dir = list_files[currentTrack];
244         //[[String] Caminho do diretorio de referencia
245         let temp_dir = path.join(__dirname, '../assets/');
246         //[[String] Caminho do audio preview temporario
247         let frag_min = document.getElementById("minFrag").value;
248         //[[Inteiro] Limite minimo de fragmentacao para cada parte do epectro
249         let frag_max = document.getElementById("maxFrag").value;
250         //[[Inteiro] Limite maximo de fragmentacao para cada parte do epectro
251         let rate = document.getElementById("sampleRate").value;

```

```

244 //[[Inteiro] Taxa de amostragem geral
    let duration = document.getElementById("duration").value;
245 //[[Inteiro] Duracao geral em segundos
    let frame_length = document.getElementById("frameSize").value;
246 //[[Inteiro] Tamanho da janela deslizante da transformada de fourier
    let fft_length = document.getElementById("nFFT").value;
//[[Inteiro] Tamanho das transformadas (A metade deste valor define a
quantidade de frequencias)
247 let stride = document.getElementById("strides").value;
248 //[[Inteiro] Tamanho dos passos de janelamento
    let subtype = 'PCM_16';
//[[String] Formato de condicao do audio resultante
249 let spec = false;
//[[Boolean] Flag para definir se os espectrogramas tambem serao
salvos

250
251 //Dicionario de argumentos para execucao do script Python
252 let options = {
253     scriptPath: path.join(__dirname, '../py/'),

//[[String] caminho do diretorio de armazenamento do script
254     args: [pvw, audio_dir, temp_dir, frag_min, frag_max, rate,
duration, frame_length, fft_length, stride, subtype, spec]
//[[Array] argumento de execucao
255     };
256
257
258     var frag_script_python = new PythonShell('fragment_audio.py',
options); //[[Objeto] Instancia de execucao do script python
259     frag_script_python.on('message', function (message) {
260         //Evento inicia a execucao do script python e aguardo um
retorno atraves do argumento message
261         //Os comando subsequentes sao executados ao fim do processo
python.
262         preview.innerHTML = 'Preview'
263         if (message == "True"){
264             //Caso o fim do processo resulte em 'True', nao houve
problema na execucao
265             playpause_preview.classList.add('active');
266             playpause_preview.classList.add("pause");
267             preview_wavesurfer.load(path.join(__dirname,
'../assets/frag_temp.wav')); //Carrega o arquivo temporario
resultante
268             playpause_preview.removeAttribute("disabled");
//Habilita o botao de play/pause da secao preview
269             slider_preview.removeAttribute("disabled");
//Habilita o botao sliders do secao preview

```

```

270         }else{
271             //Caso o script python nao tem sido executado
corretamente
272             $("#Error").modal('show');
                //Chamada do elemento modal de erro via JQuery
273         }
274         preview.disabled = false;    //Habilita o botao de preview
275         avancar.disabled = false;    //Habilita o botao de avanco
276     });
277
278 });
279
280 avancar.addEventListener('click', function(e) {
281
282     let pvw = false;
//[Boolean] Flag de preview
283     let audio_dir = path_dir;
//[String] Caminho do diretorio de referencia
284     let save_dir = document.getElementById("path").value;
//[String] Caminho do audio preview temporario
285     let frag_min = document.getElementById("minFrag").value;
//[Inteiro] Limite minimo de fragmentacao para cada parte do epectro
286     let frag_max = document.getElementById("maxFrag").value;
//[Inteiro] Limite maximo de fragmentacao para cada parte do epectro
287     let rate = document.getElementById("sampleRate").value;
//[Inteiro] Taxa de amostragem geral
288     let duration = document.getElementById("duration").value;
//[Inteiro] Duracao geral em segundos
289     let frame_length = document.getElementById("frameSize").value;
//[Inteiro] Tamanho da janela deslizante da transformada de fourier
290     let fft_length = document.getElementById("nFFT").value;
//[Inteiro] Tamanho das transformadas (A metade deste valor define
a quantidade de frequencias)
291     let stride = document.getElementById("strides").value;
//[Inteiro] Tamanho dos passos de janelamento
292     let subtype = 'PCM_16';
//[String] Formato de condicaoacao do audio resultante
293     let spec = document.getElementById("SwitchCheck").checked;
//[String] Formato de condicaoacao do audio resultante
294
295     if (save_dir != '') {
296         //Caso o diretorio de destino seja um diretorio valido
297         args = [pvw,
298                 audio_dir,
299                 save_dir,
300                 frag_min,
301                 frag_max,

```

```

302         rate,
303         duration,
304         frame_length,
305         fft_length,
306         stride,
307         subtype,
308         spec]
309
310         ipcRenderer.send('argsToMain', args);           //Envia
o array de argumentos para o script principal
311         window.location.replace("../html/loadpage.html");
//Avanca para a proxima pagina (loadpage.html)
312     }
313 });
314 });

```

### 5.2.5 ./js/loadpage.js

```

1 //Autor: Arthur Serra
2 /* ===== Loadpage.js
=====
3 * Este script nada mais e que uma barra de progresso do processo de
fragmentacao
4 * =====
5 */
6 const{ ipcRenderer } = require('electron');           //Importa funcao de
comunicacao com o processo principal
7 var $ = jQuery = require('jquery');                 //Importa comandos
jQuery
8 var path = require('path');                         //Importa sistema de
arquivos
9 const glob = require('glob').Glob;                 //Importa sistema de
arquivos
10 const fs = require('fs');                          //Importa sistema de
arquivos
11 const {PythonShell} = require('python-shell');      //Importa biblioteca de
integracao com script Python
12
13 var args = ipcRenderer.sendSync('fromMain', ""); // [Array] faz uma
chamada ao evento do processo principal para obter o diretorio geral
e sua lista de arquivos
14 var path_dir = args[0];                             // [String] caminho do
diretorio geral
15 var list_files = args[1];                          // [Array] caminho de
todos os arquivos do diretorio princial
16 delete args
17

```

```

18 var frag_args = ipcRenderer.sendSync('argsfromMain', "");    //[Array]
    faz uma chamada ao evento do processo principal para obter os
    parametros de fragmentacao
19 var dest_files = frag_args[2]+'/Audios'                      //[String]
    Complemento do diretorio de armazenamento dos audio fragmentados
20
21 async function run(){
22     /* Esta funcao recebe executa o processo de fragmentacao python de
    forma assincrona
23     * <ATRIBUTOS>
24     *      [null]
25     * <RETORNO>
26     *      [null]
27     */
28     const { success, err = '', results } = await new Promise(
29         (resolve, reject) =>{
30             //gera promessa de resolucao
31
32             //Dicionario de argumentos para execucao do script Python
33             let options = {
34                 scriptPath: path.join(__dirname, '../py/'),
35                 //[[String] caminho do diretorio de armazenamento do script
36                 args: frag_args
37                 //[[Array] argumento de execucao
38             };
39
40             PythonShell.run('fragment_audio.py', options, function
41 (err, results) {
42                 if (err){
43                     //Caso um erro seja gerado na chamada
44                     reject({ success: false, err });
45                 }
46                 if (results[0] == "False") {
47                     //Caso um erro tenha ocorrido na execucao do script
48                     $("#Error").modal('show');    //Ativa modal de
49 erro via JQuery
50                 }
51                 resolve({ success: true, results });
52             });
53         });
54
55 document.addEventListener('DOMContentLoaded', function() {
56     //Quando o documento for carregado
57     run();
58 }

```



```

57 //CONTADORES AUXILIARES
58 var frag_audio = 0; //[[Inteiro]
Quantidade de audios processados
59 var perc = 0; //[[Inteiro]
Percentual de audios processados
60 let aux_perc = perc;
61
62 let att_perc = document.getElementById("perc"); //Elemento com
o valor da porcentagem de audio processado
63
64 let delay = 1000; // 1 segundo //[[Inteiro]
Define o delay em milissegundos do processo de atualizacao de
progresso
65
66 let avancar = document.getElementById("avancar"); //Elemento do
botao de avanco
67 let voltar = document.getElementById("voltar"); //Elemento do
botao de voltar
68
69 var iID = setInterval(function(){
70 //A cada intervalo de 1 segundo
esta funcao e executada
71 */
72 if (frag_audio >=
list_files.length) {
73 //Caso o botao a quantidade de
audio processados seja igual a quantidade de arquivos enviados
74 avancar.disabled = false;
//Habilita o botao de avanco
75 clearInterval(iID);
//Encerra o loop de checagem
76 }else{
77 //Valida a existencia do
arquivo de destino
78 if (fs.existsSync(dest_files)) {
79 let files =
fs.readdirSync(dest_files); //[[Array] caminhos dos
arquivos processados
80 frag_audio = files.length;
//[[Inteiro] quantidade de arquivos
processados
81 perc =
parseInt(frag_audio/list_files.length*100) //[[Inteiro] percentual
de arquivos processados
82
83 if (aux_perc < perc){
84 //Condicao para evitar

```

```

escrita de valores repetido no formulario HTML
85                                     att_perc.innerHTML =
perc+"%"                               //Atualiza o
valor do percentual no formulario
86
$($('.progress-bar').css('width', perc+'%').attr('aria-valuenow',
perc); //Atualiza a barra de progresso via JQuery
87                                     aux_perc = perc;
88                                     }
89                                 }
90                             }
91                     },delay);
92
93     avancar.addEventListener('click', function(){
94         //Evento ativado ao clicar no botao avancar
95         glob(dest_files + '/*.wav', {}, (err, files)=>{
96             ipcRenderer.send('destToMain', files);
97             //Envia ao script principal a lista de arquivos processados
98             window.location.replace("../html/playlistpage.html");
99             //Redireciona a pagina para playlistpage.html
100         })
101     });
102
103     voltar.addEventListener("click", function(){
104         //Evento ativado ao clicar no botao voltar
105         window.location.replace("../html/fragpage.html");
106         //Redireciona a pagina para fragpage.html
107     });

```

## 5.2.6 ./js/playlistpage.js

```

1 //Autor: Arthur Serra
2 /* ===== playlistpage.js
   =====
3 * Este script apresenta ao usuario duas playlists, uma com os audio de
4 * referencia e outra com os audios fragmentados. Lado a lado de tal
   forma
5 * que o usuario possa avaliar o processo gerado.
6 *
   =====
7 */
8 const{ ipcRenderer } = require('electron'); //Importa funcao de
   comunicacao com o processo principal
9 const mm = require('music-metadata'); //Importa biblioteca
   obtencao de metadados de audios

```

```

10 var fs = require('fs'); //Importa sistema de
    arquivos
11 var $ = jQuery = require('jquery'); //Importa comandos JQuery
12 var WaveSurfer = require('wavesurfer.js'); //Importa biblioteca de
    visualizacao de ondas e espectro
13 var path = require('path'); //Importa sistema de
    arquivos
14
15 var args = ipcRenderer.sendSync('fromMain', ""); // [Array] faz uma
    chamada ao evento do processo principal para obter o diretorio geral
    e sua lista de arquivos
16 var path_dir = args[0]; // [String] caminho do
    diretorio geral
17 var list_files = args[1]; // [Array] caminho de
    todos os arquivos do diretorio princial
18 delete args
19
20 var list_generated_files = ipcRenderer.sendSync('destTofromMain', "");
    // [Array] Faz uma consulta ao processo principal para obter os
    caminho gerados pelo processo de fragmentacao
21
22 //ICONES DE PLAY E PAUSE
23 var pause_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
    height="30" fill="currentColor" class="bi bi-pause-fill" viewBox="0
    0 15 15"><path d="M5.5 3.5A1.5 1.5 0 0 1 7 5v6a1.5 1.5 0 0 1-3
    0V5a1.5 1.5 0 0 1 1.5-1.5zm5 0A1.5 1.5 0 0 1 12 5v6a1.5 1.5 0 0 1-3
    0V5a1.5 1.5 0 0 1 1.5-1.5z"/></svg>';
24 var play_icon = '<svg xmlns="http://www.w3.org/2000/svg" width="30"
    height="30" fill="currentColor" class="bi bi-play-fill" viewBox="0 0
    15 15"><path d="m11.596 8.697-6.363
    3.692c-.54.313-1.233-.066-1.233-.697V4.308c0-.63.692-1.01
    1.233-.696l6.363 3.692a.802.802 0 0 1 0 1.393z"/></svg>';
25
26 var currentTrack = 0;
27 var currentTrack_corte = 0;
28
29 var wavesurfer; // [Object] Instancia gerenciamento dos audio de
    referencia
30 var corte_wavesurfer; // [Object] Instancia gerenciamento dos audios
    fragmentados
31
32 async function getDurationTrack(track, id){
33     /* Esta funcao recebe um numero inteiro que representa a duracao
34     * de tempo em segundos de uma determinada faixa e transforma
35     * para o formato mm:ss
36     * <ATRIBUTOS>
37     * track: [String] caminho do arquivo

```

```

38     *      id:      [String] identificador no formulario infopage.html
39     * <RETORNO>
40     *      [null]
41     */
42     let metadata = await mm.parseFile(track);           //obtem os
metadados do uma arquivo de audio
43     let secs = metadata.format.duration;               //extrai a
duracao total em segundos dos metadados
44
45     var divisor_for_minutes = secs % (60 * 60);
46     var minutes = Math.floor(divisor_for_minutes / 60); //obtem os
minutos inteiros
47
48     var divisor_for_seconds = divisor_for_minutes % 60;
49     var seconds = Math.ceil(divisor_for_seconds);      //obtem os
segundos inteiro fora os minutos
50
51     $(id).text(minutes+": "+seconds)                  //Atualiza o
texto no fomulario html via JQuery
52 }
53
54 function player(audios, ws, ct){
55     /* Esta funcao produz um player de audio dado um array de caminhos
56     * <ATRIBUTOS>
57     *      audios: [Array] caminhos dos arquivo
58     *      ws:      [Object] Instancia da biblioteca de exibicao
59     *      ct:      [Inteiro] indice do audio
60     * <RETORNO>
61     *      [null]
62     */
63     ct = 0
64     // CARREGA UMA FAIXA DADOS UM INDICE ATUAL
65     let setCurrentSong = function(index) {
66         audios[ct].classList.remove('active');
67         audios[ct].childNodes[1].innerHTML = play_icon;
68         ct = index;
69         audios[ct].classList.add('active');
70         audios[ct].childNodes[1].innerHTML = pause_icon;
71         ws.load(audios[ct].attributes.href.nodeValue);
72     };
73
74     // CARREGA UM AUDIO DADO EVENTO DE CLICK
75     Array.prototype.forEach.call(audios, function(link, index) {
76         link.addEventListener('click', function(e) {
77             e.preventDefault();
78             if (audios[index].classList.contains('active')) {
79                 //CASO O AUDIO DE ESTEJA CARREGADO e FEITO APENAS UM

```

## GERENCIAMENTO DE PLAY E PAUSE

```
80         if (audios[index].classList.contains('pause')) {
81             ws.play();
82             audios[index].childNodes[1].innerHTML = pause_icon;
83             audios[index].classList.remove('pause');
84         } else {
85             ws.pause();
86             audios[index].childNodes[1].innerHTML = play_icon;
87             audios[index].classList.add('pause');
88         }
89
90     } else {
91         //CASO O AUDIO AINDA NAO ESTEJA CARREGADO
92         setCurrentSong(index);
93     }
94 });
95 });
96 }
97
98 document.addEventListener('DOMContentLoaded', function() {
99     //instacia o objeto de visualizacao do audio de referencia
100     wavesurfer = WaveSurfer.create({
101         container: '#waveform',
102         waveColor: '#4ACA4E',
103         progressColor: '#765FC9',
104         height: 100
105     });
106     //instacia o objeto de visualizacao do audio de fragmentado
107     corte_wavesurfer = WaveSurfer.create({
108         container: '#waveform-corte',
109         waveColor: '#4ACA4E',
110         progressColor: '#765FC9',
111         height: 100
112     });
113 });
114
115 document.addEventListener('DOMContentLoaded', function(){
116     let naudio = 9;
117     //[[Inteiro] Define a quantidade de audio aparecerar por vez em uma
118     //coluna da pagina
119
120     let beg_pos = 0;
121     //[[Inteiro] Indice do primeiro audio na pagina em relacao a lista de
122     //todos oa audios
123
124     let end_pos;
125     //[[Inteiro] Indice do ultimo audio na pagina em relacao a lista de
126     //todos oa audios
```

```

120
121     let playlist = document.getElementById('playlist');
122     // [Objeto] Carrega o elemento de visualizacao da playlist de
referencia
123
124     let playlist_corte = document.getElementById('playlist-corte');
125     // [Objeto] Carrega o elemento de visualizacao da playlist fragmentada
126
127     let npage = document.getElementsByClassName('npage');
128     // [Objeto] Carrega o elemento de posicao na paginacao
129
130     let currentPage = 1;
131     // [Inteiro] pagina atual
132     let lastPage = Math.ceil(list_files.length/naudio)
133
134     npage[0].innerHTML = currentPage+"/"+lastPage;
135     // Define o valor da pagina atual no formulario
136
137     if (list_files.length > naudio) {
138         // Valida de o total de arquivos e inferior ao limite de
exibicao por vez
139         end_pos = naudio;
140         for (let index = 0; index < naudio; index++) {
141             // Produz os containers de exibicao da playlist de referencia
142             $('#playlist').append('<div class="list-group-item
list-group-item-action audio d-flex" id=audio'+
143             index+' href='+list_files[index]+'
144             '> <div class="col-1" id="icone">'+
145             play_icon+
146             '</div> <div class="col-9" id="nome">'+
147             path.parse(list_files[index]).name+
148             '</div><div id="duracao" class="col-2
duracao'+index+' "></div></div>');
149
150             getDurationTrack(list_files[index], ".duracao"+index);
151
152             // Produz os containers de exibicao da playlist fragmentada
153             $('#playlist-corte').append('<div class="list-group-item
list-group-item-action audio_corte d-flex" id=audio'+
154             index+' href='+list_generated_files[index]+'
155             '> <div class="col-1" id="icone">'+
156             play_icon+
157             '</div> <div class="col-9" id="nome">'+
158             path.parse(list_generated_files[index]).name+
159             '</div><div id="duracao" class="col-2
duracao'+index+' "></div></div>');
160
161             getDurationTrack(list_generated_files[index],

```

```

155     ".duracao"+index);
156     }
157     } else {
158         end_pos = list_files.length;
159         for (let index = 0; index < list_files.length; index++) {
160             //Produz os containers de exibicao da playlist de referencia
161             $('#playlist').append('<div class="destaque list-group-item
list-group-item-action audio d-flex " id=audio'+
162             index+' href='+list_files[index]+'
163             '> <div class="col-1" id="icone">'+
164             play_icon+
165             ' </div> <div class="col-9" id="nome">'+
166             +path.parse(list_files[index]).name+
167             '</div><div id="duracao" class="col-2
duracao'+index+' "></div></div>');
168
169             getDurationTrack(list_files[index], ".duracao"+index);
170
171             //Produz os containers de exibicao da playlist fragmentada
172             $('#playlist-corte').append('<div class="list-group-item
list-group-item-action audio_corte d-flex" id=audio'+
173             index+' href='+list_generated_files[index]+'
174             '> <div class="col-1" id="icone">'+
175             play_icon+
176             ' </div> <div class="col-9" id="nome">'+
177             +path.parse(list_generated_files[index]).name+
178             '</div><div id="duracao" class="col-2
duracao'+index+' "></div></div>');
179
180             getDurationTrack(list_generated_files[index],
".duracao"+index);
181         }
182     }
183
184     let links = document.querySelectorAll('.audio');
185     // [Array] obtém os cominhos de cada container de audio de referencia
186     // em em exibicao
187     let links_corte = document.querySelectorAll('.audio_corte');
188     // [Array] obtém os cominhos de cada container de audio fragmentado
189     // em em exibicao
190
191     player(links, wavesurfer, currentTrack);
192     // inicia o player de referencia
193     player(links_corte, corte_wavesurfer, currentTrack_corte);
194     // inicia o player fragmentado
195
196     wavesurfer.on('ready', function(e) {

```

```

191         //play caso carregado
192         wavesurfer.play();
193     });
194
195     wavesurfer.on('error', function(e) {
196         //mostra warnings no console
197         console.warn(e);
198     });
199
200     wavesurfer.on('finish', function() {
201         //Pausa a faixa caso acabada
202         links[currentTrack].classList.remove('active');
203         links[currentTrack].classList.remove('pause')
204         links[currentTrack].childNodes[1].innerHTML = play_icon;
205     });
206
207     corte_wavesurfer.on('ready', function(e) {
208         //play caso carregado
209         corte_wavesurfer.play();
210     });
211
212     corte_wavesurfer.on('error', function(e) {
213         //mostra warnings no console
214         console.warn(e);
215     });
216
217     // Go to the next track on finish
218     corte_wavesurfer.on('finish', function() {
219         //Pausa a faixa caso acabada
220         links_corte[currentTrack_corte].classList.remove('active');
221         links_corte[currentTrack_corte].classList.remove('pause')
222         links_corte[currentTrack_corte].childNodes[1].innerHTML =
play_icon;
223     });
224
225     let prev = document.getElementById("prev")           //Elemento de
paginacao voltar
226     prev.addEventListener('click', function(e) {
227         if (currentPage > 1) {
228             //Caso nao seja a pagina inicial
229             currentPage -= 1;
230             npage[0].innerHTML = currentPage+"/"+lastPage;
231             playlist.innerHTML = '';
232             playlist_corte.innerHTML = '';
233
234             //Define o range de exibicao
235             end_pos = beg_pos;

```



```

236         if ((beg_pos - naudio) >= 0) {
237             beg_pos -= naudio;
238         } else {
239             beg_pos = 0;
240         }
241
242         let currentAudio = list_files.slice(beg_pos, end_pos)
                //[[Array] Lista de caminhos dos audios de referencia
em exibicao
243         let currentAudio_corte =
list_generated_files.slice(beg_pos, end_pos)           //[[Array] Lista
de caminhos dos audios fragmentados em exibicao
244
245         for (let index = 0; index < currentAudio.length; index++) {
246             //Carrega os elementos dos novos audios de referencia
247             $('#playlist').append('<div class="list-group-item
list-group-item-action audio d-flex" id=audio'+
248                 index+' href='+currentAudio[index]+
249                 '> <div class="col-1" id="icone">'+
250                 play_icon+
251                 '</div> <div class="col-9" id="nome">'+
252                 path.parse(currentAudio[index]).name+
253                 '</div><div id="duracao" class="col-2
duracao'+index+'"></div></div>');
254
255                 getDurationTrack(currentAudio[index],
".duracao"+index);           //obtem duracao de cada faixa
256
257                 //Carrega os elementos dos novos audios fragmentados
258                 $('#playlist-corte').append('<div
class="list-group-item list-group-item-action audio_corte d-flex"
id=audio'+
259                     index+' href='+currentAudio_corte[index]+
260                     '> <div class="col-1" id="icone">'+
261                     play_icon+
262                     '</div> <div class="col-9" id="nome">'+
263                     path.parse(currentAudio_corte[index]).name+
264                     '</div><div id="duracao" class="col-2
duracao'+index+'"></div></div>');
265
266                     getDurationTrack(currentAudio_corte[index],
".duracao"+index);           //obtem duracao de cada faixa
267                 }
268             }
269             links = document.querySelectorAll('.audio');
                //[[Array] Lista de audio de referencia em exibicao
270             links_corte = document.querySelectorAll('.audio_corte')

```

```

271 // [Array] Lista de audio fragmentados em exibicao
272     player(links, wavesurfer, currentTrack);
273 // inicia o player de referencia
274     player(links_corte, corte_wavesurfer, currentTrack_corte);
275 // inicia o player fragmentado
276     });
277
278     let next = document.getElementById("next") // Elemento de
279     paginacao avancar
280     next.addEventListener('click', function(e) {
281         if (currentPage < lastPage) {
282             // Caso nao seja a pagina Final
283             currentPage += 1;
284             npage[0].innerHTML = currentPage+"/"+lastPage;
285             playlist.innerHTML = '';
286             playlist_corte.innerHTML = '';
287
288             // Define o range de exibicao
289             beg_pos = end_pos;
290             if (end_pos+naudio < list_files.length) {
291                 end_pos += naudio;
292             } else {
293                 end_pos = list_files.length
294             }
295
296             let currentAudio = list_files.slice(beg_pos, end_pos)
297             // [Array] Lista de caminhos dos audios de referencia
298             em exibicao
299
300             let currentAudio_corte =
301             list_generated_files.slice(beg_pos, end_pos) // [Array] Lista
302             de caminhos dos audios fragmentados em exibicao
303
304             for (let index = 0; index < currentAudio.length; index++) {
305                 // Carrega os elementos dos novos audios de referencia
306                 $('#playlist').append('<div class="list-group-item
307                 list-group-item-action audio d-flex" id=audio'+
308                 index+' href='+currentAudio[index]+'
309                 '> <div class="col-1" id="icone">'+
310                 play_icon+
311                 '</div> <div class="col-9" id="nome">'+
312                 path.parse(currentAudio[index]).name+
313                 '</div><div id="duracao" class="col-2
314                 duracao'+index+' "></div></div>');
315
316                 getDurationTrack(currentAudio[index],
317                 ".duracao"+index); // obtem duracao de cada faixa

```

```

307
308         //Carrega os elementos dos novos audios fragmentados
309         $('#playlist-corte').append('<div
class="list-group-item list-group-item-action audio_corte d-flex"
id=audio'+
310             index+' href='+currentAudio_corte[index]+'
311             '> <div class="col-1" id="icone">'+
312             play_icon+
313             '</div> <div class="col-9" id="nome">'+
314             path.parse(currentAudio_corte[index]).name+
315             '</div><div id="duracao" class="col-2
duracao'+index+'"></div></div>');
316
317             getDurationTrack(currentAudio_corte[index],
".duracao"+index);           //obtem duracao de cada faixa
318         }
319     }
320     links = document.querySelectorAll('.audio');
//[Array] Lista de audio de referencia em exibicao
321     links_corte = document.querySelectorAll('.audio_corte')
//[Array] Lista de audio fragmentados em exibicao
322
323     player(links, wavesurfer, currentTrack);
//inicia o player de referencia
324     player(links_corte, corte_wavesurfer, currentTrack_corte);
//inicia o player fragmentado
325     });
326
327     let avancar = document.getElementById("avancar"); //Elemento do
botao avancar
328     avancar.addEventListener('click', function(e) {
329         //Ativo ao clicar
330         window.location.replace("../html/home.html"); //retorna a
pagina inicial da aplicacao
331     });
332 });

```