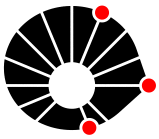

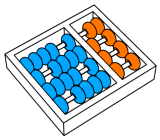

Editor Web de Podcasts

Gabriel Bueno de Oliveira *Gustavo Amgarten de Lêdo*
Matheus Yokoyama Figueiredo *Leandro Aparecido Villas*

Relatório Técnico - IC-PFG-18-05
Projeto Final de Graduação
2018 - Junho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Editor Web de Podcasts

Gabriel Bueno de Oliveira Gustavo Amgarten de Lêdo
Matheus Yokoyama Figueiredo Leandro Aparecido Villas *

Resumo

Com o intuito de aproximar alunos e professores das plataformas de tecnologia, foi realizada uma parceria com a empresa de educação Clickideia para um protótipo inicial de um ambiente de criação de podcasts web, facilitando a distribuição de conhecimento e a interação entre as partes através da internet.

Sumário

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

1 Introdução

O Instituto de Computação (IC) da Universidade Estadual de Campinas (Unicamp) vem realizando diversas parcerias para o desenvolvimento de projetos. Com esse intuito, a empresa Clickideia, atuante na área de educação no Brasil, propôs o desenvolvimento de uma plataforma web de edição e gravação de podcasts, auxiliando alunos e professores a criarem conteúdos educacionais.

Foi realizado, nas reuniões iniciais, a extração dos requisitos, importante para que fosse possível entender melhor as dificuldades e exigências do projeto, e permitiu com que houvesse um planejamento para o restante do desenvolvimento.

2 Objetivos

O objetivo desse projeto foi fornecer uma arquitetura, lógica de negócio e interface inicial, para uma ferramente utilizável por alunos e professores, a fim de criarem e gerenciarem podcasts em uma plataforma web. As edições incluem sobreposição de áudio, ajuste de volume e recorte de intervalos. Esse sistema inicial pode ser adaptado para uso junto de diversas tecnologias, de modo que futuras alterações sejam de fácil acesso e sem muitas dificuldades.

A ferramenta visa um meio para que alunos e professores possam criar um podcasts com conteúdos educacionais. O usuário deve ser capaz de, através da ferramenta, criar gravações, cortar trechos, sobrepor e concatenar múltiplos áudios.

3 Justificativa

A principal justificativa é a tentativa de aproximar professor e alunos ao meio educacional digital para que exista mais conteúdos com uma abordagem mais moderna.

Essa ferramenta visa auxiliar esse processo através dos podcasts.

4 Metodologia

Para o desenvolvimento do projeto, foram utilizadas práticas e conceitos de desenvolvimento ágil, bastante presente no mercado de desenvolvimento de software.

Foram usadas as ferramentas presentes no ambiente do GitLab que auxiliavam a aplicação da metodologia, como por exemplo adição, classificação e resolução de estórias. Reuniões periódicas entre os membros também fizeram-se presente para acompanhamento do planejamento. Dessa forma, fez-se uso de:

a) Controle de versão (Git)

b) Planejamento e releases

c) Integrações curtas

5 Arquitetura

O sistema foi desenvolvido em dois repositórios separados: um para o **Frontend** desenvolvido utilizando o framework **React**, e outro para o **Backend** baseado em uma arquitetura REST e desenvolvido fazendo uso da tecnologia **Node.js**.

5.1 Cliente

Como citado anteriormente, o cliente web foi construído utilizando o framework React, e portanto a linguagem javascript (ES6).

O design da interface foi planejada para ser uma página dividida em 3 áreas principais:

- **Header:** Topo da página contendo informações de projeto, player geral e botões de gravação/upload de arquivos.
- **Linha do tempo:** Região central que contém todos os elementos de áudio em suas respectivas posições no projeto. O tamanho de cada fatia é proporcional ao tempo audível de cada uma.
- **Editor:** Parte inferior, contém as informações de cada faixa, player individual e opções de edição, como corte, deslocamento e ajuste de volume.

Esse panorama geral é mostrado na Figura 1.

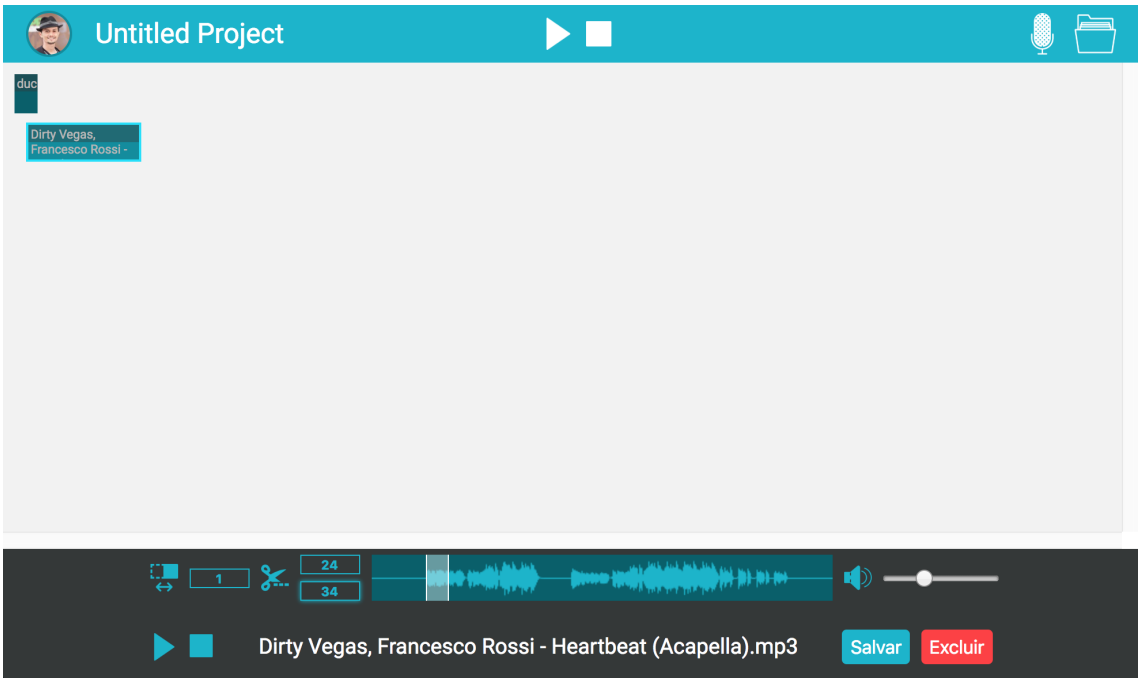

Figura 1: Header, linha do tempo e editor

Para a gravação de áudio e upload de arquivos, são exibidas popups com as funcionalidades particulares de cada caso, como mostram as Figuras 2 e 3.

Na popup de gravação temos o botão para iniciar o processo, assim como o campo para nomear o arquivo e salvá-lo.

Já na área de upload de arquivos, simplesmente é disponibilizado um botão para a seleção do mesmo.

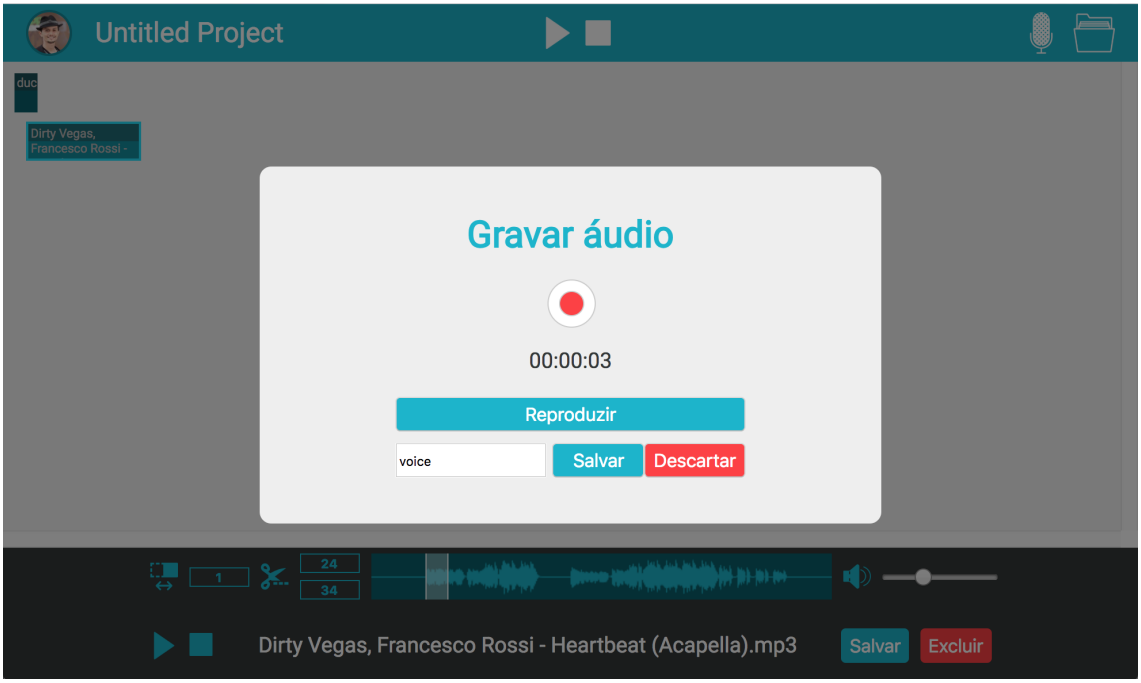

Figura 2: Popup de gravação

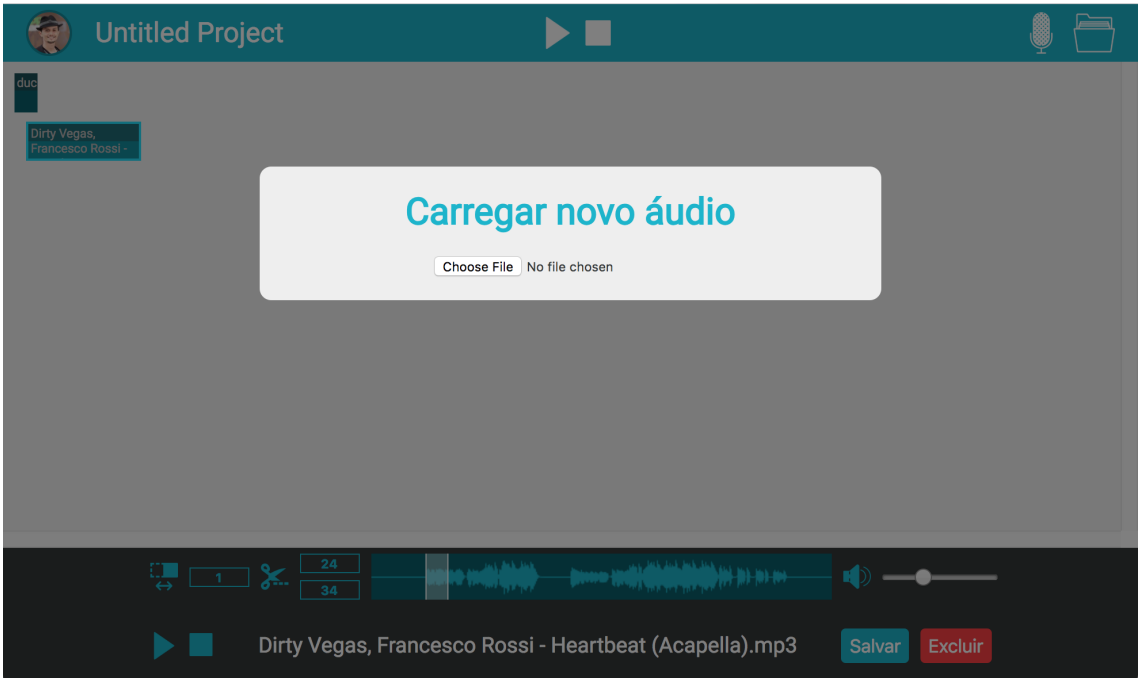

Figura 3: Linha do tempo e editor

5.2 Servidor

Foi utilizada a linguagem de programação JavaScript em cima do *runtime* NodeJS. A arquitetura da aplicação seguiu a estrutura API RESTful, podendo atender assim a aplicação em frontend feita em React e ao mesmo tempo flexibilizar a utilização da mesma em outras infraestruturas, já que não era conhecido a priori como ia se dar a utilização desta ferramenta.

Em conjunto do framework ExpressJS em uma estrutura Controller-Service-Repository, foi utilizada a biblioteca Sequelize para atender as conexões de banco de dados e ao mesmo tempo definir os modelos de entidades utilizados e também duas bibliotecas ficaram responsáveis pelo gerenciamento dos arquivos de audio, *multerS3*, uma branch da conhecida biblioteca *multer* responsável por fazer o *middleware* de requisições com arquivos e enviá-los ao servidor AWS S3, e *aws-sdk* que utilizamos para fazder o download e upload dos arquivos manipulados.

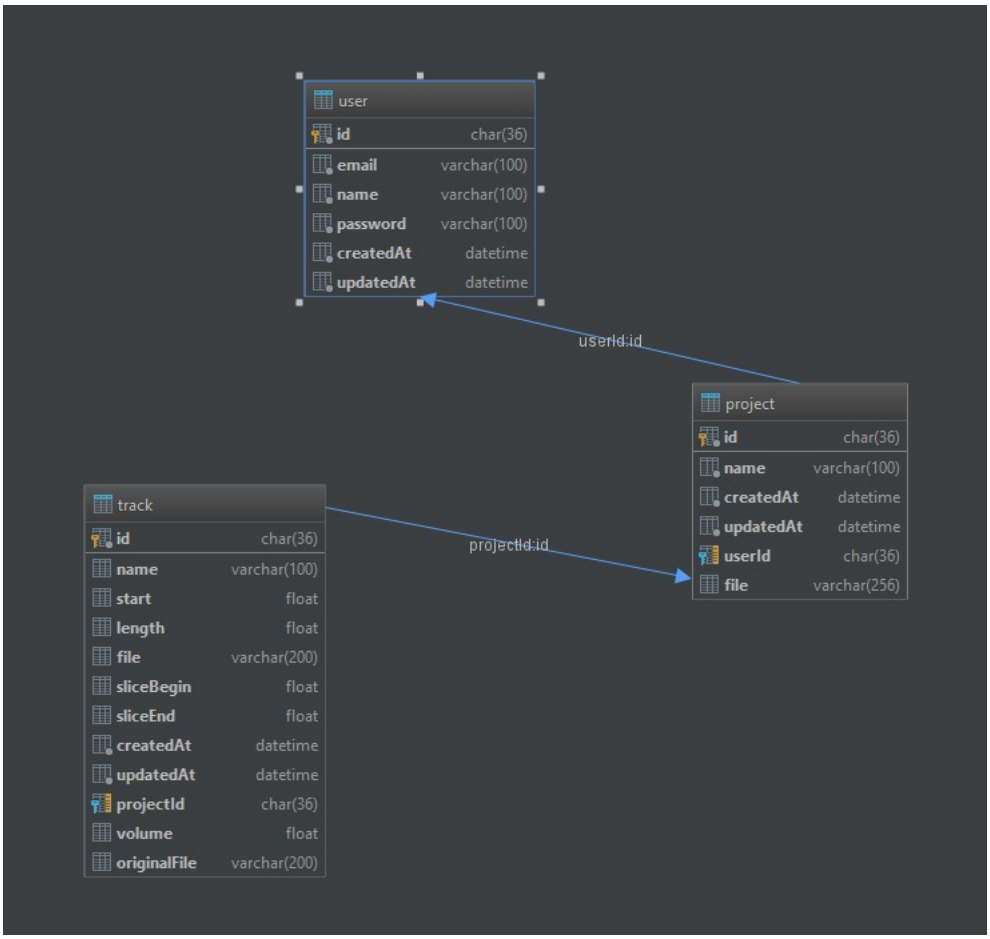

Figura 4: Modelagem da base de dados

Toda nossa infraestrutura estava modularizada nos serviços da Amazon AWS, já que este é um dos principais provedores do serviço e o que a maioria das empresas utiliza atualmente. Contávamos com um servidor, rodando em uma EC2 mini utilizando o sistema operacional Ubuntu 16.04, que se comunicava com o serviço de banco de dados RDS(Relational Database Service) em MySQL, sendo mapeado pelos models utilizando a biblioteca Sequelize. Foi também utilizado o serviço de armazenamento Amazon S3 para o armazenamento dos áudios originais, intermediários e para o arquivo de áudio final gerado pelo serviço, com todas as alterações realizadas pelo usuário.

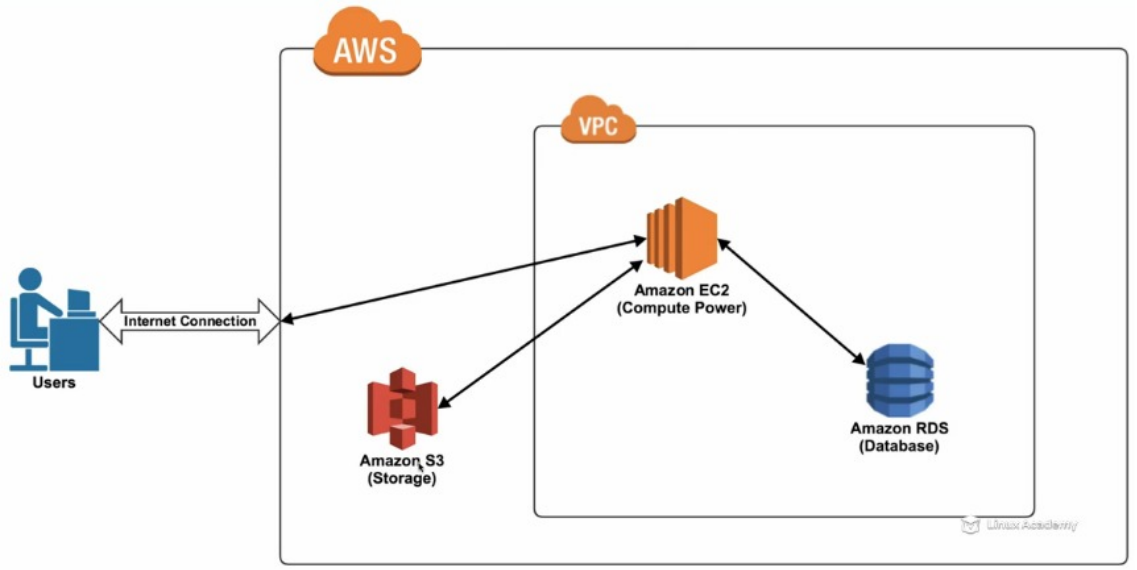

Figura 5: Modelo de organização do servidor

A biblioteca utilizada para manipulação de audio foi a *sox-audio*, que implementa um interface simplificada em JavaScript para fazer chamadas a biblioteca SOX - do próprio site: *"SoX is a cross-platform (Windows, Linux, MacOS X, etc.) command line utility that can convert various formats of computer audio files in to other formats. It can also apply various effects to these sound files, and, as an added bonus, SoX can play and record audio files on most platforms."*.

A biblioteca em JavaScript faz chamadas CLI para o SOX afim de manipular os audios. A biblioteca possibilitou a implementação de diversos efeitos de som, dentre eles volume, corte, *delay* e sobreposição, e ainda permite a possibilidade de expandir o uso da ferramenta para outros efeitos, visto que a biblioteca JavaScript viabiliza o uso de todos os comandos implementados na SOX.

6 Funcionalidades

- Criar novo projeto.
- Atualizar projeto.

- Deletar projeto.
- Gerar o arquivo intermediário/final com todas os áudios editados.
- Adicionar arquivo de áudio.
- Ajustar volume.
- Cortar um trecho do áudio.
- Reproduzir o áudio com as edições aplicadas.

7 Guia de Desenvolvimento

7.1 Cliente

Para executar o cliente é necessário a instalação do Node para o uso do NPM. Assim que for instalado, deve-se fazer o download das dependências do projeto. Para isso, acesse o diretório do projeto (onde está localizado o arquivo **package.json**) e execute o comando:

```
$ npm install
```

Após todas as dependências terem sido devidamente instaladas, execute a aplicação com o comando:

```
$ npm start
```

Por padrão, o cliente é acessado através da porta 3000.

7.2 Servidor

Recomenda-se a utilização do sistema operacional Ubuntu 16.04, que foi a versão utilizada para o desenvolvimento, foi também utilizado os serviços da Amazon AWS RDS e S3, sendo assim é necessário a criação das instâncias e a substituição das API keys no projeto. Uma outra opção é a substituição por outros serviçõe similares.

1. Executar o seguinte comando no terminal:

   ```
   $ sudo apt install sox
   $ sudo apt install libsox-fmt-mp3
   ```

2. Instalar o Node

   ```
   $ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
   $ sudo apt-get install -y nodejs
   ```

3. Ir ao diretório do projeto e instalar as dependências

   ```
   $ npm install
   ```

Os repositórios podem ser encontrados em:

- Backend: https://gitlab.ic.unicamp.br/ra139455/pfg-podcast
- Frontend: https://gitlab.ic.unicamp.br/ra139490/pfg-podcast-webapp

Referências

[1] Nodejs. https://nodejs.org/en/download/package-manager/
Acessado em 28/06/2018.

[2] SOX.http://sox.sourceforge.net/
Acessado em 28/06/2018.

[3] API.https://matheusfigueiredo1.docs.apiary.io/
Acessado em 28/06/2018.

[4] Sequelizejs: http://docs.sequelizejs.com/
Acessado em 28/06/2018.

[5] React: http://docs.sequelizejs.com/
Acessado em 28/06/2018.