INFRAESTRUTURA PARA COLETA DE VOZ VIA INTERNET

Guilherme Inácio Grandesi[1], *Ivandro Sanches*
Engenharia Elétrica, Centro Universitário FEI

ggrandesi@gmail.com / isanches@fei.edu.br

Resumo: Este projeto de iniciação visa a construção de site de coleta de voz. A intenção é facilitar a obtenção de banco de dados de voz a um custo relativamente baixo quando comparado a outras formas de construção de banco de dados de voz, as quais exigem o deslocamento de pessoas para a gravação. A formação do banco de dados é extremamente importante para o desenvolvimento de pesquisas como reconhecimento automático de fala, biometria pela voz, conversão texto-fala, redução de ruído, dentre outras. Este trabalho em andamento encontra-se na metade do cronograma proposto inicialmente.

1. Introdução

O início do trabalho está completamente pautado em programação. Para o desenvolvimento do site, é necessário conhecimento das linguagens HTML5, JavaScript e CSS, sendo esses três juntos a base da programação front-end e desenvolvimento Web. São os responsáveis pela parte que interage com o usuário. O back-end é fundamental para o armazenamento do áudio e outras informações dos usuários.

Em relação a essa última, como precisaremos de um servidor para hospedar o site, o escolhido foi o Node.js – interpretador de código JavaScript, o qual possibilita a confecção de um servidor – que cumpre esse papel e ainda direciona os arquivos do lado do cliente até o lado do servidor onde os áudios gravados serão armazenados localmente.

A segunda parte do projeto será mais científica em que faremos a estimação da relação sinal-ruído dos áudios capturados como forma de auxiliar desenvolvimentos futuros ao rotular e diferenciar áudios com boa qualidade (alta relação sinal-ruído) dos áudios de baixa qualidade.

O servidor poderá ser hospedado em algum computador mantido pela CGI da FEI como forma de facilitar o acesso de usuários internos da FEI, principalmente alunos. Logicamente o acesso externo também será habilitado

2. Metodologia

Depois de conhecimento adquirido acerca de HTML, JavaScript e CSS, o primeiro passo foi a criação da página inicial do projeto. Nela, um formulário foi criado para que o usuário possa criar um nome de usuário, fazer o login e preencher com seus dados pessoais. Após isso, poderá iniciar a sessão de gravações na plataforma de voz. Essa etapa exigiu conhecimento básico das linguagens já citadas e do protocolo HTTP, esquematizado basicamente na Figura 1.

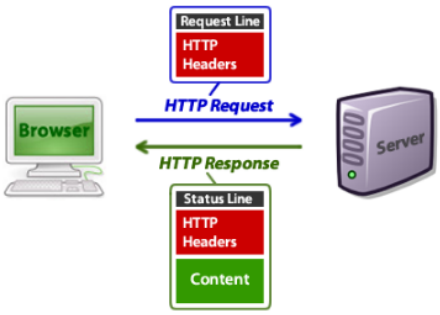

Figura 1 – Funcionamento básico do HTTP.

Entendendo como browser e servidor se comunicam e os métodos GET e POST [1], o próximo passo foi a criação do servidor através do Node.js, em JavaScript com algumas bibliotecas adicionais como o express e o multer. Trecho importante do código do servidor é apresentado na Figura 2.

O express é um framework back-end que cria abstrações de rota e participa da criação e obtenção dos dados a partir do servidor, enquanto o multer é um middleware usado para o upload de arquivos.

```
var express = require('express');        //as primeiras linhas
var cons = require('consolidate');       //do código servem para
var multer = require('multer');          //fazer a requisição das
var app = express();                     //bibliotecas externas

app.engine('html', cons.swig);           //esses comandons irão determinar
app.set('view engine','html');           //aonde o conteúdo visível/aparente
app.set('views',__dirname+'/views');     //se encontra

app.get('/',function(req,res){             //através do método 'GET', caso '/' seja o caminho
    res.render('index'));                  //o servidor deve renderizar a página index.html

app.post('/index2', function(req,res){     //assim que uma requisição do tipo 'POST'
    res.render('index2');                  //for chamada e tiver o endereço de
})                                         //'index2', essa página será renderizada

var upload = multer({ dest: __dirname + '/static/uploads/' });  //com o uso do multer, é possível
var type = upload.single("audio_data");                         //escolher em que diretório do projeto
                                                                //os uploads de gênero "audio_data"
                                                                //serão salvos

app.post('/upload', type, function (req, res) {  //essas linhas endereçam o caminho para o
    console.log(req.body);                       //upload do arquivo gerado no lado do cliente
    console.log(req.file);                       //quando seu post é requerido, uma mensagem no
 });                                             //console do servidor mostra como o arquivo foi salvo

app.use(express.static(__dirname + '/static'));  //essa linha possibilita que conteúdos de imagens,
                                                 //css e javascript consigam ser interpretados caso
                                                 //estejam dentro da pasta de nome "static" do projeto

app.listen(3000, () => {                                    //por último, essas linhas têm função
    console.log('Servidor rodando em http://localhost:3000');
    console.log('Para derrubar, ctrl + C');                 //de deixar o servidor funcionando na
});                                                         //porta 3000 e operando
```

Figura 2 – Código do servidor.

Para criar a página da coleta de voz, uma biblioteca já existente em JavaScript, Recorder.js, foi utilizada. Desenvolvida pelo americano Matt Diamond[2], esse recurso externo cumpria praticamente todas as funções que eram necessárias: solicitar o uso do microfone ao usuário, gravar sua voz e salvar como arquivo no formato .WAV.

Porém, o upload dos arquivos de áudio do código original[2] depende de PHP. Uma decisão deste projeto é reduzir o número de linguagens de programação utilizadas.

A solução encontrada foi substituir PHP por um recurso moderno do JavaScript chamado Fetch[3]. Ele fornece uma definição genérica de objetos de Request e Response, permitindo que sejam usados onde quer que seja no futuro.

Esse método tem um argumento obrigatório, o caminho para o recurso que deseja obter. Ele retorna uma promessa que resolve a Response para esta requisição, seja ela bem-sucedida ou não. Por fim, quando a Response é recuperada, o conteúdo do corpo e como será tratado é facilmente definido por outros métodos internos do Fetch, como mostrado na Figura 3.

```
var fd = new FormData();
fd.append("audio_data", blob, filename);

fetch('/upload',
{
    method: 'post',
    body: fd
});
```

Figura 3 – Parâmetros do recurso Fetch.

Com o exemplo dessa última figura, vemos que o argumento obrigatório (caminho) foi definido como /upload através do método POST, e o conteúdo do corpo é um formulário com os dados do BLOB (Binary Large Object) do arquivo de voz gravado. Agora, cabe ao servidor fazer uma Request do corpo para que os dados sejam manipulados desse lado, como ilustrado na Figura 5.

Dessa forma, com todas essas mudanças e ajustes, a plataforma já pode ser testada. Os próximos passos envolvem estudos mais detalhados que serão feitos para entender como alterar parâmetros de aquisição de áudio, como frequência de amostragem, delinear a estrutura de diretórios para organização dos áudios recebidos, formulação de estratégia de sugerir frases pré-estabelecidas para que os usuários/locutores pronunciem, aprimorar o mecanismo de login dos usuários, dentre muitas outras tarefas de testes.

3. Resultados

Com toda a infraestrutura montada, depois que a primeira versão funcional do site for implementada para teste, a participação dos usuários será fundamental. Os mesmos devem entrar no site, preencherem dados pessoais básicos, liberarem acesso ao microfone e gravarem frases pré-estabelecidas. No momento em que os botões são clicados, mensagens no console informam essas ações e alertam se a função de capturar o áudio do microfone do usuário (**getUserMedia()**)foi bem sucedida, como vemos na Figura 4. Todos os áudios gerados serão diferentes uns dos outros, já que serão salvos com as informações anteriormente fornecidas.

```
recordButton clicked
getUserMedia() success, stream created, initializing Recorder.js ...
Recording started
stopButton clicked
```

Figura 4 – Mensagem completa do que o console responde assim que a gravação é realizada.

A informação gerada depois de gravar a voz, tem que chegar no lado do servidor onde será guardada com todas as outras possibilitando sua manipulação.

```
[Object: null prototype] {}
{ fieldname: 'audio_data',
  originalname: '2019-09-03T14:27:29.389Z',
  encoding: '7bit',
  mimetype: 'audio/wav',
  destination: 'C:\\Users\\Cliente\\Desktop\\aapp/static/uploads/',
  filename: 'fd43390c08be07abc4c02de91dc6b1bd',
  path:
   'C:\\Users\\Cliente\\Desktop\\aapp\\static\\uploads\\fd43390c08be07abc4c02de91dc6b1bd',
  size: 270380 }
```

Figura 5 – Mensagem que o console (do lado do servidor) responde mostrando informações do arquivo.

Observa-se que o nome do arquivo criado ainda não está certo, já que deveria conter as informações de login do usuário, mas essa mudança é mais simples de ser atingida. Outras como o tipo de arquivo e caminho final de upload já foram definidas corretamente.

Em relação ao design estético das páginas web, ainda é apenas um esboço. Como fazer mudanças visuais é algo mais simples e imediato, é viável deixar mais para o final, mantendo foco inicial na conexão cliente-servidor, transmissão de áudio e sua organização em diretórios no servidor.

4. Conclusões

O projeto encontra-se em andamento e os resultados parciais indicam que poderá ser concluído com êxito.

O ponto crítico e consequentemente a tarefa mais difícil até o momento foi a parte do Upload em que o arquivo de som gerado e transformado em BLOB deveria ser enviado para o lado do servidor.

O caráter científico da pesquisa é alcançado principalmente na segunda metade do projeto, onde os arquivos de áudio obtidos passarão por uma análise de sinal-ruído, tornando-se disponíveis para o desenvolvimento de novos projetos como biometria vocal, reconhecimento de voz, entre outros.

Até o momento tem sido desafiador e ao mesmo tempo enriquecedor fazer parte do projeto. Enxergar utilidade no que está sendo feito e estudar sobre novos assuntos fora da zona de conforto, são motivos para seguir com o trabalho.

5. Referências

[1]https://www.w3schools.com/tags/ref_httpmethods.asp
[2]https://github.com/addpipe/simple-recorderjs-demo
[3]https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API

Agradecimentos

À instituição Centro Universitário da FEI pela realização das medidas ou empréstimo de equipamentos. Ao professor orientador pela prontidão e auxílio no desenvolver do projeto.

[1] Aluno de IC do Centro Universitário FEI. Projeto com vigência de 04/19 a 03/20.