UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AUGUSTO BENNEMANN

# Pré-visualização de Artefatos Musicais no Navegador

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcelo Soares Pimenta

Porto Alegre
2018

## AGRADECIMENTOS

À minha família por todo o apoio até aqui. Aos professores e colegas que muito contribuíram para minha formação. Ao amigo Lucas Zawacki pelo suporte e pelas ideias e discussões essenciais à concretização deste trabalho. Ao Prof. Dr. Marcelo Soares Pimenta pela orientação neste trabalho.

## RESUMO

*Musical Artifacts* é um repositório colaborativo de arquivos, conhecidos como *artefatos*, para produção musical, tais como bibliotecas sonoras (*soundfonts*) e configurações para softwares de produção (processadores de efeitos, sintetizadores e extensões). Atualmente, o processo de experimentação desses arquivos é tedioso e demorado, sobretudo quando se deseja comparar diversos artefatos, uma vez que exige baixar e configurar os arquivos em *softwares* específicos. Visando a melhorar a experiência do usuário, a proposta deste trabalho é criar um sistema *web* que possibilite a pré-visualização, diretamente no navegador, de dois tipos de artefatos: *soundfonts* e arquivos para o software *Guitarix*, um amplificador virtual de guitarra. Para o primeiro, a solução proposta permite que usuário experimente o artefato a partir de um piano virtual, que pode ser controlado, inclusive, por meio de um dispositivo MIDI. Já para o segundo, a solução envolve a criação de um servidor de processamento de áudio que, ao receber uma gravação da guitarra do usuário (realizada no navegador), aplica a ela o artefato e retorna o áudio resultante.



**Palavras-chave:** Musical Artifacts. Guitarix. Soundfont. JACK. MIDI.

**Preview Musical Artifacts in the Browser**

## ABSTRACT

*Musical Artifacts* is a collaborative repository of files, known as *artifacts*, used for music production, such as soundfonts and presets for software (effects processors, synthesizers and extensions). Currently, the process of experimenting artifacts is discouraging and time-consuming for users, in particular when comparing several artifacts, once it requires downloading and configuring the files in specific software. In order to improve the user experience, this work aims to create a web application for previewing, in the browser, two types of artifact: soundfonts and files for *Guitarix*, a virtual guitar amp. For the first, the proposed solution enables users to try out the artifact using a virtual piano, which can also be controlled from a MIDI device. For the latter, the solution involves the creation of an audio processing service, which receives a recording of the user's guitar (performed in the browser) and returns the result of applying a specified artifact to it.

**Keywords:** Musical Artifacts, Guitarix, Soundfont, JACK, MIDI, Audio.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| BLOB | Binary Large OBject |
| DOM | Document Object Model |
| DBMS | Database Management System |
| CSV | Comma-separated Values |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| JSON-RPC | JSON Remote Procedure Call |
| MA | Musical Artifacts |
| MIDI | Musical Instrument Digital Interface |
| SQL | Structured Query Language |
| SUS | System Usability Scale |
| URL | Uniform Resource Locator |
| UUID | Universally Unique Identifier |

**SUMÁRIO**

# 1 INTRODUÇÃO

O Musical Artifacts (ZAWACKI, 2015) é um *website* que funciona como um repositório colaborativo de arquivos para produção musical. Nele, artistas audiovisuais, músicos e produtores compartilham arquivos que são úteis para a realização de seu trabalho. Esses arquivos, os quais o projeto nomeia de *artefatos*, consistem tipicamente em bibliotecas sonoras (*soundfonts*), *presets*, arquivos MIDI (ASSOCIATION, 2018) e configurações para softwares de produção - a saber, processadores de efeitos, sintetizadores e extensões. A ênfase do projeto está na disponibilização de artefatos em formatos abertos e sob licenças livres - a exemplo de Creative Commons, Domínio Público e GNU GPL.

Ao redor do site há uma comunidade de usuários bastante ativa, que contribui disponibilizando novos artefatos e utilizando e avaliando os existentes. Esses usuários, no entanto, enfrentam uma limitação de usabilidade: a falta de pré-visualização sonora dos artefatos. Atualmente, uma parcela pouco significativa dos artefatos possuem arquivos de demonstração - áudios ou vídeos que tenham sido produzidos com ele. Esse tipo de pré-visualização, apesar de útil, é bastante limitado pois não é interativo.

Quando o artefato não possui arquivo de demonstração ou quando o usuário deseja utilizar seu instrumento musical para experimentá-lo, a única alternativa é testá-lo em sua máquina. Nesse caso, o usuário precisa baixar o artefato e configurá-lo em um ou mais softwares de áudio. Esse processo demanda um tempo considerável, que pode ser ainda maior no caso (não raro) de arquivos muito grandes. Somente após isso o usuário consegue avaliar se o artefato, na prática, o atende. Esse acaba sendo um grande impeditivo quando se deseja testar e comparar vários artefatos.

Diante dessas limitações, este trabalho propõe a criação de formas de pré-visualização alternativas, que sejam mais interativas e funcionem diretamente no navegador. Assim, exime-se o usuário da necessidade de realizar o *download* e a configuração do artefato em sua máquina. O foco desta implementação são dois tipos de artefatos: os para o *software Guitarix* e os *soundfonts*. A escolha desses formatos de arquivo deve-se a: $(i)$ os dois juntos representarem mais da metade dos artefatos cadastrados; $(ii)$ implementação da solução proposta para cada um deles ser bastante singular.

No caso dos artefatos para o *Guitarix*, a solução possibilita que o usuário faça uma gravação de sua guitarra diretamente no navegador e, então, escolha um conjunto de artefatos para avaliar. Por meio de uma API, a gravação é enviada a um servidor, que funciona como um serviço de processamento de áudio remoto. Esse servidor executa uma

série de softwares de áudio em ambiente GNU/Linux. Quando uma requisição é recebida, ela é processada e, em seguida, o arquivo de áudio resultante é disponibilizado ao cliente.

Já no caso dos *Soundfonts*, a produção de áudio ocorre diretamente na máquina do usuário. O servidor mantém, para cada artefato, arquivos com os sons de todas as notas musicais. Assim, quando o usuário escolhe um artefato para pré-visualizar, um arquivo com seus sons é carregado e, então, o usuário pode experimentá-lo utilizando um controlador MIDI ou o piano disponível na interface (controlável por toque, *mouse* ou teclado).

## 1.1 Objetivos

O principal objetivo deste trabalho é melhorar significativamente a experiência do usuário do Musical Artifacts. Acredita-se que a possibilidade de pré-visualizar os artefatos diretamente no navegador contribuirá muito nesse aspecto. Além de facilitar o processo, evitando o *download* e a configuração do artefato, esse recurso permitirá aos usuários do Guitarix, por exemplo, comparar diferentes artefatos de modo muito mais rápido.

Como objetivo secundário, espera-se que a solução desenvolvida apresente qualidade e potencial que a possibilite ser integrada por outros sistemas, com diferentes propósitos. Aspira-se, também, que a solução se torne uma referência em código aberto de um serviço de processamento de áudio remoto, de maneira a ser adaptada ou servir de base para outras soluções.

Espera-se também que as melhorias elevem o nível de engajamento da comunidade e incentivem novas pessoas a participarem do projeto. Assim, como consequência, se fomentará a utilização, a criação e o compartilhamento de artefatos musicais livres.

## 1.2 Organização deste trabalho

Este trabalho está organizado em sete capítulos. O primeiro capítulo, do qual esta seção faz parte, detalha a motivação e os objetivos deste trabalho. Em seguida, o capítulo 2 apresenta o Musical Artifacts, descrevendo seu propósito, seu funcionamento e sua comunidade.

O capítulo 3 apresenta conceitos e um panorama sobre produção musical em am-

biente GNU/Linux, com ênfase nos *softwares* utilizados na implementação deste trabalho. Seu objetivo é prover ao leitor a base necessária para compreender a ferramenta desenvolvida.

O capítulo 4 avalia o estado da arte, comparando trabalhos relacionados e analisando pontos comuns e divergentes. Em seguida, o capítulo 5 apresenta a solução desenvolvida, explicando a arquitetura, detalhes de implementação e seu funcionamento.

O capítulo 6 apresenta uma avaliação do processo e uma análise da avaliação do produto, realizada com usuários reais. É discutida a metodologia empregada e os resultados obtidos.

Por fim, o capítulo 7 discute as conclusões a respeito da solução desenvolvida. É apresentado um resumo das contribuições realizadas, suas limitações e possíveis direções para trabalhos futuros.

## 1.3 Arquivos complementares

O código-fonte da solução desenvolvida neste trabalho está disponível no repositório (BENNEMANN, 2018). Os processos de instalação, configuração e utilização da aplicação estão documentados na *wiki* do projeto (MUSICAL..., 2018).

# 2 MUSICAL ARTIFACTS

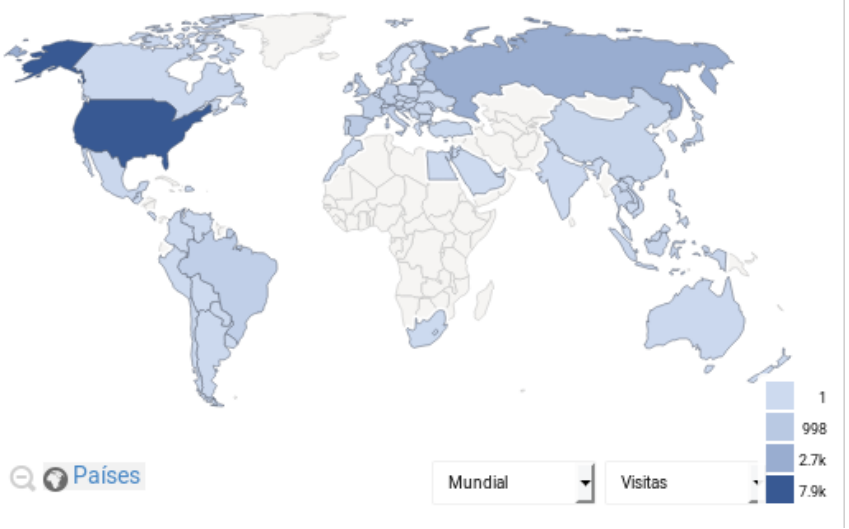
Musical Artifacts é um *website* que tem por intuito auxiliar artistas audiovisuais, músicos e produtores a encontrarem e preservarem arquivos que são necessários para a realização de seu trabalho. Esses arquivos são tipicamente bibliotecas sonoras, configurações para softwares de produção - a saber, processadores de efeitos, sintetizadores e extensões -, *presets* e arquivos MIDI.

A ênfase é em disponibilizar os arquivos - chamados pelo projeto de *artefatos* - sob licenças e formatos livres, assim como catalogar, manter e preservá-los para consulta futura. Parte do esforço consiste, também, em educar os criadores desses artefatos acerca dos benefícios que eles e a comunidade podem obter quando suas criações são disponibilizadas sob licenças livres e formatos abertos e bem documentados.

A comunidade que envolve o projeto é majoritariamente de músicos e produtores que utilizam software livre em seu trabalho. No entanto, há também uma parcela bastante significativa de contribuições e acessos por outros perfis de pessoas - a exemplo de desenvolvedores de jogos *indie*-, que se aproveitam da disponibilidade desses arquivos para usá-los em seus projetos.

Atualmente, o site hospeda aproximadamente 650 artefatos e diariamente recebe cerca de 800 acessos e um número próximo de *downloads*. Ele está disponível em três idiomas - Inglês, Francês e Português do Brasil - e é acessado por pessoas de todo o mundo. As estatísticas de acesso referentes a Outubro de 2018 revelam a distribuição por país: 42,4% Estados Unidos; 14,2% Rússia; 5,4% Reino Unido; 5% Espanha; 3,4% França; 3,4% Brasil; 3,3% Itália; 3,2% Japão; 2,8% Alemanha; 1,6% China; 1,2% Tailân-

Figura 2.1: Acessos ao Musical Artifacts em Outubro/2018.



dia; 1,2% Indonésia; 1,1% Polônia; outros 61 países dividem os 12,8% restantes. Esses números estão ilustrados no mapa da figura 2.1.

Os dados também apresentam a distribuição de usuários por navegador: 52% *Google Chrome*; 12% *Firefox*; 9,2% *Google Chrome Mobile*; 7,2% *Internet Explorer*; 4% *Opera*; 2,4% *Microsoft Edge*; os 13,2% restantes estão distribuídos em pequenas fatias entre dezenas de outros navegadores, a exemplo de *Safari Mobile*, *Yandex Browser*, *Chromium*, *Samsung Browser*, e diversas derivações do *Mozilla Firefox*.

A figura 2.2 demonstra a página do artefato *Vox Beatles Mystery Presets*, um arquivo para o *Guitarix*. Junto ao título são exibidos o nome do usuário que enviou o arquivo e as datas de adição e última atualização.

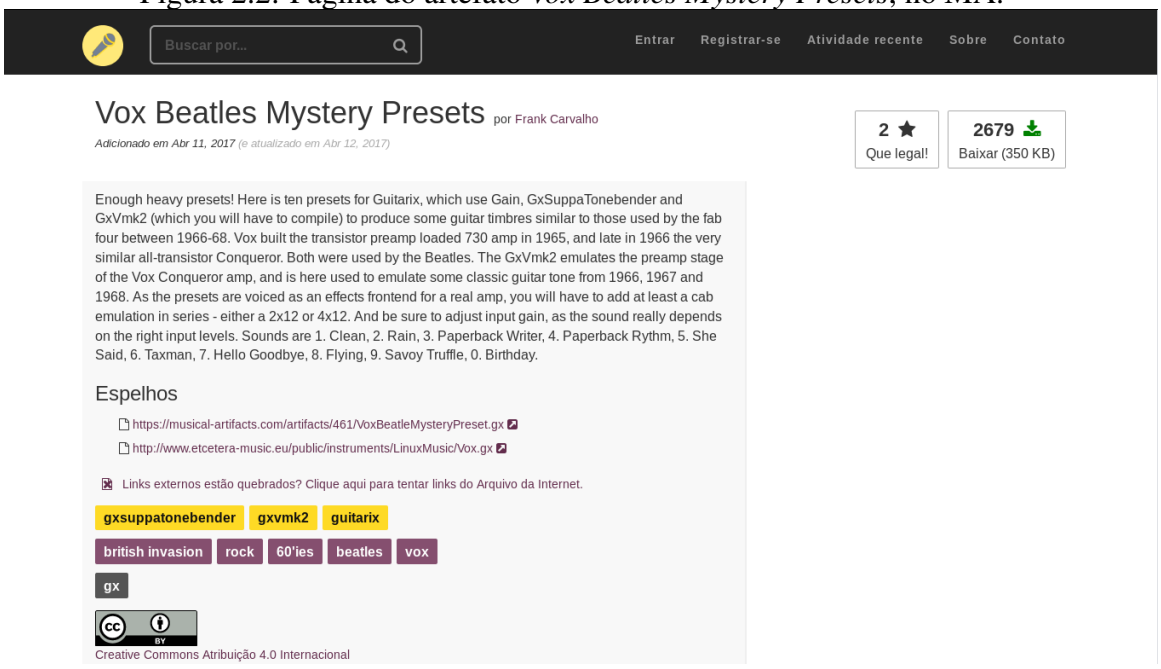
Cada artefato possui também uma descrição. Ela é exibida logo abaixo, na região de fundo cinza. Esse arquivo, assim como muitos outros, também está hospedado em servidores de terceiros. Por isso, abaixo da descrição há o item *Espelhos*, no qual são mostradas as URLs para cópias do arquivo em outros servidores.

Em seguida são exibidas outras três informações, que podem ser utilizadas na filtragem da busca: o *software* para o qual ele serve, com fundo amarelo; as *tags* que o usuário escolheu, com fundo roxo; e o formato do arquivo, com fundo cinza.

Por fim, é apresentada a licença do artefato (no caso, *CC BY*). Ademais, à direita na página estão os botões para favoritar e baixar, respectivamente, o artefato.

O Musical Artifacts foi criado em setembro de 2015 por Lucas Fialho Zawacki, di-

Figura 2.2: Página do artefato *Vox Beatles Mystery Presets*, no MA.



ante da necessidade de auxiliar a comunidade de música e software livre (FOUNDATION, 2018b) a organizar e preservar o material utilizado e produzido por todos os membros. Ele é um software livre (sob a licença MIT (MIT..., )) e oferece uma API de acesso, por meio da qual outros projetos que desejem fazer uso dos arquivos catalogados podem realizar a integração.

## 2.1 Licenças Livres

Segundo (ECAD, 2018), "direito autoral ou direito de autor é um conjunto de prerrogativas conferidas por lei à pessoa física ou jurídica criadora da obra intelectual, para que ela possa usufruir de quaisquer benefícios morais e patrimoniais resultantes da exploração de suas criações". Portanto, por padrão, todos os direitos relativos a uma obra são reservados a seu criador.

As *licenças livres* são licenças que determinam termos e condições para a utilização e distribuição de uma obra. Elas são mais permissivas, de modo que, ao licenciar um trabalho sob uma licença desse tipo, o autor concede autorizações adicionais, para além do especificado pelo direito autoral. Durante muito tempo, isso não foi possível, como afirma (LEMOS, 2005, p. 83): "até o surgimento da internet, da tecnologia digital e de um modelo jurídico como o Creative Commons, não havia meios para que esses autores pudessem indicar à sociedade que eles simplesmente não se importam com a divulgação de suas obras".

A escolha de muitos autores por esse tipo de licença deve-se a acreditarem que o compartilhamento de conhecimento e criatividade é um caminho para se "alcançar um mundo mais equânime, acessível e inovador", como defende a licença *Creative Commons* (CREATIVE..., 2018d). No caso do Musical Artifacts, em que os arquivos disponibilizados são utilizados diretamente para produção musical, o licenciamento sob licenças livres possibilita aos usuários uma maior liberdade artística, uma vez que eles possuem mais permissões sobre os artefatos.

As licenças mais utilizadas pelos usuários do MA são, em ordem, *CC BY* (CREATIVE..., 2018a), *CC BY-SA* (CREATIVE..., 2018b) e *Domínio Público*. Todas essas são consideradas livres, mas variam em aspectos em relação a atribuição de crédito ao autor, utilização em trabalhos derivados e regras de distribuição.

Em primeiro e segundo lugar entre as mais populares estão duas licenças *Creative Commons*. As licenças CC (com exceção da CC-O (CREATIVE..., 2018c), que é equi-

valente a Domínio Público), são nomeadas de acordo com 4 componentes - *BY, NC, ND* e *SA* -, que combinadas formam seis configurações.

A componente BY (Atribuição) está presente em todas as variações e significa que deve ser atribuído o crédito ao autor original. A componente NC (NãoComercial), por sua vez, determina que a obra não pode ser utilizada para fins comerciais.

As licenças que possuem a componente ND (SemDerivações) não permitem a distribuição de obras derivadas - remixadas, transformadas ou criadas a partir do material original. As que possuem a componente SA (CompartilhaIgual), por sua vez, exigem que as obras derivadas sejam distribuídas sob a mesma licença da obra original. Essas duas componentes, portanto, não podem ser combinadas.

A combinação das quatro componentes dá origem a seis licenças: CC-BY; CC-BY-SA; CC-BY-ND; CC-BY-NC; CC-BY-NC-SA; CC-BY-NC-ND. A licença mais utilizada no MA, CC BY (Creative Commons Atribuição), permite que a obra seja utilizada, adaptada e redistribuída livremente (inclusive para fins comerciais e sob outras licenças), desde que seja atribuído o devido crédito ao autor original.

A CC BY-SA (Atribuição-CompartilhaIgual), por sua vez, é a segunda licença mais utilizada no MA. Ela permite que a obra seja utilizada, adaptada e redistribuída, desde que o crédito ao autor original seja atribuído e as obras derivadas sejam licenciadas sob a mesma licença.

Em terceiro lugar estão os autores que preferem renunciar à propriedade da criação, declarando suas criações como *Domínio Público*. Assim, elas se tornam um bem comum.

# 3 SUPORTE A PRODUÇÃO MUSICAL NO GNU/LINUX

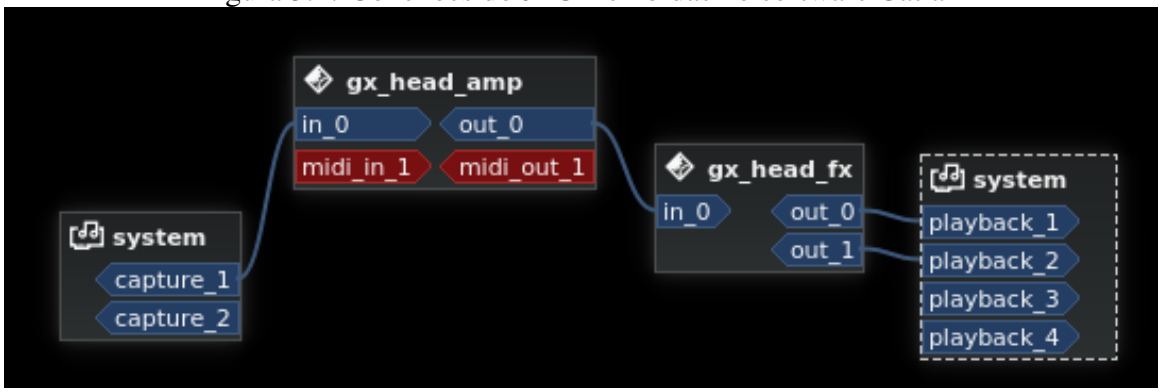
A produção musical no ambiente GNU/Linux está vigorosamente baseada no *JACK*, um servidor de áudio utilizado por uma parcela bastante significativa do softwares de áudio profissional disponíveis para o sistema operacional. Além de abordá-lo, este capítulo também apresenta outros *softwares* e formatos de arquivos utilizados na solução desenvolvida, como *Guitarix* e *soundfonts*. Ademais, também são abordadas duas APIs - *MediaStream Recording API* e *Web MIDI API* - implementadas pelos navegadores mais modernos e que possibilitam a construção de aplicações musicais mais interativas.

## 3.1 JACK

*JACK Audio Connection Kit* (JACK..., 2014) é um servidor de áudio de uso profissional compatível com sistemas operacionais que seguem o padrão POSIX (WALLI, 1995). No Linux, é uma alternativa ao PulseAudio (PULSEAUDIO, 2018). Ele provê conexão de áudio e MIDI, em tempo real (*real-time*) e com baixa latência, entre diferentes aplicações que implementam sua API. Ademais, o JACK oferece a possibilidade de execução sincronizada de todos clientes.

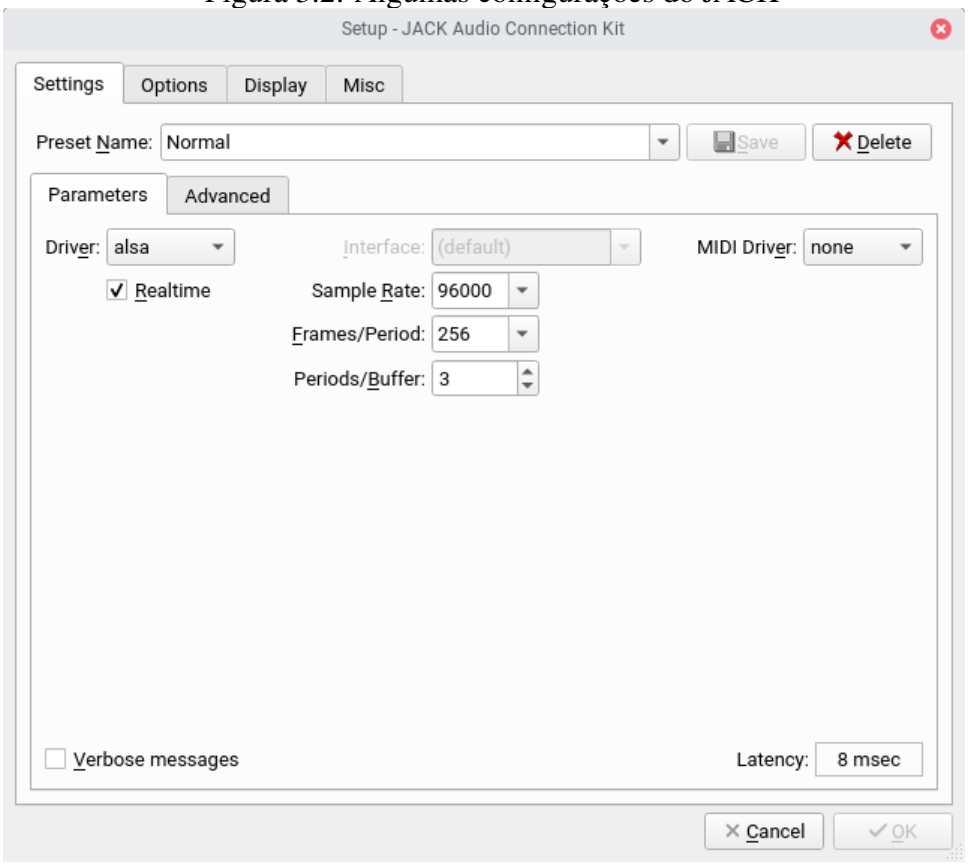
As conexões entre os clientes podem ser gerenciadas por meio de diversas aplicações. O software Catia (CATIA..., 2018), ilustrado na figura 3.1, é uma opção. Na imagem, pode-se observar os clientes (blocos em cinza escuro) e as conexões entre eles (linhas azuis). Cada cliente pode ter zero, uma ou mais portas de entrada e saída. As portas de áudio são representadas em azul e as de MIDI em vermelho. Ao se utilizar o Catia, para conectar dois clientes basta clicar na porta de origem e arrastar o cursor até a porta de destino.

Figura 3.1: Conexões do JACK exibidas no software Catia



O JACK permite a configuração de diversos parâmetros que influenciam na latência, como por exemplo a taxa de amostragem. A figura 3.2 demonstra uma tela de configuração com algumas dessas configurações. A latência, exibida no canto inferior direito da janela, varia de acordo com os valores definidos para os parâmetros.

Figura 3.2: Algumas configurações do JACK




Ainda, o JACK possui o mecanismo de sincronização *JACK Transport*. Esse mecanismo possibilita a execução sincronizada de um conjunto de clientes. Desse modo, ao iniciar, parar ou reposicionar o áudio em um dos clientes, todos os outros também sofrem essa ação. Isso possibilita um nível de integração bastante satisfatório, que torna a experiência do usuário mais fluída.

Além disso, o JACK oferece dois modos de operação: o *realtime* e o *freewheel*. No primeiro, o processamento ocorre em tempo real, enquanto que no segundo ocorre na velocidade máxima que o hardware e o sistema operacional permitirem. O modo *realtime* é utilizado quando se deseja processar um fluxo de áudio na mesma velocidade em que ele é gerado. Esse é o caso quando se está produzindo música ao vivo e deseja-se um retorno imediato (menor latência possível) do som processado, como por exemplo ao tocar guitarra e utilizar o Guitarix para aplicar efeitos.

Em contrapartida, o modo *freewheel* é o utilizado para processar fluxos de áudio

que já estão totalmente disponíveis, como gravações, por exemplo. A execução nesse modo é muito mais rápida, pois não depende de esperar o áudio ser gerado em tempo real. Por conseguinte, esse é o modo que permite a um software de gravação renderizar minutos de áudio em apenas alguns segundos.

## 3.2 Guitarix

O Guitarix (GUITARIX. . . , 2018) é um amplificador de guitarra virtual para GNU/-Linux (FOUNDATION, 2018a) que roda sobre o JACK. Por meio dele, é possível simular o som produzido por diversos modelos de amplificadores e aplicar um grande conjunto de efeitos a um fluxo de áudio (geralmente de uma guitarra) fornecido como entrada.

A figura 3.1 ilustra os clientes do JACK e suas conexões ao utilizar-se o Guitarix. Pode-se perceber que ele possui dois clientes: o amplificador (*gx_head_amp*) e o *rack* (*gx_head_fx*). No *rack* podem ser adicionados módulos de efeitos - a saber, ruído, *delay* e efeitos de modulação, como *flanger* e *phaser*.

O amplificador e o *rack* podem ser utilizados de maneira independente. No entanto, o uso ilustrado na figura é o padrão. Nesse caso, o áudio da guitarra (*system:capture_1*) é encaminhado para o amplificador (*gx_head_amp:in_0*). Depois, o áudio resultante se-

Figura 3.3: Interface do Guitarix

gue ao *rack* (*gx_head_fx:in_0*) que, por fim, encaminha o áudio produzido ao dispositivo de saída de áudio *system:playback_1*.

A interface gráfica do Guitarix é ilustrada na figura 3.3. À esquerda estão as opções de módulos de efeito que podem ser adicionados ao *rack*. Na parte inferior, há a seleção de banco e *preset* a ser utilizado. Ademais, no centro da tela são exibidos todos controles relativos ao amplificador e aos módulos do *rack*.

O Guitarix também oferece suporte a *plugins* LADSPA (LINUX. . . , 2018) e LV2 (LV2, 2018), dois padrões abertos para *plugins* de áudio. Ambos são amplamente utilizados e suportados por diversos softwares de áudio do Linux. Eles (especialmente o LV2) fornecem uma API bastante extensível, que permite a desenvolvedores construírem *plugins* de diferentes categorias, que variam desde efeitos até sintetizadores. O projeto Calf Studio Gear (CALF. . . , 2018), a saber, oferece plugins LV2 de instrumentos (como órgão, *monosynth* e *wavetable*), de efeitos de modulação (tais como *Chorus*, *Phaser*, *Flanger* e *Ring Modulator*), *delay* e distorção, além de compressores e equalizadores.

### 3.2.1 Presets para Guitarix

Os sons e efeitos produzidos pelo Guitarix variam de acordo com uma série de parâmetros de configuração. Essas configurações podem ser salvas como um *preset*, de modo que possam ser facilmente compartilhadas e utilizadas por outras pessoas.

O Guitarix salva os *presets* em arquivos de texto com a extensão *gx*. Cada arquivo desses é um banco que contém um ou mais *presets*. A figura 3.4 ilustra parte de um desses arquivos, cujo conteúdo consiste em um arquivo JSON.

Figura 3.4: Exemplo de parte de um arquivo de *preset* para Guitarix

```
["gx_head_file_version", [1, 2, "0.37.2git"],
  "livebuffer1", {
    "engine": {
      "AC-15.position": 1,
      "AC-15.pp": "pre",
      "AC-30.position": 1,
      "AC-30.pp": "pre",
      "Ampeg.position": 1,
      "Ampeg.pp": "pre",
      "Bassman.position": 1,
      "Bassman.pp": "pre",
      "Deville.position": 1,
      "Deville.pp": "pre",
      "Engl.position": 1,
      ...
    }
  }
]
```

### 3.2.2 Controle externo

O *Guitarix* oferece uma API JSON-RPC (JSON-RPC, 2018), por meio da qual é possível controlar vários de seus parâmetros durante a execução. Para habilitá-la, é necessário que o *Guitarix* seja inicializado com o argumento $p$, indicando a porta a ser utilizada por ela. Caso a porta 7000 seja escolhida, por exemplo, o comando para execução deve ser *guitarix p 7000*.

Neste trabalho, o único método da API utilizado é o *setpreset*, que define o banco e o *preset* a serem usados. O bloco de código 3.5 ilustra essa chamada, na qual *bank_name* e *preset_name* representam os nomes do banco e do *preset*, respectivamente.

Figura 3.5: Exemplo de chamada à API do Guitarix

```
{
    "jsonrpc": "2.0",
    "method": "setpreset",
    "params": ["bank_name", "preset_name"]
}
```

## 3.3 Soundfont

*SoundFont* (ROSSUM; JOINT, 1995) é um formato de arquivo que contém informações necessárias para a síntese sonora de instrumentos musicais. Esse formato toca arquivos MIDI por meio da técnica de síntese *Sample-Based Synthesis* (CANN, 2011), que consiste em, a partir de instrumentos pré-gravados, realizar modificações como mudança de frequência, volume e aplicação de filtros.

O Musical Instrument Digital Interface (MIDI) (ASSOCIATION, 2018) é um protocolo utilizado para comunicação entre instrumentos musicais eletrônicos e dispositivos de áudio. Um único arquivo MIDI pode conter partituras de até 128 instrumentos, de acordo com a definição padrão do formato. Por conta disso, esse é o número de instrumentos que um *soundfont* pode conter.

Embora existam versões mais recentes do formato, como o *SF3*, o *SF2* ainda é amplamente mais utilizada. Há também o *SFZ*, um formato de texto puro, que funciona do mesmo jeito e é mais aberto (CANN, 2009).

## 3.4 APIs do Navegador

À medida que a *web* evolui e novas tecnologias são incorporadas aos navegadores, o desenvolvimento web ganha recursos que até então só estavam disponíveis em aplicações nativas. O surgimento de novas APIs nos navegadores amplia as possibilidades de interação e possibilita a criação de novas categorias de aplicação *web*.

Esta seção aborda duas dessas APIs, essenciais para a solução proposta neste trabalho. A *MediaStream Recording API* e a *Web MIDI API* permitem aos desenvolvedores utilizar dois novos tipos de entrada do usuário: áudio e MIDI, respectivamente.

### 3.4.1 MediaStream Recording API

A *MediaStream Recording API* (BARNETT; CASAS-SANCHEZ; LEITHEAD, 2017) implementa o suporte a gravação de áudio no navegador, permitindo assim que páginas *web* capturem o fluxo de áudio de um dispositivo de entrada escolhido pelo usuário. Essa API busca prover aos desenvolvedores um mecanismo simples, removendo a necessidade de operações de *encoding* manual, até então existente.

Atualmente, de acordo com o *Can I Use*(CAN..., 2018a), os navegadores que implementam essa API são: *Mozilla Firefox* desde a versão 29, de 2014; *Google Chrome* desde a versão 29, de 2016; *Opera* desde a versão 36, de 2016; *Chrome para Android* desde a versão 69, desse ano. Outros navegadores bastante utilizados, como *Safari* e *Microsoft Edge*, ainda não possuem esse recurso.

Do ponto de vista do usuário, a utilização também é simples. Quando a API é chamada pela aplicação, o navegador exibe uma caixa flutuante (conforme figura 3.6 para que o usuário selecione o dispositivo que deseja utilizar como entrada e permita a página a acessá-lo.

Por questões de privacidade e segurança, o navegador indica quando está utilizando o microfone (ou outro dispositivo de entrada de áudio) do usuário. No caso do *Firefox* (figura 3.7), um ícone de microfone piscante é exibido na barra de endereços, à esquerda.

Após finalizada a gravação, o navegador disponibiliza o fluxo de áudio em um BLOB. O formato e o *codec* utilizados na gravação variam de acordo com o navegador.

Figura 3.6: Seleção do dispositivo de entrada a ser utilizado para gravação, no *Mozilla Firefox*.

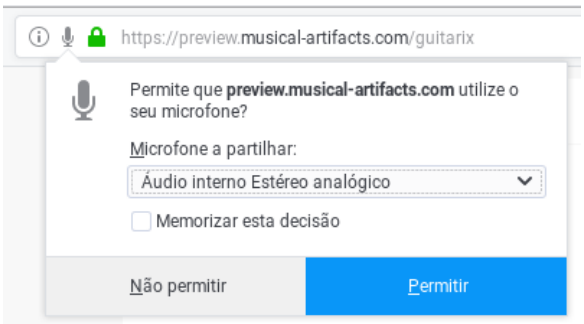
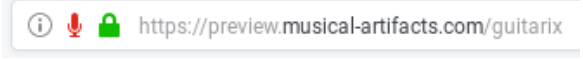



Figura 3.7: Exemplo de indicador de gravação, no *Mozilla Firefox*.



### 3.4.2 Web MIDI API

A *Web MIDI API* (KALLIOKOSKI; WILSON, 2015) implementa o suporte a dispositivos MIDI no navegador. Dessa maneira, uma aplicação *web* pode acessá-los e utilizá-los como entrada - lendo as notas produzidas e diversas outras mensagens que o MIDI suporta - ou saída. Isso possibilita a criação de aplicações com novas formas de interação como, por exemplo, utilizar um controlador MIDI para gerar música, diretamente no navegador.

No momento, apenas uma pequena parcela dos navegadores mais populares a implementa. Segundo (CAN..., 2018b), esse recurso está disponível nos seguintes navegadores desde as versões: Google Chrome 43, de 2015; Opera 30, de 2015; Android WebView / Chromium 67, de 2017; Opera Mobile Android 46, de 2016; Google Chrome Android 69, desse ano.

# 4 TRABALHOS RELACIONADOS

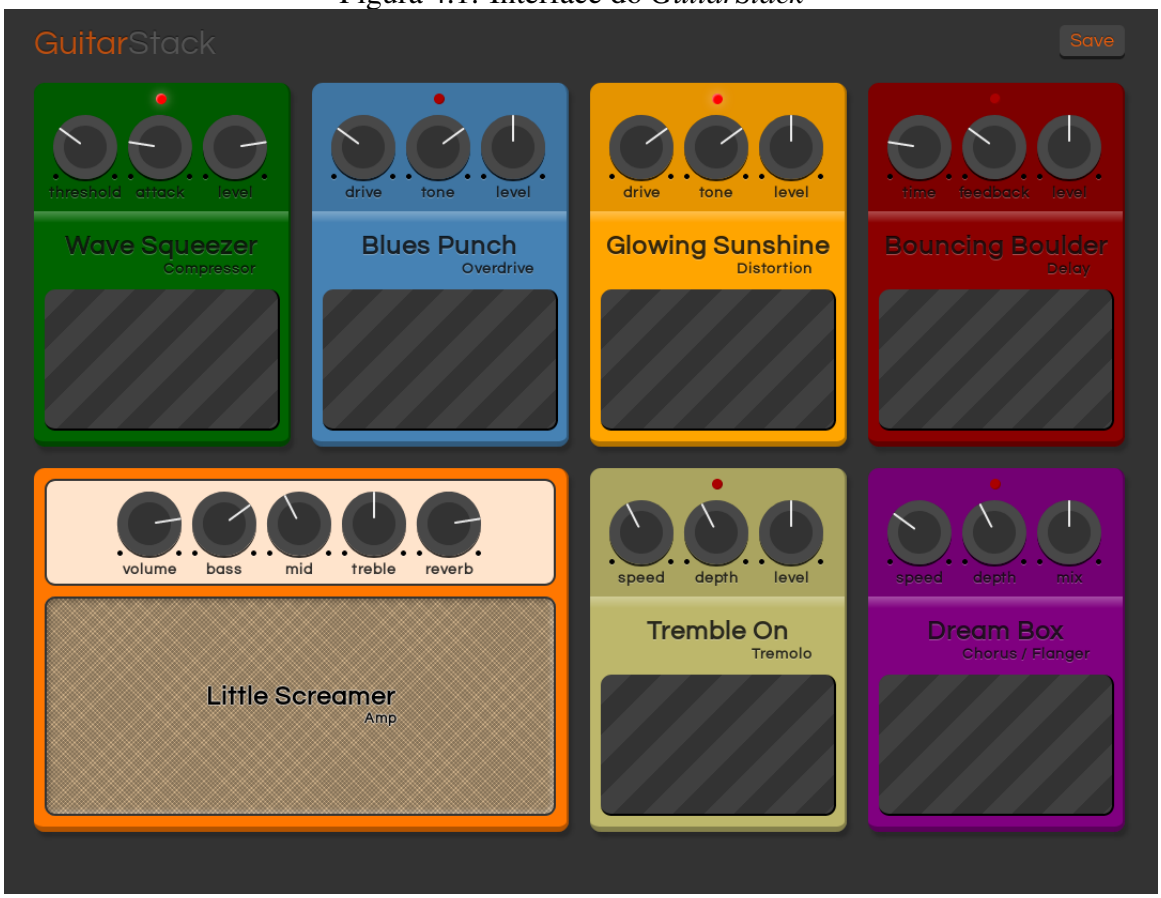
Há diversos outros trabalhos que utilizam APIs de áudio do navegador e exploram a experimentação musical. Entre esses, foram selecionados aqui os que mais se assemelham com a proposta deste trabalho.

## 4.1 GuitarStack

O GuitarStack (BLASCO, 2016) é uma aplicação *web* de código aberto que simula efeitos de pedais de guitarra. O processamento do áudio é implementado em Javascript e, portanto, executado pelo navegador em tempo real. Como fonte de áudio, é possível utilizar uma guitarra ou qualquer outro dispositivo de entrada de áudio.

A figura 4.1 demonstra a interface do GuitarStack. Nela, pode-se observar os efeitos disponíveis: compressor, *overdrive*, distorção, *delay*, *tremolo* e *chorus* / *flanger*, além de controles referentes ao amplificador. Há outros projetos similares a esse, como o

Figura 4.1: Interface do *GuitarStack*



*Pedals.io* (PEDALS..., 2018). No entanto, o GuitarStack é o mais completo.

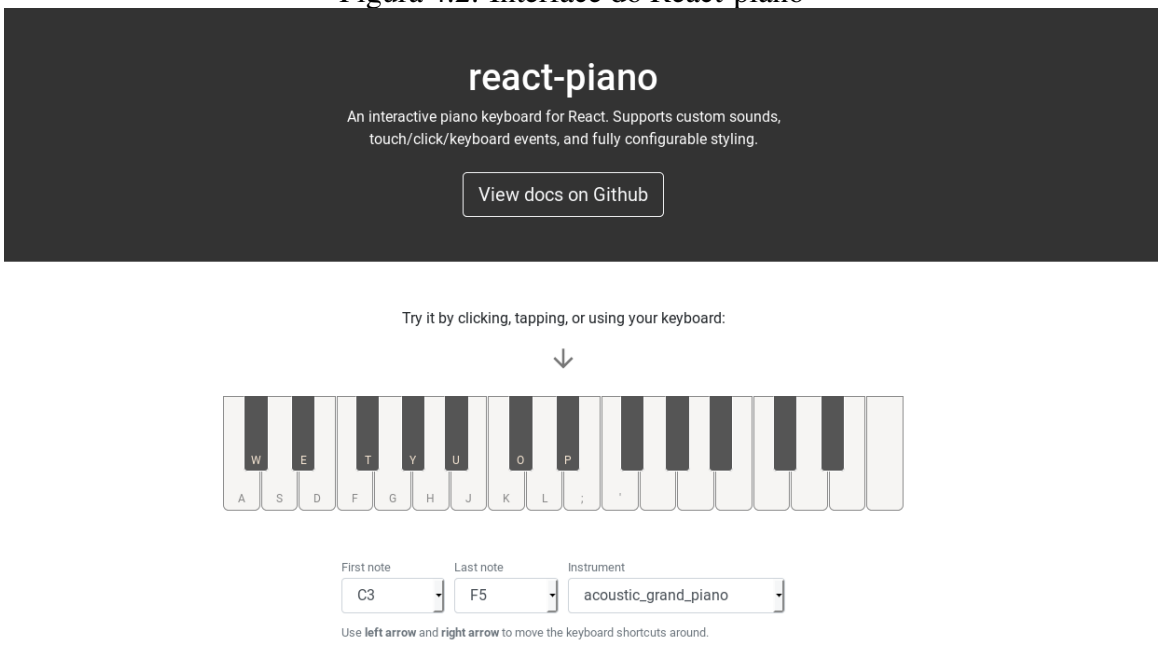
O processamento em tempo real e a facilidade de utilização, por executar no navegador, são qualidades que tornam o GuitarStack uma ótima opção para quem busca um uso mais lúdico. No entanto, em um contexto de produção musical mais profissional, os artefatos para *Guitarix* mostram-se uma opção muito mais interessante, pois executam em um *software* de áudio desenvolvido especificamente para esse propósito e que, também, propicia a utilização conjunta de outros *softwares* de qualidade profissional, por meio do JACK.

## 4.2 React Piano e SoundFont Player

O React Piano (REACT-PIANO..., 2018) é um piano virtual interativo, implementado sobre a biblioteca *React*. Sua interface bem construída propicia uma interação bastante fluída, a partir do *mouse* e do teclado. O React-piano um componente de *software* e seu propósito, portanto, é ser utilizado por outros projetos.

O *SoundFont Player* (BLASCO, 2015), por sua vez, é uma biblioteca javascript que provê aos desenvolvedores maneiras simples de carregar arquivos de sons e tocá-los no navegador, por meio da *Web Audio API* (API..., 2018). Utilizando essa biblioteca, é possível simular a execução de *soundfonts*, desde que anteriormente seja realizado um

Figura 4.2: Interface do React-piano



processo de extração dos sons para um formato específico.

Devido à qualidade dessas duas bibliotecas, considera-se que sua utilização é um bom ponto de partida para o desenvolvimento da pré-visualização de artefatos do tipo *soundfont*. Portanto, este trabalho as utilizará como base para isso. Adicionalmente, será implementado o suporte a dispositivos MIDI para controle do piano.

# 5 DETALHES DA IMPLEMENTAÇÃO

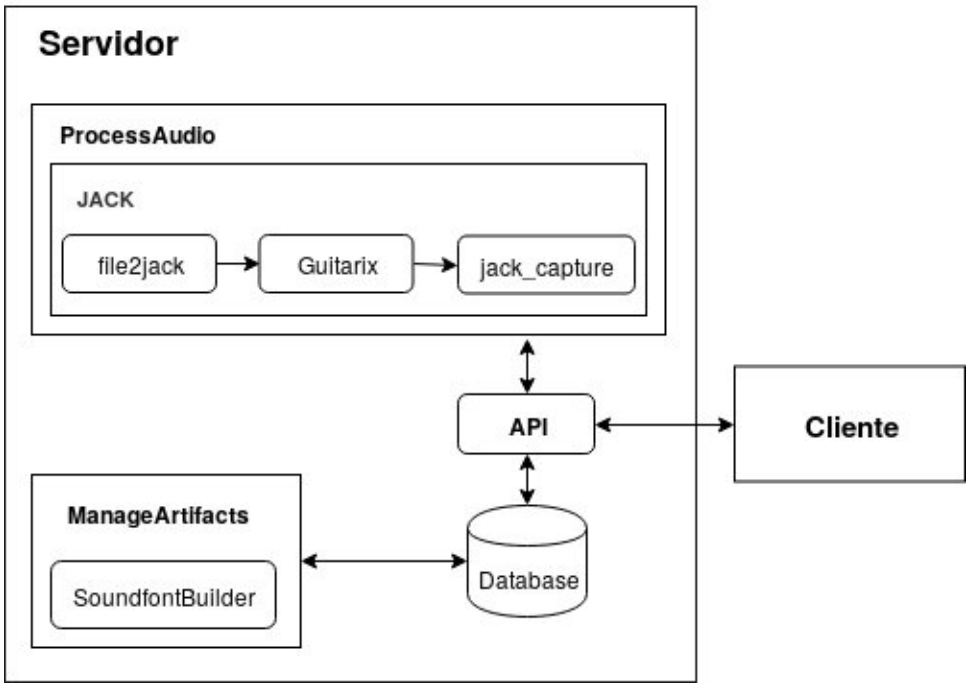
A solução desenvolvida neste trabalho consiste em uma aplicação com modelo cliente-servidor (BERSON, 1992), na qual a comunicação entre as duas partes é realizada por meio de uma API provida pelo servidor. Ao longo deste capítulo, a arquitetura e seus componentes, ilustrados na figura 5.1, são detalhados.

Esta solução possibilita a pré-visualização de dois tipos de artefatos: *soundfonts* e arquivos para o *Guitarix*. A implementação e o funcionamento das duas formas de pré-visualização são bastante distintos. No primeiro, o áudio é gerado, no computador do usuário. Enquanto que, no segundo, o processamento do áudio é realizado remotamente: o áudio do usuário é enviado ao servidor, que o processa e após retorna ao cliente.

O cliente consiste em uma aplicação *web*, na qual o usuário pode testar, interativamente, dois tipos de artefatos. É possível experimentá-los utilizando como fonte de entrada um instrumento musical, como guitarra ou um dispositivo MIDI.

O servidor, por sua vez, é o responsável por manter e fornecer ao cliente todos os arquivos, referentes a artefatos, necessários para a pré-visualização. Esses arquivos incluem listas de artefatos disponíveis e arquivos de áudio (no caso de pré-visualização de *soundfonts*). Ademais, na pré-visualização de artefatos para *Guitarix*, o servidor também desempenha a função de processamento do áudio.

Figura 5.1: Arquitetura



## 5.1 Cliente

Nesta seção, primeiramente são descritos o funcionamento básico da interface e a tela inicial de seleção de instrumento. Em seguida, são apresentadas as telas referentes aos dois instrumentos disponíveis: Guitarra / Baixo e MIDI.

### 5.1.1 Interface

O cliente é uma aplicação web do tipo *Single-page Application* (MIKOWSKI; POWELL, 2013). Portanto, ela se comporta como uma página única que, à medida que o usuário interage, carrega e modifica os componentes exibidos na tela, ao invés de carregar e renderizar uma página inteira nova a cada vez. Assim, possibilita-se aos usuários uma experiência muito mais fluída e rápida.

Esse comportamento deve-se ao uso das bibliotecas *javascript React* (REACT..., 2018a) e *React Router* (REACT..., 2018). A primeira é a responsável por, durante a execução, remontar apenas os elementos do DOM que devem ser alterados (REACT..., 2018c). Do ponto de vista de código, a utilização dessa biblioteca também é muito benéfica, pois ela é pensada e oferece mecanismos de comunicação simples e consistentes para aplicações organizadas em muitos componentes. Desse modo, sua utilização favorece o desenvolvimento de códigos modulares com baixo acoplamento e alta coesão, características muito desejadas (MEYER, 1988).

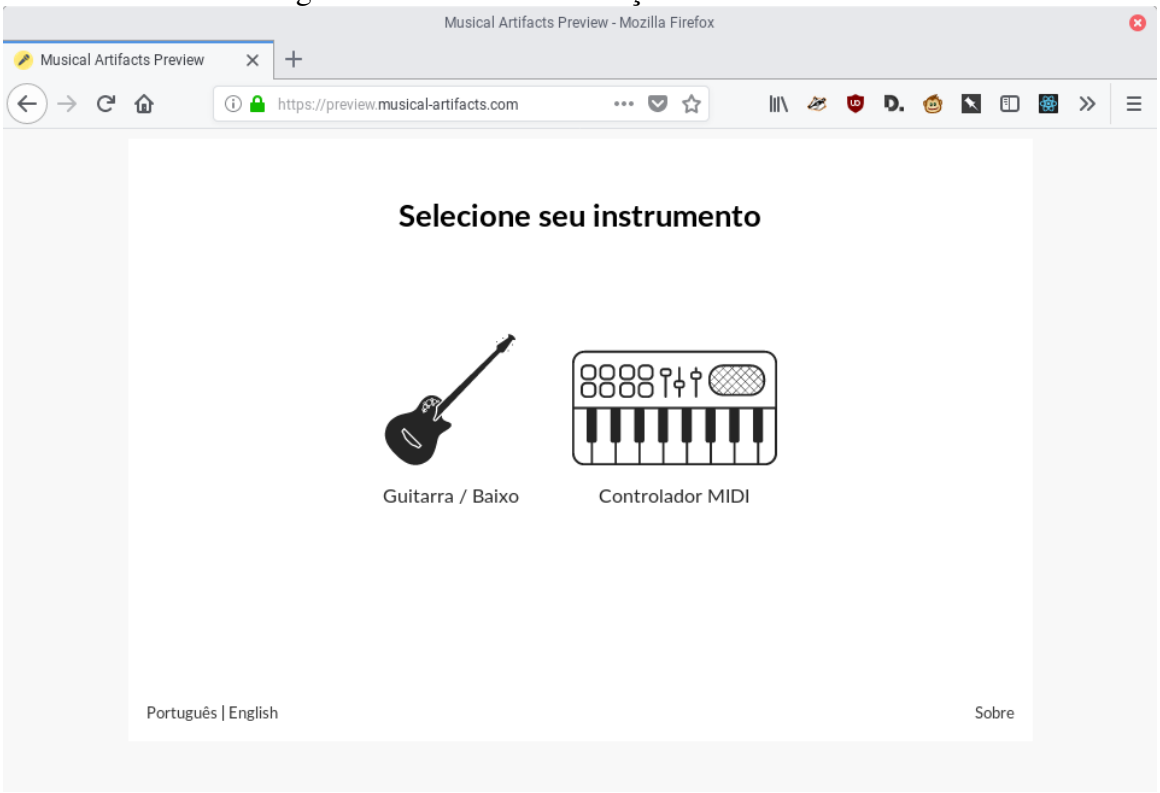
Essa característica propicia a reutilização de componentes. Como o *React* é amplamente utilizado (BUILT..., 2018), é possível beneficiar-se da imensa quantidade de componentes criados e disponibilizados por outros desenvolvedores. Ademais, a biblioteca é capaz de gerar *builds* de produção, que consistem, basicamente, em códigos bastante otimizados para serem usados em produção (REACT..., 2018b).

A interface do cliente é responsiva e pode, portanto, ser utilizada satisfatoriamente tanto em dispositivos móveis como em *desktops*. Pode-se dizer que a aplicação possui três modelos de página: Página Inicial / Seleção de Instrumento; Sobre; e Pré-visualização dos Artefatos.

A figura 5.2 demonstra a página inicial. Nessa tela, o usuário define o tipo de artefato que deseja pré-visualizar a partir da seleção de seu instrumento: Guitarra / Baixo, para testar os artefatos para *Guitarix*; Controlador MIDI, para testar os *soundfonts*.

Após selecionar um instrumento, a página de pré-visualização relativa a ele ar-

Figura 5.2: Tela inicial: seleção de instrumento




tefatos é carregada. O conteúdo dessa página, variável de acordo com o instrumento escolhido, é descrito nas subseções 5.1.2 e 5.1.3. Por fim, a página *Sobre* apenas contém informações e créditos do projeto.

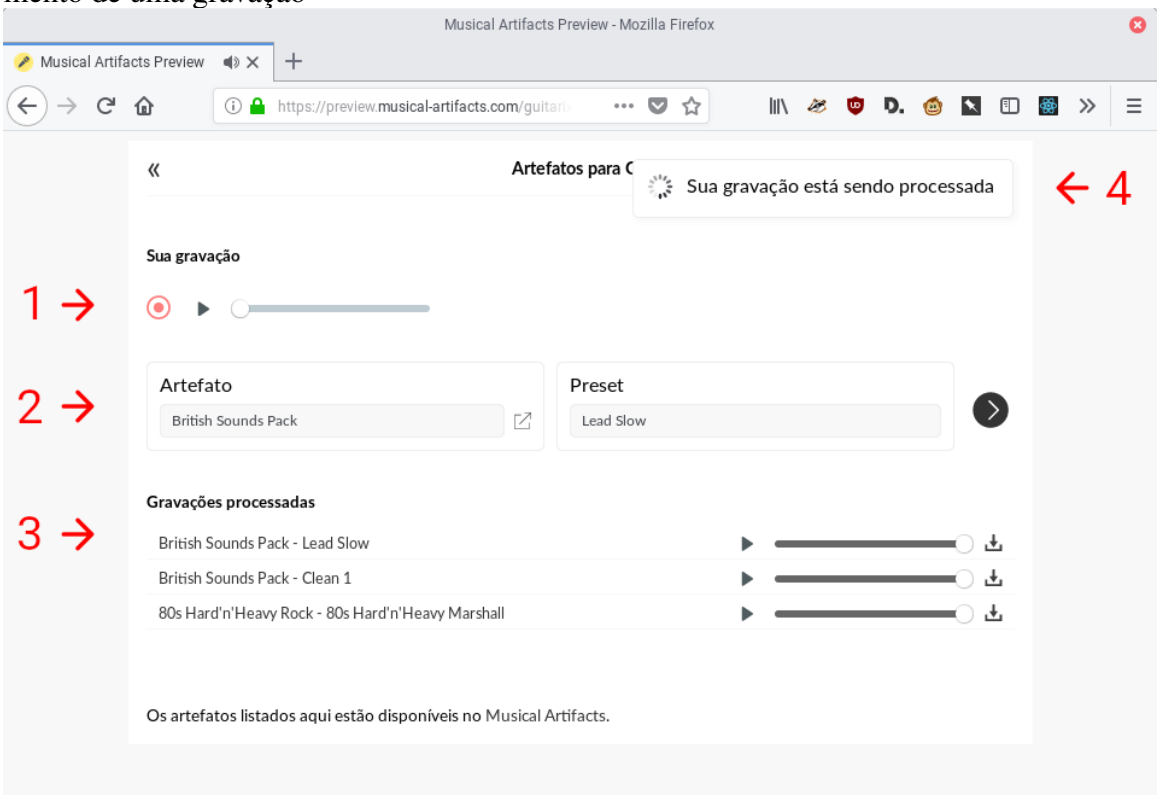
Visto que o Musical Artifacts é um site internacional, acessado por pessoas de dezenas de países, essa interface também possui suporte à internacionalização. Na tela inicial, no canto inferior esquerdo, há uma seleção de idioma, por meio da qual o usuário pode alternar entre Inglês e Português.

### 5.1.2 Instrumento: Guitarra / Baixo

Quando o usuário escolhe *Guitarra / Baixo* como seu instrumento, a tela de pré-visualização de artefatos para o *Guitarix* é carregada. Na pré-visualização desse tipo de artefato, o usuário realiza, pelo navegador, uma pequena gravação de sua guitarra ou seu baixo). Em seguida, a gravação é enviada ao servidor, que aplica a ela o artefato escolhido e, por fim, retorna o áudio resultante.

A figura 5.3 ilustra a página no momento em que uma gravação está sendo processada. Em 1 está a área de gravação. Quando o usuário pressiona o botão vermelho,

Figura 5.3: Interface da pré-visualização de artefatos para Guitarix, durante o processamento de uma gravação




o navegador pergunta qual dispositivo de entrada de áudio deve ser gravado (vide figura 3.6). A gravação é iniciada assim que um dispositivo é selecionado. Para encerrá-la, deve-se clicar novamente no botão vermelho.

Uma vez finalizada a gravação, é possível ouvi-lá no *player*, ou realizar uma nova gravação clicando no botão vermelho e repetindo o processo. Essa gravação é efetuada por meio da *MediaStream Recording API* e, portanto, só está disponível nos navegadores que a implementam, conforme descrito na subseção 3.4.1.

Em 2 está a seleção do artefato a ser testado. Conforme visto na subseção 3.2.1, cada artefato para o *Guitarix* consiste em um banco de *presets*. Assim, na primeira caixa de seleção o usuário escolhe o artefato a ser testado. Após, a segunda caixa de seleção é atualizada com as opções de *preset* que fazem parte desse artefato. Depois de selecionar o artefato e o *preset*, o usuário pode solicitar o processamento de sua gravação clicando no botão preto, à direita.

Ao pressionar o botão, uma requisição de processamento, contendo a gravação do usuário e a informação sobre artefato e *preset* escolhidos, é enviada ao servidor. Caso existam outras requisições à frente na fila de processamento, a área de notificações (4) exibe uma mensagem informando a posição do usuário na fila. Assim que o processa-

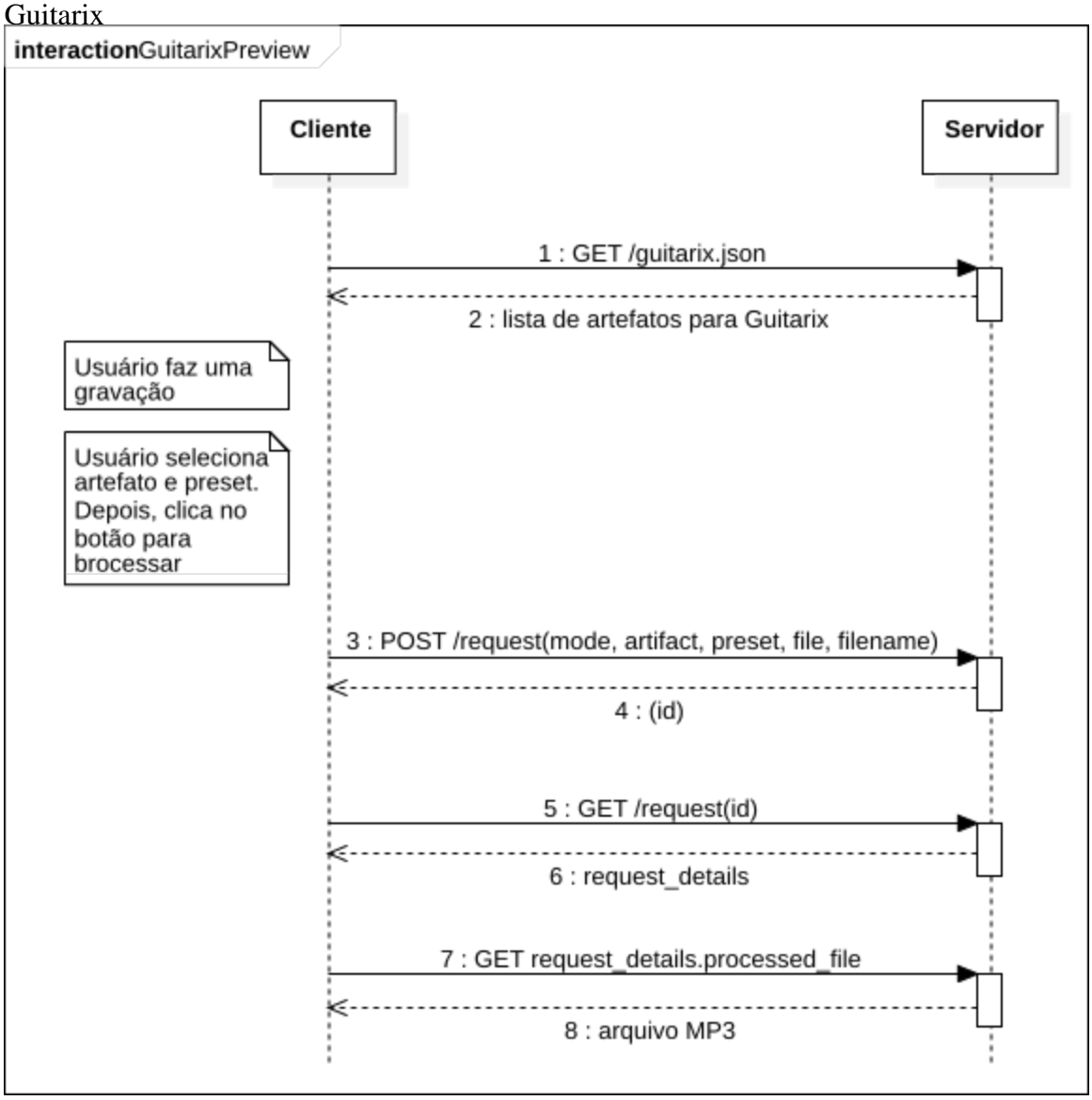
mento é iniciado, a notificação passa a exibir a mensagem vista na figura.

Uma vez finalizado o processamento, o áudio resultante aparece na lista de gravações processadas (3). Cada item dessa lista possui um tocador, a partir do qual o usuário pode ouví-lo, e um botão para baixá-lo. Pode-se, também, clicar em cima do nome do artefato para visualizar sua página no Musical Artifacts.

No rodapé, há um *link* que aponta para a página do Musical Artifacts que contém os artefatos disponíveis nessa pré-visualização. Também, no canto superior esquerdo, na região de fundo branco, há um botão para retornar à página de seleção de instrumento.

A comunicação entre cliente e servidor, na pré-visualização deste tipo de artefato, é exemplificada no diagrama 5.6. Na primeira chamada, efetuada durante o carregamento

Figura 5.4: Comunicação entre cliente e servidor na pré-visualização de artefatos para Guitarix



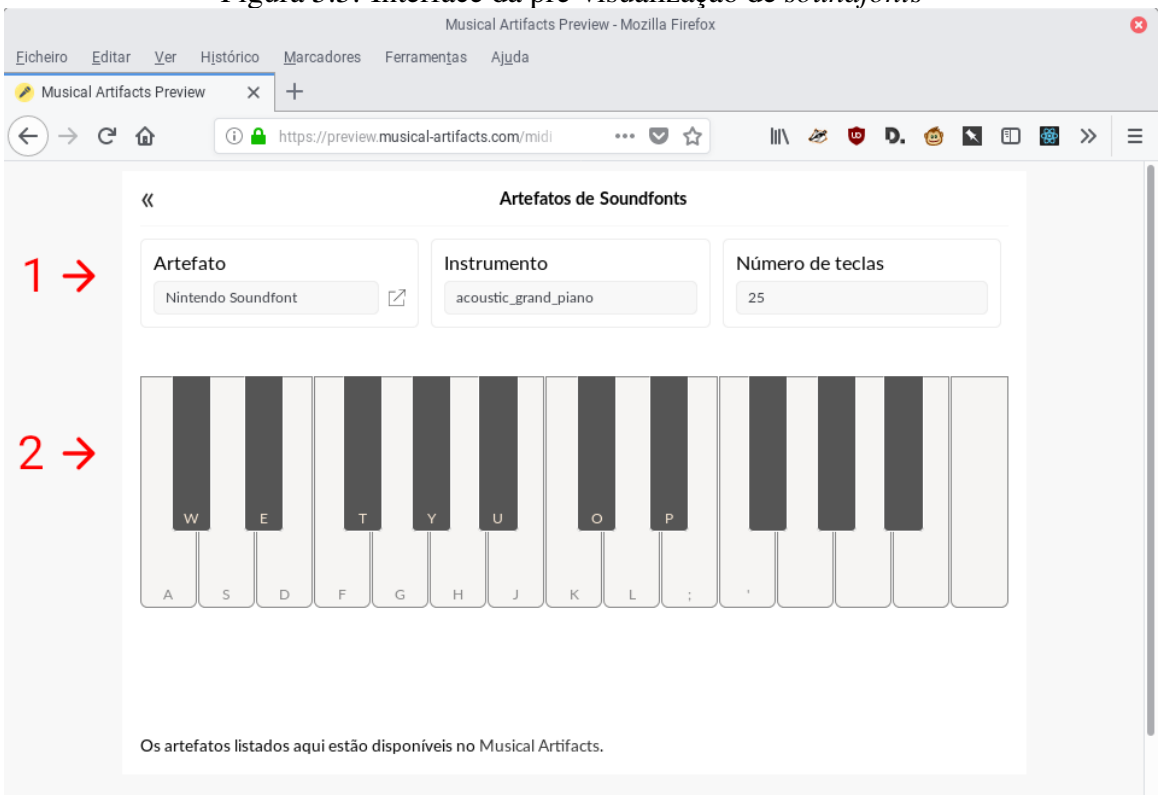
da tela de pré-visualização, o cliente requisita ao servidor a lista de artefatos disponíveis. A segunda chamada ocorre quando o usuário clica no botão para processar uma gravação. A partir de então, por meio da terceira chamada, o cliente continuamente verifica o *status* do processamento, até que ele seja finalizado. Após isso, uma última chamada é realizada a fim de obter o arquivo de áudio resultante.

### 5.1.3 Instrumento: Controlador MIDI

Quando o usuário escolhe *Controlador MIDI* como seu instrumento, a tela de pré-visualização de *soundfonts* é carregada. Esses artefatos correspondem a cerca de 40% do total de arquivos disponíveis no site. Ao contrário da pré-visualização de artefatos para *Guitarix*, nesta o som é produzido localmente, na máquina do usuário.

A base da interface é a mesma da seção anterior, portanto o botão de voltar à tela inicial e o *link* para a página do Musical Artifacts com os artefatos, no rodapé, possuem o mesmo comportamento. Na figura 5.5, similarmente, em 1 há uma caixa para a seleção do artefato a ser testado. Ao selecionar um artefato, a caixa de seleção de instrumento, ao lado, é atualizada com as opções de instrumento do artefato. Adicionalmente, nessa

Figura 5.5: Interface da pré-visualização de *soundfonts*



tela há uma terceiro campo, que permite ao usuário escolher o tamanho do piano exibido abaixo (2). Pode-se optar entre 25, 49 e 88 teclas.

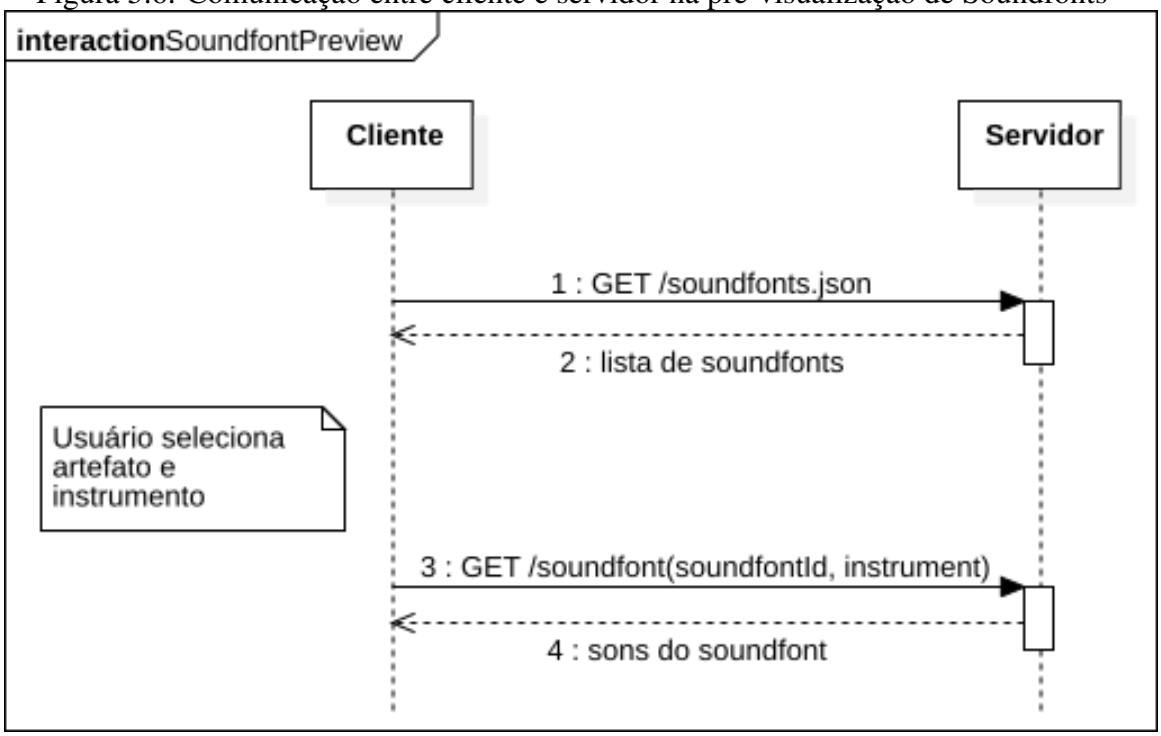
Após selecionar um artefato e um instrumento, o arquivo contendo todas as gravações de todas suas notas musicais é baixado do servidor. A partir de então, cada tecla do piano é associada ao som de sua respectiva nota.

O usuário pode, então, utilizar o piano para testar o artefato. No computador, é possível controlá-lo pelo *mouse* ou pelo *teclado* - de acordo com as letras exibidas em cima de cada tecla do piano. Em dispositivos com tela de toque, a experiência torna-se ainda mais interessante, pois há suporte para *multitouch*.

Ademais, em navegadores com suporte a *Web MIDI API* (vide 3.4.2), é possível utilizar um dispositivo MIDI como entrada. Em termos de usabilidade, essa é a opção mais interessante, pois, além de permitir a utilização de um instrumento confortável com o qual o usuário já está acostumado, beneficia-se dos níveis de intensidade oferecidos por esse tipo de dispositivo.

O diagrama 5.6 exemplifica a comunicação entre cliente e servidor nessa forma de pré-visualização. A primeira chamada, que ocorre assim que a tela de pré-visualização é carregada, requisita ao servidor a lista de artefatos disponíveis. Após as seleções de artefato e instrumento, uma segunda chamada, para obter os arquivos sonoros, é realizada.

Figura 5.6: Comunicação entre cliente e servidor na pré-visualização de Soundfonts



## 5.2 Servidor

Conforme a figura 5.1, o servidor está dividido em 4 componentes: Banco de Dados; API; ManageArtifacts; e ProcessAudio. O funcionamento de cada um é descrito a seguir.

### 5.2.1 Banco de Dados

Para manter o registro de seus artefatos, o servidor utiliza um banco de dados SQLITE (SQLITE..., 2018). A manipulação desse banco é realizada somente pelo *ManageArtifacts*, componente responsável por atualizar e configurar os artefatos localmente.

Segundo (APPROPRIATE..., 2018), o SQLITE é um banco de dados desenvolvido para oferecer armazenamento de dados local, ao contrário de DBMSs como MySQL (MYSQL, 2018) e PostgreSQL (POSTGRESQL, 2018), que seguem um modelo cliente-servidor. Enquanto esses objetivam escalabilidade, concorrência, centralização e controle, as características buscadas pelo SQLITE são economia, eficiência, confiabilidade, independência e simplicidade. Um possível problema na utilização do SQLITE é que seu desempenho é equivalente a operações de leitura no disco.

No caso deste trabalho, há um único componente (*ManageArtifacts*) que manipula o banco de dados, e ele não realiza operações concorrentes. Ademais, esse componente realiza operações de escrita e leitura no banco de dados somente quando os artefatos são atualizados, o que acontece poucas vezes por dia. Por esses motivos, escalabilidade e concorrência não são características necessárias à tecnologia de banco de dados escolhidos. De acordo com (APPROPRIATE..., 2018), essa é uma situação em que a escolha do SQLITE é adequada.

Na figura 5.1, o banco de dados aparece conectado à API. Esta é uma ligação virtual e possui apenas fins didáticos - ela representa somente o relacionamento entre os dados. O único momento em que o *ManageArtifacts* realiza operações de escrita no banco de dados é quando os artefatos do servidor são atualizados. No final desse processo, por motivos de otimização, são gerados dois arquivos JSON, um para cada tipo de artefato, contendo a lista dos artefatos registrados no banco de dados. Assim, evita-se que a API realize consultas SQL desnecessárias e facilita-se que esses arquivos estáticos sejam mantidos em *cache*.

O esquema da tabela desse banco de dados está descrito na tabela 5.1. Cada re-

Tabela 5.1: Esquema da tabela *artifacts* do Banco de Dados

| *Coluna* | *Tipo* | *Descrição* |
|---|---|---|
| id | INTEGER PRIMARY KEY AUTOINCREMENT | ID |
| ma_id | INTEGER | ID do artefato no MA |
| name | TEXT | Nome do artefato |
| file_hash | TEXT | *Hash* do arquivo |
| options | TEXT | Lista com presets (no caso de guitarix) ou instrumentos (soundfonts) |
| filetype | TEXT | Tipo do arquivo (*guitarix* ou *soundfont*) |
| updated_at | DATE | Data de atualização do arquivo local |

gistro do banco se refere a um artefato disponível no servidor. A coluna *id* armazena um identificador único, que possui a função de chave primária. A coluna *ma_id*, por sua vez, mantém o identificador único do artefato no Musical Artifacts.

As colunas *name* e *file_hash* referem-se ao nome e ao *hash* do arquivo, respectivamente, e são obtidas na consulta à API do Musical Artifacts. A coluna *options* possui dois usos distintos, de acordo com o tipo do artefato: para um arquivo do *Guitarix* é armazenada uma lista com seus *presets*; para um *soundfont* é armazenada a lista de seus instrumentos.

A coluna *filetype* registra o tipo do artefato - (*guitarix* ou *soundfont*). A última coluna, *updated_at*, mantém registro da data em que o artefato foi atualizado no servidor pela última vez.

### 5.2.2 ManageArtifacts

O gerenciamento de artefatos do servidor é realizado pelo utilitário ManageArtifacts. Este é o responsável pela comunicação com a API do Musical Artifacts e pela atualização dos artefatos no servidor. Além de baixar e configurar os novos artefatos e atualizar os existentes, ele gerencia o banco de dados que mantém o registro dos artefatos. Ademais, também gera os arquivos JSON, com as listas de artefatos, que são posteriormente disponibilizados pela API.

Esse utilitário possui quatro modos de operação, definido de acordo com o parâmetro passado na execução. Os modos existentes são: *create_database*; *list*; *update_guitarix*; e *update_soundfonts*.

#### *5.2.2.1 Modo create_database*

Este modo é o responsável por criar o banco de dados. Por esse motivo, deve ser executado somente na configuração inicial do servidor. Ao utilizá-lo, é criado o banco de dados *db.sqlite*, com uma única tabela *artifacts*, de acordo com o esquema da tabela 5.1.

#### *5.2.2.2 Modo list*

Este modo exibe uma listagem dos artefatos existentes no banco de dados. Esses são os artefatos já configurados e, portanto, disponíveis para o cliente.

#### *5.2.2.3 Modo update_guitarix*

Este modo realiza a atualização dos artefatos para *Guitarix*. Inicialmente, ele requisita à API do Musical Artifacts o conjunto de artefatos cujo tipo de arquivo é *gx*. Em seguida, verifica quais desses artefatos ainda não estão no banco de dados e, portanto, devem ser instalados. Também verifica quais já estão mas possuem *hash* diferente - esses devem ser atualizados.

Ambos, então, são baixados e salvos no local apropriado, seguindo uma nomenclatura específica - no caso do artefato de *ID* 123, */.config/guitarix/banks/ma-123*. Em seguida, atualiza-se o banco de dados para que ele reflita os artefatos disponíveis no servidor.

Por fim, o arquivo *guitarix.json*, contendo todas as entradas no banco de dados referentes aos artefatos para *Guitarix* é gerado. Esse arquivo é posteriormente fornecido pela API ao cliente.

#### *5.2.2.4 Modo update_soundfonts*

Este modo realiza a atualização dos artefatos do tipo *soundfont*. Similarmente ao *update_guitarix*, inicialmente é requisitado à API do Musical Artifacts o conjunto de artefatos de formato *sf2*. Após verificar quais desses são novos ou devem ser atualizados, eles são baixados.

Em seguida, para cada um dos artefatos, executa-se o *SoundfontBuilder*. Esse é um *script* em Ruby utilizado para extrair os sons (de todos instrumentos) dos *soundfonts*. Seu funcionamento consiste em, basicamente, utilizar o *FluidSynth* (FLUISYNTH, 2018) para gravar o som produzido pela execução de cada nota por três segundos, obtendo como

resultado, para cada uma delas, um arquivo *wav*.

Uma vez finalizado esse processo, os arquivos obtidos são comprimidos para *MP3* e reunidos em um único arquivo *JavaScript*. Esse arquivo mantém, para cada nota musical, uma variável que armazena seu som codificado em *base64* (JOSEFSSON, 2006).

Por fim, similarmente ao processo dos artefatos para *Guitarix*, é gerado o arquivo *soundfonts.json*. Esse arquivo contém todas as entradas no banco de dados de artefatos do tipo *soundfont*.

### 5.2.3 API

O servidor executa uma aplicação *Node.JS* (NODE. . . , 2018) que provê uma API. Essa API é consumida pelo cliente e possui duas funções. A primeira é servir ao cliente as informações referentes aos artefatos disponíveis no servidor: os arquivos *guitarix.json*, *soundfonts.json*) e os arquivos com sons extraídos de cada *soundfont*.

Sua segunda função é manter e gerenciar uma fila de requisições de processamento de áudio, referente à pré-visualização de artefatos para *Guitarix*. A capacidade de processamento dessa fila é de uma requisição por vez. Portanto, sempre que há algum item na fila o componente *ProcessAudio* é requisitado para processá-lo.

A seguir estão descritas as rotas da API. As duas primeiras referem-se aos arquivos estáticos, e as seguintes às requisições de processamento de áudio.

#### *5.2.3.1 GET /guitarix.json*

Esta rota retorna um arquivo JSON contendo a lista de artefatos para Guitarix. Cada artefato possui três campos: *ma_id*, o identificador do artefato no Musical Artifacts; *name*, o nome do artefato; *presets*, uma lista dos presets disponíveis no artefato.

Figura 5.7: Exemplo de retorno da rota *GET /guitarix.json*

```
[
    {
        "ma_id": 578,
        "name": "Jcm Style",
        "presets": [
            "jcm style",
            "jcm style 2",
            "Jcm style 3",
            "jcm style 4",
            "jcm style 5",
            "jcm style 6"
        ]
    },
    {
        "ma_id": 572,
        "name": "Guitarix hi-gain  sound preset",
        "presets": [
            "jcm1",
            "jcm2",
            "jcm3",
            "jcm3 lead",
            "jcm4",
            "jcm5",
            "pure jcm"
        ]
    }
]
```

#### *5.2.3.2 GET /soundfonts.json*

Similar à rota anterior, esta retorna a lista de artefatos do tipo *Soundfont*. A diferença está em que, no lugar de *presets*, há *instruments*. Esse campo indica os instrumentos existentes no *soundfont*. Exemplo:

Figura 5.8: Exemplo de retorno da rota *GET /soundfonts.json*

```
[{
    "ma_id": 602,
    "name": "Soundfont test 1",
    "instruments": ["000 - instrument one", "012 - another instrument"]
}, {
    "ma_id": 603,
    "name": "Soundfont test 2",
    "instruments": ["030 - instrument example", "100 - instrument 2nd
    example"]
}]
```

#### *5.2.3.3 POST /request*

Esta rota é utilizada para solicitar o processamento de um arquivo de áudio. Seus parâmetros são:

- *mode* - Modo de processamento. No momento a única opção implementada é *guitar*
- *artifact* - ID do artefato a ser utilizado
- *preset* - *Preset* a ser utilizado
- *file* - Arquivo de áudio a ser processado, em formato binário (BLOB)
- *filename* - Parâmetro opcional. Indica o nome com que o arquivo foi salva no servidor anteriormente

Os três primeiros parâmetros são obrigatórios. No caso de o arquivo já ter sido enviado ao servidor anteriormente, o parâmetro *filename* é usado alternativamente ao *file*, e o arquivo não é reenviado.

O retorno dessa rota é um objeto com dois atributos: *id*, um UUID para a requisição de processamento, a ser utilizado posteriormente para consultar seu estado; *success*, um campo booleano que indica se a requisição foi realizada com sucesso.

Figura 5.9: Exemplo de retorno da rota *POST /request*

```
{
    "id": "ae051ad5-d22c-4353-90a7-28924d2571d3",
    "success": true
}
```

#### *5.2.3.4 GET /request/:id*

Retorna dados sobre um pedido de processamento, de acordo com o parâmetro *id* fornecido.

Figura 5.10: Exemplo de retorno da rota *GET /request/:id*

```
{
    "mode": "guitarix",
    "status": "processing",
    "artifact": "465",
    "preset": "80s Hard'n'Heavy Marshall",
    "filename": "738f87c5348c2e012faa5ab29a725249"
}
```

A maior parte dos atributos são os que foram enviados pelo cliente na requisição inicial, com exceção de quatro:

- *filename* - Nome utilizado no servidor para armazenar o áudio fornecido pelo usuário. É enviado nas requisições seguintes para evitar o reenvio da mesma gravação.
- *status* - Status da requisição. Pode ser: *queue* quando na fila; *processing* quando em processamento; *done* após finalizado o processamento; *error* em caso de ocorrência de algum erro.
- *processed_file* - URL para o arquivo de áudio processado. Presente apenas quando o *status* da requisição é *done*.
- *position_in_queue* - Posição da requisição na fila. Existente somente quando o *status* é *queue*.

#### *5.2.3.5 GET /requests*

Similar à rota anterior, mas o JSON retornado contém todos os pedidos de processamento que estão em memória - portanto, desde a inicialização da aplicação.

#### *5.2.3.6 GET /soundfont/:id/:instrument*

Retorna um arquivo que contém os sons de todas as notas musicais de um determinado instrumento de um *soundfont*. O parâmetro *id* refere-se ao identificador do artefato no MA, e o *instrument* ao número do instrumento (de 0 a 127) no *soundfont*.

### 5.2.4 ProcessAudio

O *ProcessAudio* é o componente que, junto à API, fornece à solução desenvolvida o caráter de serviço de processamento de áudio. Sua função é, a partir de um arquivo de áudio e de um artefato para Guitarix fornecidos como entrada, obter o áudio resultante da aplicação do artefato à gravação e retorná-lo.

#### *5.2.4.1 Softwares utilizados*

A função desse componente é realizada por meio de controle e utilização de quatro *softwares*, indicados na figura 5.1: JACK, Guitarix, file2jack (JACK-FILE..., 2018) e jack_capture (JACK-CAPTURE..., 2018). Esses *softwares* foram executados em ambiente GNU/Linux - Debian 9 stretch (DEBIAN..., 2018) no ambiente de desenvolvimento e Ubuntu 18.04 LTS (UBUNTU, 2018) no de produção.

Com exceção do file2jack, todos os *softwares* são obtidos do repositório do *KXStudio* (KXSTUDIO, 2018). Esse é um projeto que mantém pacotes com as últimas versões de *softwares* de áudio e outros pacotes que não fazem parte dos repositórios oficiais das distribuições.

O file2jack e o jack_capture são dois softwares que suportam o JACK e são utilizados para funções opostas. O file2jack é um tocador, enquanto o jack_capture é um gravador. Portanto, o file2jack lê um arquivo de áudio e envia seu fluxo para outro cliente do JACK. O jack_capture, por sua vez, recebe como entrada um fluxo de áudio e o salva em um arquivo de áudio.

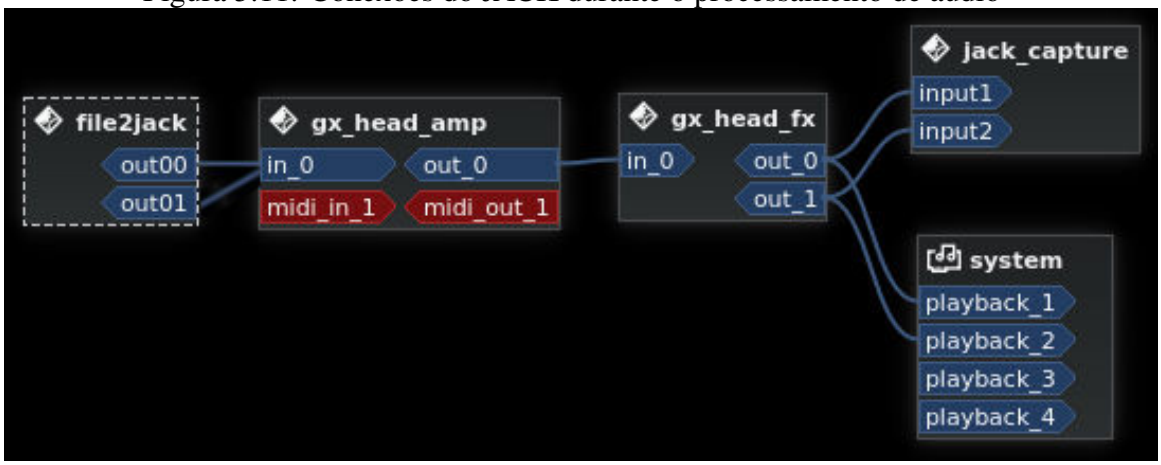
A comunicação com a API do JACK é realizada por meio da biblioteca jackclient-python (JACKCLIENT-PYTHON..., 2018). Essa é uma das poucas bibliotecas, em linguagens diferentes de C e C++, que implementam a comunicação com essa API. Sua existência é a principal razão para o ProcessAudio ser escrito em Python.

#### *5.2.4.2 Processamento de uma requisição*

Todas as requisições de processamento utilizam o JACK e o Guitarix. Por esse motivo, os dois são sempre mantidos em execução. O Guitarix é inicializado com o parâmetro *-N*, que desabilita sua interface gráfica, uma vez que ela não é utilizada.

O ProcessAudio é chamado pela aplicação da API toda vez que há, na fila dela, alguma requisição de processamento de áudio. Ao ser executado, esse componente verifica

Figura 5.11: Conexões do JACK durante o processamento de áúdio




se o JACK e o Guitarix estão em execução. Em caso positivo, é realizada uma chamada JSON-RPC à API do Guitarix (vide figura 3.5) para definir o banco e o *preset* a serem utilizados.

Ao implementarem a *Media Stream Recording*, os navegadores têm liberdade para escolher o *codec* utilizado nas gravações. Diante disso, para evitar problemas, o ProcessAudio converte o arquivo recebido por meio do FFmpeg (FFMPEG, 2018). Assim, após passarem por esta etapa, todos os arquivos possuem o mesmo formato e *codec*.

Em seguida, uma chamada à API do JACK define seu modo de operação como *freewheel*. Depois, o file2jack (com o arquivo de áudio) e o jack_recorder são executados. Nesse momento, todos os clientes do JACK já estão em execução, falta apenas conectá-los. Então, por meio de algumas chamadas à API do JACK, as conexões ilustradas na figura 5.11 são estabelecidas.

Após, uma chamada de *start* à API dá início à execução síncrona de todos os clientes do JACK. Quando o fim do arquivo de áudio é alcançado, uma chamada de *stop* à API encerra a execução de todos os clientes. Neste momento o jack_recorder fecha o arquivo processado, que em seguida é convertido para MP3 (com uma taxa de 192kbps) e retornado à API.

# 6 EXPERIMENTOS E AVALIAÇÃO

Após a implementação da aplicação, foram realizados testes com usuários, a fim de validar a usabilidade da solução. A metodologia e os resultados são detalhados a seguir.

## 6.1 Metodologia

O SUS - System Usabilty Scale (USABILITY.GOV, ) é um método de averiguação do nível de usabilidade de sistemas. Criado em 1986, ele pode ser utilizado para avaliar uma variedade de serviços e produtos incluindo *hardwares* e *softwares*.

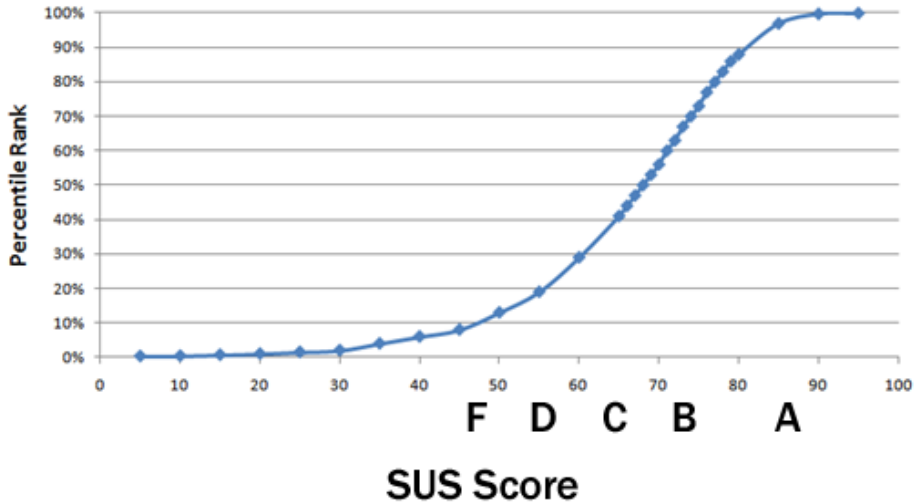
A aplicação do questionário do SUS é simples e rápida. Ele é composto por 10 questões, para as quais os perguntados especificam seu nível de concordância, de acordo com a escala *Likert* - de 1 (discordo completamente) a 5 (concordo completamente).

Para metade das questões, espera-se que a resposta seja a mais próxima de 5 - concordância - , enquanto que para a outra metade espera-se o contrário. O questionário do SUS, propositalmente, apresenta as questões de modo alternado: as questões ímpares são questões de concordância e as pares de discordância.

Para calcular a pontuação de um participante, primeiramente subtrai-se 1 de todas as questões ímpares. Em seguida, para cada uma das questões pares, subtrai-se seu valor de 5. Por fim, basta somar todos os números resultantes e multiplicar o resultado por 2,5 para obter a pontuação.

Embora a pontuação obtida esteja no intervalo de 0 a 100, ela não é uma porcen-

Figura 6.1: Curva de conversão de pontuação SUS para percentil




Fonte: Jeff Sauro (MeasuringU) - https://measuringu.com/sus/

tagem. A interpretação desse resultado deve ser realizada a partir da análise do percentil de uma curva de *ranking* do SUS, como a da figura 6.1. De acordo com pesquisas, uma pontuação acima de 68 é considerada acima da média (USABILITY.GOV, ).

## 6.2 Análise dos Resultados

A avaliação foi realizada com 7 participantes: dois usuários do Musical Artifacts; três pessoas que não conhecem o MA, mas já utilizaram *softwares* de produção musical; duas pessoas que tocam algum instrumento, mas nunca utilizaram um *software* de produção musical.

Primeiramente, cada avaliador realizou um pequeno conjunto de tarefas, que envolveu utilizar a aplicação para pré-visualizar dois artefatos específicos, um *soundfont* e outro para *Guitarix*. Depois, o avaliador respondeu o questionário do SUS.

O mapa de calor da figura 6.2 ilustra a distribuição das respostas obtidas para cada uma das perguntas, enquanto a tabela 6.1 apresenta o enunciado dessas perguntas e sumariza os resultados, seguindo uma escala simplificada para 3 níveis (concordância, neutralidade e discordância).

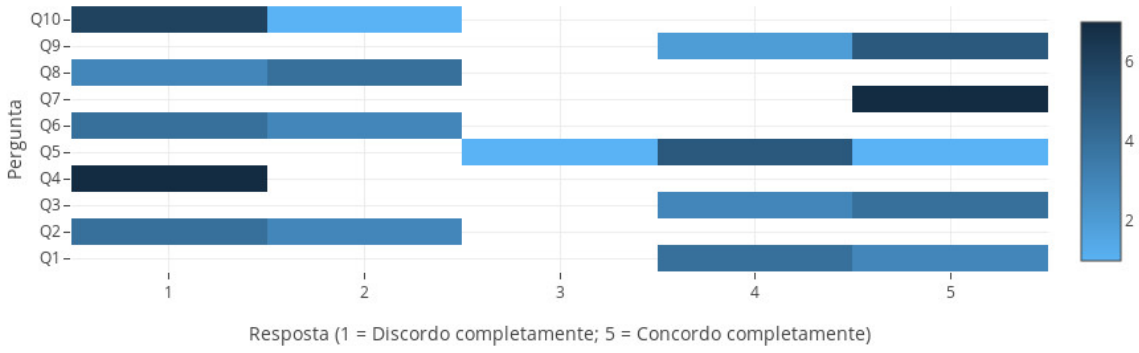
Analisando-se o gráfico e a tabela, percebe-se que houve uma variação pequena entre as respostas dos usuários. A questão de número 5 (*eu acho que as várias funções do sistema estão muito bem integradas*) foi a única que recebeu alguma resposta considerada neutra.

A pontuação média obtida foi de 90,3 pontos, sendo que as pontuações individuais ficaram no intervalo de 82,5 a 95 pontos. Esse é um resultado bastante acima da média. A partir do *ranking* da figura 6.1, verifica-se que essa pontuação equivale a um percentil próximo de 98%, correspondente à nota A. Por conseguinte, conclui-se que a usabilidade do sistema é altamente satisfatória.

Tabela 6.1: Sumário das respostas ao questionário SUS

| *Questão* | *Afirmação* | *Concorda* | *Neutro* | *Discorda* |
|---|---|---|---|---|
| Q1 | Eu acho que gostaria de usar esse sistema com frequência | 100% | 0% | 0% |
| Q2 | Eu acho o sistema desnecessariamente complexo | 0% | 0% | 100% |
| Q3 | Eu achei o sistema fácil de usar | 100% | 0% | 0% |
| Q4 | Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema | 0% | 0% | 100% |
| Q5 | Eu acho que as várias funções do sistema estão muito bem integradas | 85.8% | 14.2% | 0% |
| Q6 | Eu acho que o sistema apresenta muita inconsistência | 0% | 0% | 100% |
| Q7 | Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente | 100% | 0% | 0% |
| Q8 | Eu achei o sistema atrapalhado de usar | 0% | 0% | 100% |
| Q9 | Eu me senti confiante ao usar o sistema | 100% | 0% | 0% |
| Q10 | Eu precisei aprender várias coisas novas antes de conseguir usar o sistema | 0% | 0% | 100% |

Figura 6.2: Mapa de calor com as respostas da aplicação do SUS



# 7 CONCLUSÃO E TRABALHOS FUTUROS

Neste capítulo são apresentadas as conclusões do trabalho. A seção 7.1 expõe os resultados obtidos. A seção 7.2 aborda as limitações da implementação e a seção 7.3 discorre sobre possíveis futuras extensões ao trabalho.

## 7.1 Resultados

Este trabalho surgiu diante de uma dificuldade enfrentada pelos usuários do Musical Artifacts. Para testar e avaliar se um artefato de fato atendia às suas necessidades, era necessário que o usuário baixasse e configurasse o arquivo em um ou mais *softwares* de áudio. O desenvolvimento deste trabalho removeu essa necessidade, de modo que agora é possível pré-visualizar dois tipos de artefatos diretamente no navegador. Como consequência, o processo de experimentação de artefatos tornou-se mais fluído e passou a demandar menos tempo do usuário.

Ademais, a solução implementada está disponível como código aberto e é bastante original. Não foi possível encontrar outro projeto que oferecesse, por exemplo, uma solução similar ao serviço de processamento de áudio desenvolvido que, rodando sobre o JACK em ambiente GNU/Linux, possibilita a utilização de uma série de *softwares* de áudio de qualidade. Acredita-se que isso, somado aos fatos de o código ser bem estruturado, possuir bom nível de modularidade e estar bem documentado, possibilitarão que ele seja utilizado, sem grandes dificuldades, como base para outros projetos.

De acordo com as avaliações realizadas com usuários, pode-se concluir que a experiência com a solução desenvolvida foi bastante positiva. Dada a receptividade da comunidade, acredita-se que a médio prazo seu nível de engajamento aumentará e, por conseguinte, o compartilhamento de artefatos musicais livres será intensificado.

## 7.2 Limitações

Apesar de a solução apresentada ter alcançado ótimos resultados e atingido seus objetivos, ela possui características que podem representar limitações à adaptabilidade e escalabilidade do projeto. Essas características são resultado de decisões de projeto tomadas ao longo do desenvolvimento.

Na pré-visualização de *soundfonts*, o som de cada uma das teclas do piano é obtido a partir da reprodução de arquivos sonoros - previamente extraídos do artefato por meio de um processo de gravação - que possuem duração de três segundos. Por conta disso, após tocar uma nota do piano por mais de três segundos, seu som é interrompido abruptamente. A escolha de limitar a três segundos a duração de cada nota beneficia a maioria dos usuários, pois evita que os arquivos necessários à pré-visualização fiquem pesados demais. No entanto, essa é uma limitação que pode afetar a experiência de usuários que desejem testar o artefato tocando as notas por um longo tempo.

Uma segunda limitação da solução desenvolvida é sua capacidade de processamento simultâneo. Embora o processamento de um arquivo de áudio com 15 segundos de duração demore menos que um segundo - em uma máquina virtual bastante simples, com apenas 1 core -, há um limite de processamento. Mesmo em um processador com diversos núcleos, não é possível processar gravações em paralelo, pois há somente uma instância do JACK e uma do Guitarix em execução. Estima-se que esse gargalo não represente um problema a médio prazo para a pré-visualização de artefatos do MA. Contudo, possíveis soluções baseadas nesta podem vir a enfrentar essa limitação. Para resolvê-la, acredita-se que o caminho seja a execução de múltiplas instâncias desses processos, se possível, ou a distribuição do processamento entre um conjunto de máquinas.

Um artefato para *Guitarix* consiste em uma série de efeitos e, para cada um desses, há diversos parâmetros configuráveis que definem suas características e intensidades. Na solução desenvolvida, não é possível controlar esses parâmetros. Por conseguinte, uma terceira limitação é a impossibilidade de a pré-visualização propiciar ao usuário uma experiência tão rica como a utilização do *Guitarix* (e sua interface gráfica) em seu próprio computador.

## 7.3 Trabalhos futuros

Este trabalho pode ser estendido futuramente de muitas formas. Algumas dessas possibilidades são descritas a seguir.

### 7.3.1 Incorporação ao Musical Artifacts

A solução desenvolvida é uma aplicação à parte do Musical Artifacts e, atualmente, está disponível em um subdomínio próprio. Com o objetivo de melhorar a experiência dos usuários do MA, planeja-se em breve incorporá-la ao Musical Artifacts, de modo que seja possível pré-visualizar cada artefato diretamente em sua página no *site*.

### 7.3.2 Pré-visualização de outros formatos de arquivo

Apesar de a solução desenvolvida contemplar dois dos formatos de arquivos mais populares do Musical Artifacts, ainda há outros formatos de arquivo para os quais se poderia implementar alguma forma de pré-visualização interativa. Exemplos desses artefatos são os arquivos com formato *h2drumkit* e *sfz*.

O primeiro candidato a possuir pré-visualização no navegador é o *h2drumkit*. Esse é um formato de arquivo para o Hydrogen (HYDROGEN, 2018), um *software* de *drum machine* bastante popular. A princípio, pensa-se que a interface dessa pré-visualização poderia ser similar à do Hydrogen.

Um segundo candidato é o *sfz*. Esse é um formato de *soundfont* de texto puro e, portanto, mais aberto que o *sf2*. Estima-se que parcela bastante considerável da implementação dessa pré-visualização seria similar à dos artefatos *sf2*. Por conseguinte, espera-se que sua implementação não apresente grandes dificuldades.

### 7.3.3 Estender e integrar o serviço de processamento de áudio

A solução desenvolvida para a pré-visualização de artefatos para o *Guitarix* incluiu a criação de um serviço de processamento de áudio. Essa aplicação foi construída de maneira bastante modular, visando a ser futuramente estendida e adaptada para outros propósitos.

Uma possível extensão é adicionar outros processadores de efeitos, para desempenhar uma função similar à do *Guitarix*. Ademais, o serviço de processamento pode ser integrado, por meio de sua API, em outras aplicações, como plataformas de prototipação musical- a exemplo do *C.O.D.E.S* (MILETTO et al., 2005).

# REFERÊNCIAS

API Web Audio - MDN. 2018. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/API_Web_Audio>. Acesso em: 5 dez. 2018.

APPROPRIATE Uses For SQLite. 2018. Disponível em: <https://www.sqlite.org/whentouse.html>. Acesso em: 18 nov. 2018.

ASSOCIATION, M. M. **General MIDI 1 and 2 Specification**. 2018. Disponível em: <https://www.midi.org/specifications/category/gm-specifications>. Acesso em: 3 dez. 2018.

BARNETT, J.; CASAS-SANCHEZ, M.; LEITHEAD, T. **MediaStream Recording**. [S.l.], 2017. Https://www.w3.org/TR/2017/WD-mediastream-recording-20170621/.

BENNEMANN, A. **Musical Artifacts Preview**. [S.l.]: GitHub, 2018. <https://github.com/gutobenn/musical-artifacts-preview>.

BERSON, A. **Client-server architecture**. [S.l.]: McGraw-Hill, 1992.

BLASCO, D. G. **Soundfont-player: Quick soundfont loader and player for browser**. [S.l.]: GitHub, 2015. <https://github.com/danigb/soundfont-player>.

BLASCO, D. G. **GuitarStack - Electric guitar effect stack in the browser, using the Web Audio API**. [S.l.]: GitHub, 2016. <https://github.com/lucaong/guitarstack>.

BUILT With - React Usage Statistics. 2018. Disponível em: <https://trends.builtwith.com/javascript/React>. Acesso em: 18 nov. 2018.

CALF Studio Gear. 2018. Disponível em: <https://calf-studio-gear.org/>. Acesso em: 4 dez. 2018.

CAN I Use... MediaRecorder API? 2018. Disponível em: <https://caniuse.com/#feat=mediarecorder>. Acesso em: 2 nov. 2018.

CAN I Use... Web MIDI API? 2018. Disponível em: <https://caniuse.com/#feat=midi>. Acesso em: 2 nov. 2018.

CANN, S. **Cakewalk synthesizers: from presets to power user**. [S.l.]: Nelson Education, 2009.

CANN, S. **How to Make a Noise: Sample-Based Synthesis**. [S.l.]: BookBaby, 2011.

CATIA - KXStudio Applications. 2018. Disponível em: <https://kxstudio.linuxaudio.org/Applications:Catia>. Acesso em: 4 dez. 2018.

CREATIVE Commons - Atribuição 4.0 Internacional. 2018. Disponível em: <https://creativecommons.org/licenses/by/4.0/deed.pt_BR>. Acesso em: 21 nov. 2018.

CREATIVE Commons - Atribuição-CompartilhaIgual 4.0 Internacional. 2018. Disponível em: <https://creativecommons.org/licenses/by-sa/4.0/deed.pt_BR>. Acesso em: 21 nov. 2018.

CREATIVE Commons - CC-0 1.0. 2018. Disponível em: <https://creativecommons.org/publicdomain/zero/1.0/deed.pt_BR>. Acesso em: 21 nov. 2018.

CREATIVE Commons - When we share, everyone wins. 2018. Disponível em: <https://creativecommons.org/>. Acesso em: 21 nov. 2018.

DEBIAN - O Sistema Operacional Universal. 2018. Disponível em: <https://www.debian.org/index.pt.html>. Acesso em: 19 nov. 2018.

ECAD. **O que é Direito Autoral?** 2018. Disponível em: <http://www2.ecad.org.br/pt/direito-autoral/o-que-e-direito-autoral/Paginas/default.aspx>. Acesso em: 3 dez. 2018.

FFMPEG. 2018. Disponível em: <https://www.ffmpeg.org/>. Acesso em: 19 nov. 2018.

FLUISYNTH. 2018. Disponível em: <http://www.fluidsynth.org/>. Acesso em: 18 nov. 2018.

FOUNDATION, F. S. **Linux e GNU**. 2018. Disponível em: <https://www.gnu.org/gnu/linux-and-gnu.html>. Acesso em: 3 dez. 2018.

FOUNDATION, F. S. **O que é o software livre?** 2018. Disponível em: <https://www.gnu.org/philosophy/free-sw.pt-br.html>. Acesso em: 2 dez. 2018.

GUITARIX - GNU/Linux Virtual Amplifier. 2018. Disponível em: <http://guitarix.org/>. Acesso em: 20 nov. 2018.

HYDROGEN. 2018. Disponível em: <http://hydrogen-music.org/>. Acesso em: 4 dez. 2018.

JACK Audio Connection Kit. 2014. Disponível em: <http://jackaudio.org/files/refman.pdf>.

JACK-CAPTURE - A program for recording soundfiles with jack. 2018. Disponível em: <https://github.com/kmatheussen/jack_capture>. Acesso em: 19 nov. 2018.

JACK-FILE - Jack transport-centric utilities for audio playback. 2018. Disponível em: <https://danmbox.github.io/jack-file/>. Acesso em: 19 nov. 2018.

JACKCLIENT-PYTHON - JACK Audio Connection Kit (JACK) Client for Python. 2018. Disponível em: <https://github.com/spatialaudio/jackclient-python/>. Acesso em: 5 dez. 2018.

JOSEFSSON, S. **The Base16, Base32, and Base64 Data Encodings**. [S.l.], 2006. <http://www.rfc-editor.org/rfc/rfc4648.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc4648.txt>.

JSON-RPC. 2018. Disponível em: <https://www.jsonrpc.org/>. Acesso em: 20 nov. 2018.

KALLIOKOSKI, J.; WILSON, C. **Web MIDI API**. [S.l.], 2015. Http://www.w3.org/TR/2015/WD-webmidi-20150317/.

KXSTUDIO. 2018. Disponível em: <https://kxstudio.linuxaudio.org/>. Acesso em: 19 nov. 2018.

LEMOS, R. **Direito, Tecnologia e Cultura**. [S.l.]: Editora FGV, 2005. 211 p.

LINUX Audio Developer's Simple Plugin API. 2018. Disponível em: <https://www.ladspa.org/>. Acesso em: 4 dez. 2018.

LV2. 2018. Disponível em: <http://lv2plug.in/>. Acesso em: 4 dez. 2018.

MEYER, B. **Object-oriented software construction**. [S.l.]: Prentice hall New York, 1988.

MIKOWSKI, M.; POWELL, J. **Single Page Web Applications: JavaScript End-to-end**. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2013. ISBN 1617290750, 9781617290756.

MILETTO, E. M. et al. Codes: Um ambiente para prototipação musical cooperativa baseado na web. **XXXII Seminário Integrado de Software e Hardware, São Leopoldo**, 2005.

MIT License. Disponível em: <https://mit-license.org/>.

MUSICAL Artifacts Preview - Wiki. 2018. Disponível em: <https://github.com/gutobenn/musical-artifacts-preview/wiki>. Acesso em: 2 dez. 2018.

MYSQL. 2018. Disponível em: <https://www.mysql.com/>. Acesso em: 4 dez. 2018.

NODE.JS. 2018. Disponível em: <https://nodejs.org>. Acesso em: 4 dez. 2018.

PEDALS.IO - Guitar effects in the cloud. 2018. Disponível em: <https://pedals.io/>. Acesso em: 2 dez. 2018.

POSTGRESQL. 2018. Disponível em: <https://www.postgresql.org/>. Acesso em: 4 dez. 2018.

PULSEAUDIO. 2018. Disponível em: <https://www.freedesktop.org/wiki/Software/PulseAudio/>. Acesso em: 4 dez. 2018.

REACT - A JavaScript library for building user interfaces. 2018. Disponível em: <https://reactjs.org>. Acesso em: 18 nov. 2018.

REACT - Optimizing Performance - Use the Production Build. 2018. Disponível em: <https://reactjs.org/docs/optimizing-performance.html#use-the-production-build>. Acesso em: 18 nov. 2018.

REACT - Rendering Elements. 2018. Disponível em: <https://reactjs.org/docs/rendering-elements.html>. Acesso em: 18 nov. 2018.

REACT-PIANO: An interactive piano keyboard for React. 2018. Disponível em: <https://github.com/gleitz/MIDI.js/tree/master/soundfont-generator/>. Acesso em: 2 dez. 2018.

REACT Router: Declarative Routing for React.js. 2018. Disponível em: <https://reacttraining.com/react-router>. Acesso em: 1 dez. 2018.

ROSSUM, D.; JOINT, E. The soundfont® 2.0 file format. **Joint E-Mu/Creative Tech Center white paper**, 1995.

SQLITE Home Page. 2018. Disponível em: <https://www.sqlite.org/index.html>. Acesso em: 18 nov. 2018.

UBUNTU. 2018. Disponível em: <https://www.ubuntu.com/>. Acesso em: 19 nov. 2018.

USABILITY.GOV. **System Usability Scale**.

WALLI, S. R. The posix family of standards. **StandardView**, ACM, v. 3, n. 1, p. 11–17, 1995.

ZAWACKI, L. F. **Musical Artifacts**. 2015. Disponível em: <https://musical-artifacts.com>.