UNIVERSIDADE DO SUL DE SANTA CATARINA

GERSON MORAES

SOFTWARE DE GERENCIAMENTO PARA SHOWS MUSICAIS

Palhoça

2015

GERSON MORAES

SOFTWARE DE GERENCIAMENTO PARA SHOWS MUSICAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Prof. Sheila Santisi Travessa, Dra.

Palhoça

2015

GERSON MORAES

SOFTWARE DE GERENCIAMENTO PARA SHOWS MUSICAIS

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina.

Palhoça, 16 de novembro de 2015.

Professora e orientadora Sheila Santisi Travessa, MEng.
Universidade do Sul de Santa Catarina

Prof. Saulo Popov Zambiasi, Dr.
Universidade do Sul de Santa Catarina

Prof. Flávio Ceci, Dr.
Universidade do Sul de Santa Catarina

Este trabalho é dedicado à minha esposa, amiga e companheira, Rosana Tabalipa Moraes, por todo o amor, companheirismo e confiança; à minha família, pelo apoio e compreensão pelos muitos dias de ausência; aos amigos da Unisul, que estavam sempre a postos para ajudar; e aos parceiros e amigos da Banda MagHigh, Nilton Albieri Ferreira, Rodrigo Mussato, Maikel Daian, Daniel Andreazza e André Milaneze, por serem os precursores da idéia.

AGRADECIMENTOS

É difícil dizer algo nesse momento. Uma série de pensamentos surgem ao final da jornada acadêmica, com a conclusão do presente trabalho. E, o mais gratificante, é saber que uma pessoa mais do que especial esteve ao meu lado durante toda essa jornada, acreditando no meu potencial, mesmo quando eu não acreditava, que me trouxe das profundezas do inferno e caminhou comigo, lado a lado, rumo ao paraíso. Essa pessoa é minha esposa, minha amada, minha amiga, minha parceira, minha rosa graciosa, Rosana.

Errar é humano. Confiar em alguém que errou é angelical. E ela foi esse anjo, durante os cinco anos de Unisul. Não teria conseguido, sem esse apoio... fatalmente, teria desistido no meio do percurso. A essa pessoa maravilhosa, o meu amor, a minha alma e a minha vida!

Aos meus pais, José Carlito e Paula, que sempre estiveram ao meu lado, rezaram muito pelo meu sucesso, e compreenderam os meus muitos momentos de ausência. Certamente, as orações surtiram efeito positivo durante esse tempo. Pessoas que, em sua simplicidade, ensinaram-me os valores mais complexos que, infelizmente, o ser humano têm esquecido constantemente. Valores como fé, amor, esperança, humildade, compaixão, trabalho e coragem... sem esses valores, não seria quem sou. A vocês o meu amor, carinho e admiração eterna. Podem ter certeza de que compensarei esses momentos de ausência, a partir de agora!

Aos meus familiares, que foram extremamente importantes ao longo da jornada. Nos momentos mais complicados, sempre tive esse carinho, amor e apoio, que foram uma injeção de ânimo e motivação. Agradeço por todas as dicas, sugestões e opiniões, pois foram extremamente importantes, sendo que, algumas delas, fazem parte do presente projeto. Vocês são incríveis!

Aos amigos da Unisul, pelos momentos de alegria, diversão, aprendizado, amizade e companheirismo durante os cinco anos de universidade. Foram muitas histórias, alguns churrascos, muito código compartilhado e dicas essenciais para a sobrevivência acadêmica. Impossível lembrar nominalmente de todos, mas sintam-se abraçados!

Entretanto, há aquelas pessoas que foram muito especiais durante todos esses anos, que não poderiam ficar de fora dessas páginas: Daiane, Luana, Alcieny, Jonas, Luiz, Vinícius, Guilherme, Daniel, Alexandre, Alex, Francisco, Danyelle, Diego, Pablo, Roberto, Romeu, Eduardo, Lauro, ..., aprendi muito com vocês! Desejo a todos o melhor que a vida puder

oferecer, pois são pessoas incríveis e profissionais excepcionais, que terão uma carreira vitoriosa pela frente!

Aos amigos e amigas que, mesmo distantes, sempre torceram pelo meu sucesso e deram aquele apoio... saibam que são muito importantes na minha vida e que, mesmo distantes, estão sempre perto em meus pensamentos e em meus desejos de paz, amor e sucesso. Contem sempre comigo!

Aos novos amigos e amigas que conheci ao longo desses cinco anos... Henrique, Michelle, Isadora, Silvane, Felipe, Renata, Dra. Vânia, Anne, Marcela, Soraia, Jully, Fernando, Janaína, Renata... a convivência com vocês, independente de tempo, foi primordial para que eu pudesse evoluir profissional e pessoalmente. Dessa forma, impossível não agradecê-los formalmente neste trabalho. Tenham a certeza de que estarei sempre pronto a ajudá-los no que for preciso!

Aos amigos do Tribunal de Justiça, empresa Quality's e Seal Telecom, entre outros muitos com quem tenho convivido durante esses cinco anos... vocês são referência de pessoas e profissionais e são algumas das minhas fontes de inspiração. É uma honra trabalhar e conviver com vocês!

Aos professores com os quais tive aulas na Unisul. Alguns, já saíram, outros, permanecem ..., mas eles jamais deixarão de fazer parte da minha história. Anderson, Sidnei, Orlando, Maureci, Fernanda, Saulo, Luiz Otávio, Márcio, Ricardo, Flávio, Edson, Rafael, Ivo, Richard, Cleiton, Jean, entre tantos outros, compartilharam conhecimento, experiências e profissionalismo. E, mesmo com toda a cobrança por eles imposta, não deixaram de ser pessoas amáveis e cordiais.

À professora Maria Inês, coordenadora do curso de Ciência da Computação e à professora Sheila Santisi Travessa, orientadora deste projeto, que se mostraram sempre empolgadas e interessadas pelo trabalho ora apresentado. Muitas vezes, mostraram-se mais entusiasmadas que eu. Sempre com um sorriso no rosto, e prontas a ajudar, no que fosse preciso. A vocês, um carinho e admiração mais que especial!

Aos professores Saulo e Flávio por aceitarem ser membros da banca examinadora deste projeto. Desde o início, vocês foram a minha primeira opção para a banca examinadora deste trabalho, haja vista a admiração, respeito e amizade que tenho por vocês. Com certeza, vocês são, e sempre serão, duas das minhas maiores referências em termos profissionais.

Não poderia deixar de agradecer a dois grandes amigos, que me apoiaram muito nessa jornada, principalmente, durante o curso do presente projeto. São eles: Marcelino Hable

e Nilton Albieri Ferreira. Dois excelentes músicos e pessoas ainda maiores com as quais aprendi muito.

Marcelino, você sempre foi a minha principal referência musical, e a razão de eu ter me tornado músico. Se o tema escolhido foi este, certamente, foi por causa da sua influência, mesmo que sem querer, para que eu me tornasse o profissional que sou hoje. E, Nilton, você é uma daquelas pessoas com quem se pode contar em todos os momentos, seja para tocar algumas músicas, seja para soldar placas de Arduíno e *shields* MIDI ou para conversar sobre assuntos diversos. A vocês, toda a minha admiração, carinho, respeito e amizade!

Um agradecimento mais do que especial aos amigos e parceiros da Banda MagHigh: Rodrigo Mussato, Maikel Daian, Daniel Andreazza e André Milaneze, por esses dois anos de grandes alegrias, ao som de muito rock! E, um agradecimento ainda maior, por terem sido os precursores da idéia deste trabalho. Se não se lembram, foi por um questionamento do Daniel e, após conversas que surgiram depois desse ensaio, que eu decidi comprar a idéia. Infelizmente, não foi possível a realização do projeto completo, mas, para o próximo ano, a pedaleira Hades entrará, definitivamente, no setup da banda!

A todos aqueles amigos e amigas que, porventura, não tiveram o nome mencionado nesses agradecimentos, não se sintam excluídos do meu carinho, admiração e respeito. Impossível lembrar dos nomes de todos os que passaram pelo meu caminho. Alguns, tão rápido que nem tive tempo de conhecer melhor, mas, que deixaram uma marca especial!

Por fim, um agradecimento aos seres superiores, à natureza, à vida e às forças do universo, que conspiraram, por vezes misteriosamente, ao meu favor, mesmo quando parecia que não. Por vezes, acontecimentos alheios ao meu planejamento ocorreram, e me fizeram recuar ou avançar passos para os quais não havia me preparado, mas que foram extremamente importantes para que eu me tornasse um profissional e uma pessoa melhor. Quero retribuir cada presente que me foi dado, com muito trabalho, amor, humildade, dedicação, carinho, simplicidade e emoção, para ajudar a fazer deste mundo um lugar melhor. Respire fundo e voe alto!

“A sabedoria não se transmite, é preciso que nós a descubramos fazendo uma caminhada que ninguém pode fazer em nosso lugar e que ninguém nos pode evitar, porque a sabedoria é uma maneira de ver as coisas. ” (PROUST, 1920).

RESUMO

Uma das maiores dificuldades para músicos e instrumentistas que utilizam dois ou mais equipamentos é a dificuldade em efetuar a troca simultânea de programações de ambos os equipamentos, em tempo de execução. Com vistas a auxiliar nessa questão, foi desenvolvido o presente projeto, que consiste em um software de gerenciamento para shows musicais. Esse software permite o gerenciamento de bandas, músicas, programações, equipamentos, blocos de músicas, *setlists* e shows. Ainda, é possível avançar e retroceder músicas, blocos de músicas e programações, em tempo real. Durante a execução de um show, o artista pode exibir ou ocultar a letra de uma música, o que facilita muito àqueles músicos que são vocalistas, pois evita o uso de dois ou mais programas ou arquivos simultâneos, como, por exemplo, uma planilha para visualizar o repertório e um arquivo de PDF para visualizar as letras. No presente software, a letra já fica carregada com a música atual, bastando um clique para exibi-la ou ocultá-la. Após a definição do tema e dos objetivos gerais e específicos, bem como, a definição do problema e a justificativa, foi realizada uma pesquisa bibliográfica sobre os assuntos envolvidos. Entre as pesquisas, constam: desenvolvimento de software, engenharia de software, linguagens de programação, banco de dados, fundamentos de programação web, pedaleira controladora, tecnologia musical, comunicação serial e protocolo MIDI. Depois da definição da metodologia utilizada, foi feita a modelagem da proposta do projeto, bem como, apresentado o desenvolvimento da solução proposta. Ao final, foram apresentadas as conclusões, as dificuldades encontradas e os trabalhos futuros. Concluiu-se que o software desenvolvido atendeu às expectativas inicialmente propostas.

Palavras-chave: Tecnologia musical, programação web, MIDI, comunicação serial, pedaleira controladora.

ABSTRACT, RÉSUMÉ OU RESUMEN

One of the biggest difficulties for musicians who use two or more musical equipments or instruments is the difficulty in effecting the simultaneous exchange of programs of both devices at runtime. In order to assist in this issue, this project was developed, which consists of a management software for musical shows. This software allows the management of bands, songs, programs, equipment, music blocks, setlists and shows. It's possible fast forward and rewind songs, music blocks and programs in real time. During the execution of a show, the artist can show or hide the lyrics of a song, which makes it much easier for those musicians who are vocalists, because it avoids the use of two or more programs or concurrent file. For example, a spreadsheet to view the repertory, and a PDF file to view the lyrics. In this software, the lyric is already loaded with the current song, and simply just a click to display or hide it. After the theme definition and the general and specific objectives as well, the problem definition and justification, a bibliographic research on the issues involved was held. Between researches, included: software development, software engineering, programming languages, databases, web programming basics, controller pedalboard, music technology, serial communication and MIDI protocol. After the definition of the methodology, the modeling of the project proposal was designed and the development of the proposed solution was executed Finally, the results, the difficulties encountered and the future works were presented. It was concluded that the software developed has met the expectations initially proposed.

Keywords: Musical technology, web programming, MIDI, serial communication, controller pedalboard.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

GLOSSÁRIO

ABSTRAÇÃO: processo no qual somente os atributos particulares de um objeto são considerados, não se levando em conta os atributos comuns ao grupo de entidades.
AMOSTRAGEM: processo de leitura de dados de valores de tensão a cada intervalo de tempo.
AMPLITUDE: variação de uma onda sonora, entre picos positivos e negativos.
ARRANJADOR: tipo de teclado que possui ritmos e timbres.
ATRIBUTO: característica associada a uma entidade ou grupo de entidades.

BIBLIOTECA: trechos de código que executam funções específicas.
BOOLEAN: tipo de operador lógico, que assume os valores verdadeiro (*true*) ou falso (*false*).

CLASSE: estrutura que agrupa um conjunto de objetos com características similares, definindo atributos e comportamentos (métodos).
CONTAINER: tipo de objeto que engloba outros objetos, permitindo operações dinâmicas (em tempo de execução).
CONTROLADOR: tipo de teclado que não possui timbres internos, mas que permite o controle de outros equipamentos via tecnologia MIDI.

DLS (*DownLoadable Sounds*): formato de arquivo para instrumentos musicais, que engloba especificações detalhadas de como instrumentos musicais vão renderizar os sons, através da tecnologia MIDI.

ENTIDADE: representação lógica de um objeto ou fenômeno existente no mundo real.

FILTRAGEM HARMÔNICA: ação que possibilita a eliminação de ruídos ou distorções em ondas sonoras, através de um conjunto de componentes elétricos (capacitores, resistores indutores e varistores, por exemplo).
FIREWIRE: tipo de interface serial, utilizada para equipamentos de áudio e vídeo que necessita de comunicação em alta velocidade e em tempo real.
FRAMEWORK: conjunto de códigos que provém funcionalidades específicas a um sistema.
FREQUÊNCIA: quantidade de ciclos (ocorrências) de um evento em determinado período de tempo.

FULL-DUPLEX: tipo de configuração de transmissão de dados bidirecional, no qual esses dados podem ser enviados e recebidos ao mesmo tempo por ambas as pontas.

HALF-DUPLEX: tipo de configuração de transmissão de dados unidirecional, no qual esses dados somente podem ser ou enviados ou recebidos, um de cada vez.

HERANÇA MÚLTIPLA: possibilidade que permite a uma classe filha herdar atributos de uma classe mãe.

HUB: espécie de adaptador que possui duas ou mais portas de conexão, permitindo o envio e recebimento de dados para muitos receptores, ao mesmo tempo.

IMPLEMENTAÇÃO: composição ou desenvolvimento de um código, em determinada linguagem de programação, que deverá possuir os módulos básicos para o seu funcionamento.

INJEÇÃO DE DEPENDÊNCIA: técnica utilizada para otimização de códigos, visando a uma melhor manutenção e evolução.

INTERFACE: equipamento utilizado como intermediário na comunicação de dois ou mais equipamentos, de modo a permitir uma maior confiabilidade na transmissão de dados.

INTERPRETADOR: programa que recebe um código feito em uma linguagem de programação, e o converte para um código executável.

LOOP: laço, volta ou percurso, onde o ponto inicial é o final, para fins de avanço de uma etapa ou fase de um processo de desenvolvimento de software.

LUTHIER: profissional ou artesão com habilidades para construção ou reparo de instrumentos musicais com cordas (piano, guitarra, contrabaixo e violino, entre outros).

MASTER: dispositivo principal de um hardware.

MASTERIZAÇÃO: processo de pós-produção musical, com a finalidade de corrigir eventuais deficiências ou problemas.

MICROCONTROLADOR: microprocessador que pode ser programado para funções específicas.

MICROPROCESSADOR: circuito integrado constituído por milhares ou milhões de transistores, que permite a manipulação de dados e execução de funções.

MIDI DIN: conector fêmea de cinco pinos, utilizado para a comunicação e transferência de informações ou dados específicos para um instrumento ou equipamento musical.

MIDI IN: conexão que permite o recebimento de dados.

MIDI OUT: conexão que permite o envio de dados.

MIDI THRU: conexão que permite a comunicação direta entre três ou mais equipamentos MIDI (o MIDI IN do primeiro equipamento será ligado ao MIDI OUT do segundo, e o MIDI THRU do segundo ao MIDI OUT do terceiro equipamento, por exemplo).

MIDI (*Musical Instruments Digital Interface*): protocolo ou tecnologia que permite a comunicação de equipamentos ou instrumentos musicais.

MIXAGEM: processo no qual múltiplos canais de fontes sonoras são ajustados, nos mais diversos parâmetros, de forma a fornecer uma melhor qualidade de áudio.

MLAN: protocolo de nível de transporte, utilizado para gerenciamento e manipulação de sinais de áudio e vídeo digital, controle e comunicação multiportas MIDI via rede.

MÓDULO DE SOM: equipamento musical sem teclas, utilizado em um teclado via conexão MIDI, com todas as funcionalidades e especificações de um modelo específico (como exemplo, o módulo KORG i5M é similar ao teclado KORG i5S, porém, sem teclas).

OPEN HARDWARE: conceito similar ao de software livre, onde o próprio desenvolvedor de um componente de hardware disponibiliza o diagrama esquemático, a lista de componentes, o *layout* da placa e demais informações relacionadas ao hardware em questão.

OPEN SOURCE: software livre, com licença de código-fonte aberto, permitindo alterações conforme o conhecimento e as necessidades de um usuário específico.

ORIENTAÇÃO A OBJETO: técnica de modelagem e desenvolvimento de sistemas de software, com foco na relação e interação entre os objetos (classes) do sistema.

PATCH: configuração pertencente a um som, também conhecido como “programa”.

PATCHBAY: cabo utilizado para interligar dois equipamentos ou módulos.

PEDALEIRA CONTROLADORA: pedal interligado a um equipamento via MIDI, com a finalidade de acionar recursos ou parâmetros do respectivo equipamento com os pés.

PLUGIN: programa ou módulo que pode ser acoplado a um programa existente, adicionando recursos ou funções específicas ao sistema.

POLIFÔNICO: característica de um instrumento ou equipamento relacionada à polifonia, ou seja, a capacidade de executar uma ou mais notas simultaneamente.

PONTO-A-PONTO: ligação feita entre dois equipamentos, de modo direto, sendo que as informações a serem trafegadas ficarão vinculadas exclusivamente a esses dois equipamentos.

PRESET: assemelha-se ao *patch*, podendo ser um timbre específico, com as características inerentes a esse timbre, por exemplo.

PROTOCOLO: normativa ou regulamento para controle de conexão, comunicação e transferência de dados entre equipamentos e/ou sistemas.
PROTÓTIPO: versão de teste de um produto ou programa.

REDUNDÂNCIA: sistema "espelho" que visa a garantir a disponibilidade de um sistema, caso haja problemas.
REQUISITO: característica ou condição que um sistema deve ter para que possa atender a uma necessidade específica.

SAMPLER: equipamento que permite o armazenamento de sons em formato WAV em memória digital, para reproduzi-los posteriormente, um a um ou em grupos.
SCSI (*Small Computer System Interface*): tecnologia de comunicação para periféricos.
SERVLET: classe Java utilizada para estender funcionalidades e comportamentos de um servidor de aplicação.
SHIELD: módulo de expansão utilizado em placas Arduíno, com a finalidade de disponibilizar funções específicas.
SÍNTESE ADITIVA: técnica de sobreposição de várias faixas de frequências sonoras simultaneamente, de modo a criar novas texturas sonoras.
SINTETIZADOR: teclado utilizado para produzir sons gerados artificialmente, através de técnicas específicas.
SLAVE: dispositivo secundário de um hardware.
SOFTWARE EMBARCADO: também conhecido por sistema embarcado, é um programa implementado em um chip programável, de modo a permitir a execução de funções ou ações diretamente em um equipamento.
SYSEX (*System Exclusive Messages*): mensagens exclusivas de sistema, transmitidas e/ou recebidas via protocolo MIDI, cuja estrutura é definida pelo aparelho receptor.

TAG: estrutura de linguagem de marcação, com marcas de início e fim, que permitem a um navegador de internet renderizar uma página específica.
TCI/IP (*Transmission Control Protocol/Internet Protocol*): protocolo de transmissão de dados via rede *ethernet*.
TEMPLATE: modelo de documento, composto de cabeçalho padrão, além de instruções básicas para a correta elaboração de uma página ou documento.

TIMBRE: característica sonora que permite distinguir as diferenças entre duas fontes sonoras, tocando uma mesma nota de mesma frequência.

VSTi (*Virtual Studio Technology Instrument*): interfaces que integram sintetizadores e efeitos de áudio, utilizando processamento digital e técnicas de *sampler* para captar sons de instrumentos reais e transformá-los em digitais, e que oferecem as mesmas características e particularidades dos instrumentos musicais reais.

WEB DESIGNER: profissional habilitado para criação e elaboração de projeto visual de um site ou aplicativo de internet.

WORKSTATION: tipo de teclado que, além de possuir características de sintetizadores, envolve funcionalidades de sequenciamento e gravação, sendo chamado de "estação de trabalho".

WORLD WIDE WEB: sistema de documentos em hipermídia, interligados e utilizados no ambiente da internet.

XHTML (*eXtensible Hypertext Markup Language*): reformulação do padrão HTML (*HyperText Markup Language*), combinando esse padrão com regras do padrão XML (*eXtensible Markup Language*).

SUMÁRIO

1 INTRODUÇÃO

A música, sendo uma arte das mais antigas na história da humanidade, se redescobre a cada novo dia. Desde os primórdios da humanidade, passando pelo amadurecimento da inteligência, as primeiras experiências e descobertas com sons e notações musicais e, avançando com a descoberta da eletricidade, a criação do transistor, dos microprocessadores e todo o impulso da tecnologia, a criatividade e curiosidade humanas vêm sendo atraídas pelas infinitas possibilidades abertas ao longo do caminho. (RATTON, 1991, p.2).

Atualmente, vê-se uma ampla variedade de equipamentos musicais, instrumentos virtuais, teclados controladores, arranjadores, sintetizadores, *workstations* e outros equipamentos, que propiciam uma gama quase que infinita de possibilidades musicais. Com tantas possibilidades disponíveis, é extremamente comum que muitos músicos utilizem mais de um equipamento em shows ou apresentações. O uso de diversos equipamentos permite ao profissional da área obter os sons, ritmos, recursos e efeitos necessários para a execução das mais variadas músicas.

De encontro a essa diversidade, surge um problema: a dinâmica de troca de ritmos, sons, efeitos e recursos durante a execução das músicas. Quanto mais equipamentos são utilizados, maior o tempo gasto para a troca das programações em tempo real, o que pode atrapalhar consideravelmente o músico durante a execução de determinado trecho. Além disso, é preciso visualizar as letras e o repertório do show ou apresentação. São diversos fatores simultâneos, que podem levar o profissional a cometer erros durante o exercício de sua atividade.

Para auxiliar nessa questão, o uso de softwares multifuncionais e multiplataformas é algo fundamental. Para a criação de tais softwares, uma das linguagens de programação mais utilizadas é o Java, pela sua adaptabilidade e facilidade de comunicação com a internet. (LEMAY *et al.*, 1999, p.11).

Com a popularização da internet e sua ampla utilização, torna-se interessante a utilização de aplicações Web, haja vista que possui como característica mais marcante o dinamismo da informação, ou seja, as informações acessadas podem ser personalizadas, alteradas e atualizadas para cada usuário, de acordo com as necessidades. (FIELDS *et al*., 2000, p.3).

Para facilitar o armazenamento dessas informações, são utilizados os Sistemas Gerenciadores de Bases de Dados Relacionais (SGBDR's), tais como o PostgreSQL, amplamente utilizado em sistemas web. (PEREIRA NETO, 2003, p.28). Já, para fins de

integração entre a aplicação web e o banco de dados, utiliza-se um servidor Apache Tomcat, que é um *servlet* e container de *servlets*, servindo de apoio para desenvolvimento de páginas JSP. (FIELDS *et al*., 2000, p.477).

Outra ferramenta amplamente utilizada neste tipo de projeto é o Hibernate, que pode ser considerado como um framework cuja finalidade é a integração da aplicação com o banco de dados, efetuando a persistência das informações. (GONÇALVES, 2008, p.88) Para fins de desenvolvimento desse tipo de aplicação, são utilizadas as plataformas de desenvolvimento, tais como NetBeans e Eclipse. Dentre elas, a plataforma Eclipse é uma das mais utilizadas, haja vista a alta integração com os mais diversos *plugins*, incluindo HTML. (HEMRAJANI, 2007, p.140).

Pode-se, ainda, utilizar um hardware externo, para facilitar a operação do software desenvolvido. Para tanto, pode-se utilizar o Arduíno, que é, segundo Roberto Brauer Di Renna e outros (2013, p. 3), "um kit de desenvolvimento que pode ser visto como uma unidade de processamento capaz de mensurar variáveis do ambiente externo, transformadas em um sinal elétrico correspondente, através de sensores ligados aos seus terminais de entrada. ".

No caso de utilização com instrumentos ou equipamentos musicais, há a necessidade de utilização de um protocolo específico para que essa comunicação possa ser realizada. É nesse momento que entra em cena o protocolo MIDI (*Musical Instrument Digital Interface*), inicialmente orientado para sintetizadores, mas com suas aplicações expandidas para outros equipamentos. (RATTON, 2005, p.9).

1.1 PROBLEMÁTICA

A ampla diversidade de equipamentos utilizados por músicos, principalmente em apresentações ao vivo, gera uma grande dificuldade em efetuar alterações de programas, timbres, ritmos e efeitos, em tempo real, durante a execução de músicas. Além disso, há a dificuldade em encontrar um sistema capaz de gerenciar repertórios e letras de músicas, haja vista que muitos dos músicos são cantores, e necessitam ter à mão as letras da maioria das músicas executadas, além do repertório em si, de modo simples, fácil e prático.

Essas dificuldades vão ao encontro de outro problema: na maioria das vezes, as mudanças em tempo real das programações ocorrem em tempo de execução das músicas, não havendo uma maneira mais objetiva de efetuar essas mudanças para todos os equipamentos simultaneamente, sem deixar de tocar naquele determinado trecho.

A finalidade do presente trabalho visa à construção de uma solução de software, trabalhando de modo dinâmico e integrado, para facilitar o gerenciamento de todos os fatores acima elencados, de modo rápido, preciso e em tempo real.

Futuramente, será realizada a integração do presente software a uma pedaleira controladora, permitindo que o músico possa efetuar trocas de programações sem precisar interromper a execução da música.

1.2 OBJETIVOS

Para o presente trabalho, ficam definidos os seguintes objetivos, separados em objetivo geral e objetivos específicos.

1.2.1 OBJETIVO GERAL

O presente trabalho tem como objetivo desenvolver um software web para gerenciamento de shows musicais, com recursos para comunicação serial.

1.2.2 OBJETIVOS ESPECÍFICOS

Em relação aos objetivos específicos, este trabalho pretende:

- ✓ desenvolver uma aplicação web, utilizando IDE de desenvolvimento, banco de dados, servidor de aplicação e persistência em BD;
- ✓ implementar recursos de comunicação serial;
- ✓ realizar um estudo sobre aplicativos web, integração com banco de dados e persistência;
- ✓ realizar um estudo sobre comunicação serial;
- ✓ realizar um estudo sobre protocolo MIDI, pedaleira controladora, Arduíno e *shields* MIDI, para fins de desenvolvimento de trabalhos futuros com integração de hardware/software;
- ✓ apresentar os resultados obtidos.

1.3 JUSTIFICATIVA

A utilização de um sistema que permita ao músico gerenciar todo o rol de equipamentos, músicas, programações e *setlists*, traz maior segurança, praticidade e dinamismo durante a execução de um show.

Muitos músicos utilizam dois, três ou mais arquivos para efetuarem todo o gerenciamento das informações, o que pode aumentar a possibilidade de erros na troca de uma programação ou na visualização de uma letra, por exemplo.

Um aplicativo web apresenta como principal característica o dinamismo. As informações podem ser mais bem distribuídas e apresentadas em uma interface amigável ao usuário, facilitando a operação nas mais variadas situações.

Utilizando um conjunto de componentes e ferramentas específicas, pode-se desenvolver um software capaz de suprir grande parte das necessidades de um músico, de modo a facilitar a criação, gerenciamento e execução de repertório, músicas e programações em tempo real. Pode-se, da mesma forma, ter informações sobre a banda da qual o músico participa, redirecionamento para perfis em redes sociais, além das funções principais para o músico (gerenciar dados sobre a banda, músicas, shows, programações e repertórios).

Para facilitar ainda mais, há a possibilidade de utilização de um hardware externo, que efetue comunicação com o software e com os equipamentos que o músico utiliza, efetuando, rapidamente, troca de músicas e programações durante a execução das músicas, sem que o artista precise deixar de tocar em algum trecho para efetuar essas mudanças. E essa facilidade fica ainda mais explícita nos casos em que o músico utiliza, por exemplo, três, quatro ou mais equipamentos simultaneamente.

Haja vista todos os motivos anteriormente elencados, além do grande interesse pessoal, entende-se por relevante o desenvolvimento desta proposta de solução, pois há muitos músicos que enfrentam, a cada show ou apresentação, os problemas já apontados anteriormente. E, com a utilização massiva de notebooks nas performances ao vivo, a solução apontada torna-se ainda mais interessante e atrativa, pois apresenta uma estrutura completa de funcionalidades e recursos que são necessários para uma boa execução dos trabalhos dos profissionais da área musical.

Vale ressaltar que o escopo principal do presente projeto foi o desenvolvimento do software, sendo que a integração entre esse software e eventuais pedaleiras controladoras é o foco dos trabalhos futuros deste autor.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho é dividido em 6 capítulos. O primeiro capítulo abordou a introdução, a problemática, os objetivos gerais e específicos, além da justificativa do tema apresentado.

No capítulo 2, é feita uma abordagem teórica sobre os temas e assuntos condizentes com o projeto, tais como, IDE de desenvolvimento, servidor Apache Tomcat, base de dados PostgreSQL e Hibernate. Ainda, são estudados os assuntos: Arduíno, comunicação serial, protocolo MIDI e pedaleira controladora.

Já, no capítulo 3, é dada ênfase ao método científico. Para o capítulo 4, há a apresentação da modelagem para a solução do problema.

O capítulo 5 trata do método de desenvolvimento, descrição das etapas e arquitetura do sistema proposto.

Finalmente, no capítulo 6, são apresentados os resultados finais, as dificuldades encontradas e as perspectivas para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, é feita uma abordagem teórica sobre os assuntos relacionados ao presente trabalho, com um breve histórico da evolução tecnológica dos tópicos envolvidos, paralelamente à tecnologia e ao protocolo MIDI, haja vista que esse histórico, por vezes, caminha lado a lado.

São abordados os principais conceitos sobre processos de desenvolvimento de software, linguagens de programação, focando em Java, IDEs de desenvolvimento, focando em Eclipse, servidor de aplicação Apache Tomcat, base de dados PostgreSQL, Hibernate e desenvolvimento web, entrando em conceitos sobre web design, JSP, JSF, HTML, XHTML e CSS. Ainda, são revisados os tópicos Arduíno, comunicação serial e protocolo MIDI, além de conceitos sobre pedaleiras controladoras e sobre a evolução da tecnologia musical.

Um aplicativo web apresenta como principal característica o dinamismo. As informações podem ser mais bem distribuídas e apresentadas em uma interface amigável ao usuário, facilitando a operação nas mais variadas situações. Grannell (2009, p.14) destaca a importância do surgimento do Cascading Style Sheets (CSS), que é um padrão de definição visual para uso em páginas JSP e HTML, e que facilitou e melhorou a comunicação visual entre aplicativo e usuário, tornando os aplicativos e páginas web bastante interessantes.

Os aplicativos *web*, como o nome já menciona, têm por finalidade troca e armazenamento de informações através da Web. Contudo, pode ser utilizado localmente em notebooks, utilizando uma base de dados para armazenamento das informações. Neste caso, uma boa alternativa é o PostgreSQL pois, além de ser Open Source (Código Aberto), é orientado a objetos e possui uma interface amigável ao usuário. (PEREIRA NETO, 2003, p.25).

Entre aplicação e BD, o servidor Apache Tomcat é um dos mais utilizados e populares, por ser mais leve e, também, com uma interface mais amigável para operação. (HEMRAJANI, 2007, p.7). Já, para fins de persistência em BD, o framework Hibernate se encontra no patamar de solução mais viável, principalmente, no quesito Entity Beans, que facilita o relacionamento entre as classes existentes.

Utilizando esse conjunto de componentes, pode-se desenvolver um software capaz de suprir todas as necessidades de um músico, de modo a facilitar a criação, gerenciamento e execução de repertório, músicas e programações em tempo real. Pode-se, da mesma forma, ter informações sobre a banda da qual o músico participa, redirecionamento para perfis em redes sociais, além das funções principais para o músico (gerenciar dados sobre a banda, músicas, shows, programações e repertórios).

Para facilitar ainda mais, há a possibilidade de utilização de um hardware externo, que efetue comunicação com o software e com os equipamentos que o músico utiliza, efetuando, rapidamente, troca de músicas e programações durante a execução das músicas, sem que o artista precise deixar de tocar em algum trecho para efetuar essas mudanças. E essa facilidade fica ainda mais explícita nos casos em que o músico utiliza, por exemplo, três, quatro ou mais equipamentos simultaneamente.

O Arduíno é componente mais importante na construção desse tipo de hardware. Essa unidade pode ser programada, de modo a gerenciar e atuar no controle de equipamentos conectados. Ainda, possui a característica de ser totalmente adaptável às mais diversas situações e necessidades, o que o torna ainda mais versátil. (DI RENNA *et al*., 2013, p.4).

Para tornar toda essa comunicação entre hardware/software/equipamentos possível, entra em cena o protocolo MIDI que, na opinião de Prado (2006, p.2):

> [...] é uma ferramenta familiar a milhares de pessoas, não apenas a músicos profissionais. E apesar da chegada da internet e da evolução dos programas que tratam áudio, MIDI continua vivo e poderoso, permitindo o controle de simples placas de computadores até sofisticados sistemas com diversos equipamentos.

Ainda, segundo o mesmo autor: "... Por suas características, MIDI é um protocolo, e não uma linguagem, pois estabelece padrões para a comunicação." (PRADO, 2006, p.2). Assim, o uso desse protocolo permite que a comunicação entre os componentes envolvidos seja precisa, rápida e objetiva, de modo a dar maior dinamismo e segurança aos músicos que utilizam a solução hardware/software apresentada.

A parir deste ponto, inicia-se à abordagem teórica dos assuntos acima elencados.

2.1 DESENVOLVIMENTO DE SOFTWARE

Nos dias atuais, dificilmente há um local que não possua algum tipo de mecanismo tecnológico para manipulação de dados e informações dos mais diversos tipos e modos. Esses mecanismos são os chamados *softwares,* que permitem ao usuário uma forma simples e objetiva de efetuar as mais diversas atividades. Grandes empresas utilizam sistemas altamente complexos; por outro lado, usuários domésticos também se utilizam de sistemas mais simples, para as mais diversas finalidades. (SOMMERVILLE, 2003, p.4).

Para uma melhor compreensão do que é um software e quais os processos envolvidos, é necessária a apresentação de alguns conceitos importantes, que constam nos próximos tópicos.

2.1.1 A Engenharia de Software

Dentro das áreas da engenharia, a engenharia de software é a responsável pelas características envolvidas nos processos de produção de um software, seja ele genérico ou personalizado. Todas as etapas, desde a coleta de requisitos, o desenvolvimento da proposta, a implantação e posterior manutenção são realizadas pela engenharia de software. (SOMMERVILLE, 2003, p.5).

Para o desenvolvimento de um software, são utilizados os processos de software, que são um conjunto de atividades ou procedimentos, organizados e estruturados, que geram um produto final (software). De acordo com Sommerville (2003, p.7), são quatro os principais deles, quais sejam:

1. **especificação do software:** são as características e funcionalidades do software, bem como restrições envolvidas;
2. **desenvolvimento do software:** a construção do software, de acordo com as especificações;
3. **validação do software:** checagem por parte do cliente, para verificar se o software atende às necessidades desse cliente;
4. **evolução do software:** deve atender às alterações que o cliente necessitar futuramente.

2.1.2 Modelos de processo de software

Para o desenvolvimento desses sistemas, há vários modelos de processo que podem ser adotados. Um modelo de processo de software representa as etapas de desenvolvimento de software, de acordo com a perspectiva do modelo, mostrando apenas as informações pertinentes a ele. (SOMMERVILLE, 2003, p.37).

O modelo de processo de software adotado no presente trabalho é descrito a seguir.

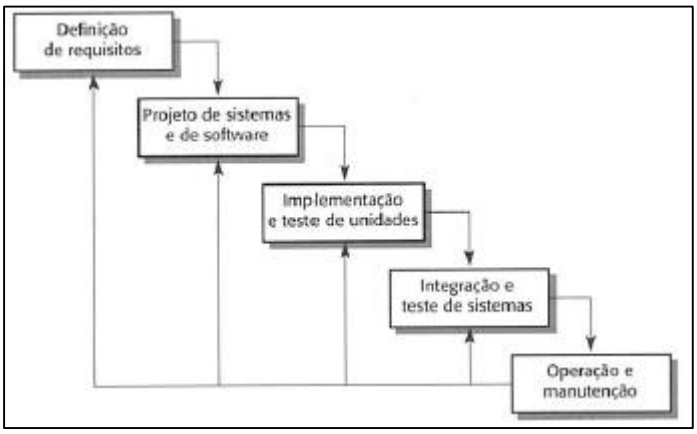
2.1.2.1 Modelo cascata

Nesse modelo, cada fase deve ser finalizada para que a próxima seja iniciada. Apesar de não ser um modelo linear, envolve várias iterações das atividades desenvolvidas, gerando maiores dificuldades na relação custo/benefício. Ainda, esse modelo apresenta grande dificuldade na questão dos requisitos que surgem posteriormente, por ser um modelo que não permite uma flexibilidade maior para o atendimento de novos requisitos, com retorno às fases anteriores somente no final do ciclo. (SOMMERVILLE, 2003, p.39). O modelo cascata trabalha, basicamente, com as seguintes atividades:

1. **análise e definição de requisitos:** funções, restrições, características e atividades do sistema que, posteriormente, servirão como especificação do sistema;
2. **projeto de sistemas e de software:** separa as especificações entre requisitos de hardware e software, definindo a arquitetura necessária para o sistema como um todo;
3. **implementação e testes de unidade:** o software é testado componente a componente, de forma individual, para verificar se atende a todas as necessidades especificadas;
4. **integração e testes de sistema:** os componentes são integrados e são realizados testes no sistema já integrado, para posterior aprovação e entrega ao cliente, ou correção de eventuais erros;
5. **operação e manutenção:** pode ou não ocorrer, sendo que é uma das etapas mais longas do processo, sendo identificados eventuais erros não descobertos anteriormente, efetuando as devidas correções, ou criando novas funcionalidades a partir da descoberta de novos requisitos.

Um exemplo de modelo cascata é apresentado na Figura 1.

Figura 1 – Ciclo de vida do software no modelo cascata



Fonte: Sommerville (2003, p.38).

Há uma variação do modelo cascata, também bastante utilizada, chamada de modelo cascata com retroalimentação. Nesse modelo, há a possibilidade de retorno a uma etapa anterior sem que, obrigatoriamente, seja alcançada a última etapa. (SOMMERVILLE, 2003, p.39)

2.2 LINGUAGENS DE PROGRAMAÇÃO

Para fins de construção de um bom software, vários requisitos devem ser levados em conta. E, entre os requisitos não funcionais, um deles é o de implementação, que apresenta as informações sobre a linguagem de programação que será utilizada para desenvolvimento do software, de acordo com o ambiente no qual será utilizado. (SOMMERVILLE, 2003, p.85).

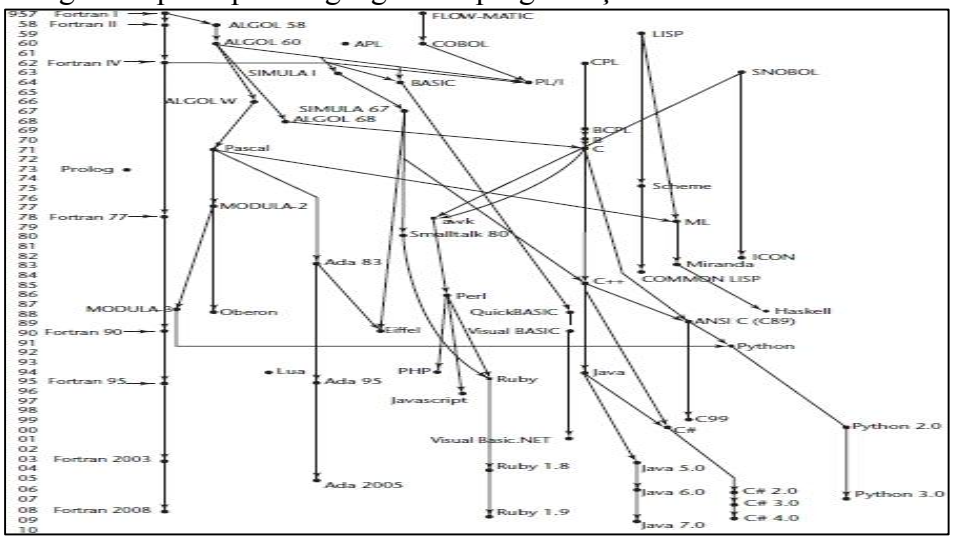
Uma linguagem de programação pode ser definida como uma notação formal, utilizada para a descrição e compreensão de algoritmos, que serão executados por um computador. Ela se constitui, basicamente, de duas etapas: sintaxe, que descreve os caracteres pertencentes à linguagem e as regras de composição, utilizando esses símbolos. Já, a semântica é uma espécie de "dicionário", que apresenta o significado das palavras da linguagem ou o mecanismo das mesmas. (GHEZZI *et al.,* 2008, p.49).

No próximo tópico, é apresentado um histórico sintetizado das linguagens de programação.

2.2.1 Um breve histórico das linguagens de programação

Para fins de conhecimento e, antes de abordar a linguagem de programação que será estudada neste trabalho, a Figura 2 traz um breve histórico genealógico das linguagens de programação existentes. (SEBESTA, 2012, p.37).

Figura 2 – Genealogia das principais linguagens de programação de alto nível



Fonte: Sebesta (2012, p.37).

Para fins de abstração de conteúdo, não são abordadas as demais linguagens no presente trabalho, restando, tão somente, a linguagem Java a ser estudada mais adiante.

2.3 BANCO DE DADOS

Com a evolução da tecnologia e o aumento das informações, muitos dados vêm sendo trafegados e acessados. Empresas necessitam armazenar dados de produtos, clientes, compras, vendas e outras atividades em meios digitais, e todas essas informações devem ser guardadas e protegidas, de modo a não haver perdas de dados, redundância ou discrepância de informações. (HEUSER, 1998, p.14).

Para evitar problemas, como a redundância, uma solução é o compartilhamento desses dados. Mantendo esses dados armazenados em um único local, todos aqueles que necessitarem do acesso às informações podem efetuá-lo em um único local. Assim, surge o conceito de banco de dados (BD), ou seja, “conjunto de dados integrados que têm por o objetivo atender a uma comunidade de usuários. ” (HEUSER, 1998, p.14).

A fim de facilitar o controle da estrutura interna dos arquivos compartilhados, são utilizados os sistemas de gerência de banco de dados (SGBDs). Segundo Heuser (1998, p.15), um sistema de gerência de banco de dados é um "software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados". Esses sistemas modulares trazem benefícios como facilidade na manutenção e aumento na produtividade, por exemplo. (HEUSER, 1998, p.16).

Para um projeto de banco de dados, dois níveis de abstração são utilizados: o conceitual (descrição de um banco de dados de forma independente de implementação) e o lógico (descrição de um banco de dados no nível de abstração visto pelo usuário). (HEUSER, 1998, p.16).

Para fins de conceito lógico, a abordagem mais utilizada é a entidade-relacionamento (ER). Já, para o conceito lógico, o modelo é aquele referente ao SGBD relacional. O termo entidade pode ser descrito como uma representação de um objeto ou coisa real, com as respectivas características, que deverão ser armazenadas em um banco de dados. Já o termo relacionamento é a associação entre essas entidades. (HEUSER, 1998, p.24).

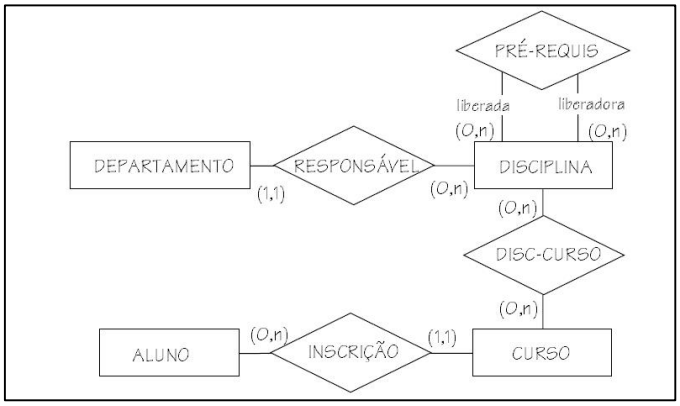
Esses relacionamentos necessitam de uma informação: a quantidade de ocorrências dessas entidades em relação a uma ocorrência de outra entidade via relacionamento. Essa propriedade é conhecida como cardinalidade. (HEUSER, 1998, p.26).

Para cada entidade, há uma série de características envolvidas, que poderão ou não ser informadas e armazenadas em um banco de dados. Por exemplo, uma pessoa (entidade) pode ter armazenada um código de identificação, o nome, CPF e telefone (atributos), sendo que um desses atributos deverá ser único, conhecido como chave primária. (HEUSER, 1998, p.34).

Pode haver ocorrências de entidades de relacionamentos com chaves primárias compostas, herdadas de outra entidade, ou entidades com uma única chave primária, mas cujos atributos necessitem de uma chave primária de outra entidade, chamada de chave estrangeira. (HEUSER, 1998, p.89).

Na Figura 3, é apresentada uma representação gráfica, em formato de diagrama, das entidades e relacionamentos que representam, por exemplo, o controle acadêmico de uma universidade.

Figura 3 – Diagrama de ER de um controle acadêmico



Fonte: Heuser (1998, p.32).

Para fins de manipulação dessas informações, é utilizada uma *Structured Query Language* (SQl, ou, Linguagem Estruturada). É originária no início dos anos 80, pela IBM, mas teve a primeira padronização pela ANSI (*American Nacional Standarts Institute*) e pela ISO (*International Standarts Organization*) em 1986 e 1987, respectivamente. (PEREIRA NETO, 2003, p.25).

2.4 PROGRAMAÇÃO WEB

A história da programação web teve seu início nos anos 60, após experiências militares de comunicação. Desde então, muitas ferramentas foram sendo desenvolvidas e muitos avanços foram sendo alcançados. Com o lançamento do protocolo TCP/IP (*Transmission Control Protocol / Internet Protocol*), a comunicação entre dispositivos acabou se tornando melhor, o que incentivou o desenvolvimento de aplicações web. (GRANNELL, 2009, p.2).

Nos tópicos seguintes, são apresentados detalhes sobre o surgimento da programação voltada para o ambiente web, bem como algumas fermentas que auxiliam no desenvolvimento de aplicativos para tal ambiente.

2.4.1 Um contexto geral

No início dos anos 60, havia uma grande preocupação em questões de segurança, como, por exemplo, no caso de um ataque nuclear. Assim, surgiu a necessidade de um sistema

que permitisse a comunicação das autoridades militares com autoridades do governo, logo após um acontecimento desse porte. (GRANNELL, 2009, p. 2).

Um sistema inicial foi proposto, através de comunicação ponto-a-ponto, porém sem sucesso. Um sistema no formato de uma "teia" era a melhor solução, pois, caso um ponto fosse rompido, haveria outro caminho para transmitir a informação. Entretanto, essa ideia inicial de ponto-a-ponto acabou sendo usada em universidades norte-americanas, posteriormente. (GRANNELL, 2009, p. 2).

Já, nos anos 80, a internet ganhou um destaque maior, com sua abertura ao público. A criação do protocolo TCP/IP auxiliou nessa propagação da internet. Paralelamente ao surgimento da internet, o protocolo HTML ia sendo desenvolvido, ganhando destaque a partir do início dos anos 90, com a popularização maciça da internet, a melhoria nos sistemas de comunicação e a criação dos primeiros *browsers*, como o Mosaic, e se expandindo até os dias atuais. (GRANNELL, 2009, p. 3).

Com o crescimento das aplicações web, foi-se constatando a necessidade de incorporação de arquivos de mídia, tais como vídeos, fotos, cores, a animações, entre outros detalhes, exigindo um aprimoramento dos ambientes e, consequentemente, das ferramentas utilizadas para o desenvolvimento. Conceitos de usabilidade e acessibilidade foram sendo colocados em prática mais recentemente, que focam na relação usuário-sistema, ou seja, na facilidade que o usuário encontra ao utilizar determinado sistema.

Inúmeros projetos ou serviços podem ser construídos e oferecidos em um ambiente web, tais como projetos musicais, artísticos, divulgação de trabalhos, pesquisas, órgãos públicos e compartilhamento de conhecimento, entre outros exemplos. Tudo vai depender da finalidade da aplicação, do público alvo e das particularidades envolvidas, mas, em regras gerais, aplicações web podem ser usadas para praticamente todas as finalidades. (GRANNELL, 2009, p.8).

2.5 ARDUÍNO

Alguns softwares, embora possuindo maior facilidade na operação, podem ser ainda mais práticos com o auxílio de um hardware externo. Esses hardwares podem receber e transmitir informações para outros dispositivos ou equipamentos, executar comandos ou emitirem informações ao usuário. E um dos componentes mais utilizados para o desenvolvimento de hardware é o Arduíno. (MCROBERTS, 2010, p.20).

O Arduíno faz parte do conceito de hardware e software livre, e surgiu na Itália, em 2005, com a finalidade de ser uma opção mais prática e menos onerosa para protótipos. (DI RENNA *et al.*, 2013, p.3).

Ele foi projetado com a intenção de ser multiplataforma, além de ser facilmente instalado, programado e configurado, com um custo relativamente baixo. Além disso, se encontra vinculado a uma comunidade de filosofia open-source, o que tem ajudado muito nas questões de divulgação e apresentação dos projetos realizados. (DI RENNA *et al.*, 2013, p.3).

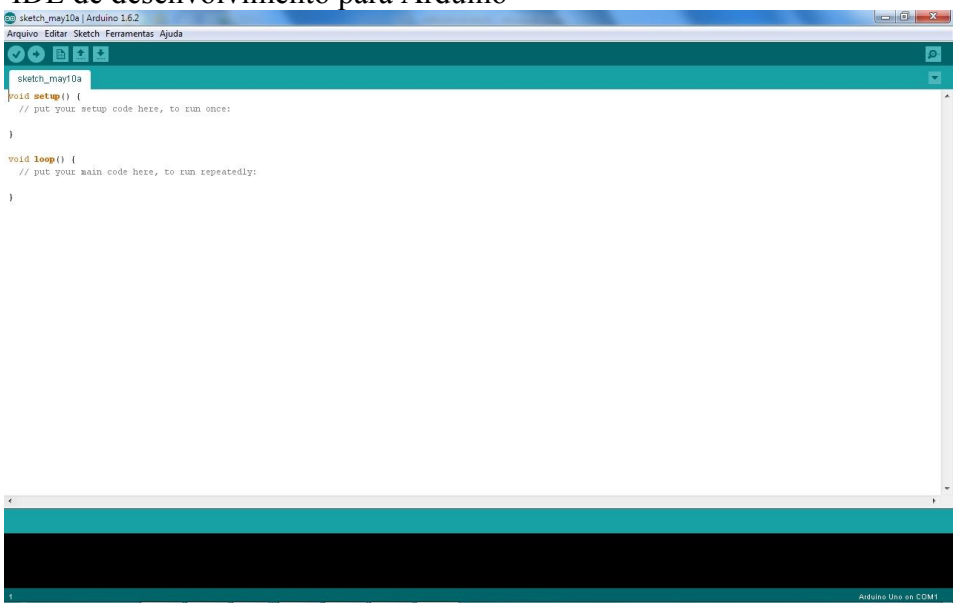
O Arduíno é baseado em um microcontrolador com portas de entrada/saída (I/O), com a presença de várias bibliotecas de funções, escritas em linguagem similar a Java, C e C++. Esse microcontrolador é, em tese, um computador dentro de um chip, com microprocessador, memória e dispositivos de entrada/saída, podendo ser embarcado em outros dispositivos eletrônicos. (DI RENNA *et al.*, 2013, p.3).

2.5.1 Sistema

Possui uma IDE própria para desenvolvimento, além de uma linguagem própria, com funções, delimitadores e bibliotecas, que tornam a sua manipulação mais legível, mesmo para aqueles mais leigos em programação. Assim, o Arduíno pode ser utilizado não somente para fins de projetos, mas, também, para fins educacionais. (DI RENNA *et al.*, 2013, p.4).

Na Figura 4, é mostrada a IDE de desenvolvimento do Arduíno.

Figura 4 – IDE de desenvolvimento para Arduíno



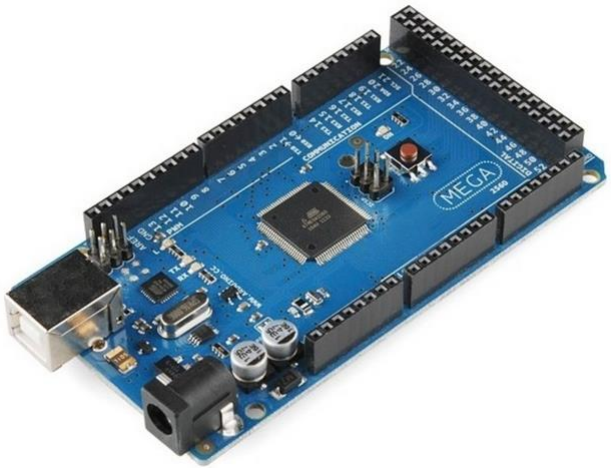
Fonte: Do autor.

2.5.2 Tipos de Arduíno

Dentre os vários tipos de Arduíno existentes, há opções para todos os tipos de projetos, dependendo das particularidades, exigências e orçamentos. O modelo a ser utilizado deve ser escolhido de acordo com as necessidades do hardware que se está desenvolvendo. (MCROBERTS, 2010, p.25).

O Arduíno Mega 2560 é uma versão *open-hardware* do original. Ele é uma atualização do Arduíno Mega, e possui um microcontrolador baseado no ATMega2560, contendo 54 pinos de entrada/saída, 16 entradas analógicas, 256 KB de memória e uma corrente de 40mA por pino. (ARDUINO.CC, 2015) A Figura 5 ilustra um exemplo de placa Arduíno Mega 2560 *open-hardware*.

Figura 5 –Arduíno Mega 2560, na versão open-hardware



Fonte: Filipeflop.com (2015).

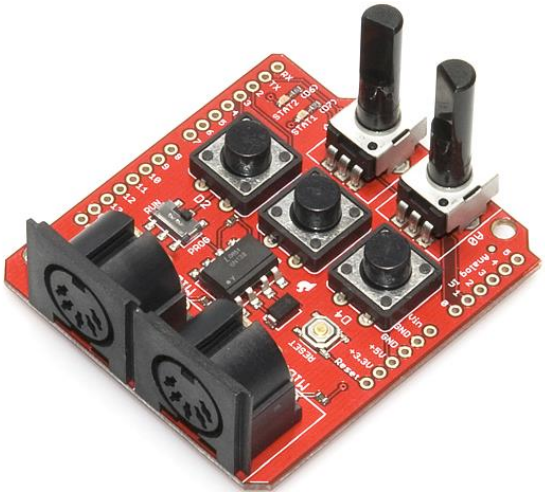
Caso necessário, pode ser efetuada a expansão do número de pinos RX e TX, mediante a criação de portas virtuais. (VIEIRA, 2015).

2.5.3 Shields

Para fins de uma maior expansibilidade do Arduíno, são utilizados os *shields*, que são acessórios que incorporam funcionalidades ao Arduíno. Esses *shields* não possuem processamento próprio e podem ser utilizados vários em um mesmo Arduíno, dependendo do projeto. (VIEIRA, 2015).

Dentre os vários Shields existentes, está o Arduíno MIDI Shield, que é mostrado na Figura 6.

Figura 6 – Arduíno MIDI Shield



Fonte: Multilógica Shop (2015).

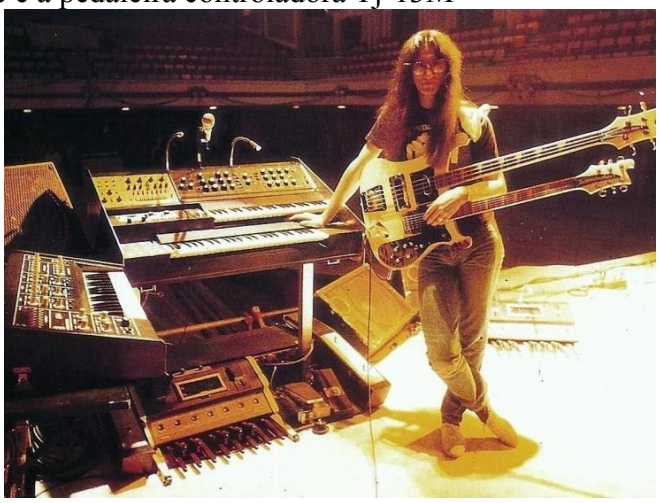
Esse *shield* é responsável pela comunicação entre o Arduíno e os instrumentos e/ou equipamentos musicais conectados a ele.

2.6 PEDALEIRA CONTROLADORA

Dentro da área musical, é bastante comum encontrar esse tipo de equipamento. Basicamente, uma pedaleira controladora é um pedal que controla parâmetros e comandos MIDI, enviados por um software ou equipamento externo através de comunicação serial, que podem ser controle de volume, mudança de *presets* (configurações pré-estabelecidas), efeitos, timbres (sons), entre outros. (CALIMARO, 2015).

Ainda, podem trabalhar com mecanismos que permitam que um músico possa tocar notas musicais com os pés, ampliando a área de cobertura de um instrumento musical, por exemplo. Bandas, como Rush e Genesis, utilizam-se desse tipo de pedal, bem como músicos como Rick Wakeman e Keith Emerson, por exemplo. Na Figura 7, pode-se observar o músico Geddy Lee, da banda Rush, ao lado de teclados e do sintetizador Bass Pedal Moog Taurus. (PADRINI, 2015). Alguns exemplos de pedaleiras controladoras são mostrados nas Figuras 8 (FS13), 9 (FC-7), 10 (Tj-13M) e 11 (FBV Shortboard MKII Line 6).

Figura 7 – Geddy Lee e a pedaleira controladora Tj-13M



Fonte: MA (2015).

Essa pedaleira permite a execução de notas musicais, simulando a ação de um contrabaixo, enquanto o músico utiliza as duas mãos nos teclados.

Figura 8 – Pedaleira controladora Ketron FS13

Fonte: Do autor.

A pedaleira FS13 é específica para teclados da marca italiana Ketron (antiga Solton), e é interligada ao teclado através de porta paralela. No teclado, são configuradas as ações de cada pedal.

Figura 9 – Pedaleira controladora Roland FC-7

Fonte: Teclacenter (2015).

Já, a pedaleira FC-7 funciona em teclados da marca Roland. Entretanto, as funções são fixas, e as variações são feitas progressivamente, ou seja, iniciam na variação A, passando para a B, C e D, respectivamente.

Figura 10 – Pedaleira controladora Tj-13M

Fonte: MA (2015).

A pedaleira Tj-13M é utilizada como uma espécie de contrabaixo, além de avançar ou retroceder programações no teclado ao qual estiver conectada.

Figura 11 – Pedaleira controladora FBV Shortboard MKII Line 6

Fonte: CALIMARO (2015).

Finalmente, a pedaleira FBV Shortboard MKII trabalha em um conceito diferenciado, com a incorporação de pedal de volume e 10 pedais de funcionalidades, além de três específicos para menus e funções internas. Os 10 pedais e o pedal de volume podem ter suas funcionalidades alteradas via software próprio de gerenciamento.

2.7 TECNOLOGIA MUSICAL

Não há como negar que a música está ao nosso redor 24 horas por dia. Desde *tablets, iPods, iPhones, smartphones*, aparelhos de CD e DVD, MP4, internet, instrumentos musicais, softwares de áudio, ..., enfim, a música está presente em diversos locais e de diversas formas ao acesso de todos. (BARBOSA, 1999, p.1).

Com todo o avanço tecnológico ao longo dos anos, a área da produção musical deixou de ser privilégio apenas para estúdios profissionais e indústria fonográfica. Com o surgimento de softwares e instrumentos musicais mais acessíveis, qualquer pessoa, desde que com um mínimo de conhecimento, é capaz de fazer trabalhos de alta qualidade. (BARBOSA, 1999, p.1).

Ao encontro dessa evolução tecnológica, instrumentos musicais, pedaleiras, módulos de som, acessórios e periféricos externos também foram se tornando produtos mais acessíveis, expandindo as possibilidades para músicos, instrumentistas e aventureiros. Assim, a própria indústria musical se reinventou e se reinventa cada vez mais. (GOHN, 2001, p.1).

Para uma melhor compreensão do que é o fenômeno da tecnologia musical e de um dos maiores responsáveis por essa evolução (ou revolução) – o MIDI -, é necessária uma breve abordagem em alguns tópicos importantes, que serão apresentados a seguir.

2.7.1 O que é transmissão serial

Segundo Ratton (1997, p.31), a transmissão serial é aquela na qual os dados são transmitidos *bit* a *bit*, ou seja, um *bit* de cada vez, sequencialmente, em um canal de transmissão. E, para que essa transmissão serial seja efetuada da melhor forma possível, alguns parâmetros devem ser levados em conta.

Inicialmente, deve ser estabelecida a velocidade de transmissão, que é o tempo em que o *bit* deve permanecer na linha até que seja lido e interpretado pelo receptor. Essa velocidade pode variar entre 2.400 a 14.400 *bits*/segundo, mas, para o padrão MIDI, que será visto em seguida, essa velocidade é de 31.250 *bits*/segundo. (RATTON, 1997, p. 32)

Além da velocidade, deve ser definido o protocolo de comunicação e o padrão de formatação dos dados. Ou seja, é necessário definir um padrão de comunicação e tratamento dos dados enviados e/ou recebidos. (RATTON, 1997, p. 32).

Há duas formas de se efetuar uma transmissão serial: a *síncrona*, e a *assíncrona*. Na transmissão *síncrona*, são utilizadas duas linhas de transmissão, sendo uma para marcar a

cadência da transmissão e outra para enviar os *bits*. Já, na *assíncrona*, é informado apenas o *bit* de partida, sendo que o receptor, a partir da leitura desse *bit*, utiliza circuitos próprios para a transmissão e sincronia da informação transmitida. Ao final do *bit* e, caso todos os dados tenham sido devidamente recebidos, o transmissor insere um bit de parada na linha, encerrando a transmissão. (RATTON, 1997, p. 32).

Ratton (1991, p.33) alerta para o fato de que, na transmissão via MIDI, todos os dados devem ser transmitidos instantaneamente, sem nenhum tipo de atraso ou perdas, decorrentes do processo de transmissão. Ou seja, uma informação de mudança de timbre, programação ou uma nota tocada, transmitida a um equipamento externo, deve ser imediatamente executada e controlada por esse dispositivo, sem atrasos perceptíveis ou perdas.

Com o conceito de transmissão serial já definindo, pode-se dar início aos estudos sobre a tecnologia musical, pois a comunicação serial faz parte dessa história.

2.7.2 Os primeiros passos

O primeiro protótipo de instrumento a funcionar com eletricidade foi o Dynamophone, ou Telharmonium, construído por Thaddeus Cahill, em 1906. Apesar das idéias desse instrumento não terem sido muito aceitas na época, anos mais tarde, pesquisadores deram continuidade, resultando na criação do Theremin, criado por Leon Theremin, em 1920, vindo a ser comercializado em 1929. Desde então, outros instrumentos acústicos passaram a ser eletrificados, como o violino, o piano, o órgão e a guitarra, entre outros. (GOHN, p.3) A Figura 12 ilustra um Theremin.

Figura 12 – O músico Jean-Michel Jarre e o Theremin

Fonte: Do autor.

2.7.3 Do analógico para o digital

Certamente, o estopim para uma das maiores mudanças no cenário musical foi a invenção do transistor, em 1948, e, posteriormente, a dos microprocessadores. Foi entre os anos 30 e 40 que as grandes marcas que hoje conhecemos começaram a ganhar força, passando a eletrificar os instrumentos, como exemplos, os clássicos órgãos Wurlitzer e Hammond, as guitarras elétricas de Adolf Rickenbacker e Léo Fender e o piano elétrico de Harold Rhodes, entre outros. (RATTON, 1997, p.5) Na Figura 13, um desses instrumentos clássicos: o órgão Hammond.

Figura 13 – Jon Lord (tecladista da banda Deep Purple) e Hammond B3

Fonte: Do autor.

Em 1955, os sintetizadores Mark I e Mark II foram criados e ocupavam uma sala inteira. Já, entre os anos 50 e 60, um dos mais importantes foi o Mellotron, sendo usado até hoje. Mas foi entre as décadas de 60 e 70 que começou a haver uma profunda mudança na concepção de instrumentos musicais, com a utilização dos transistores dentro de circuitos integrados, ou chips, reduzindo consumo e custos dos equipamentos. (RATTON, 1997, p.6).

Foi nessa época que um dos instrumentos mais legendários surgiu: o sintetizador Moog, criado por Robert Moog, no final dos anos 60. Nos anos 70, depois de várias mudanças, foi desenvolvido o Minimoog, um modelo mais compacto, consolidando o instrumento e o nome Moog no mercado mundial. (RATTON, 1997, p.6). Na Figura 14, um Moog modular. Já na Figura 15, um Minimoog em ação.

Figura 14 – Keith Emerson e o Moog modular

Fonte: Do autor.

Entre os Moogs modulares e os Minimoogs, houve uma redução considerável de espaço físico, o que facilitou na utilização desses equipamentos. (RATTON, 1997, p.6).

Figura 15 – “A lenda” Rick Wakeman e o Minimoog

Fonte: Do autor.

Foi a partir do final dos anos 70 e início dos anos 80, que os primeiros sintetizadores aproveitaram o desenvolvimento dos microprocessadores. Os primeiros sintetizadores eletrônicos eram construídos com circuitos analógicos, com os sons obtidos a partir de sinais eletrônicos. Eram inicialmente monofônicos e adotavam o processo de síntese aditiva, em que um sinal inicial de som podia ser manipulado em tempo real, por meio de controladores de forma de onda, frequência, amplitude e filtragem harmônica. Muitos músicos e bandas eram adeptos a esses sintetizadores, como Kraftwerk, Genesis, Jean-Michel Jarre e Vangelis, por exemplo. (RATTON, 1997, p.6).

Nos anos 80, os sintetizadores já eram polifônicos, implementando, inclusive, funções de manipulação em tempo real dos parâmetros do som e teclas sensíveis ao toque. Além disso, a capacidade de memória foi ampliada, junto com a utilização da tecnologia de *sampler*, que permite a gravação de instrumentos acústicos e a gravação digital desses sons para incorporação e utilização em outros instrumentos. Modelos como o Yamaha DX7 e o KORG M1 começaram a ganhar preferência entre os músicos. (RATTON, 1997, p.5).

Com todas essas inovações tecnológicas no meio musical e o custo reduzido dos componentes digitais, os equipamentos desenvolvidos posteriormente já começaram a ser desenvolvidos com a síntese de sons de modo digital, e não mais analógico. E é nesse momento que surgiu um componente fundamental nesse processo: o MIDI. (RATTON, 1997, p.7).

2.7.4 A tecnologia MIDI

O MIDI (*Musical Instrument Digital Interface*, ou Interface Digital para Instrumentos Musicais) é um sistema de comunicação digital para instrumentos e equipamentos musicais. Criado em 1983, é uma tecnologia de domínio público. Inicialmente, foi criada para ser orientada a objetos de teclado, várias mudanças o tornaram disponível para diversos outros equipamentos. (RATTON, 1997, p.37).

Haja vista que o MIDI não trafega sinais sonoros, mas, sim, dados digitais, permite a edição de notas musicais, estruturas de música, edição de trechos, andamento, tonalidade, entre outras características. Os arquivos sequenciados no formato MIDI podem ser executados em qualquer dispositivo que dê suporte a ele e pode ser gravado em qualquer tipo de mídia (disquetes, *pen drives*, CDs, entre outros), pois geram arquivos extremamente pequenos, da ordem de *quilobytes* (KB) a alguns *megabytes* (MB). (RATTON, 2015, p.9).

As utilizações da tecnologia MIDI são as mais variadas, desde o controle de um instrumento através de outro, transmissão e recebimento de parâmetros de manipulação de sons, mudanças de *patches*, acionamento e desligamento de recursos, entre muitas outras funcionalidades, com uma grande facilidade de operação e custo baixo, por ser de domínio público. (RATTON, 2005, p.20).

2.7.4.1 Especificações

A tecnologia MIDI possui um conjunto de especificações, criado em 1983, para fins de padronização das especificações de transmissão, formatos de arquivos, instrumentos,

tecnologias, sons e outras características importantes para o funcionamento. (RATTON, 1997, 37). Ratton (2005, p. 23) descreve essas especificações:

- ✓ **MIDI 1.0 Specification (1983):** define protocolo de comunicação, programação, formato de mensagens e características de hardware e interface;
- ✓ **Standart MIDI Files 1.0 (1988):** define as características dos formatos de arquivo MIDI;
- ✓ **General MIDI System Level 1 (1991):** define o conjunto de características mínimas que os equipamentos devem possuir para a criação de músicas MIDI dentro do padrão General MIDI (GM), de modo a serem executadas em equipamentos diferentes, mas com as mesmas sonoridades;
- ✓ **MIDI Show Control 1.0 (1991):** define as condições de comunicação com equipamentos de controle de iluminação, visuais e multimídia;
- ✓ **MIDI Machine Control 1.0 (1992):** define especificações para comunicação e controle com equipamentos de gravação de áudio e sistemas de produção musical;
- ✓ **General MIDI 2 Specifications (1994):** além de reafirmar as especificações anteriores, insere novas funcionalidades e especificações;
- ✓ **Downloadable Sounds (DLS) Level 1 Specifications (1999):** define arquitetura de dispositivos para fins de reprodução de timbres carregados a partir de arquivos que contenham formas de onda e parâmetros de execução;
- ✓ **MIDI Media Adaptation Layer for IEEE-1394 (2000):** define requisites para transmissão e recepção de mensagens MIDI, via conexão *FireWire*;
- ✓ **RMID File Format (2000):** define as formas de integração entre dados DLS em arquivos Standart MIDI Files (SMF);
- ✓ **MIDI 1.0 Detailed Specification (2001):** compila e descreve todas as mensagens MIDI aprovadas, além de procedimentos recomendados;
- ✓ **General MIDI Lite Specificaction and Guidelines for Mobile Applications (2001):** define as especificações para comunicação MIDI via dispositivo móvel;
- ✓ **Downloadable Sounds (DLS) Level 2.1 Specifications (2001):** define arquitetura de dispositivos para fins de reprodução DLS e mostra as extensões ao formato;

- ✓ **Scalable Polyphony MIDI Specifications (2002):** define parâmetros de execução de dados MIDI em dispositivos com capacidades de polifonia diferentes;
- ✓ **XMF (eXtensible Music Format) Specifications (2003):** define o padrão de formato que engloba todos os recursos de mídia (e/ou endereçamento para recursos externos) necessários para a execução de músicas;
- ✓ **MIDI XML Specification (2003):** define os requisitos e permite a representação de dados MIDI em formato XML.

2.7.4.2 Características

As mensagens MIDI que são transmitidas ou recebidas seguem um protocolo previamente determinado. As mensagens podem ser classificadas em mensagens de canal (estão relacionadas ao canal MIDI em questão) e de sistema (aspectos globais do sistema MIDI). (RATTON, 2005, p.27).

O Quadro 1 mostra a classificação dessas mensagens, que podem ser de canal ou de sistema. (RATTON, 2005, p.28).

Quadro 1 – Mensagens MIDI

Mensagens de canal	
Mensagens de voz	Note On
	Note Off
	Pitch bender
	Program Change
	Control Change
	Channel Aftertouch
	Polyphonic Aftertouch
Mensagens de Modo	Local Control
	All Notes Off
	Omni On / Off
	Poly Mode
	Mono Mode
Mensagens de Sistema	
Mensagens Comuns	MIDI Time Code (posicionamento)
	Song Position Pointer
	Song Select
	Tune Request
	EOX (End of SysEx)
Mensagens de Tempo-Real	Timing Clock
	Continue

	Start / Stop
	Active Sensing
	Reset
Mensagens Exclusivas	Manufacturer Data Dump
Mensagens Exclusivas Universais	MIDI Time Code (sincronismo e edição)
	MIDI Show Control
	Notation Information
	MIDI Machine Control
	MIDI Tuning Standart
	Sample Dump Standart
	General MIDI

Fonte: Ratton (2005, p.28).

Para fins de manipulação dessas mensagens, são utilizados os sistemas binário e hexadecimal. A maioria das mensagens MIDI, à exceção das mensagens de *SysEx*, trabalha de um a três *bytes* para a codificação dessas mensagens. O primeiro é o *byte* de status, que identifica qual a mensagem; os demais, quando houver, são os *bytes* de dados. (RATTON, 2005, p.33)

Dos oito *bits* de cada *byte*, apenas sete deles vão transmitir dados. Sendo assim, com sete *bits* para manipulação dos dados, um padrão de faixa de valores para os equipamentos musicais é representado através dos valores 0 e 127. (RATTON, 2005, p.33).

A transmissão dos dados é efetuada da forma serial, ou seja, de bit a bit. Contudo, a taxa de transferência é de 31.250 *bps* (bits por segundo), um tempo considerado rápido para o padrão MIDI. A razão principal para a transmissão ser via serial são os custos envolvidos. (RATTON, 2005, p.36).

A transmissão física é feita através de cabo MIDI DIN, de cinco pinos, sendo que os equipamentos possuem três tipos de portas MIDI: MIDI IN, por onde as mensagens são recebidas; MIDI OUT, por onde as mensagens são enviadas; e MIDI THRU, que atua como uma ponte entre um MIDI OUT e um MIDI IN. Alguns equipamentos possuem ambas as portas MIDI (IN, OUT e THRU), outros somente MIDI OUT, outros, várias portas MIDI IN e várias MIDI OUT e outros, nenhuma delas. Isso vai depender do tipo de equipamento em questão. (RATTON, 2005, p.40).

A forma de transferência de dados MIDI é *half-duplex*, ou seja, unilateral. Por regra, liga-se a porta MIDI OUT de um transmissor, chamado de *master* (mestre), à porta MIDI IN de um receptor, chamado de *slave* (escravo). Dependendo dos equipamentos envolvidos e das portas disponíveis, várias formas de conexão podem ser efetuadas, desde ligação entre dois

equipamentos, até a ligação entre vários instrumentos simultâneos junto a um computador, via *PatchBay*, que é uma espécie de *hub* MIDI. (RATTON, 2005, p.48).

A transmissão de dados MIDI pode ser feita através de outros meios, como as portas Serial-Com (do tipo *mini-din*), USB, SCSI, mLAN, via conexão virtual (entre dois softwares dentro de um mesmo computador) e via wireless. (RATTON, 2005, p.70).

Independente de equipamento, uma tabela de eventos de recepção e transmissão de dados MIDI deve ser reconhecida por esse equipamento, que é chamada de *MIDI Implementation Chart*. (RATTON, 2005, p.353). Essa tabela é mostrada na Figura 16.

Figura 16 – Exemplo de MIDI Implementation Chart

Function		Transmitted	Recognized
Basic Channel	Default Changed	1 1~16	1~16 1~16
Mode	Default Messages Altered	 × **********	×
Note Number:	 True voice	0~127 **********	×
Velocity	Note ON Note OFF	○ v=0~127 ○ v=0~127	× ×
Aftertouch	Key's Ch's	× ○	× ×
Pitch Bend		○	×
Control Change		0~127	○
Prog Change:	 True #	○ **********	×
System Exclusive		○	○
System Real Time	Clock Commands	○ ○	× ×
System command	Song position	○	×
Aux Messages	Active Sense	○	×

○: Yes ×: No

Fonte: Do autor.

2.7.4.3 Timbres

Em relação aos sons que podem ser trafegados via MIDI, há uma padronização chamada General MIDI (GM), que especifica uma lista de 128 timbres pré-fabricados, comuns aos equipamentos que utilizam o padrão GM, bem como para os timbres de bateria e percussão. (RATTON, 2005, p.179). As Figura 17, 18 e 19 mostram essas listas.

Figura 17 – Mapa de Timbres GM

Fonte: Do autor.

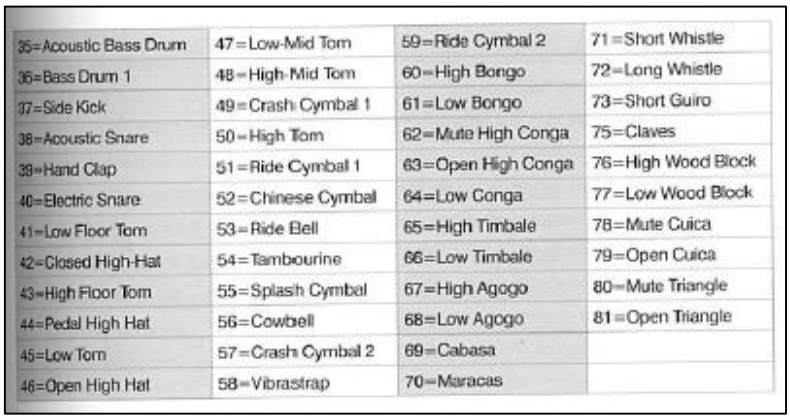
Caso o instrumento possua mais de 128 timbres, a contagem numérica é reiniciada e, internamente, o equipamento sabe distinguir que o primeiro timbre 1 não é igual ao próximo timbre 1. (RATTON, 2005, p.179).

Figura 18 – Mapa de Percussão GM

Fonte: Do autor.

Os timbres de percussão são fixos, não havendo necessidade de algoritmos para interpretação do número do timbre, uma vez que vão, somente, ao número 81. (RATTON, 2005, p.179).

Figura 19 – Mapa de Bateria GM



35=Acoustic Bass Drum	47=Low-Mid Tom	59=Ride Cymbal 2	71=Short Whistle
36=Bass Drum 1	48=High-Mid Tom	60=High Bongo	72=Long Whistle
37=Side Kick	49=Crash Cymbal 1	61=Low Bongo	73=Short Guiro
38=Acoustic Snare	50=High Tom	62=Mute High Conga	75=Claves
39=Hand Clap	51=Ride Cymbal 1	63=Open High Conga	76=High Wood Block
40=Electric Snare	52=Chinese Cymbal	64=Low Conga	77=Low Wood Block
41=Low Floor Tom	53=Ride Bell	65=High Timbale	78=Mute Cuica
42=Closed High-Hat	54=Tambourine	66=Low Timbale	79=Open Cuica
43=High Floor Tom	55=Splash Cymbal	67=High Agogo	80=Mute Triangle
44=Pedal High Hat	56=Cowbell	68=Low Agogo	81=Open Triangle
45=Low Tom	57=Crash Cymbal 2	69=Cabasa	
46=Open High Hat	58=Vibrastrap	70=Maracas	

Fonte: Ratton (2005, p.179).

Da mesma forma que os timbres de percussão, os timbres de percussão são fixos, não havendo necessidade de algoritmos para interpretação do número do timbre. (RATTON, 2005, p.179).

2.7.5 As novas tendências

Segundo Ratton (2015, p.8), a nova tendência vinculada à tecnologia MIDI são os chamados instrumentos virtuais (*VSTi's*). Um instrumento virtual é uma "cópia fiel e original" da sonoridade e características de um instrumento real. É um instrumento totalmente implementado por computador, que usa os métodos de amostragem e sampleamento para capturar o som e características de um instrumento acústico (um piano de cauda, por exemplo), sendo esse software acionado via MIDI por um equipamento controlador externo.

As grandes vantagens dos instrumentos virtuais são o custo (visto que há uma infinidade de softwares e VSTi's gratuitos na internet), a mobilidade (não há equipamentos físicos, mas, sim, equipamentos virtuais em um notebook), a diversidade (a maioria dos softwares musicais podem trabalhar com vários VSTi's abertos simultaneamente), a qualidade e a facilidade de operação. (RATTON, 2015, p.8).

Com apenas um computador, um software de áudio, alguns VSTi's e um equipamento controlador MIDI, um músico pode compor, arranjar, mixar, masterizar e gravar músicas com qualidade de estúdio, com arranjos de coral, orquestra e instrumentos típicos de bandas, incluindo os famosos sintetizadores analógicos, já mencionados neste trabalho, além de muitos outros existentes no mercado. (RATTON, 2015, p.8).

Um bom exemplo de VSTi é o Quantum EastWest Symphonic Choirs. Esse instrumento virtual é capaz de reproduzir, com incríveis níveis de realismo, um coral inteiro, seja ele infantil, uníssono (uma voz) ou com quatro ou seis naipes de vozes. Ainda, o instrumento conta com um editor interno, chamado WordBuilder, que permite ao músico escrever a letra e ensiná-la ao Symphonic Choirs, para que essa letra possa ser "cantada". (SOUNDSONLINE, 2015).

Para os amantes de violino, uma opção bastante recomendada é o 8Dio Solo Studio Violin. Ele possui mais de 400 frases executadas por violinistas profissionais, em ambientes variados, e com todo o tratamento acústico necessário. Com esse tratamento sonoro, aliado a controles de *pitch*, vibrato, *sustain*, compressão/descompressão de tempo, controles de *reverb*, rotor, equalizador, *gate*, filtros e *delay*, entre outros, oferece ao músico uma performance extremamente realista, como se estivesse tocando em um violino real. (8DIO.COM, 2015)

Já, o VSTi EZ Drummer2 herdou de seu antecessor toda uma gama de sons, timbres, kits de bateria e percussão, *mixer* e efeitos, e somou a toda essa herança novas funcionalidades. Nessa nova versão, foi inserido um controle de velocidade, sensibilidade e condução enquanto se está tocando, o que permite ao músico ajustar o instrumento virtual de acordo com a sua dinâmica, em tempo de execução. Além disso, houve uma melhoria nos controles de *mixer* e efeitos, além de uma ampliação na biblioteca de frases musicais. (TOONTRACK, 2015).

Em relação aos softwares existentes no mercado, duas opções se destacam: o Kontakt e o FLStudio. Tanto o Kontakt, quanto o FLStudio permitem gravação multicanal, sendo que, em cada canal, pode ser utilizado um VSTi diferente, com controles, efeitos e parâmetros dedicados para cada canal, dependendo do tipo de canal selecionado. Além disso, permitem a utilização de inúmeros VSTi's e a utilização de teclados controladores, tanto como teclados em si, como *interfaces* de controle do próprio *software*. (IMAGE-LINE, 2015).

Finalmente, em relação aos teclados controladores, uma das inúmeras opções existentes no mercado é o teclado controlador Novation Impulse. Tal qual outros modelos controladores, o Novation Impulse pode, além de atuar como controlador para módulos, outros teclados e instrumentos virtuais, atuar como superfície de controle para softwares, como o Kontakt, o FLStudio, Pro Tools e Logic Studio. Com controles de *knobs, faders, sliders, pitch, modulation, bender, pads* e *buttons,* oferece ao músico inúmeras possibilidades de controle em tempo real, facilitando na dinâmica e no realismo das execuções. (GLOBAL, 2015).

Com todas essas inovações, o cenário musical, bem como o número de novos aventureiros, têm aumentado consideravelmente. Entretanto, Luiz Schiavon, tecladista do

RPM, comenta sobre o novo cenário musical (MENA, [1997?], p.36), principalmente sobre as composições musicais, com a seguinte opinião:

> [...] Hoje em dia, vejo passar pelo meu estúdio uma série de músicos que até uma certa dificuldade em tocar. Muitos não têm técnica pianística nenhuma, mas comandam o computador muito bem e arranjam com facilidade, concluindo seus trabalhos tão bem quanto qualquer outro músico mais técnico ao tocar. Na verdade, o software ampliou a gama de músicos no mercado. Hoje em dia, temos uma nova categoria de músico: o "micreiro", que não sabe tocar nada, mas compõe, arranja e faz tudo baseado no micro... [...].

Segundo essa visão, músicos amadores podem efetuar trabalhos relativamente bons, mesmo sem grande conhecimento teórico musical. Contudo, esses mesmos músicos podem realizar trabalhos sem a qualidade mínima necessária, resultando em prejuízos ao próprio músico. Além disso, esse novo tipo de "músico" se insere no mercado musical, cobrando, na maioria das vezes, valores abaixo do preço de mercado, o que impacta em prejuízos a músicos mais profissionais, saturando o mercado de trabalho. E esse é o principal ponto negativo dessas novas tendências. (RATTON, 2015, p. 13).

Recentemente, um novo tipo de instrumento musical vem chamando à atenção de músicos e leigos curiosos, devido à forma de funcionamento: o ROLI Seaboard. Diferentemente dos controladores atuais, esse instrumento possui teclas com silicone macio e táctil, integradas a sensores específicos, permitindo a utilização de efeitos como *pitch bend, modulation, pitch* e *vibrato*, sem o uso de dispositivos extras. (GEAR4MUSIC.PT, 2015).

Essa nova tecnologia permite execuções ainda mais realistas, com a execução de efeitos e modulações para cada nota individualmente, interagindo com a pressão e sensibilidade aplicada à nota. Além disso, traz as mesmas funcionalidades físicas e lógicas encontradas em um piano, por exemplo, mas sem a estrutura física de um teclado com sistema de teclas do tipo martelo. Ou seja, o peso do instrumento reduz consideravelmente, chegando a ser, em alguns casos, três vezes mais leve que um teclado convencional de 88 teclas, muito usado por pianistas e tecladistas em seus *setups*. (GEAR4MUSIC.PT, 2015).

As Figuras 20, 21 e 22 mostram alguns desses VSTi's. Já, as Figuras 23 e 24, dois dos softwares mais utilizados para rodar esses VSTi's. A Figura 25 mostra o teclado controlador ROLI Seaboard nas versões 25, 61 e 88 teclas. (GEAR4MUSIC.PT, 2015) Finalmente, a Figura 26 mostra outro modelo de controlador.

Figura 20 – VSTi Quantum EastWest Symphonic Choirs

Fonte: Do autor.

O VSTi's Symphonic Choirs foi desenvolvido a partir do uso da técnica de sampleamento, e é um simulador de coral. Esse instrumento virtual é capaz de reproduzir frases e letras inteiras, tanto em uníssono (masculino, feminino, infantil masculino e infantil feminino), quanto a quatro ou a seis vozes.

Figura 21 – VSTi 8Dio Solo Studio Violin

Fonte: Do autor.

Já, O VSTi's 8Dio SS Violin apresenta uma dinâmica quase que perfeita de um violino Stradivarius, com todas as nuances sonoras do instrumento.

Figura 22 – VSTi EZ Drummer2

Fonte: Do autor.

Para fins de gravações de trilhas sonoras de bateria, o VSTi's EZDrummer 2 possui como principal característica a interface visual, que permite ao músico identificar qual peça está sendo tocada em tempo real, evitando, assim, a execução de trechos nos quais seis peças da bateria sejam tocadas ao mesmo tempo, por exemplo.

Figura 23 – Software Kontakt

Fonte: Do autor.

O software Kontakt possui um bom ambiente para a execução dos VSTI's, permitindo a utilização de vários instrumentos virtuais em uma mesma sessão.

Figura 24 – Software FLStudio 11

Fonte: Do autor.

O software FLStudio10, por sua vez, possui um modo de Piano Roll bastante prático e eficiente, facilitando a criação de músicas e arranjos.

Figura 25 – Teclado Controlador ROLI Seaboard GRAND

Fonte: Do autor.

Entre os teclados controladores, o Roli Seaboard vem ganhando destaque, pela questão mecânica e pela praticidade e versatilidade em qualquer tipo de timbre, com maior integração entre o músico e o instrumento.

Figura 26 – Teclado Controlador Novation Impulse 61 teclas

Fonte: Do autor.

Por fim, o teclado controlador Novation Impulse apresenta uma mecânica bastante confortável, e uma gama ampla de controles de tempo real.

Entre os softwares de gerenciamento existentes no mercado, há alguns voltados para a área musical. Esses softwares permitem o cadastro de músicas, inclusão de letras e gerenciamento de programações, além de outras funcionalidades. Entre esses softwares, destacam-se:

- ✓ **Setlist Master:** permite a criação, exportação/importação e compartilhamento de *setlists*, vinculação de arquivos de áudio nas músicas cadastradas, e impressão de repertórios. Não gerencia programações nem cadastra shows, blocos de músicas ou equipamentos;
- ✓ **BandBro':** permite a criação de setlists, vinculação de letras às músicas e indicação de tempo do *setlist*. Não gerencia programações, shows, blocos de músicas ou equipamentos;
- ✓ **iGig Book:** permite a criação de setlists, vinculação de letras, partituras e tablaturas às músicas, além de envio de comandos MIDI a um instrumento conectado através de acessório específico. Pode controlar apenas um instrumento, e é compatível apenas com iPad e iPhone (versão para Android em desenvolvimento);

- ✓ **Setlist Organizer:** permite a criação e compartilhamento de *setlists*, cadastro de timbres e envio de comandos *ProgramChange* e *Volume* para um instrumento externo, através de acessório específico. Não incorpora letras, tablaturas e partituras nem gerencia blocos de músicas. Ainda, não permite a incorporação de arquivos de áudio. Compatível apenas em iPad e iPhone;
- ✓ **MainStage:** permite a criação e compartilhamento de *setlists*, cadastro de timbres e envio de comandos *ProgramChange*, *Volume*, *Transpose* e *Pitch*, entre vários comandos, para instrumentos externos, através de acessório específico. Não incorpora letras, tablaturas e partituras nem gerencia blocos de músicas. Permite o controle de, no máximo, dois equipamentos simultâneos. Ainda, não permite a incorporação de arquivos de áudio. Compatível apenas em sistemas Mac OSX.

Pode-se notar que as ferramentas apresentadas possuem diversas funcionalidades, que auxiliam os músicos na execução dos trabalhos. Entretanto, fica evidente que não há no mercado uma ferramenta que incorpore, em um único ambiente, as funcionalidades mais utilizadas por músicos e instrumentistas, quais sejam: cadastro de shows, *setlists*, músicas, blocos de músicas, equipamentos, programações, incorporação de letras, partituras, tablaturas e arquivos de áudio às músicas, bem como acesso rápido e prático para troca de programações dentro de uma música.

3 METODOLOGIA DE PESQUISA

Segundo Prodanov (2013, p.14), a metodologia pode ser compreendida como a disciplina que analisa, descreve, estuda e avalia métodos de avaliação de pesquisas acadêmicas. Em um sentido mais aplicado, ela descreve, analisa e avalia métodos e técnicas utilizadas para a coleta de informações, com a finalidade de comprovar a utilidade e veracidade das informações.

Esse conceito de metodologia está diretamente ligado ao conceito de ciência que, segundo Lakatos e Marconi, *apud* Prodanov (2013, p.14), não é somente uma sistematização de conhecimentos, mas, também, um conjunto de proposições lógicas, ligadas diretamente a fenômenos relacionados a determinado assunto que se deseja estudar. Dessa forma, é atingido o conhecimento, que é o conjunto de informações sobre determinado assunto.

Então, o método científico pode ser considerado como o conjunto de ferramentas ou procedimentos utilizados para se alcançar o conhecimento buscado pela metodologia. (PRODANOV, 2013, p.24).

3.1 CARACTERIZAÇÃO DO TIPO DE PESQUISA

Para Prodanov (2013, p.42), pesquisa é a realização de um estudo sistemático e aprofundado sobre um determinado assunto, com o intuito de coletar informações relevantes e, dessa forma, obter respostas para questões relacionadas ao método científico. Ainda, para o mesmo autor, toda pesquisa parte de uma teoria, que deve ser estudada.

Há inúmeros tipos de pesquisa, que serão diferentes de acordo com o assunto abordado, quais sejam: enfoque dado, interesses, campos, metodologias, situações e objetos de estudo. (PRODANOV *et al.*, 2013, p.43).

Já, critérios como tempo disponível para realização, espaço onde será realizado, recursos materiais necessários e recursos humanos disponíveis estão entre aqueles relacionados ao planejamento da pesquisa, ou seja, a estruturação básica necessária antes do início de uma pesquisa científica. (PRODANOV *et al.*, 2013, p.43).

Lakatos e Marconi (2007, *apud* PRODANOV *et al.*, 2013, p.48) se referem à pesquisa como um procedimento sistemático, que fornece um caminho definido. Sendo assim, uma sequência lógica de passos é adotada. Esses passos são os seguintes: preparação da pesquisa (definição do tema, preparação dos recursos necessários e elaboração de teorias e hipóteses), trabalho de campo (coleta dos dados), processamento dos dados (com a

sistematização e classificação dos mesmos), análise e interpretação dos dados (gerando informação) e elaboração do relatório de pesquisa (conclusões finais sobre a pesquisa efetuada).

Para Rodrigues (2007, p.5), os tipos de pesquisas que podem ser adotados são:

1. **quanto à área da ciência:** pesquisa teórica, pesquisa metodológica, pesquisa empírica e pesquisa prática;
2. **quanto à natureza:** trabalho científico original e resumo de assunto;
3. **quanto aos objetivos:** pesquisa exploratória, pesquisa descritiva e pesquisa explicativa;
4. **quanto aos procedimentos:** pesquisa de campo e pesquisa de fonte de papel;
5. **quanto ao objeto:** pesquisa bibliográfica, pesquisa de laboratório e pesquisa de campo;
6. **quanto à forma de abordagem:** pesquisa quantitativa e pesquisa qualitativa.

Quanto à área da ciência, o presente trabalho encaixa-se no tipo prática ou aplicada, pois se trata do desenvolvimento de uma solução de software para a área musical. O embasamento teórico, visto até agora, servirá como base para o desenvolvimento prático da solução proposta, para o atendimento de interesses locais. (PRODANOV, 2013, p.51).

Em relação à natureza, é de caráter de resumo de assunto, pois é realizada uma revisão bibliográfica dos tópicos que serão utilizados no desenvolvimento da solução proposta, com base em soluções já existentes. (PRODANOV, 2013, p.51).

No caso dos objetivos, a pesquisa exploratória é a que mais se aproxima do presente trabalho, haja vista que são realizados estudos e coleta de informações quanto ao assunto pesquisado, com a finalidade de definir o enfoque que será dado ao tema. Ainda, esse tipo de pesquisa envolve entrevistas com pessoas que passam pelo mesmo tipo de problema. (PRODANOV, 2013, p.52).

Para os procedimentos, ambos são utilizados, ou seja, a pesquisa em livros e mídias e a aplicação de um questionário de avaliação, uma vez que, segundo Prodanov (2013, p.54), a coleta de dados é a parte fundamental para a construção de uma solução e, quanto mais dados puderem ser coletados, melhor será a solução proposta.

A pesquisa bibliográfica foi o objeto adotado para o presente projeto, pois o foco será na coleta de informações, partindo de materiais impressos e existentes na internet, de forma

a documentar formalmente os aspectos envolvidos, com a observância das advertências feitas por Prodanov quanto à confiabilidade e veracidade do material pesquisado. (2013, p.54).

Finalmente, quanto à forma de abordagem, será feita pesquisa qualitativa, com base em um estudo de caso, haja vista que, para o tipo de problema envolvido, não é interessante efetuar uma pesquisa quantitativa, pois o foco não está nos números em si, mas, sim, na relação entre o problema e o sujeito envolvido. (PRODANOV, 2013, p.70).

3.2 ETAPAS

A organização de um projeto em etapas, além de facilitar o entendimento do problema e da solução a ser adotada, fornece ao pesquisador uma sequência lógica e fundamentada para a realização dos objetivos propostos. Para Prodanov (2013, p.73), são três as principais fases de um projeto, nas quais as etapas deverão ser ajustadas: decisória (escolha do tema, definição e delimitação do problema), construtiva (construção e execução do plano de pesquisa) e redacional (análise dos dados obtidos na fase construtiva, mediante organização das idéias sistematicamente e dentro de normas acadêmicas).

Para o presente projeto, foram adotadas as seguintes etapas, que estão detalhadas no Apêndice A:

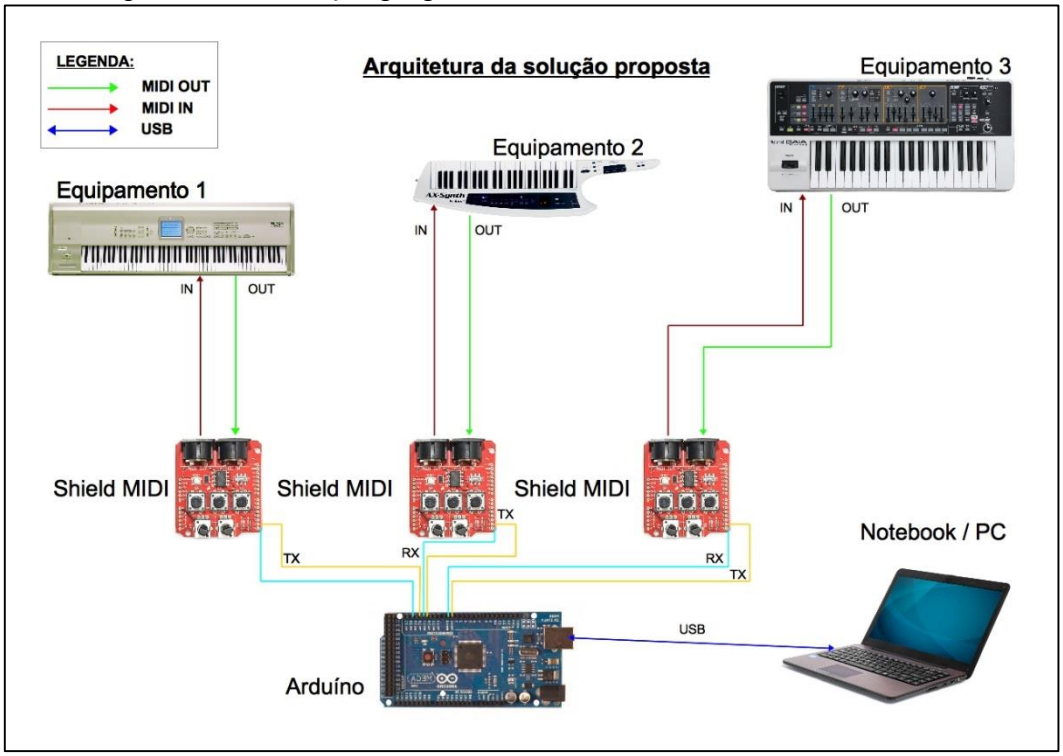
- ✓ definição e delimitação do problema;
- ✓ definição do tema;
- ✓ definição da orientadora;
- ✓ levantamento de informações sobre o problema;
- ✓ levantamento de requisitos;
- ✓ modelagem do problema;
- ✓ instalação de softwares e aplicativos necessários;
- ✓ estudo dos softwares e aplicativos envolvidos;
- ✓ coleta de material escrito necessário;
- ✓ pesquisa de referências de internet;
- ✓ elaboração do Capítulo 1;
- ✓ elaboração do Capítulo 2;
- ✓ elaboração do Capítulo 3;
- ✓ elaboração do Capítulo 4;
- ✓ elaboração do Capítulo 5;

- ✓ elaboração do Capítulo 6;
- ✓ estudo sobre comunicação serial;
- ✓ estudo sobre protocolo MIDI, pedaleiras controladoras, Arduíno e *shields,* para posterior desenvolvimento de hardware;
- ✓ desenvolvimento do software;
- ✓ avaliação do software a partir do seu uso;
- ✓ preparação do material para a defesa do TCC (slides);
- ✓ finalização da monografia e simulação de defesa do projeto;
- ✓ defesa da monografia;
- ✓ correções da monografia apresentada;
- ✓ entrega da monografia com as correções.

3.3 ARQUITETURA DA SOLUÇÃO PROPOSTA

Na Figura 27, é apresentada a proposta de solução do problema, bem como uma breve apresentação dos pontos envolvidos. Tanto a placa Arduíno e *shields* MIDI quanto os equipamentos são meramente ilustrativos, e farão parte de posterior desenvolvimento.

Figura 27 – Arquitetura da solução proposta



Fonte: Do autor.

- ✓ **equipamento:** qualquer equipamento ou instrumento musical que possua as conexões MIDI IN e OUT (teclados, controladores, sintetizadores, *workstations*, pedaleiras de voz e efeitos e módulos de som);
- ✓ ***shield* MIDI:** componente que recebe as informações de timbres e canais MIDI transmitidas pelo Arduíno e envia aos respectivos equipamentos conectados ;
- ✓ **Arduíno:** recebe as informações de timbres e canais MIDI oriundas do software e as repassa aos Shields MIDI, além de controlar os pedais que acionam as funções do software;
- ✓ **notebook / PC:** possui o software ora desenvolvido e envia as informações de timbres e canais MIDI ao Arduíno.

3.4 DELIMITAÇÕES

Especificações técnicas e detalhes mais aprofundados da tecnologia MIDI não serão abordados na revisão bibliográfica do presente projeto.

O software não fará comunicação com outros aplicativos de áudio, nem permitirá a utilização de plugins ou VSTi's externos.

O software não será um serviço web. Inicialmente, será apenas um aplicativo web local, com o servidor de aplicação e banco de dados na própria máquina na qual o software estiver rodando.

Inicialmente, foi proposto o desenvolvimento de um protótipo de pedaleira controladora MIDI. Entretanto, devido a problemas técnicos, esse protótipo não mais faz parte do presente projeto, ficando como uma das opções para trabalhos futuros.

O software desenvolvido não rodará, inicialmente, em ambientes MacOSX, com as mesmas especificações utilizadas no ambiente Windows. Para rodar em Mac, a versão do sistema operacional deve ser a 10.7, ou superior, sendo, que a versão 10.6, oferece suporte, apenas, ao Apache Tomcat 6.

Quanto à revisão bibliográfica sobre sistemas embarcados, também não será contemplada na revisão bibliográfica do presente trabalho.

4 MODELAGEM

Para Guedes (2015, p.13), a UML (*Unified Modeling Language*, ou Linguagem de ModelagemUnificada) não é uma linguagem de programação, mas, sim, uma linguagem de modelagem para software, envolvendo características físicas, de relacionamento entre objetos, comportamentos, requisitos e regras de negócio, entre outros aspectos.

Essa modelagem utiliza-se de diversos diagramas, que são visões diferentes do sistema, de modo a facilitar a compreensão do funcionamento desse software, auxiliando, ainda, na localização de eventuais falhas do sistema. (GUEDES, 2015, p.13).

4.1 MODELAGEM DE SOFTWARE

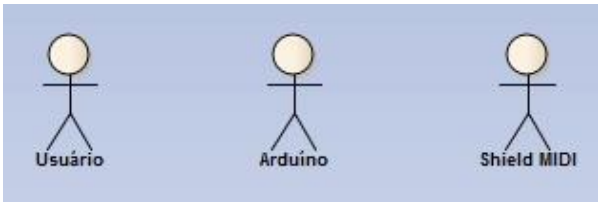
Para uma melhor visualização do software, os seguintes diagramas são utilizados:

- ✓ **Atores:** representam os personagens que efetuam alguma interação;
- ✓ **Requisitos funcionais:** apresentam as principais funções que o sistema deve apresentar;
- ✓ **Requisitos não-funcionais:** descrevem recursos adicionais que o software deve apresentar;
- ✓ **Regras de negócio:** Mostram as características e comportamentos de cada requisito funcional;
- ✓ **Casos de uso:** apresentam as principais interações dos atores envolvidos;
- ✓ **Diagrama de tabelas:** Mostram a interação entre as tabelas relacionais, bem como as respectivas granularidades;
- ✓ **Diagrama de sequência:** mostra o passo-a-passo de certas atividades, de modo a facilitar a compreensão do funcionamento das mesmas.

4.1.1 Atores

Guedes (2015, p. 15) explica que os atores são os personagens que efetuarão alguma interação com o sistema. Esses atores são mostrados na Figura 28.

Figura 28 – Atores



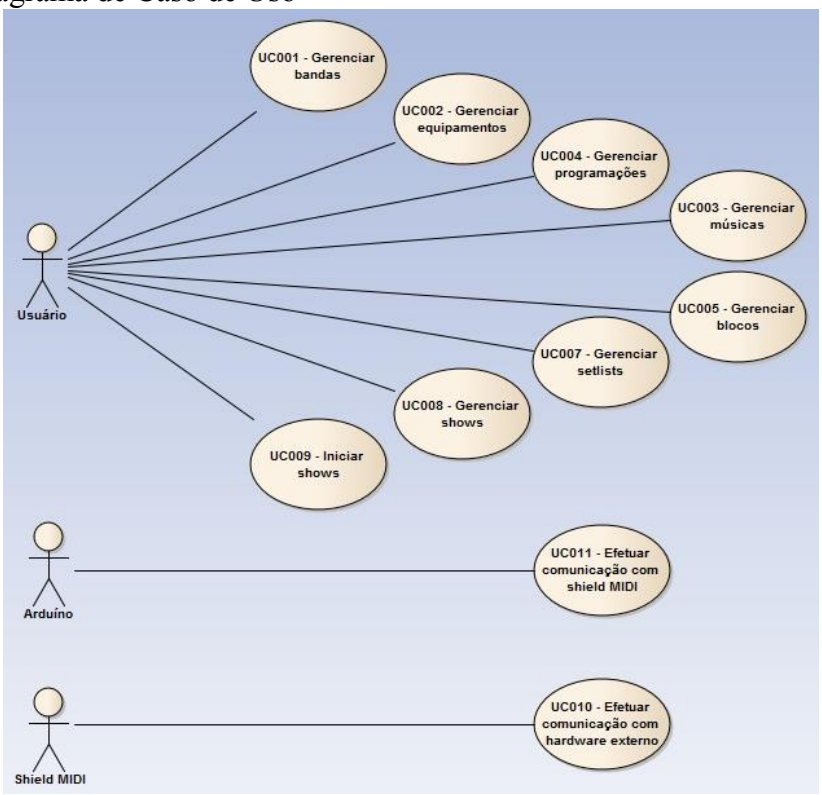
Fonte: Do autor.

O usuário utiliza o sistema, mediante *login* e senha, e efetua todas as manipulações necessárias e disponíveis. O Arduíno, quando acionado, recebe e envia as informações das programações aos *shields* MIDI. Já, os *shields* MIDI recebem as programações e as repassam aos instrumentos e equipamentos externos.

4.1.2 Casos de Uso

Guedes (2015, p. 15) informa que o Diagrama de Casos de Uso tem por finalidade facilitar a visualização do comportamento do sistema, como um todo, bem como das interações dos atores envolvidos com esse sistema. Na Figura 29, é apresentado o respectivo diagrama.

Figura 29 – Diagrama de Caso de Uso



Fonte: Do autor.

O usuário realiza as principais interações dentro do sistema: cadastro de bandas, músicas, blocos de músicas programações, equipamentos, *setlists* e shows. Ao Arduíno, cabe efetuar a comunicação entre ele e o *shield* MIDI. Já, ao *shield* MIDI cabe a comunicação entre ele e os equipamentos e instrumentos conectados.

4.1.3 Requisitos

Segundo Wazlawick (2011, p. 29), é na análise de requisitos que o analista de software tem uma idéia melhor de como o sistema deverá funcionar, com base nos requisitos e regras de negócio levantadas junto ao cliente. A seguir, são relacionados os requisitos funcionais, não funcionais e regras de negócio existentes no presente projeto.

4.1.3.1 Requisitos Funcionais

No Quadro 2, são apresentados os requisitos funcionais do presente projeto.

Quadro 2 – Requisitos Funcionais

CÓDIGO	NOME	DESCRIÇÃO
RF001	Efetuar Login	O sistema deverá permitir autenticação do usuário, através de *login* e senha
RF002	Cadastrar Novo Usuário	O sistema deverá permitir o cadastro de um novo usuário, no caso de primeiro acesso
RF003	Editar Dados do Usuário	O sistema deverá permitir a edição dos dados de um usuário, após logado no sistema.
RF004	Visualizar Dados do Usuário	O sistema deverá permitir a visualização dos dados de um usuário, após logado no sistema.
RF005	Gerenciar Bandas	O sistema deverá permitir o cadastro, edição, remoção e visualização dos dados de uma banda
RF006	Gerenciar Equipamentos	O sistema deverá permitir o cadastro, edição, remoção e visualização dos equipamentos do usuário
RF007	Gerenciar Músicas	O sistema deverá permitir o cadastro, edição, remoção e visualização das músicas
RF008	Gerenciar Programações	O sistema deverá permitir o cadastro, edição, remoção e visualização das programações utilizadas nos equipamentos previamente cadastrados
RF009	Manipular Programações	O sistema deverá permitir a inclusão, exclusão ou alteração de programações em uma música
RF010	Gerenciar Blocos	O sistema deverá permitir o cadastro, edição, remoção e visualização de blocos

RF011	Manipular Blocos	O sistema deverá permitir a inclusão, exclusão ou alteração de músicas em um bloco
RF012	Gerenciar Setlists	O sistema deverá permitir o cadastro, edição, remoção e visualização de *setlists* (repertórios)
RF013	Manipular Setlists	O sistema deverá permitir a inclusão, exclusão ou alteração de músicas ou blocos em um *setlist* (repertório)
RF014	Gerenciar Shows	O sistema deverá permitir o cadastro, edição, remoção e visualização de shows
RF015	Manipular Shows	O sistema deverá permitir a inclusão, exclusão ou alteração de bandas e *setlists* em um show
RF016	Carregar Show	O sistema deverá permitir a carga dos dados de um show para o módulo de execução
RF017	Executar Show	O sistema deverá permitir a troca dinâmica de blocos, músicas e/ou programações, em tempo real, avançando ou retrocedendo esses blocos/músicas/programações
RF019	Visualizar letras	O sistema deverá permitir que o usuário exiba ou oculte a letra de uma música executada, em tempo real.
RF020	Selecionar blocos e músicas	O sistema deverá permitir a seleção aleatória de um bloco ou música, em tempo real

Fonte: Do Autor.

Os requisitos funcionais trazem os principais recursos que o software deve possuir, de modo a satisfazer as necessidades mínimas do usuário.

4.1.3.2 Requisitos Não-Funcionais

No Quadro 3, são relacionados os requisitos não-funcionais do presente projeto.

Quadro 3 – Requisitos Não-Funcionais

CÓDIGO	NOME	DESCRIÇÃO
RNF001	Permitir envio de *login* e senha por e-mail	O sistema deverá permitir o envio de *login* e senha de um usuário por e-mail, após três tentativas frustradas de acesso, mediante informação de endereço de e-mail
RNF002	Utilizar Banco de Dados PostgreSQL	O sistema deverá persistir os dados em um banco PostgreSQL
RNF003	Utilizar Servidor de Aplicação Tomcat 7	O sistema deverá utilizar o servidor de aplicação web Apache TomCat 7
RNF004	Utilizar JSP e HTML	O sistema deverá rodar em ambiente web
RNF005	Permitir carga dos dados do show em memória	O sistema deverá efetuar a carga dos dados de um show cadastrado em memória, sem efetuar acesso ao banco de dados durante o módulo de execução

Fonte: Do Autor.

Os requisitos não-funcionais são restrições ou detalhes que permitem melhor funcionamento do sistema, complementando algumas exigências dos requisitos funcionais.

4.1.3.3 Regras de Negócio

As regras de negócio são descritas no Quadro 4.

Quadro 4 – Regras de Negócio

CÓDIGO	NOME	DESCRIÇÃO
RN001	Login	O *login* de um usuário será efetuado mediante *login* e senha. Ainda, a tela de *login* oferecer opções para cadastro de um novo usuário, saída do sistema ou envio de senha por e-mail.
RN002	Tela Inicial	A tela inicial deverá conter botões para o cadastro de bandas, equipamentos, músicas, programações, blocos, *setlists* e shows. Ainda, deverá ter a indicação de *login*, configurações do sistema, dados de usuário e saída do sistema.
RN003	Bandas	As telas de cadastro e edição dos dados de uma banda deverão ter o nome da banda, descrição, integrantes e página do Facebook. Já, a tela de visualização deverá ter botões específicos para ver os dados de descrição, integrantes e acesso direto à página da banda no Facebook
RN004	Equipamentos	As telas de cadastro e edição dos dados dos equipamentos deverão ter a marca, modelo, quantidade de teclas e canal MIDI
RN005	Programações	As telas de cadastro e edição das programações deverão ter o número da programação, nome do timbre e registro, além de permitir a vinculação a um equipamento previamente cadastrado
RN006	Músicas	As telas de cadastro e edição dos dados de uma música deverão ter o número, nome da música, nome do artista e letra. Já, a tela de visualização deverá ter botão específico para ver a letra da música, além de uma opção para vincular, desvincular e/ou editar uma programação em uma respectiva música
RN007	Blocos	As telas de cadastro e edição dos dados de um bloco deverão ter o número e nome do bloco. Já, a tela de visualização deverá ter botão específico para ver os dados de uma música específica, além de uma opção para vincular, desvincular e/ou editar uma música em um respectivo bloco
RN008	Setlists	As telas de cadastro e edição dos dados de um *setlist* deverão ter o número e nome do bloco. Já, a tela de visualização deverá ter opções de seleção para utilização de músicas ou blocos, além de vincular, desvincular e/ou editar músicas ou blocos em um respectivo *setlist*
RN009	Shows	As telas de cadastro e edição dos dados de um show deverão ter a banda, data, hora, local e informações sobre o show. Já, a

		tela de visualização deverá ter botão específico para ver as informações do show, bem como de uma opção para vincular, desvincular e/ou editar um *setlist* em um respectivo show
RN010	Módulo de Execução	A tela de execução deverá modificar os parâmetros, dependendo do *setlist*. Para o caso de *setlist* com músicas, deverá mostrar a listagem das músicas com um botão de acesso direto para cada música; música atual, com botões para retroceder e avançar a música; música anterior e próxima música; e as programações por sequencial, com botões para retroceder e avançar a programação. Já, para o caso de *setlist* com blocos, deverá mostrar a listagem dos blocos com um botão de acesso direto para cada bloco; bloco atual, com botões para retroceder e avançar o bloco; bloco anterior, e próximo bloco, a listagem das músicas do bloco atual, com um botão de acesso direto para cada música; música atual, com botões para retroceder e avançar a música; música anterior e próxima música; e as programações por sequencial, com botões para retroceder e avançar a programação. Ainda, para a música atual, deverá ter um botão dinâmico, que exiba ou oculte a letra. Finalmente, deverá haver um botão para retorno à tela inicial do sistema.

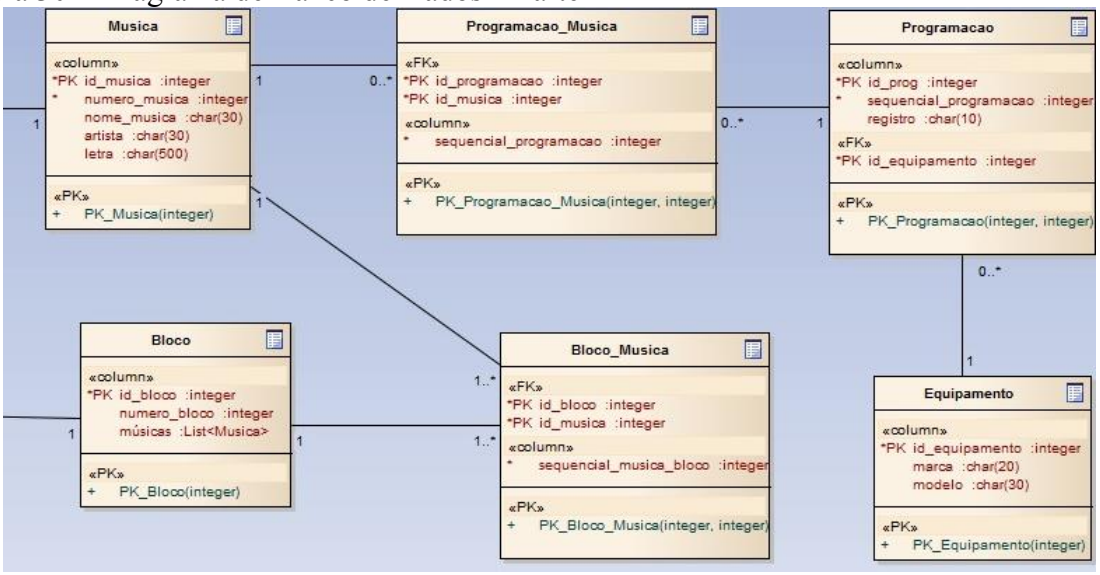
Fonte: Do Autor.

As regras de negócio descrevem de que forma funcionalidades do software são executadas e como os dados são manipulados.

4.1.4 Diagrama de Banco de Dados

Os Diagramas de Banco de Dados apresentam a estruturação das tabelas geradas pelo software, bem como as granularidades envolvidas. Eles permitem melhor visualização do relacionamento entre as classes do sistema. O respectivo diagrama é mostrado nas Figuras 30 e 31. (GUEDES, 2015, p. 17).

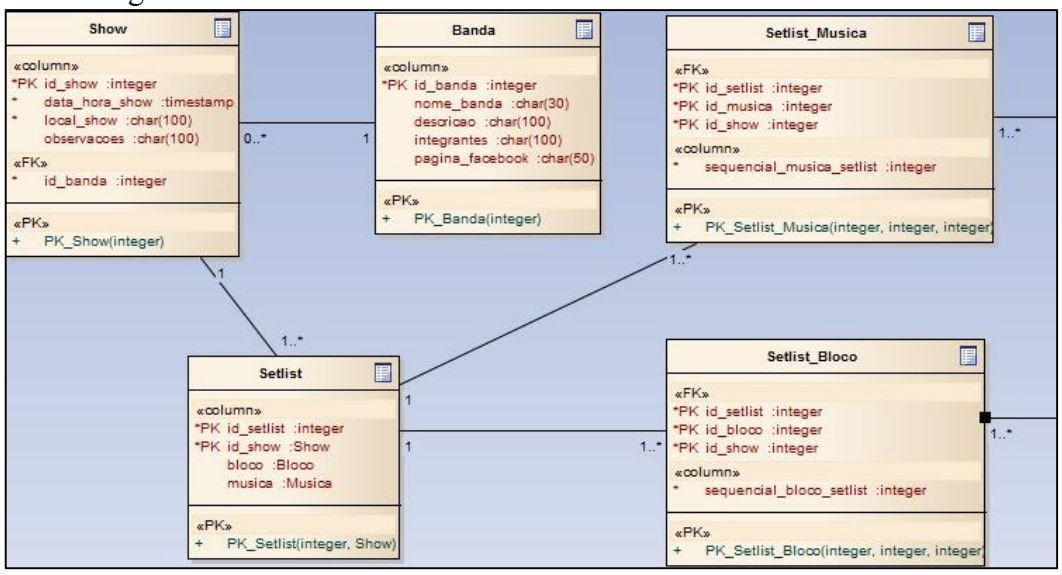
Figura 30 – Diagrama de Banco de Dados – Parte 1



Fonte: Do autor.

As entidades Música, Programação, Equipamento e Bloco, bem como os respectivos relacionamentos entre elas, são apresentados na figura acima, destacando-se o relacionamento entre Música-Programação e Bloco-Música.

Figura 31 – Diagrama de Banco de Dados – Parte 2



Fonte: Do autor.

Na figura supra, destacam-se as classes Setlist, Banda e Show, sendo que a entidade Setlist se relaciona com as entidades Música e Bloco, dependendo do tipo de *setlist* utilizado, que pode ser somente com músicas ou com blocos de músicas.

O atributo *midiChannel* da entidade *Equipamento* recebe um canal MIDI, que corresponde ao canal MIDI do instrumento conectado a uma porta de uma eventual pedaleira controladora. Já, o atributo *registro* da classe *Programacao* recebe a programação específica do equipamento.

Tanto o *midiChannel* quanto o *registro* são os principais parâmetros que a placa Arduíno recebe e transmite aos equipamentos conectados aos *shields*. Os atributos referentes às funções dos pedais são transmitidos somente à placa Arduíno.

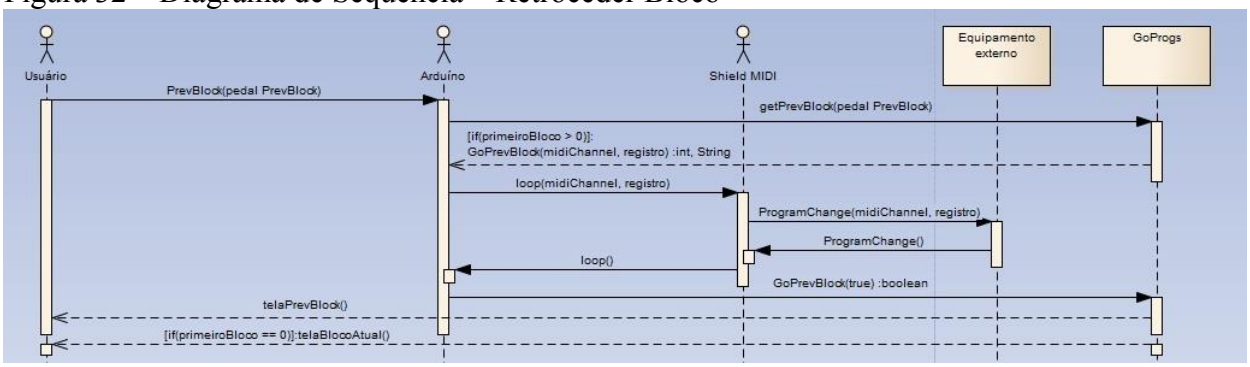
4.1.5 Diagrama de Sequência

Os Diagramas de Sequência mostram, com maior facilidade, as trocas de mensagens entre os atores e os objetos do sistema. Esses diagramas facilitam na compreensão de como determinada funcionalidade do sistema irá atuar. (GUEDES, 2015, p. 17).

Nas Figuras 32, 33, 34, 35, 36 e 37, são apresentados os Diagrama de Sequência do módulo principal do software, que é efetivamente utilizado durante um show. Os demais módulos e funcionalidades do software, por serem módulos básicos de CRUD e não apresentarem grande complexidade, não são mostrados através dos Diagramas de Sequência.

Quanto à placa Arduíno, ao *shield* MIDI e ao equipamento externo, cabe ressaltar que ambos foram inseridos nos diagramas apenas como exemplo de eventual aplicação prática. Todos os exemplos mostrados a seguir partem do princípio de que está sendo utilizado um *setlist* com blocos de músicas.

Figura 32 – Diagrama de Sequência – Retroceder Bloco



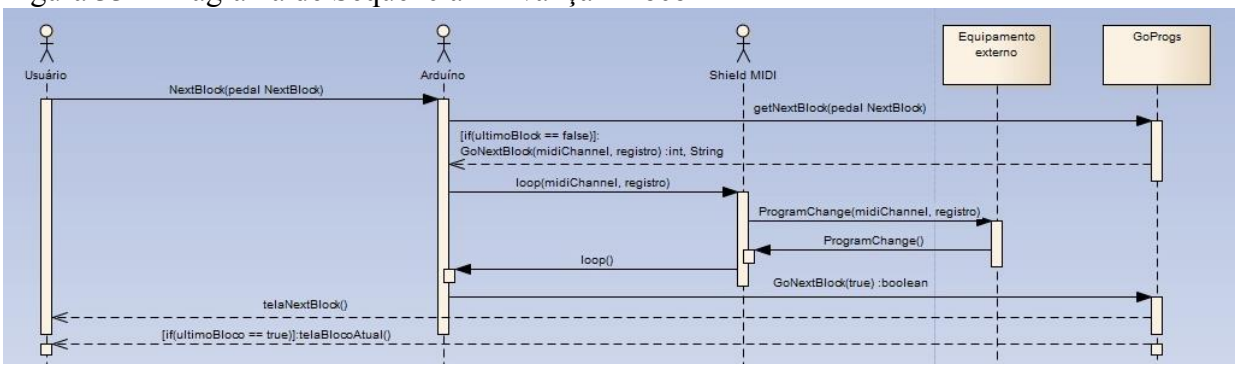
Fonte: Do autor.

No caso acima, o usuário solicita o bloco anterior. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica se o bloco atual é o primeiro bloco.

Caso seja o primeiro bloco, permanece no bloco atual; se não for o primeiro bloco, é acionada a função para retroceder o bloco, o bloco anterior se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

Figura 33 – Diagrama de Sequência – Avançar Bloco



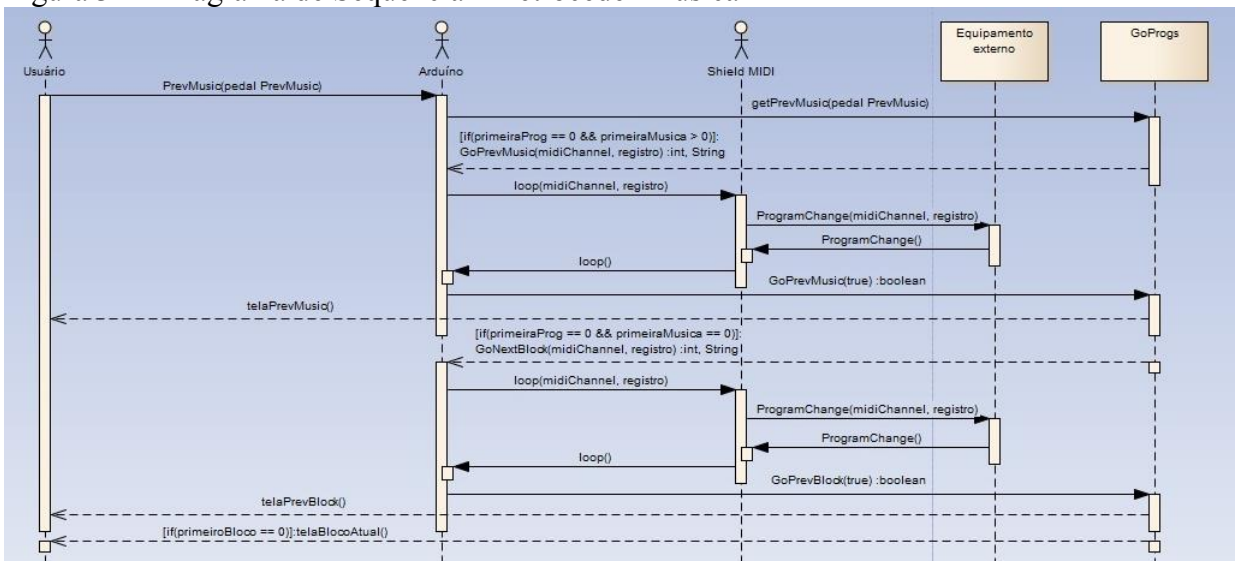
Fonte: Do autor.

Já, no caso de avançar um bloco, o usuário solicita o próximo bloco. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica se o bloco atual é o último bloco.

Caso seja o último bloco, permanece no bloco atual; se não for o último bloco, é acionada a função para avançar o bloco, o próximo bloco se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

Figura 34 – Diagrama de Sequência – Retroceder Música



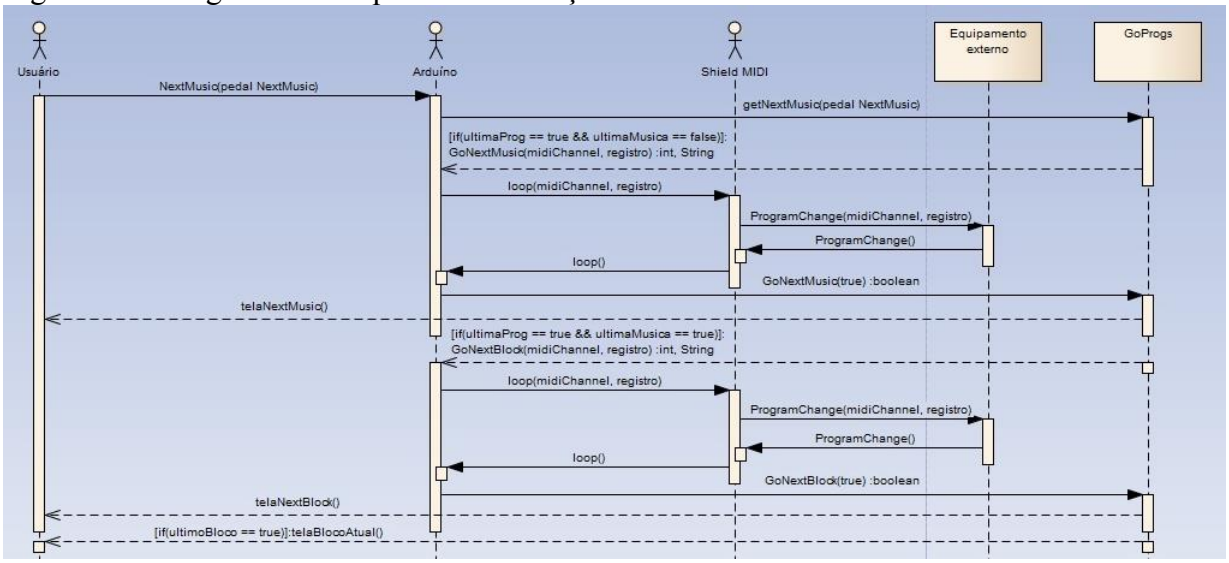
Fonte: Do autor.

No caso acima, o usuário solicita a música anterior. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica duas possíveis situações:

- ✓ **Se a primeira programação da música for igual que 0 e a primeira música do bloco for maior que 0:** é acionada a função para retroceder a música, e a música anterior se torna a música atual;
- ✓ **Se a primeira programação da música for igual que 0 e a primeira música do bloco for igual a 0:** é acionada a função para retroceder o bloco, o bloco anterior se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

Figura 35 – Diagrama de Sequência – Avançar Música



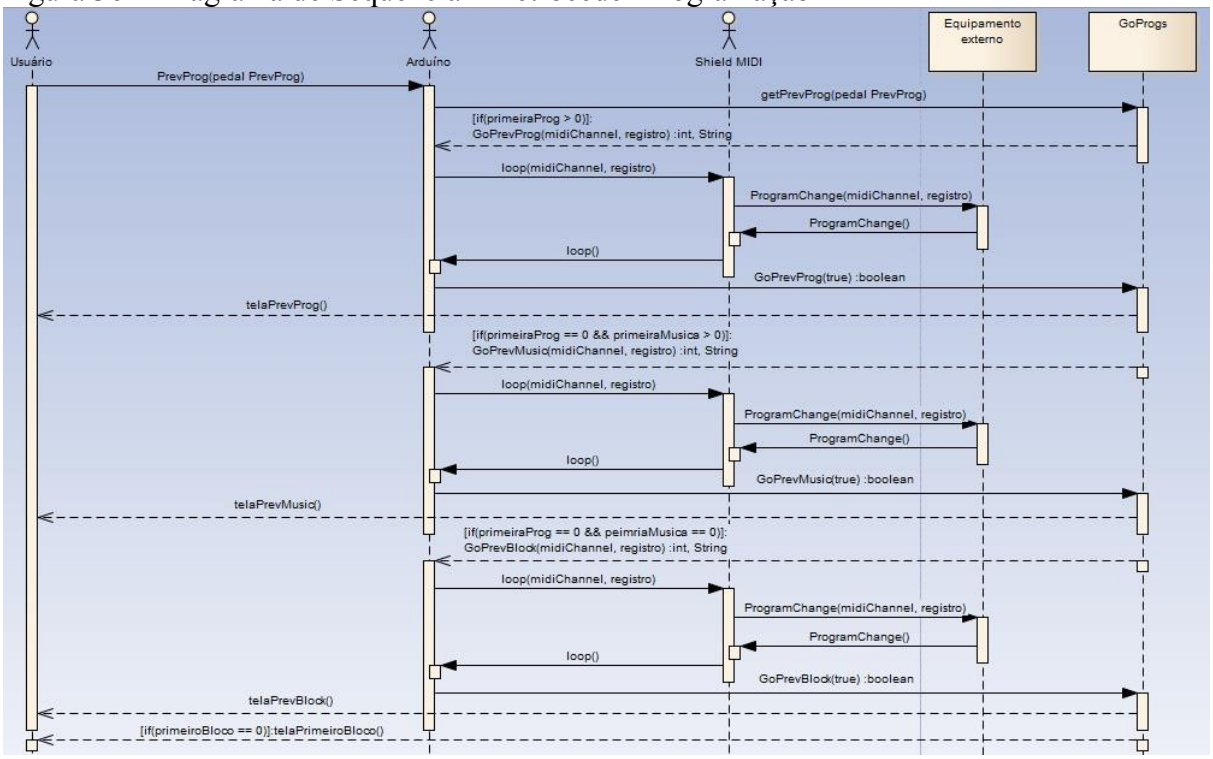
Fonte: Do autor.

No caso acima, o usuário solicita a próxima música. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica duas possíveis situações:

- ✓ **Se for a última programação da música atual e se não for a última música do bloco atual:** é acionada a função para avançar a música, e a próxima música se torna a música atual;
- ✓ **Se for a última programação da música atual e se for a última música do bloco atual:** é acionada a função para avançar o bloco, o próximo bloco se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

Figura 36 – Diagrama de Sequência – Retroceder Programação



Fonte: Do autor.

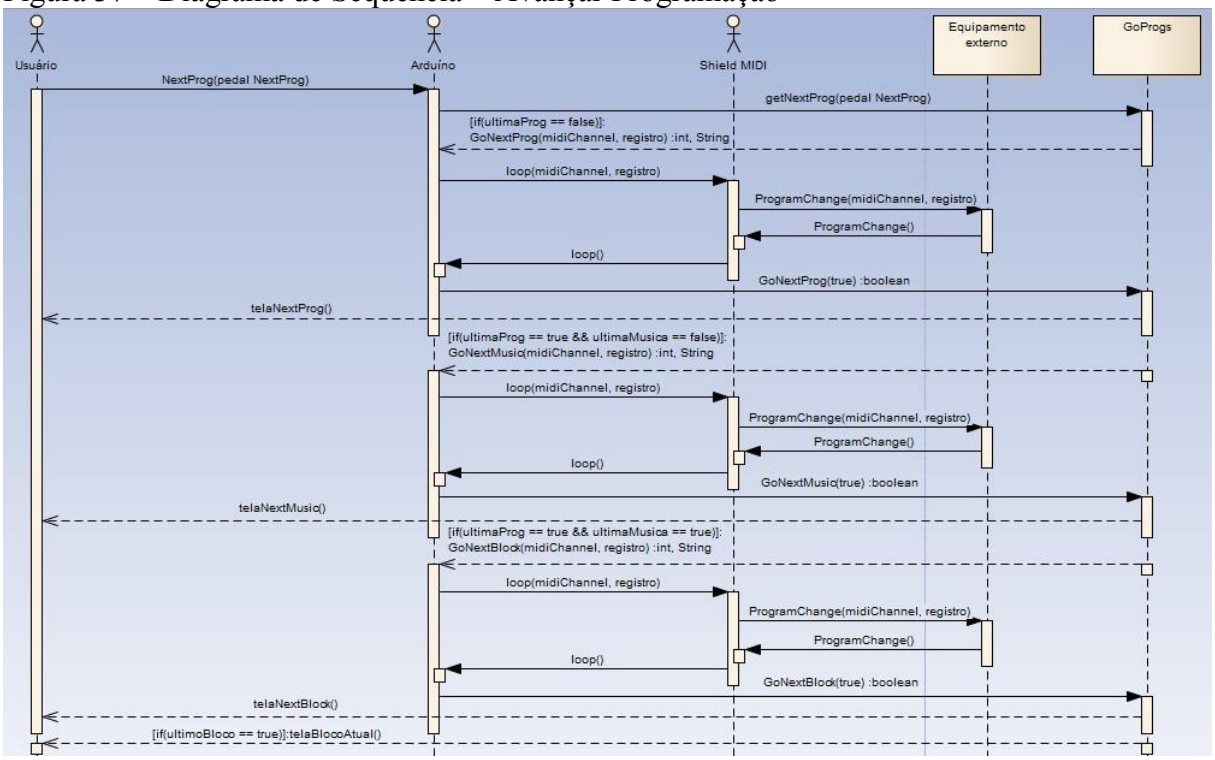
No caso acima, o usuário solicita a programação anterior. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica três possíveis situações:

- ✓ **Se a primeira programação da música for maior que 0:** é acionada a função para retroceder somente as programações da música atual;
- ✓ **Se a primeira programação da música for igual que 0 e a primeira música do bloco for maior que 0:** é acionada a função para retroceder a música, e a música anterior se torna a música atual;
- ✓ **Se a primeira programação da música for igual que 0 e a primeira música do bloco for igual a 0:** é acionada a função para retroceder o bloco, o bloco anterior se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas

de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

Figura 37 – Diagrama de Sequência – Avançar Programação



Fonte: Do autor.

No caso acima, o usuário solicita a próxima programação. A placa Arduíno reconhece que o botão/pedal específico foi pressionado, e envia a solicitação ao software, via comunicação serial. O software verifica três possíveis situações:

- ✓ **Se não for a última programação da música:** é acionada a função para avançar somente as programações da música atual;
- ✓ **Se for a última programação da música atual e se não for a última música do bloco atual:** é acionada a função para avançar a música, e a próxima música se torna a música atual;
- ✓ **Se for a última programação da música atual e se for a última música do bloco atual:** é acionada a função para avançar o bloco, o próximo bloco se torna o bloco atual e a música atual é a primeira música do novo bloco.

Após a checagem, o software envia para a placa Arduíno as informações referentes aos timbres e aos canais MIDI. Ao recebê-las, a placa Arduíno, através dos *shields* MIDI, envia

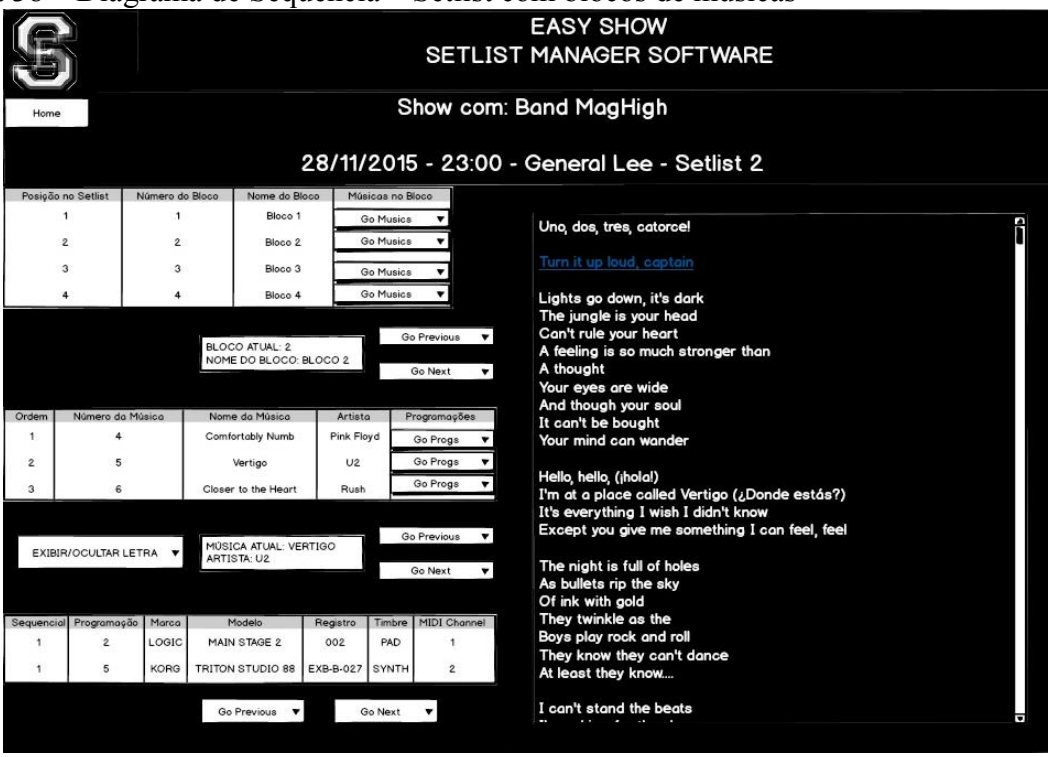
essas informações aos instrumentos conectados, e devolve ao software a confirmação das trocas de programações. Finalmente, o software recebe essas informações e envia em tela ao usuário as novas informações.

4.1.6 Prototipação de tela

Para Melo (2010, p.72), a prototipação é um recurso bastante interessante para uma melhor visualização e compreensão de determinados cenários, auxiliando na descoberta de requisitos que não foram detectados nas fases anteriores.

Nesse sentido, a Figura 38 retrata uma prototipação da tela de execução, na qual ocorrerão as principais interações entre o usuário e o software envolvidos no presente projeto.

Figura 38 – Diagrama de Sequência – Setlist com blocos de músicas



Fonte: Do autor.

Nesse ambiente, é possível visualizar a relação de blocos de um show, as músicas do bloco atual, e as programações da música atual, a partir de um número sequencial. Ainda, é possível exibir ou ocultar a letra da música atual.

Com a etapa da modelagem concluída, passa-se a dar início ao desenvolvimento do projeto.

5 DESENVOLVIMENTO

A partir da apresentação da modelagem, iniciou-se a fase de desenvolvimento do projeto. Neste tópico, são apresentadas as fases do desenvolvimento de hardware e software, tecnologias utilizadas, orçamentos e princípio de funcionamento.

5.1 CENÁRIO

A idéia inicial do projeto surgiu após um show realizado no mês de agosto de 2014. Ao ser questionado pelo baterista da banda sobre a existência ou não de um equipamento que pudesse alterar todas as programações de todos os teclados utilizados, em tempo de execução, foi feita uma análise de mercado.

Através de conversas informais em comunidades de Facebook e WhatsApp, além de pesquisas em fóruns de discussão na internet, foram observadas reclamações e sugestões de diversos músicos, que comentavam sobre produtos existentes, pontos fortes e fracos, funcionalidades que consideram importantes e detalhes que são irrelevantes.

Foram feitos contatos informais com diversos músicos, questionando sobre necessidades, características, problemas e outros detalhes pertinentes. Após esse levantamento preliminar de requisitos, foi desenvolvida a modelagem da solução.

Nesse ponto, várias ferramentas passaram a ser estudadas e analisadas, para a definição de quais seriam utilizadas. Em relação ao software, o mesmo foi desenvolvido como uma aplicação web, com armazenamento das informações em banco de dados e em memória, para fins de otimizar a atuação junto ao futuro hardware. Para fins de comunicação entre software e hardware, foram utilizadas comunicação serial e protocolo MIDI, por ser um protocolo destinado à comunicação de instrumentos e equipamentos musicais.

5.2 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do presente projeto, algumas tecnologias foram utilizadas, que são apresentadas na Figura 39.

Figura 39 – Tecnologias utilizadas no projeto

Fonte: Do autor.

- ✓ **Apache Tomcat 7:** por ser um servidor de aplicação web em versão *free*, bastante leve e objetivo, o Apache Tomcat 7 foi utilizado no presente projeto;
- ✓ **HTML e HTML5:** é uma linguagem de marcação padrão, amplamente utilizada em páginas e aplicações web;
- ✓ **jQuery:** ferramenta bastante utilizada para criação de sites interativos, foi utilizada em algumas telas, para fins de criação de alguns efeitos visuais;
- ✓ **Java:** essa linguagem foi utilizada durante todo o período acadêmico, o que proporcionou maior conforto no quesito programação. Além disso, é uma linguagem voltada a orientação a objetos e a aplicações web, possuindo um número considerável de bibliotecas para as mais diversas finalidades. Ainda, possui uma interface gráfica bastante amigável;
- ✓ **JSP:** foi amplamente utilizada no presente projeto, graças ao fato de ser bastante flexível. Ela permite a utilização de *scripts* Java, além de incorporar elementos HTML, o que proporciona uma gama considerável de opções;
- ✓ **CSS:** utilizada, inicialmente, para a parte visual. Com a utilização do Bootstrap para essa finalidade, o CSS passou a ser pouco utilizado, mas permaneceu ativo no projeto, haja vista a utilização em algumas páginas JSP específicas;

- ✓ **JavaScript:** para algumas funcionalidades do projeto, fez-se necessária a utilização de código Java dentro das classes JSP. Para tanto, foram criados *scripts* em Java, permitindo ações mais dinâmicas dentro das páginas JSP;
- ✓ **Bootstrap:** o Bootstrap é um *framework* amplamente utilizado para desenvolvimento responsivo, ou seja, que possa se adaptar a todo e qualquer dispositivo, independentemente de resolução ou tamanho de janela. Ainda, permite uma série de facilidades e opções para aprimoramento do aspecto visual; (BOOTSTRAP, 2015);
- ✓ **PostgreSQL:** esse banco de dados foi bastante utilizado pelo presente autor, durante as atividades acadêmicas, o que permitiu uma melhor adaptação, principalmente, em relação à manipulação dos dados e sintaxe de criação dos *scripts* SQL. Ainda, o PostgreSQL é um SRGBD *open-source,* e possui uma adaptabilidade maior com aplicações *web*;
- ✓ **Eclipse:** essa foi a plataforma de desenvolvimento utilizada durante grande parte do período acadêmico, sendo que o autor possui maior domínio e segurança na utilização da mesma. Além disso, possui uma gama de recursos que facilitam no desenvolvimento, verificação de erros e otimização de código, além de possuir maior integração a projetos *web*;
- ✓ **Notepad++:** esse aplicativo foi bastante utilizado durante o desenvolvimento do projeto, principalmente, para testes de funções e métodos isoladamente, possuindo uma interface bastante amigável;
- ✓ **Balsamiq Mockups:** foi utilizado para a prototipagem das telas do software, por possuir boa usabilidade, rapidez e dinamismo na concepção inicial das telas;
- ✓ **Enterprise Architect:** esse software teve ampla utilização para a elaboração dos diagramas do projeto. Sendo uma ferramenta bastante completa, possui recursos variados para a confecção de, praticamente, todos os artefatos utilizados dentro dos processos de produção de software, desde a patê de requisitos, até a realização de testes;

5.2.1 Java

A linguagem de programação Java surgiu em meados dos anos 90, baseada na linguagem C++, com várias alterações. A linguagem Java surgiu da necessidade de uma

linguagem que fosse mais simples, segura e, principalmente, orientada a objetos, haja vista que deveria ser utilizada na programação de dispositivos eletrônicos embutidos, como micro-ondas, geladeiras, TVs e afins. (SEBESTA, 2006, p.102).

Muito embora o Java tenha sido criado para ser usado em dispositivos de consumo eletrônicos, nenhum desses produtos chegou a ser efetivamente comercializado. Contudo, com a grande utilização da *World Wide Web* (WWW), a partir de 1993, os especialistas chegaram à conclusão que o Java era a linguagem ideal para desenvolver os programas para a web, graças a diversas características, dentre elas, a interface gráfica. (SEBESTA, 2006, p.102).

5.2.1.1 Características

O Java é uma linguagem orientada a objetos, oriunda do C++, mas projetada para ser menor, mais prática e segura. Possui tipos e classes, com os objetos acessados por variáveis de referência. As matrizes são instâncias de classe predefinida. Possui um tipo booleano, usado, principalmente, nas operações envolvendo *if* e *while*. (SEBESTA, 2006, p.103).

Suporta herança múltipla e utiliza um modificador, chamado *synchronize*, além de trabalhar com a técnica de "coletor de lixo", também conhecida como "*garbage collection*", eliminando variáveis e/ou elementos desnecessários, após a execução, liberando mais espaço para o processamento. Ainda, possui coerções de tipo, partindo, obrigatoriamente, do menor para o maior (*int* para *float*, mas nunca no sentido contrário, por exemplo), e um interpretador Java, conhecido como Java Virtual Machine (JVM). (SEBESTA, 2006, p.104).

Certamente, a característica mais marcante da linguagem Java é a sua utilização para a Web. Com todas as melhorias efetuadas desde a primeira versão, as várias bibliotecas de classes, interfaces gráficas, acesso a bases de dados e o sistema interpretador/compilador mais eficiente, transformaram o Java na linguagem mais crescente no mercado, com menor propensão aos erros ocorrentes nas outras linguagens. (SEBESTA, 2006, p.104). Na Figura 40, um exemplo de código em Java é apresentado.

Figura 40 – Exemplo de código em Java

```
// Java Example Program
//  Input: An integer, listlen, where listlen is less
//         than 100, followed by length-integer values
// Output: The number of input data that are greater than
//         the average of all input values
import java.io.*;
class IntSort {
public static void main(String args[]) throws IOException {
  DataInputStream in = new DataInputStream(System.in);
  int listlen,
      counter,
      sum = 0,
      average,
      result = 0;
  int[] intlist = new int[99];
  listlen = Integer.parseInt(in.readLine());
  if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum  */
    for (counter = 0; counter < listlen; counter++) {
      intlist[counter] =
            Integer.valueOf(in.readLine()).intValue();
      sum += intlist[counter];
    }
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
    for (counter = 0; counter < listlen; counter++)
      if (intlist[counter] > average) result++;
/* Print result */
      System.out.println(
          "\nNumber of values > average is:" + result);
    }  //** end of then clause of if ((listlen > 0) ...
    else System.out.println(
              "Error-input list length is not legal\n");
  }  //** end of method main
}  //** end of class IntSort
```

Fonte: Sebesta (2012, p.94).

Haja vista a grande quantidade de bibliotecas para a linguagem Java, o trabalho de codificação pode ficar mais rápido e eficiente.

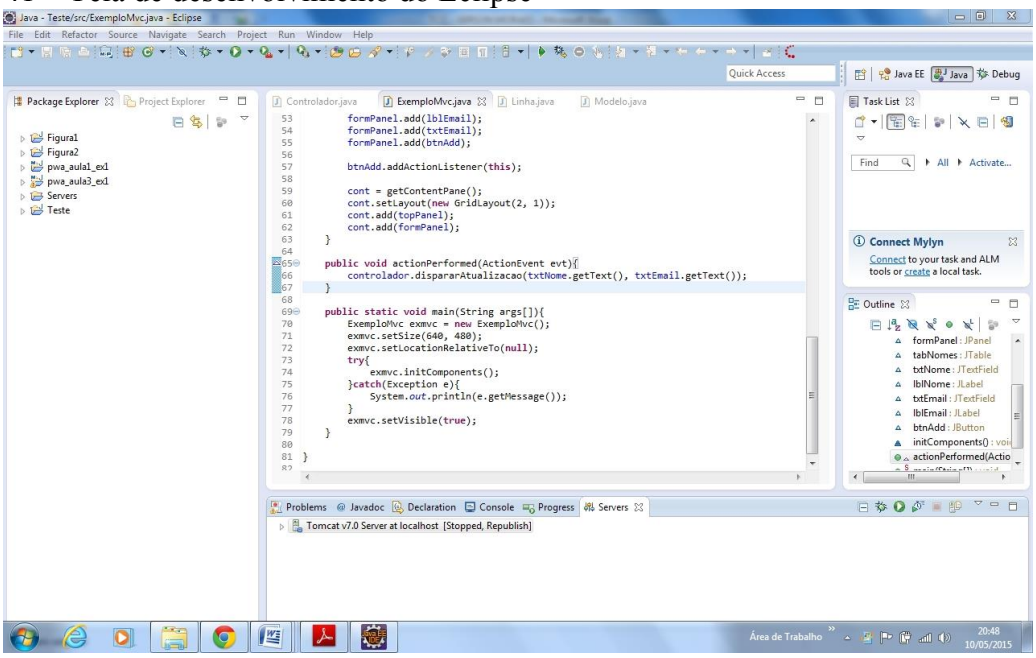
5.2.1.2 Ambiente de Desenvolvimento Eclipse

Para que a linguagem Java possa ser utilizada com maior facilidade, rapidez e eficiência, a utilização de um ambiente de desenvolvimento é de suma importância. Nesse quesito, a plataforma Eclipse é a mais recomendada. Desenvolvida pela IBM e, sendo uma plataforma *open-source*, multiplataforma e extensível, esse ambiente de desenvolvimento integrado *(Integrated Development Environment* – IDE) possui diversos *plugins*, ferramentas e assistentes de código, que tornam a tarefa de programação menos problemática, adaptando-se a qualquer situação. (GONÇALVES, 2008, p.3).

Desenvolvido inicialmente pela Object Technology International (OTI), foi comprada pela IBM posteriormente e cedida pela própria IBM para organizações de código

fonte aberto. (HEMRAJANI, 2007, p.139). Na Figura 41, vê-se a tela de desenvolvimento da IDE Eclipse.

Figura 41 – Tela de desenvolvimento do Eclipse



Fonte: Do autor.

A IDE Eclipse tem como grande característica se adaptar facilmente aos mais diversos plug-ins e frameworks utilizados no mercado.

5.2.2 PostgreSQL

Com uma linguagem definida para a manipulação de informações em banco de dados, foi necessária a criação de ambientes que propiciassem uma maior facilidade para essa manipulação. Um desses ambientes é o PostgreSQL, que é um SRGBD (Sistema gerenciador de Base de Dados Relacional), implementando os padrões ANSI-92, 96 e 99. (PEREIRA NETO, 2003, p.27).

Originário em 1986, na Califórnia, somente teve seu primeiro interpretador SQL em 1995, após inúmeros avanços e alterações no projeto inicial. Teve o código-fonte aberto na metade de 1996, com a liberação do código-fonte original, mais o código-fonte do interpretador SQL. (PEREIRA NETO, 2003, p.28).

Justamente pelo fato de ser *open-source*, vem conquistado um grande espaço no mercado, com ampliações e correções constantes em sua estrutura, o que torna seu desempenho

melhor que o de alguns concorrentes, sendo amplamente utilizado em projetos Web. (PEREIRA NETO, 2003, p.28).

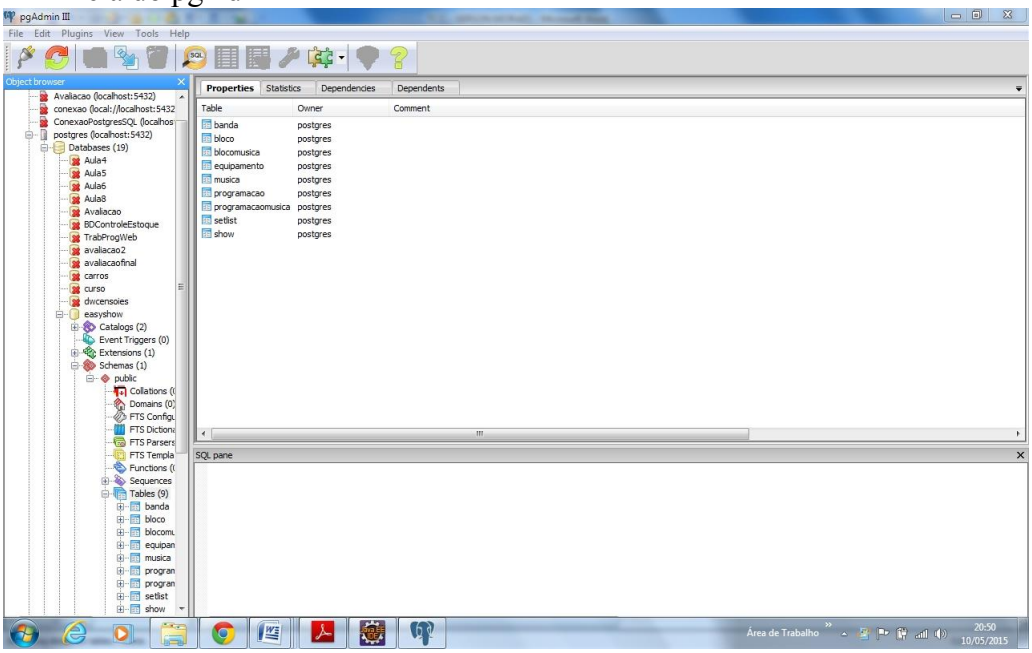
5.2.2.1 Características do PostgreSQL

O PostgreSQL trabalha, basicamente, com duas categorias funcionais, no quesito declarações: DDL (*Data Definition Language*, ou Linguagem de Definição de Dados) e DML (*Data Manipulation Language,* ou Linguagem de Manipulação de Dados). (PEREIRA NETO, 2003, p.29)

Em relação à DDL, pertencem os comandos *ALTER TABLE, CREATE TABLE, CREATE INDEX, CREATE USER*, entre outros, relacionados à manipulação das entidades do banco de dados. Já, em relação à DML, encontram-se os quatro comandos básicos de manipulação dos dados dessas entidades, que são: *SELECT, INSERT, DELETE* e *UPDATE*. (PEREIRA NETO, 2003, p.29).

Na Figura 42, vê-se a tela principal do *pgAdmin*, que é um módulo de acesso e controle do PostgreSQL.

Figura 42 – Tela do pgAdmin



Fonte: Do autor.

O uso desses módulos facilita consideravelmente a manipulação dos dados dentro de um BD, tornando essas manipulações mais precisas, seguras e eficientes.

5.2.3 Alguns componentes

Para que uma aplicação web possa ser desenvolvida e disponibilizada de forma a ser eficiente, simples e objetiva, há a necessidade de utilização de componentes extras, que facilitam na programação. Essas ferramentas trazem um nível de flexibilidade considerável ao programador, haja vista que podem ser utilizadas para inúmeras finalidades, dependendo de cada situação. (GRANNELL, 2009, p. 9).

A seguir, são apresentadas algumas dessas ferramentas.

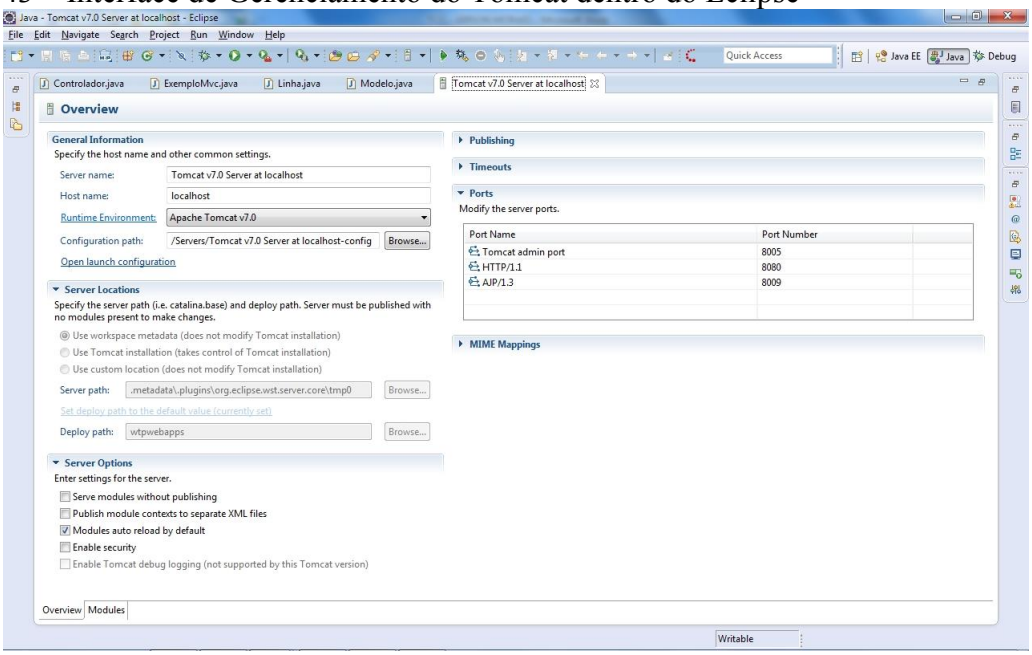
5.2.3.1 Apache Tomcat

O Apache Tomcat é um servidor de aplicação web, desenvolvido pela Sun Microsystems e pela Apache Software Foundation via projeto Jakarta, e baseado em Java e container de *servlets*. (HEMRAJANI, 2007, p. 7).

Relativamente leve, tem apresentado um grande crescimento no número de utilizadores, também motivado pelo fato de ser um servidor *free* (gratuito). Pode ser utilizado tanto para implementação de referência quanto de JSP. (FIELDS *et al*., 2000, p.32).

A seguir, a Figura 43 mostra a interface de controle do Apache Tomcat.

Figura 43 – Interface de Gerenciamento do Tomcat dentro do Eclipse



Fonte: Do autor.

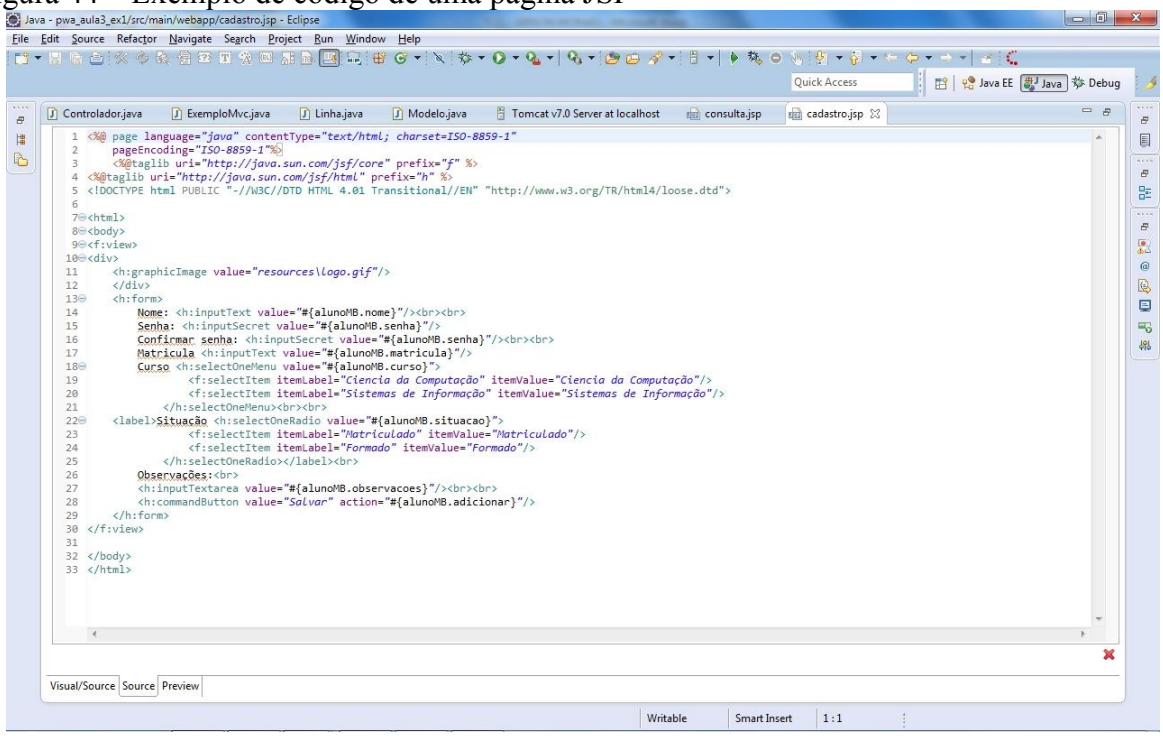
5.2.3.2 Java Server Pages (JSP)

Segundo Fields e Kolb (2000, p.2), o JSP é uma linguagem de programação, baseada em Java, mas que permite uma maior facilidade em termos de programação para conteúdo dinâmico. Ou seja, o JSP incorpora elementos da linguagem Java (lógicas de negócio e armazenamento em objetos Java, por exemplo), mas sem a obrigação do conhecimento das complexidades envolvidas na linguagem Java.

Trabalhando com a lógica de *tags* (marcações), o JSP permite uma grande flexibilidade a web designers e programadores HTML, pois atua com conteúdo HMTL tradicional e incorpora conteúdo Java, mas de um modo mais objetivo. (FIELDS *et al.*, 2000, p.2).

Além da facilidade na manipulação de conteúdo dinâmico, trabalha com o uso de scripts, com a utilização de linguagem de programação propriamente dita (no caso do JSP, é utilizada a linguagem Java). (FIELDS *et al.*, 2000, p.9). Um exemplo de código JSP é exibido na Figura 44.

Figura 44 – Exemplo de código de uma página JSP



Fonte: Do autor.

5.2.3.3 Java Server Faces (JSF)

Outro framework bastante utilizado para aplicações Web, baseadas em Java, é o JSF. O ponto principal do JSF é a separação das camadas lógica e de apresentação. Ainda, o conceito principal do JSF é ser baseado em componentes, além de possuir o escopo restringido às camadas de apresentação. (GOLÇALVES, 2008, p.40).

Assim, como o JSP, o JSF também possui *tags* (marcações) específicas, por exemplo, para conversão de dados ou manipulação e validação de erros. Permite o uso de bibliotecas, tais como a HTML e a Core, que permitem uma maior facilidade na manipulação das informações. (GOLÇALVES, 2008, p.46).

Conforme Gonçalves (2008, p.48), possui um ciclo de vida definido, compreendido por seis fases distintas e igualmente importantes, quais sejam:

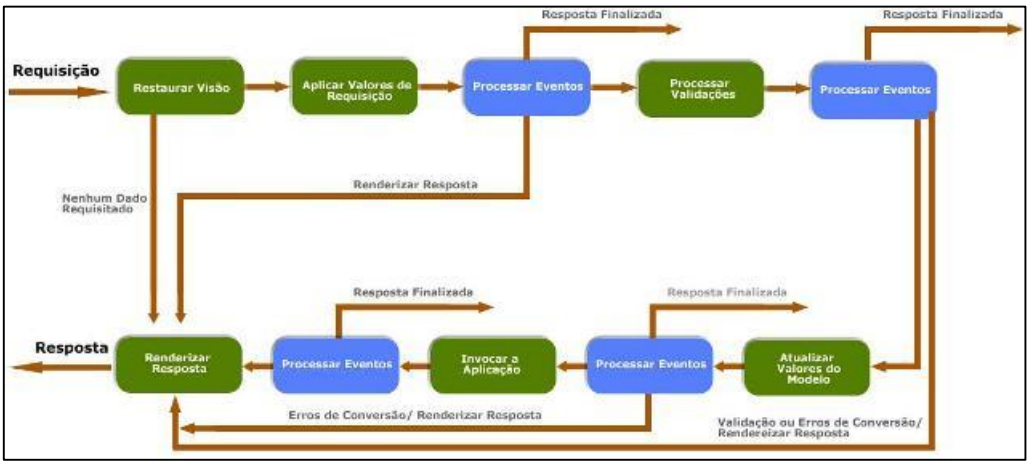
1. **restaurar a aplicação:** restaura a *view* (tela) atual. A página é construída pelo JSF e armazenada em uma instância *Faces-Context* para processamento posterior;
2. **aplicar os valores da requisição:** os valores são extraídos e armazenados nos respectivos componentes. Caso não seja uma String (variável do tipo texto), é convertido para String. Caso haja erros de conversão, pode ser apresentada uma mensagem ao usuário, ou o ciclo de vida pode pular para a fase de renderização, quando completa a conversão;
3. **processar validações:** os dados recebidos pelo passo 2 são validados, caso necessário, sendo que deverá implementar lógicas de validação para os respectivos dados, quando for o caso;
4. **atualizar valores do modelo:** os dados validados são atualizados e armazenados na classe *Bean*, que é aquela responsável por armazenar os objetos, segundo as regras de negócio. Caso os dados violem as regras de negócio, podem ser validadas novamente na próxima fase;
5. **invocar a aplicação:** após todo o processo de validação e atualização dos dados, os eventos são disparados, conforme regras implementadas, através do arquivo *faces-config.xml*, que possui vínculos para navegação com os respectivos eventos disparados. Por exemplo: supondo que, em um formulário, todos os dados foram validados e atualizados, um evento de retorno para o usuário será disparado, devidamente associado a um

mapeamento no arquivo *faces-config.xml*, que fará o devido redirecionamento e feedback ao usuário;

6. **renderizar a resposta:** a página final é renderizada e apresentada para o usuário. Caso seja uma invocação inicial, os dados serão acrescentados na apresentação da tela; senão, significa que os dados já foram somados à apresentação da tela. Ainda, caso haja erros de validação ou conversão, serão apresentados na tela.

Na Figura 45, é apresentado o ciclo de vida do JSF.

Figura 45 – Ciclo de vida do JSF



Fonte: Do autor (2015).

Basicamente, após a requisição é feita: a restauração da *view*; a restauração da árvore de componentes da página, a validação de todos os valores e validadores com o atributo *required=true*; a atualização desses valores, já validados, no modelo de negócio; a manipulação dos eventos da aplicação; e a construçãoao da página de resposta e o envio da mesma ao usuário. Se, em alguma das etapas de processamento houver algum problema, é enviada imediatamente, uma mensagem de *response* ao usuário. (GONÇALVES, 2008, p.48).

5.2.3.4 Facelets

Embora o framework JSF seja bastante eficiente para o desenvolvimento de projetos web, o uso das *tags* pode gerar complicações para o desenvolvedor. Para isso, é utilizado o framework Facelets, que é uma ferramenta de *templates* JSF, transformando *tags* (X)HTML em páginas JSF, sem precisar utilizar essas *tags*. (GOLÇALVES, 2008, p.53).

5.2.3.5 Hibernate

Ao desenvolver uma aplicação que necessite que os dados sejam armazenados, uma estratégia amplamente utilizada é a utilização de bancos de dados. Entretanto, a aplicação não gerencia somente essas informações, mas também as lógicas de programação, regras de negócio, entre outras características. (GONÇALVES, 2008, p.88).

Para essa finalidade, uma das ferramentas mais utilizadas é o Hibernate, que é uma solução para gerenciamento de dados persistentes em Java. Com suporte para múltiplos bancos de dados, o Hibernate apresenta, ainda, características como mapeamento de componentes, associações bidirecionais, mapeamento de heranças e armazenamento em cachê, entre outras facilidades. (HEMRAJANI, 2007, p.242).

5.2.3.6 HTML e XHTML

Segundo Grannell (2009, p.9), um documento HTML (*HyperText Markup Language*, ou Linguagem de Marcação de HiperTexto) é um arquivo de texto com marcações (*tags*), sendo que o conteúdo dessas marcações é acessado pelos navegadores (browsers), cuja exibição deverá estar em conformidade com os padrões determinados pela World Wide Web Consortium (W3C).

O padrão HTML não é tão rigoroso, pois permite a utilização de *tags* com letras maiúsculas ou minúsculas, por exemplo. Uma boa prática recomendada é o fechamento de todas as *tags*; entretanto, há situações em que *tags* não são fechadas, sem causar, entretanto, danos maiores à execução do sistema, mas afetando comportamentos esperados. (GRANNELL, 2009, p.11).

Um exemplo de *tag* não fechada é: Teste... . Nota-se que, neste caso, a *tag* não foi fechada corretamente, o que pode permitir a um teto subsequente ser formatado em itálico. Já um padrão correto de *tags* fechadas é: Teste... . (GRANNELL, 2009, p.11).

O padrão HTML permite o aninhamento de *tags*, conforme exemplo mostrado no parágrafo anterior, além da utilização de CSS (*Cascading Style Sheets*), para fins de uma melhor formatação das páginas. (GRANNELL, 2009, p.10).

Apesar de serem muito parecidos, há diferenças consideráveis entre HTML e XHTML. Considerada como uma evolução do HTML, o XHTML (*Extendible HyperText Markup Language*, ou Linguagem de Marcação de HiperTexto Estendível) é mais rigoroso,

com várias regras adicionais para o fechamento obrigatório de *tags* (com uma barra dentro da *tag* de fechamento e antes do nome da *tag*), o nome dos rótulos (sempre com letra minúscula), declarações de abertura e fechamento, entre outras regras. (DUCKETT, 2010, p.3)

Ao contrário de HTML, todos os atributos em XHTML devem vir entre aspas duplas, sempre com um valor atribuído. Um exemplo de *tag* com atributo em XHTML é: <td colspan=”2” nowrap=”nowrap”>. (GRANNELL, 2009, p.12).

5.2.3.7 CSS

Uma das preocupações da W3C era separar as atribuições de formato e layout das páginas do HTML e XHTML, uma vez que a eles pertenciam às atribuições visuais da página. A poluição visual dentro das páginas HTML, haja vista a alta demanda de informações que foram surgindo, além da complexidade que as páginas HTML e XHTML ganharam, foi o estopim para o surgimento do CSS. (DUCKETT, 2010, p.255)

Assim, a idéia base para a criação do CSS, segundo Grannell (2009, p.15), era “... eliminar a apresentação e separar o desenho do conteúdo.”. Em outras palavras, deixar toda a manipulação da camada visual para o CSS e manter o HTML e XHTML somente com a estrutura da página.

Um dos grandes problemas encontrados no desenvolvimento do CSS foi o suporte por parte dos navegadores da época. Além disso, outro problema foi convencer os *web designers* de que essa nova forma de manipulação visual era mais interessante que deixar tudo dentro do HTML e XHTML. Ainda, hoje, há designers que restam relutantes, mas a grande maioria já utiliza o CSS, haja vista que os navegadores atuais suportam a maior parte dos valores e atributos do CSS. (GRANNELL, 2009, p.14).

O CSS consiste de um conjunto de regras que vão definir a exibição das informações em uma página web. Basicamente, a estrutura consiste em um seletor e uma declaração, podendo conter, ainda, pares de propriedades/valores, definindo propriedades específicas. Pode-se definir, por exemplo, as características de exibição de um parágrafo, uma tabela ou um arquivo de mídia. (GRANNELL, 2009, p.16).

Com a definição das tecnologias e ferramentas a serem utilizadas, a fase de desenvolvimento do software teve seu início.

5.3 DESENVOLVIMENTO DO SOFTWARE

O desenvolvimento do software foi iniciado em janeiro de 2015, após o levantamento de alguns requisitos preliminares. A linguagem escolhida para o desenvolvimento foi o Java, por ser a linguagem de programação utilizada ao longo de toda a fase acadêmica. O software foi desenvolvido como uma aplicação web, haja vista maior identificação deste autor com o assunto.

Ainda, um dos motivos que impulsionou a decisão de se desenvolver uma aplicação web foi o fato de que, futuramente, há a intenção de fragmentar o software em dois módulos distintos: cliente e servidor. Apenas os módulos de carga e execução de shows ficariam na máquina do cliente, sendo o restante um *web service*.

Para fins de otimizar o desenvolvimento, algumas ferramentas foram utilizadas, tais como: o Bootstrap, JSP, JavaScript e HTML. Em relação ao banco de dados, foi utilizado o PostgreSQL, por apresentar melhor facilidade de manipulação e maior integração com o projeto proposto. Já, as plataformas de desenvolvimento utilizadas foram o Eclipse JEE Kepler e o Notepad++.

As classes Java foram separadas nos pacotes DAO, Servlet, Entities, Persistência e E-mail. Essa organização em pacotes ficou restrita ao código Java, uma vez que, ao usar essa sistemática usando o Bootstrap, as classes não reconheciam as folhas de estilos do Bootstrap. Ao deixá-las todas no mesmo local, essas folhas de estilos foram reconhecidas, melhorando, consideravelmente, o aspecto visual.

O primeiro passo foi definir quais as entidades envolvidas, seus atributos e os métodos DAO envolvidos. Após, passou-se ao desenvolvimento das classes JSP. Inicialmente, foi trabalhado com as folhas CSS de estilos normais, sendo que, depois da parte lógica finalizada, passou-se a utilizar o Bootstrap, para fins de melhorar o aspecto visual.

A classe *GoProgs* é aquela que, efetivamente, faz a comunicação com o hardware externo, pois é responsável por toda a manipulação que ocorre durante o show. Todas as trocas de músicas, blocos, programações e timbres passam por essa classe.

A classe *Programacao* possui um atributo *registro*, que é o timbre propriamente dito. Esse timbre é enviado como parâmetro ao evento *ProgramChange,* utilizado pelo Arduíno e transmitido via *shields* MIDI. Já, a classe *Equipamento* possui o atributo *midiChannel*, que é o canal MIDI do instrumento ou equipamento. Da mesma forma que o atributo *registro*, o atributo *midiChannel* também é passado como parâmetro ao evento *MidiChannel,* utilizado pelo Arduíno e transmitido via *shields* MIDI, posteriormente.

Finalmente, a classe *Config* é responsável por setar a configuração dos pedais que são utilizados no protótipo da pedaleira controladora MIDI em desenvolvimento.

A passagem dos parâmetros para Arduíno é feita nas páginas *jsp,* através de uma rotina escrita na linguagem PHP. Essa rotina recebe os parâmetros referentes aos pedais, aos canais MIDI e aos timbres dos instrumentos.

Ainda, foram desenvolvidas três classes específicas para o tratamento da comunicação serial, entre o software e um hardware externo, a ser desenvolvido utilizando a plataforma Arduíno. Essas classes são: *SerialCom*, que manipula as informações sobre as portas seriais para comunicação; *SerialComLeitura*, que faz o monitoramento do sistema, com a finalidade de identificar quando um evento foi disparado; e *SerialSend*, que aciona os métodos de comunicação e passa os parâmetros para um hardware externo. O método *SerialSend* é disparado pela classe *GoProgs*.

5.4 HISTÓRICO DO DESENVOLVIMENTO

A idéia inicial do projeto era a de utilizar uma pedaleira Ketron FS-13, já adquirida em anos anteriores. Essa pedaleira utiliza comunicação serial e é de uso exclusivo em teclados da marca italiana Ketron. Haja vista complicações não previstas inicialmente, a criação da pedaleira como um produto final não foi realizada, e o foco passou a ser o desenvolvimento de uma solução de software/hardware, utilizando um protótipo com Arduíno e *shields* MIDI.

Os primeiros componentes adquiridos foram os Shields MIDI, através da loja SparkFun Electronics, em 15 de maio de 2015, através do site http://www.sparkfun.com/products/12898. Contudo, essa primeira compra, por motivos desconhecidos, não chegou ao destino até a presente data. Passados dois meses da primeira aquisição, foi efetuado contato com a empresa, no sentido de identificar o paradeiro dos produtos.

Após ser informado da não-localização do pedido, a empresa se prontificou a enviar outros seis *shields* MIDI, utilizando o FedEx como meio de transporte, impactando em custo adicional (frete e impostos).

Entretanto, em 31 de julho, fui informado de que os *shields* MIDI não se encontravam mais em estoque, e que seria necessário aguardar a confecção de um novo lote dos produtos. Esse novo lote só foi finalizado em meados de outubro, e seis novos *shields* MIDI foram recebidos em 15 de outubro do corrente ano. A Figura 46 mostra a chegada dos seis *shields* MIDI, sendo que os mesmos vêm desmontados.

Figura 46 – Primeira compra de *shields* MIDI

Fonte: Do autor.

Para o Arduíno, foi adquirida a versão Mega2560, devido ao número de portas e capacidade de processamento, em versão *open-hardware*. O pedido foi efetuado em 15 de maio de 2015, e recebido em 20 de maio.

Contudo, houve equívoco no envio, sendo enviado o modelo Uno. No dia do recebimento, foi feita comunicação junto à empresa, que se prontificou em efetuar a troca do componente. No dia 27 de maio, a situação já estava resolvida, com o recebimento do Arduíno Mega2560. A Figura 47 traz o Arduíno Mega 2560, após a troca.

Figura 47 – Arduíno Mega 2560

Fonte: Do autor.

Para que não houvesse prejuízos ao desenvolvimento do projeto, foram adquiridas duas unidades na loja Multilógica Shop, através do site https://www.multilogica-

shop.com/shield-midi. Essas unidades foram adquiridas em 17 de setembro e entregues em 21 do mesmo mês. Na Figura 48, os dois *shields* MIDI já montados.

Figura 48 – Segunda compra de *shields* MIDI

Fonte: Do autor.

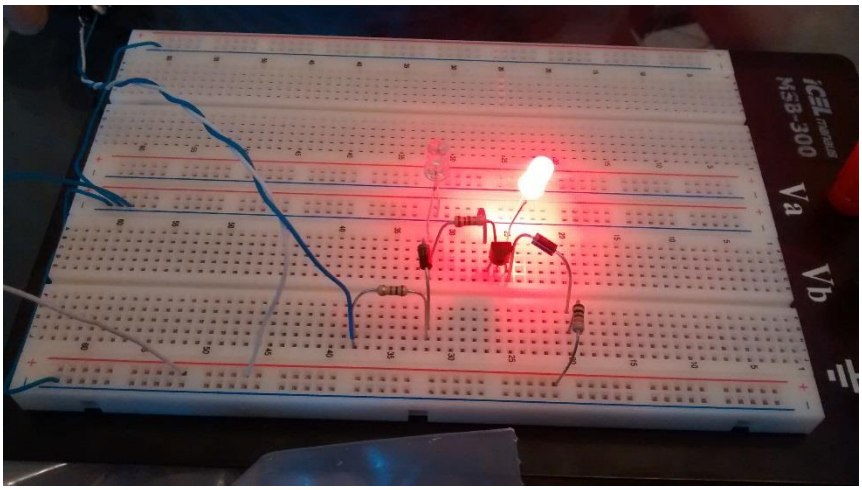
Já, os componentes eletrônicos foram adquiridos em uma loja no centro de Florianópolis e através de compras na loja Multilógica Shop, através do site oficial. As Figuras 49, 50 e 51 apresentam alguns dos componentes adquiridos.

Figura 49 – Componentes eletrônicos adquiridos

Fonte: Do autor.

Os componentes eletrônicos foram adquiridos em uma loja especializada em eletrônica. Na ocasião, os dois *shields* MIDI foram montados pelo atendente do estabelecimento.

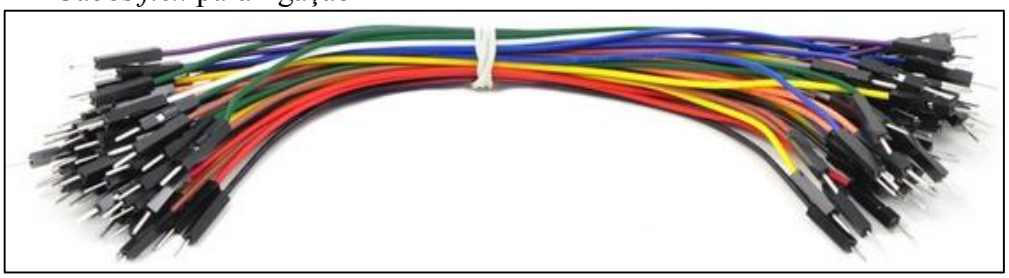
Figura 50 – Protoboard



Fonte: Do autor.

A placa de *protoboard*, usada para a elaboração do protótipo, foi comprada em outro estabelecimento comercial, também no centro de Florianópolis.

Figura 51 – Cabos *flex* para ligação



Fonte: Do autor.

Em 27 de outubro, foram comprados os cabos *flex* para prototipação, bem como, pinos para ligação da placa Arduíno e dos Shields MIDI. Os pinos não foram utilizados e, por isso, não estão retratados no presente projeto.

Os materiais adquiridos foram utilizados para uma tentativa de prototipação de hardware, compatível com o software ora desenvolvido. O Quadro 5 apresenta a relação desses materiais e componentes adquiridos, e o Quadro 6, dos materiais e componentes efetivamente utilizados no protótipo, bem como os custos envolvidos.

Quadro 5 – Materiais adquiridos

ITEM	QTDE	DESCRIÇÃO	VALOR UNITÁRIO	VALOR TOTAL	SITUAÇÃO
1	2	Resistor 110 *ohms*	R$ 0,08	R$ 0,16	**COMPRADO**
2	1	Led verde	R$ 1,40	R$ 1,40	**COMPRADO**
3	1	Led vermelho	R$ 1,40	R$ 1,40	**COMPRADO**
4	1	Chave ON/OFF	R$ 8,00	R$ 8,00	**COMPRADO**
5	1	Transistor NPN	R$ 10,00	R$ 10,00	**COMPRADO**
6	2	Diodo	R$ 0,30	R$ 0,60	**COMPRADO**
7	1	Placa Arduíno Mega 2560	R$ 90,00	R$ 90,00	**COMPRADO**
8	8	Shield MIDI	R$ 103,00	R$ 824,00	**COMPRADO**
9	3m	Fio 0,7mm	R$ 0,95	R$ 2,85	**COMPRADO**
10	4	Resistor 10K	R$ 0,35	R$ 1,40	**COMPRADO**
11	6	Botão *push-button*	R$ 2,00	R$ 12,00	**COMPRADO**
12	14	Resistor 220 *ohms*	R$ 0,25	R$ 3,50	**COMPRADO**
13	50	Cabos *flex* para prototipação	R$ 1,20	R$ 60,00	**COMPRADO**
TOTAL GASTO				**R$ 1015,31**	

Fonte: Do Autor.

O protótipo inicial pretendia implementar seis botões, um para cada funcionalidade do software (avançar/retroceder blocos de músicas, avançar/retroceder músicas e avançar/retroceder programações). Contudo, o protótipo foi reduzido para os botões de avançar/retroceder programações.

Quadro 6 – Materiais efetivamente utilizados no desenvolvimento do protótipo

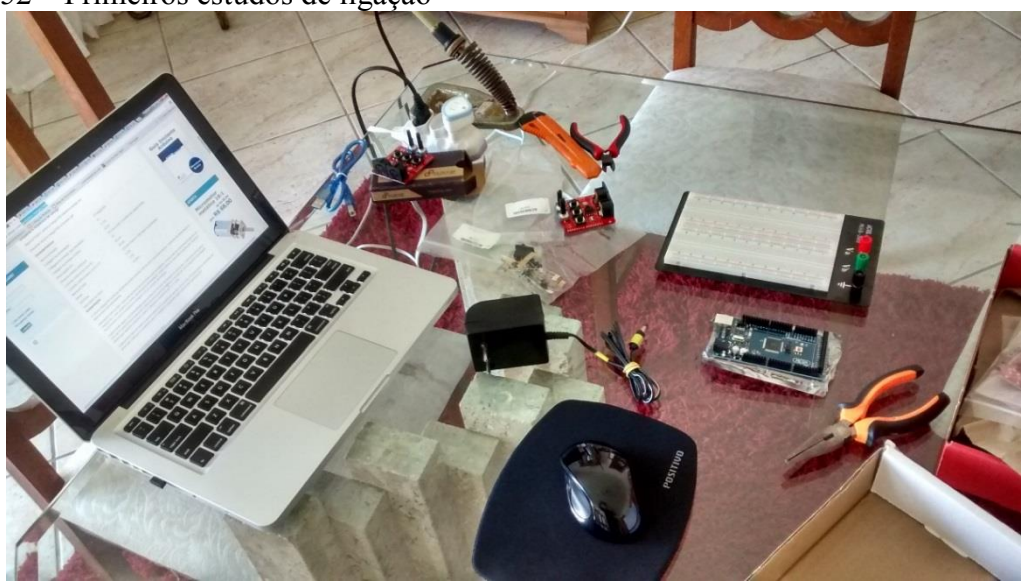
ITEM	QTDE	DESCRIÇÃO	VALOR UNITÁRIO	VALOR TOTAL	SITUAÇÃO
1	1	Led verde	R$ 1,40	R$ 1,40	**COMPRADO**
2	1	Led vermelho	R$ 1,40	R$ 1,40	**COMPRADO**

3	1	Chave ON/OFF	R$ 8,00	R$ 8,00	**COMPRADO**
4	1	Transistor NPN	R$ 10,00	R$ 10,00	**COMPRADO**
5	2	Diodo	R$ 0,30	R$ 0,60	**COMPRADO**
6	1	Placa Arduíno Mega 2560	R$ 90,00	R$ 90,00	**COMPRADO**
7	3	Shield MIDI	R$ 103,00	R$ 309,00	**COMPRADO**
8	1,5m	Fio 0,7mm	R$ 0,95	R$ 1,45	**COMPRADO**
9	2	Resistor 10K	R$ 0,35	R$ 0,70	**COMPRADO**
10	2	Botão *push-button*	R$ 2,00	R$ 4,00	**COMPRADO**
12	5	Resistor 220 *ohms*	R$ 0,25	R$ 1,25	**COMPRADO**
13	20	Cabos *flex* para prototipação	R$ 1,20	R$ 24,00	**COMPRADO**
TOTAL GASTO				**R$ 451,80**	

Fonte: Do Autor.

Após todos os componentes em mãos, iniciaram-se as tentativas de ligação do hardware, resultando nos testes e ligações que são apresentados nas Figuras 52, 53, 54 e 55.

Figura 52 – Primeiros estudos de ligação



Fonte: Do autor.

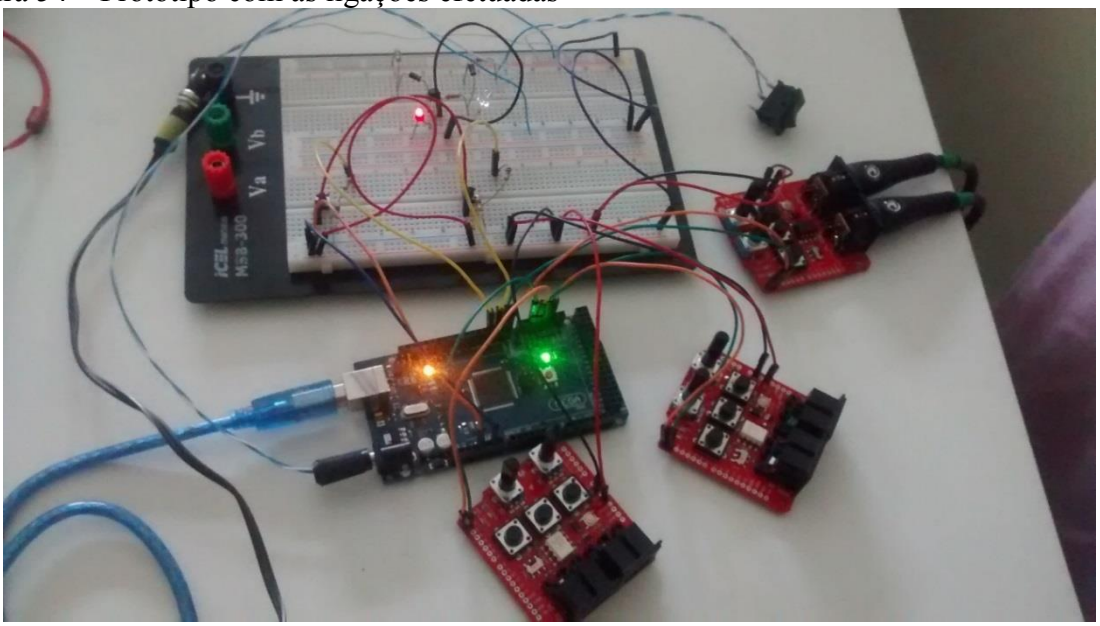
Os primeiros testes para montagem do protótipo, com todos os materiais em mãos, se iniciaram no final de outubro.

Figura 53 – Utilização de osciloscópio nos testes

Fonte: Do autor.

Esses testes incluíram o uso de osciloscópio, para fins de detecção dos sinais enviados do Arduíno para os *shields* MIDI e dos *shields* para o Arduíno. Na ocasião, ficou constatado que as placas estavam funcionando perfeitamente, enviando e recebendo as informações

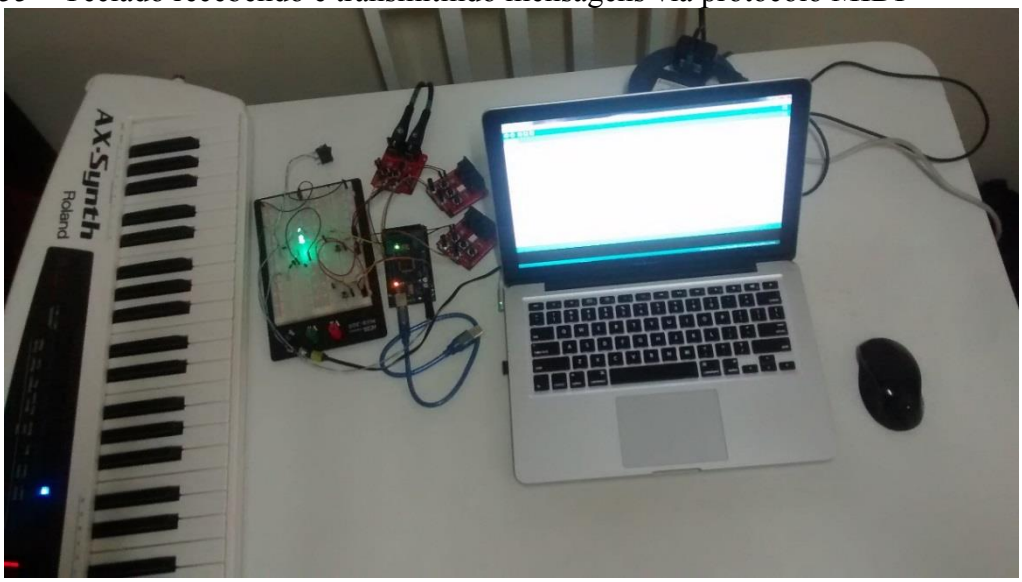
Figura 54 – Protótipo com as ligações efetuadas



Fonte: Do autor.

Com as placas funcionando, os primeiros testes foram realizados, utilizando alguns exemplos já prontos que acompanham a IDE de desenvolvimento do Arduíno.

Figura 55 – Teclado recebendo e transmitindo mensagens via protocolo MIDI



Fonte: Do autor.

Inicialmente, foi utilizado apenas um teclado nos testes. Ao se utilizar um exemplo padrão do Arduíno, houve a troca de mensagens entre o teclado conectado ao *shields* e o Arduíno, disparando uma sequência de notas musicais com um timbre fixo. Posteriormente, foi possível disparar a mesma sequência de notas em três teclados diferentes, com timbres, *pitch* e volume específicos para cada um deles.

Contudo, ao tentar executar o código oficial do projeto, não foi possível a mudança de programações. Os botões responsáveis pelo avanço e retrocesso das programações eram reconhecidos pelo Arduíno, piscando os *leds* indicadores de conexão RX e TX, mas os teclados não executavam a função de mudança de programações.

Outra dificuldade encontrada foi efetuar a troca de informações entre o Arduíno e o software externo. Mesmo com o software possuindo suporte a comunicação serial, e o Arduíno com uma função específica para receber os parâmetros do software externo, não foi possível efetuar a comunicação entre os mesmos, inviabilizando o prosseguimento dos testes posteriores.

Entre as maiores dificuldades encontradas, o fator tempo foi a maior delas. Obviamente, esse fator não deve ser utilizado como desculpa; entretanto, em relação ao segundo semestre, foi um fator decisivo para que o projeto inicial não pudesse ser desenvolvido a bom tempo e modo. Haja vista problemas não previstos pelo autor, o tempo hábil acabou sendo

consideravelmente reduzido. Essa redução do tempo, aliada à dificuldade de programação do hardware, causou grande preocupação e incômodo.

Como toda rosa possui espinhos, o presente projeto teve alguns percalços, que fizeram com que não fosse possível o desenvolvimento integral da solução inicialmente proposta.

Dessa forma, e devido à falta de tempo hábil para finalizar a etapa de comunicação do hardware com o software, o escopo do projeto foi alterado, ficando restrito, tão somente, ao software gerenciador de shows musicais.

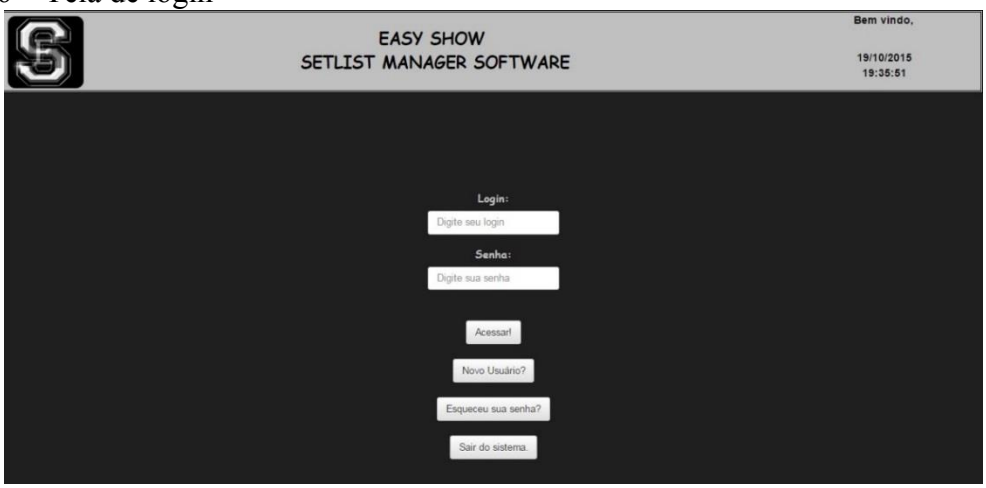
5.5 APRESENTAÇÃO DO SISTEMA

Nesse momento, é feita a apresentação do software. Não são mostradas todas as telas do sistema, haja vista que, muitas delas, são repetitivas e efetuam funções básicas de *CRUD*. Estaremos focando, apenas, aquelas que tiverem algum ponto a ser destacado.

5.5.1 Tela de login

Na tela de *login*, há opções para: inserir *login*/senha, recuperar a senha, informando um endereço de e-mail válido, efetuar um novo cadastro ou sair do sistema. O usuário só poderá acessar o sistema mediante autenticação. A Figura 56 apresenta a tela de *login*.

Figura 56 – Tela de login



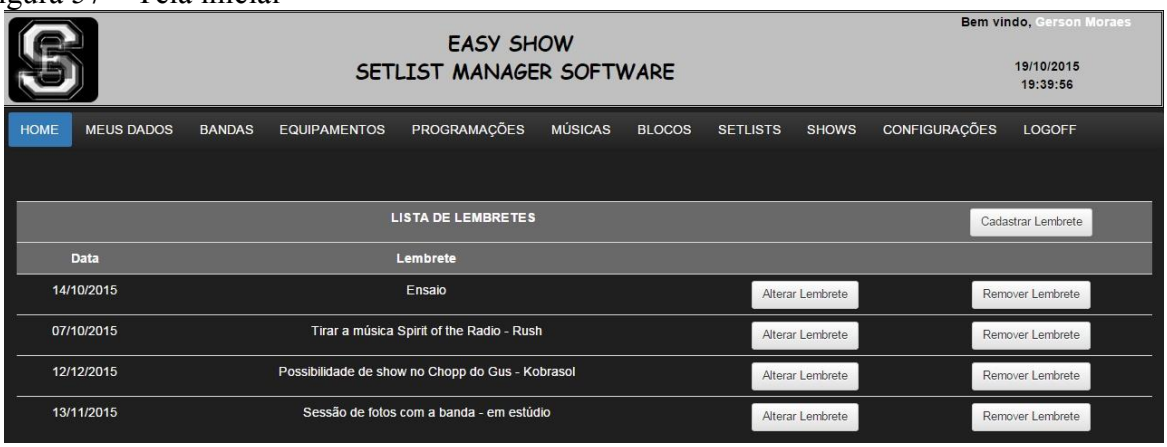
Fonte: Do autor.

Caso sejam realizadas três tentativas erradas, o usuário é redirecionado para a tela de recuperação de senha, na qual ele informa o endereço de e-mail, e recebe as informações de *login* e senha para o endereço informado.

5.5.2 Tela inicial

Na tela inicial, o usuário tem, logo abaixo do cabeçalho, um menu com todas as opções do sistema. Na área de trabalho, há uma lista de lembretes, com opções de edição de lembrete já existente, cadastro de um novo lembrete ou exclusão. A Figura 57 nos mostra essa tela.

Figura 57 – Tela inicial



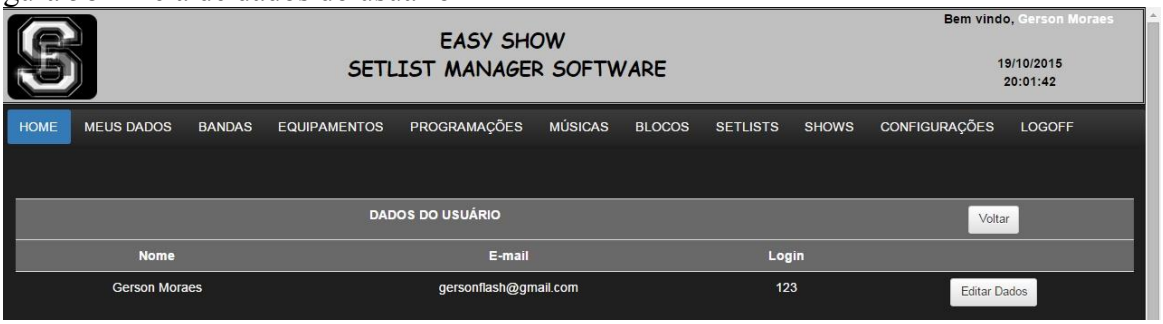
Fonte: Do autor.

A partir da opção *Cadastrar Lembrete*, é possível criar um novo lembrete, informando a data e o lembrete em si. Com a opção, *Alterar Lembrete*, os dados podem ser alterados.

5.5.3 Tela de dados do usuário

Na tela de dados do usuário, o usuário apenas poderá visualizar os dados, ou editá-los, clicando no botão correspondente. Essa tela é apresentada na Figura 58.

Figura 58 – Tela de dados do usuário



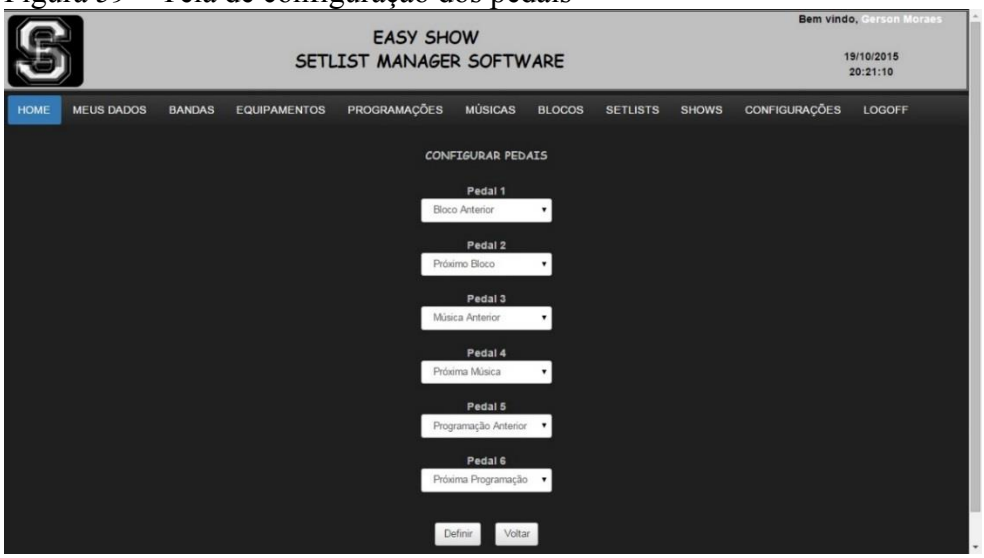
Fonte: Do autor.

Não há a opção para exclusão do usuário logado, sendo que o mesmo pode alterar os dados clicando em *Editar Dados*.

5.5.4 Tela de configuração

Na tela de configuração, o usuário pode definir qual função cada pedal vai exercer. Qualquer função pode ser atribuída a qualquer pedal, mas um pedal só poderá ter uma função setada. Essas configurações serão utilizadas no hardware que será em desenvolvimento. A tela é mostrada na Figura 59.

Figura 59 – Tela de configuração dos pedais



Fonte: Do autor.

A função atual de cada pedal aparecerá destacada na cor vermelha, para facilitar a identificação.

5.5.5 Tela de bandas

Na tela de bandas, há opções para: inserir uma nova banda, alterar dados de uma já existente ou remover. Essa tela é apresentada na Figura 60.

Figura 60 – Tela de bandas

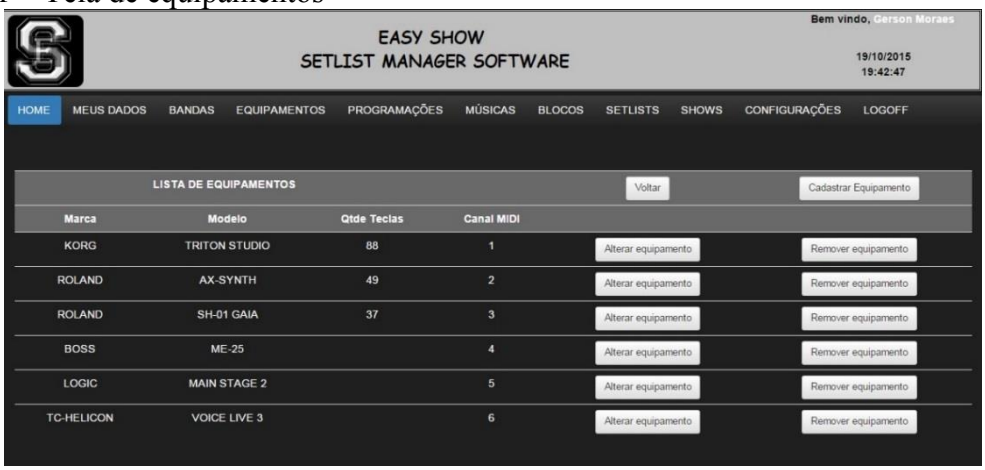
Fonte: Do autor.

Tanto o campo *Descrição* quanto o campo *Integrantes* abrem uma tela de visualização dos dados, na qual não é possível efetuar alterações. A opção *Visitar a página* permite o redirecionamento automático para a página da banda no Facebook, caso exista uma.

5.5.6 Tela de equipamentos

Na tela de equipamentos, há opções para: inserir um novo equipamento, alterar dados de um existente ou remover. Essa tela é apresentada na Figura 61.

Figura 61 – Tela de equipamentos



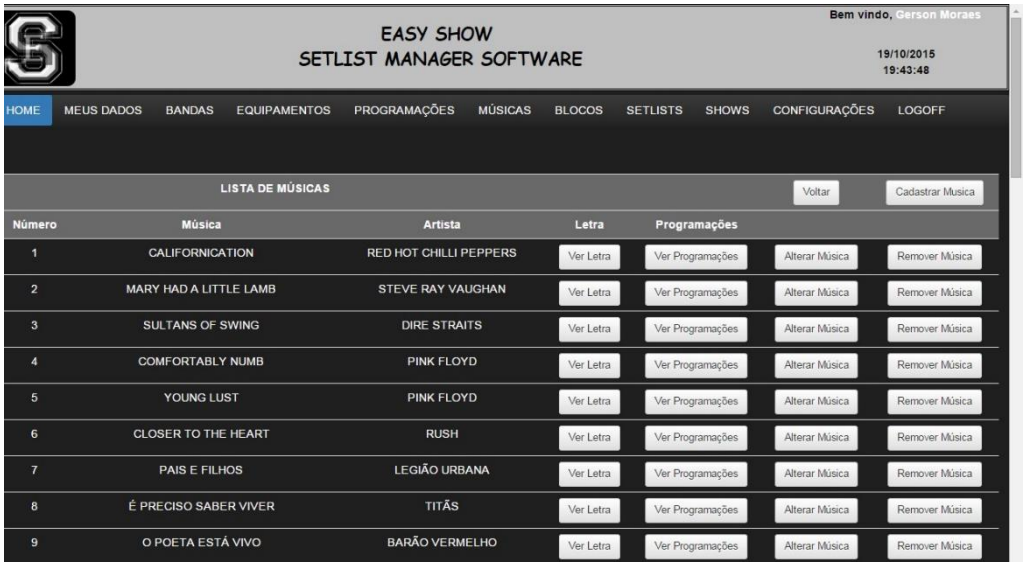
Fonte: Do autor.

Ao cadastrar um equipamento, o usuário informa a marca, o modelo, a quantidade de teclas (nos casos de pedais, não é necessário informar) e o canal MIDI. A informação do canal MIDI é utilizada como parâmetro para que eventual hardware externo possa identificar o destino das informações referentes a programação.

5.5.7 Tela de músicas

Na tela de músicas, o usuário pode incluir uma nova música, alterar uma já existente ou excluir. Para cada música, há um botão que permite consultar as programações vinculadas. A Figura 62 mostra a tela principal das músicas e a Figura 63, a listagem de programações vinculadas a uma música específica.

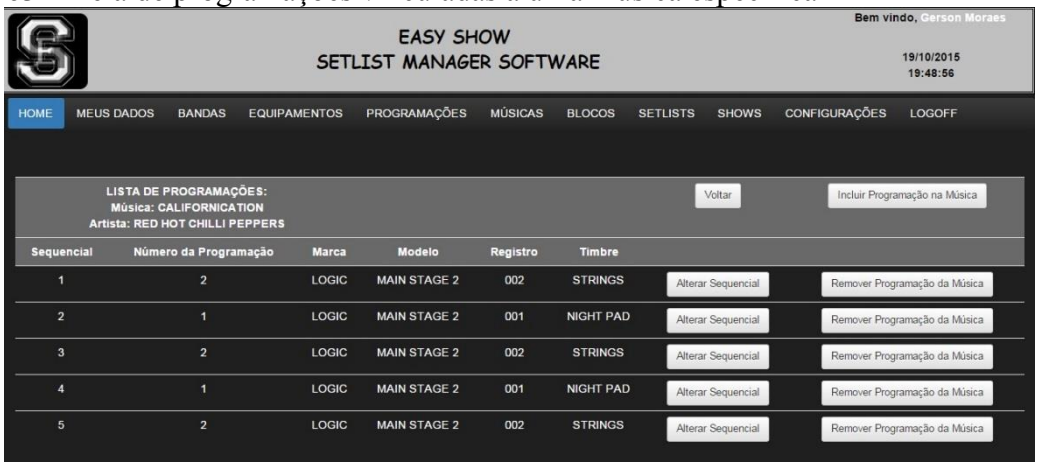
Figura 62 – Tela de músicas



Fonte: Do autor.

A partir da opção *Ver Letra*, é possível visualizar a letra da música. Na opção *Ver Programações*, é possível visualizar todas as programações utilizadas na respectiva música.

Figura 63 – Tela de programações vinculadas a uma música específica



Fonte: Do autor.

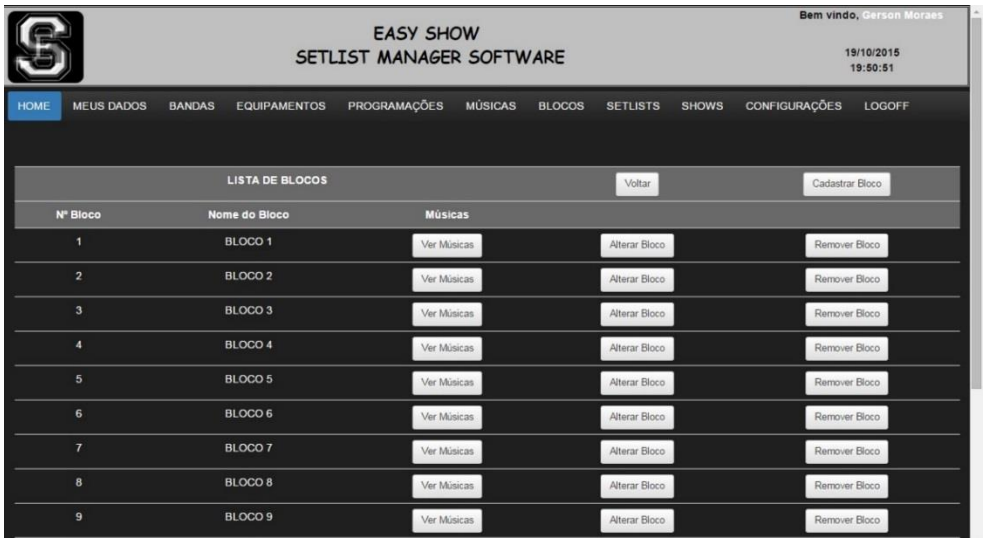
A partir dessa tela, é possível incluir programações à respectiva música. Cada música pode ter vários sequenciais e, cada sequencial, pode ter 8 instrumentos diferentes. Não é possível cadastrar uma programação de um mesmo instrumento no mesmo sequencial, ou seja, apenas uma programação por instrumento pode ser cadastrada em um sequencial.

Nos casos em que não haja a necessidade de efetuar a troca de programações para um instrumento no próximo sequencial, não é necessário incluir essa programação. O instrumento mantém o timbre anterior e só terá o timbre alterado quando tiver uma próxima programação cadastrada.

5.5.8 Tela de blocos

Na tela de blocos, o usuário pode incluir um novo bloco, alterar um já existente ou excluir. Para cada bloco, há um botão que permite consultar as músicas vinculadas. A Figura 64 mostra a tela principal dos blocos e a Figura 65, a listagem de músicas vinculadas a um bloco específico.

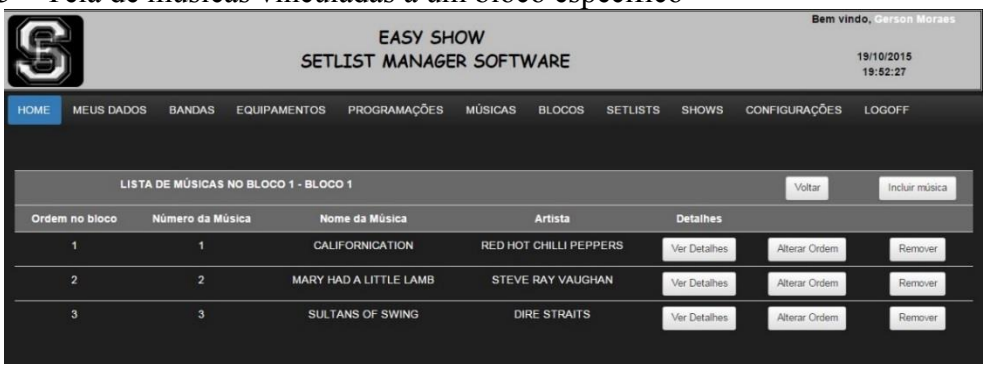
Figura 64 – Tela de blocos



Fonte: Do autor.

A partir da opção *Ver Músicas*, é possível visualizar a relação de músicas do bloco selecionado. Não há limite de músicas por bloco.

Figura 65 – Tela de músicas vinculadas a um bloco específico



Fonte: Do autor.

Ao clicar na opção *Ver Detalhes*, é possível visualizar os dados da música selecionada, e efetuar todas as operações referentes àquela música. A opção *Alterar Ordem* permite modificar apenas a ordem da música dentro do bloco.

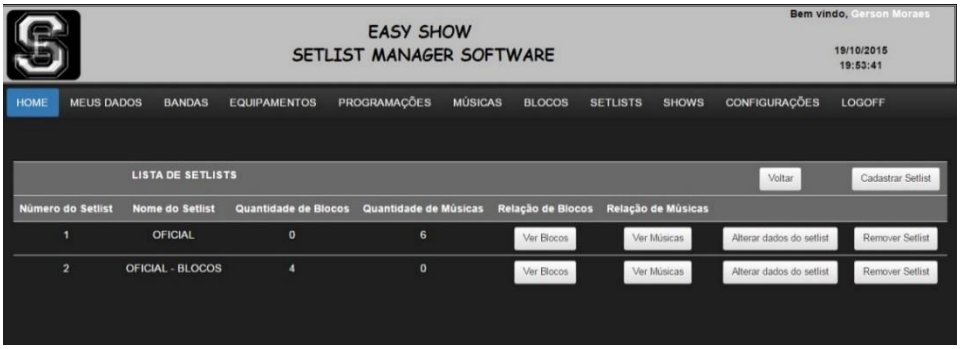
5.5.9 Tela de setlists

Na tela de *setlists*, o usuário pode incluir um novo *setlist*, alterar um já existente ou excluir. Para cada *setlist*, há um botão que permite consultar as músicas ou blocos vinculados.

Dentro dessas classes de vinculação, é possível incluir uma música ou bloco no *setlist*. Entretanto, uma vez que o *setlist* seja definido como um *setlist* de blocos, somente blocos poderão ser incluídos nele. A mesma lógica vale para as músicas.

A Figura 66 mostra a tela principal dos *setlists* e as Figuras 67 e 68, a listagem de blocos e/ou músicas vinculadas a um *setlist* específico.

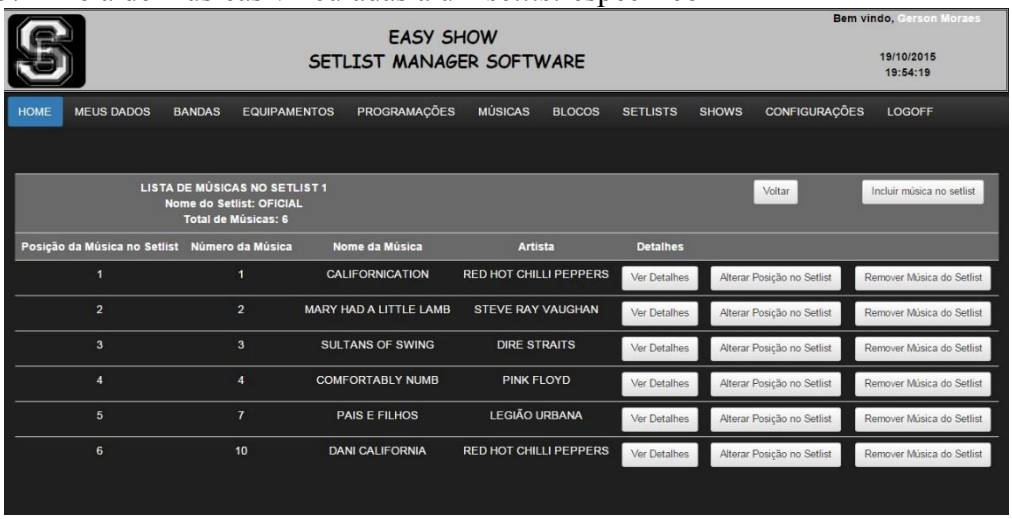
Figura 66 – Tela de *setlists*



Fonte: Do autor.

Dois tipos de *setlists* podem ser criados: *setlists* com músicas ou *setlists* com blocos de músicas. Há bandas que utilizam blocos de músicas; outras, preferem utilizar apenas listagem de músicas. Pensando nesse tipo de situação, o software desenvolvido contempla esses dois tipos de situações.

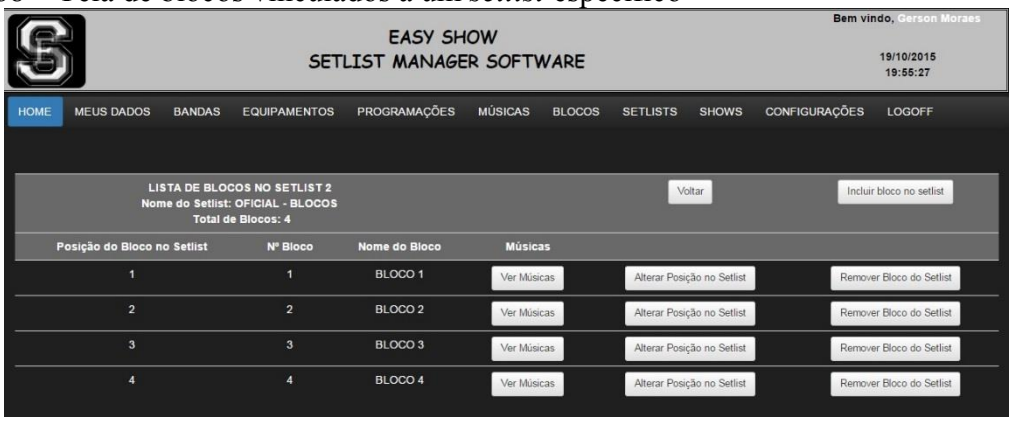
Figura 67 – Tela de músicas vinculadas a um *setlist* específico



Fonte: Do autor.

Ao clicar na opção *Ver Detalhes*, é possível visualizar os dados da música selecionada, e efetuar todas as operações referentes àquela música. A opção *Alterar Posição no Setlist* permite apenas a alteração da posição da música dentro do *setlist*.

Figura 68 – Tela de blocos vinculados a um *setlist* específico



Fonte: Do autor.

Da mesma forma, ao clicar na opção *Ver Músicas*, é possível visualizar a relação de músicas dentro do bloco selecionado, e efetuar todas as operações referentes àquele bloco. A opção *Alterar Posição no Setlist* permite apenas a alteração da posição do bloco dentro do *setlist*.

5.5.10 Tela de shows

Na tela de shows, o usuário pode incluir um novo show, alterar um já existente ou excluir. Para cada show, há um botão que permite consultar o *setlist* vinculado, sendo que, cada show só pode ter um *setlist* vinculado. Clicando no botão *Carregar Show*, todos os dados do respectivo show (banda, blocos/músicas, configurações de pedais, *setlist*, equipamentos e programações) são carregados em memória e o usuário é redirecionado para uma tela de *preview* do show.

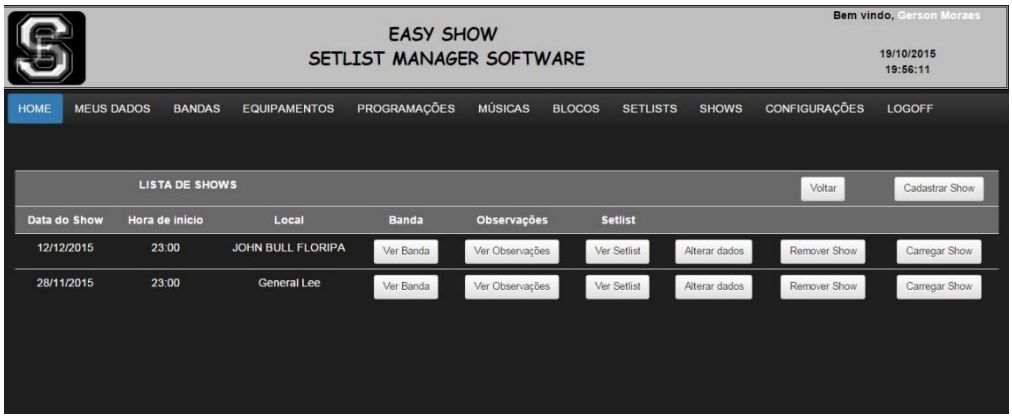
A Figura 69 mostra o *setlist* vinculado a um show específico e a Figura 70, a tela de gerenciamento dos shows.

Figura 69 – Tela de *setlist* vinculado a um show específico

Fonte: Do autor.

Os detalhes do *setlist* utilizado no show são apresentados ao usuário. A opção *Alterar dados do setlist* permite modificar o tipo de *setlist* utilizado no show.

Figura 70 – Tela de shows



Fonte: Do autor.

A opção *Ver Banda* permite visualizar os dados da banda que realizará o show. Clicando em *Ver Observações*, o usuário pode visualizar todas as observações referentes ao show. Na opção *Ver Setlist* é possível verificar qual é o *setlist* utilizado para o respectivo show.

Quando o usuário clica na opção *Carregar Show,* todos os dados referentes àquele show (setlists, blocos/músicas, programações, equipamentos e bandas) são carregados em memória, utilizando o conceito de serialização de arquivos.

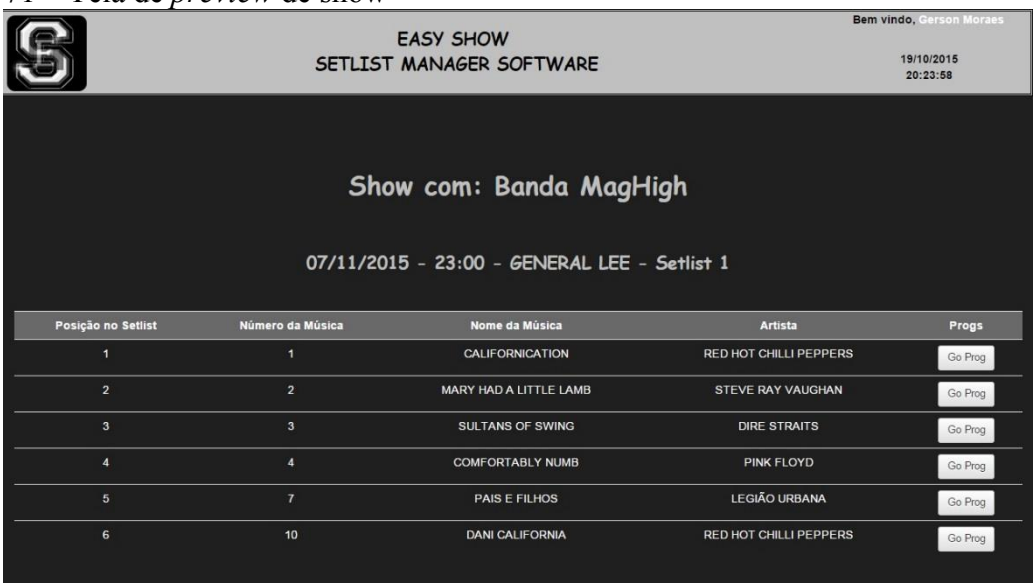
Essa serialização gera arquivos de dados (como se fossem arquivos de bloco de notas) em formato *.dat*, criando um arquivo para cada tipo de entidade envolvida.

Após a serialização e criação dos arquivos, é apresentada uma tela de *preview* do show selecionado.

5.5.11 Tela de *preview* de show

Na tela de *preview*, o usuário visualiza uma tela de apresentação do show carregado. A partir dessa tela, todos os dados do banco já se encontram gravados em memória e são esses dados em memória que serão acessados, para fins de otimização com o hardware. A Figura 71 apresenta o *preview* do show carregado.

Figura 71 – Tela de *preview* de show



Fonte: Do autor.

A partir dessa tela, os dados encontram-se serializados em memória, não necessitando mais da conexão com o banco de dados.

O usuário pode iniciar o show com qualquer música, independentemente da posição no *setlist*. Ao clicar na opção *Go Prog,* os dados são desserializados e carregados para a próxima tela, que é a tela principal de execução do show.

5.5.12 Telas de show

Dependendo do *setlist* do show, uma tela diferente será aberta. Para o caso de *setlists* apenas com músicas, serão visualizadas a relação de músicas no *setlist*, com opção para acesso direto; ponteiros indicando qual é a música anterior, música atual e próxima música; lista de programações da música atual por número sequencial e opções para ir para a programação anterior ou para a próxima programação.

Já, para o caso de *setlists* com blocos de músicas, serão visualizadas a relação de blocos no *setlist*, com opção para acesso direto; ponteiros indicando qual é o bloco anterior, bloco atual e próximo bloco; relação de músicas no bloco, com opção para acesso direto; ponteiros indicando qual é a música anterior, música atual e próxima música; lista de programações da música atual por número sequencial e opções para ir para a programação anterior ou para a próxima programação.

Ainda, para a música atual, há um botão que exibe ou esconde a letra. Caso seja exibida, fica visível no lado direito da tela. A Figura 72 mostra a tela usada para um *setlist* com músicas. Já, a Figura 73 apresenta a tela para um *setlist,* usando blocos de músicas. Finalmente, vemos, na Figura 74, a letra da música sendo exibida na tela.

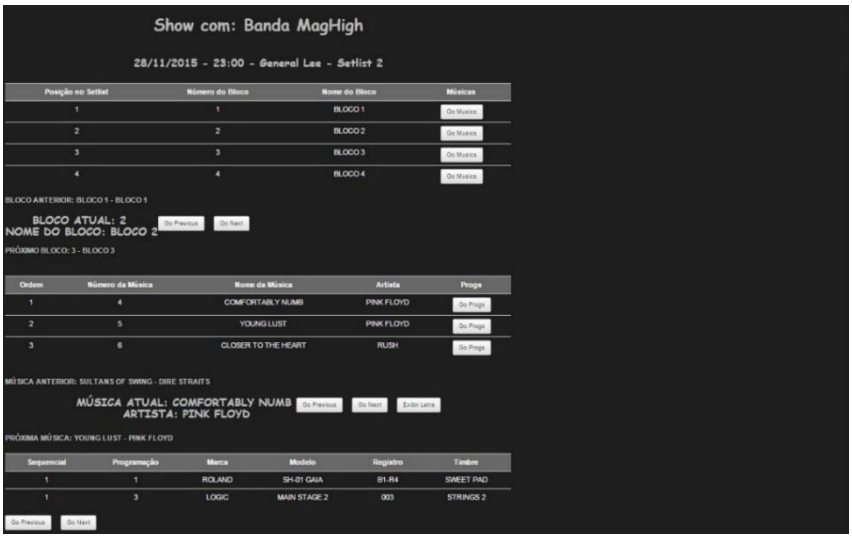
Figura 72 – Tela de show usando *setlist* com músicas

Fonte: Do autor.

No presente caso, é utilizado um *setlist* apenas com músicas. Qualquer música que esteja no *setlist* poderá ser selecionada a qualquer tempo, clicando na opção *Go Progs*. Na opção *Go Previous,* é carregada a música anterior.

Já, na opção *Go Next* é carregada a próxima música. Sempre que uma nova música é carregada, a programação assumirá o sequencial 1, caso haja programações vinculadas à musica carregada.

Figura 73 – Tela de show usando *setlist* com blocos de músicas

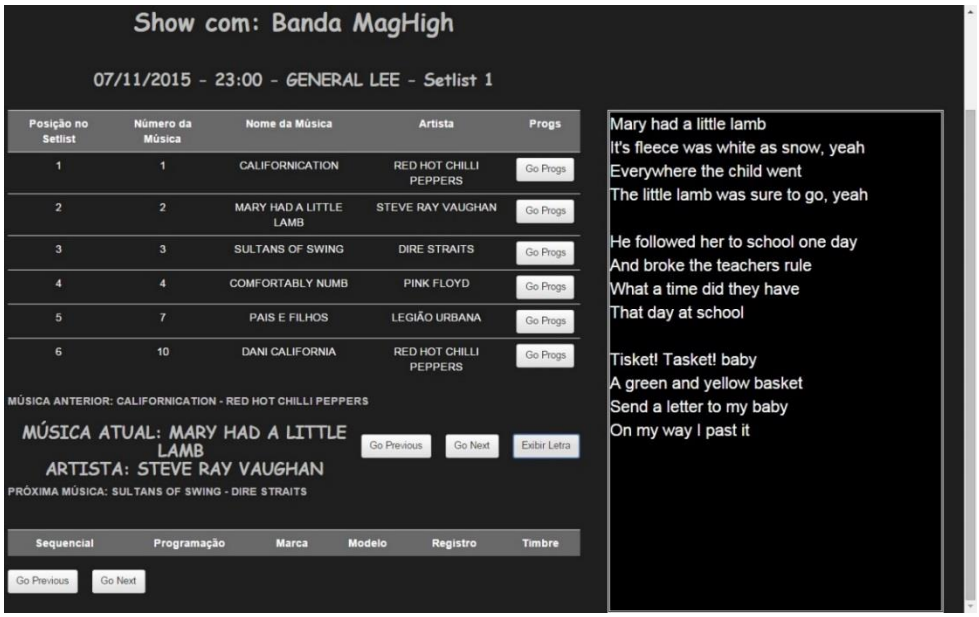


Fonte: Do autor.

Ao carregar um *setlist* com blocos de músicas, qualquer bloco ou música que esteja no *setlist* poderá ser selecionada a qualquer tempo. Ao clicar na opção *Go Musics*, é possível carregar a listagem das músicas do bloco selecionado. Clicando na opção *Go Progs*. Na opção *Go Previous,* é carregado o bloco ou a música anterior.

Já, na opção *Go Next* é carregado o próximo bloco ou a próxima música. Sempre que um novo bloco é carregado, a música atual será a primeira música daquele bloco. No caso de uma nova música ser carregada, a programação assumirá o sequencial 1, caso haja programações vinculadas à musica carregada.

Figura 74 – Letra da música atual sendo exibida



Fonte: Do autor.

A letra da música atual pode ser exibida a qualquer momento, e permanece ativa até que seja ocultada ou a música seja alterada. O botão *Exibir Letra* permite ocultar/exibir a letra da música. Por padrão, a letra fica oculta.

5.6 AVALIAÇÃO

Para fins de avaliação do software, foi realizado o cadastramento de todos os dados utilizados pelo tecladista da Banda MagHigh, que é o autor do presente projeto. Os dados cadastrados foram os seguintes: dados da banda, músicas, equipamentos utilizados, programações, vinculação das programações às músicas, criação dos blocos de músicas, criação do *setlist* oficial da banda e criação dos dados de um show.

Após o cadastramento de todos os dados, o software foi apresentado aos demais integrantes da banda. Por questões de agenda, não foi possível testar o software em um show.

Após os integrantes da banda analisarem o software, os mesmos responderam a um questionário, constante no Apêndice B deste trabalho. Este questionário possui um total de 14 perguntas, sendo 11 delas com 6 opções cada, onde é permitida somente uma opção por questão. Essas opções possuem os seguintes valores: muito bom (5 pontos), bom (4 pontos), regular (3 pontos), ruim (2 pontos), péssimo (1 ponto) e não analisado (0 pontos), sendo que a última opção não interfere na pontuação. As outras três questões são discursivas.

Ao todo, foram efetuadas cinco avaliações, com os demais integrantes da banda. Entre as questões 1 a 11, a pontuação máxima por questão é de 5 pontos, com somatório máximo de 25 pontos.

Com as respostas obtidas, foi possível efetuar uma avaliação do software, com o seguinte resultado:

1. Em relação ao visual, o software obteve 22 pontos (três avaliações com o conceito *bom* e duas avaliações com o conceito *muito bom*);
2. Em relação às funcionalidades de gerenciamento de bandas, o software obteve 21 pontos (quatro avaliações com o conceito *bom* e uma avaliação com o conceito *muito bom*);
3. Em relação às funcionalidades de gerenciamento de equipamentos, o software obteve 22 pontos (três avaliações com o conceito *bom* e duas avaliações com o conceito *muito bom*);

4. Em relação às funcionalidades de gerenciamento de programações, o software obteve 21 pontos (quatro avaliações com o conceito *bom* e uma avaliação com o conceito *muito bom*);
5. Em relação às funcionalidades de gerenciamento de músicas, o software obteve 24 pontos (quatro avaliações com o conceito *muito bom* e uma avaliação com o conceito *bom*);
6. Em relação às funcionalidades de gerenciamento de blocos de músicas, o software obteve 24 pontos (quatro avaliações com o conceito *muito bom* e uma avaliação com o conceito *bom*);
7. Em relação às funcionalidades de gerenciamento de *setlists*, o software obteve 23 pontos (duas avaliações com o conceito *bom* e três avaliações com o conceito *muito bom*);
8. Em relação às funcionalidades de gerenciamento de shows, o software obteve 21 pontos (quatro avaliações com o conceito *bom* e uma avaliação com o conceito *muito bom*);
9. Em relação às funcionalidades de gerenciamento de execução dos shows, o software obteve 22 pontos (três avaliações com o conceito *bom* e duas avaliações com o conceito *muito bom*);
10. Em relação à dinâmica das trocas de programações, blocos e músicas durante a execução do show, o software obteve 23 pontos (duas avaliações com o conceito *bom* e três avaliações com o conceito *muito bom*);
11. Em relação à exibição/ocultação da letra da música, o software obteve 24 pontos (quatro avaliações com o conceito *muito bom* e uma avaliação com o conceito *bom*);

Para a questão 12, foi questionado um ponto forte do sistema. Entre as respostas obtidas, foram destacadas a possibilidade de fragmentação do software em dois módulos distintos (cliente e servidor), a adaptação do sistema às necessidades reais de performances ao vivo e a facilidade no processo de criação e execução de *setlists* de shows, bem como a troca de *patches* de programação de equipamentos conectados.

Em relação à questão 13, foi questionado um ponto fraco do sistema. Foi destacada a necessidade de melhorias no aspecto visual, com melhorias na *interface* gráfica e mudança de posicionamento de alguns componentes.

Finalmente, na questão 14 puderam ser apresentadas considerações, críticas ou sugestões para o sistema apresentado. Entre as respostas obtidas, algumas sugestões muito interessantes surgiram, tais como a inclusão de metrônomo, a possibilidade de incluir/editar partituras, a inclusão do tempo de execução de cada música e a utilização de mecanismos para

tornar a opção de criação de *setlists* mais inteligente, abrindo a possibilidade de o próprio sistema "sugerir" um *setlist*, com base no local do show e no tipo de público presente, utilizando algoritmos baseados em inteligência artificial.

Com as respostas obtidas e a análise dos dados, nota-se que os pontos mais fortes do sistema são o gerenciamento de músicas, blocos de músicas e *setlists*, bem como a dinâmica das trocas de programações durante as músicas e a possibilidade de exibir/ocultar a letra de uma música em tempo de execução.

Entretanto, a parte visual do software, bem como as funcionalidades referentes a bandas, programações e shows não obtiveram pontuação expressiva, haja vista que mais parâmetros podem ser incluídos nesses itens, de modo a tornar o sistema ainda mais completo.

Quanto a sugestão de incluir/editar partituras, a mesma é muito importante, principalmente para tecladistas, pois a maioria deles trabalha diretamente com arranjos, orquestrações e composições, e necessitam de um ambiente no qual possam ler essas partituras. Em relação à sugestão do tempo de execução, é uma funcionalidade que ajuda na avaliação do tamanho de cada bloco e de cada *setlist*.

A função de metrônomo é amplamente utilizada por bandas e músicos que trabalham com VS (instrumentos virtuais gravados, tais como orquestras ou naipes de metais) e sistemas de retorno de som *in-ear*, ou seja, usando fones de ouvido. Já, a utilização de mecanismos e algoritmos de inteligência artificial para a criação de *setlists* é uma opção bastante interessante, que possibilita melhor distribuição dos blocos/músicas dentro de um *setlist,* de acordo com o público presente no local do show.

Com a coleta das informações, novas funcionalidades, características e sugestões foram analisadas e consideradas importantes. O material obtido será devidamente estudado e fará parte dos trabalhos futuros, para fins de posterior implementação.

Ao final do questionário e, para fins de cálculo, os pontos obtidos nas onze primeiras questões foram somados e divididos pelo total de questões de múltipla escolha, ou seja, 11. Assim, o software obteve como pontuação média 22,5 pontos. Para o cálculo da nota de avaliação, foi utilizado um cálculo matemático comum (regra de três simples). Com o cálculo efetuado, o software obteve a nota 9.

Dessa forma, constatou-se que solução ora apresentada apresenta alguns benefícios, tais como: melhor organização, praticidade, velocidade de execução e dinamismo. O software desenvolvido apresenta várias opções, que são bastantes utilizadas por músicos. Essa gama de opções em um único produto possibilita um ambiente agradável e prático, principalmente, durante shows, onde a dinâmica de troca de programações e músicas é essencial.

Uma das funções que não constaram nos objetivos do projeto, mas, que foi desenvolvida e testada com sucesso, foi a funcionalidade de *onTouch,* que permite a utilização do software em notebooks e monitores *touchscreen*, ou seja, sensíveis ao toque.

Percebeu-se que há vários produtos, com características semelhantes às do presente projeto. Entretanto, este apresenta como diferencial a possibilidade de se alterar programações para mais de um instrumento dentro de uma mesma música, em várias cenas dentro dessa mesma música.

Por exemplo, utilizando um hardware específico, que será desenvolvido após a conclusão deste projeto, pode-se alterar a programação de, até, 8 instrumentos ou equipamentos musicais simultaneamente. O fato de não se utilizar as mãos para troca de programações possibilita ao profissional da área maior liberdade para a execução dos trabalhos exercidos.

Em uma próxima etapa, o foco será a construção de uma pedaleira controladora MIDI, compatível com o software desenvolvido, de modo a comprovar a eficácia do software, atuando em conjunto com um hardware.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, são apresentadas as conclusões finais sobre o presente projeto, as dificuldades encontradas, a mudança de escopo em relação ao projeto inicial e os trabalhos futuros a serem desenvolvidos.

6.1 DISCUSSÕES

Ao final do presente projeto, chega-se à conclusão de que o objetivo foi alcançado, ou seja, o software apresenta as funcionalidades inicialmente objetivadas e se mostra bastante eficiente, prático e objetivo. Entre essas funcionalidades, foram implantadas as seguintes: gerenciamento de bandas, músicas, equipamentos, programações, blocos, *setlists* e shows; vinculação de programações a músicas; vinculação de músicas a blocos; criação de *setlists* com músicas ou com blocos de músicas; e exibição/ocultação da letra da música executada.

A funcionalidade de exibir/ocultar a letra da música se mostrou bastante agradável, haja vista o fato de ser muito rápido o acionamento da função. O acesso à letra é imediato, por estar pré-carregada na página.

Outra característica que se mostrou muito útil é a disponibilidade das informações na tela de execução. O músico sabe, exatamente, tudo o que está acontecendo durante a execução: qual é a música anterior, atual e próxima; bloco anterior, atual e próximo bloco; quais as programações ativas no momento; data e hora atual; show, banda e *setlists* utilizados.

Um ponto que apresentou grande dificuldade foi, justamente, a manipulação das informações. No caso de uso de um *setlist* com blocos de músicas, foi necessária a manipulação de várias informações, bem como, a realização de várias pesquisas em memória. Por exemplo:

- ✓ verificar se o bloco atual era o primeiro ou o último do *setlist*;
- ✓ verificar se o bloco anterior era o primeiro do *setlist*;
- ✓ verificar se o próximo bloco era o último do *setlist*;
- ✓ verificar se a música atual dentro do bloco era a primeira ou a última;
- ✓ verificar se a música anterior era a primeira do bloco;
- ✓ verificar se a próxima música era a última do bloco;
- ✓ verificar se o sequencial da programação atual era o primeiro ou o último.

A cada verificação, era necessária a alteração/atualização dos ponteiros envolvidos. Essas manipulações acabaram por tomar um tempo considerável, visto que o autor não tem experiência em programação e que não possui técnica suficientemente apurada para desenvolver um sistema com grande velocidade de codificação.

Em relação à codificação, foi um grande desafio, mas trouxe um acréscimo considerável de conhecimento e experiência. O estudo sobre algumas ferramentas utilizadas trouxe uma bagagem considerável e despertou interesse no aprofundamento sobre as mesmas

6.2 CONSIDERAÇÕES FINAIS

Ao final do presente projeto, o autor encerra esse ciclo com um sentimento de tristeza, por não ter concluído o projeto de hardware que, desde o início da escolha do tema, foi a motivação principal. Por outro lado, há um sentimento de alegria, por superar as dificuldades de programação e implementar, com sucesso, as funcionalidades propostas desde o início do projeto.

Ao final da fase de desenvolvimento, entende-se que o projeto pode vir a se tornar um produto comercialmente rentável. Vários músicos já demonstraram interesse na solução proposta, por ser um facilitador imenso no desenvolvimento das atividades musicais.

Dessa forma, conclui-se o projeto com maior motivação e empolgação para continuidade. Com mais calma, os problemas que ocorreram serão resolvidos e um grande produto pode ser gerado a partir do presente projeto.

6.3 TRABALHOS FUTUROS

Mesmo com a conclusão do software, que se mostrou uma ferramenta bastante versátil, rápida e objetiva para uso em shows, muitas funcionalidades ficaram fora do desenvolvimento.

Somadas à continuidade do projeto de hardware e às respostas ao questionário de avaliação, algumas funcionalidades serão desenvolvidas posteriormente, quais sejam:

- ✓ revisão da programação, separando-a em dois módulos distintos: o módulo de gerenciamento, que será um serviço web, e o de execução, que será utilizado, efetivamente, no notebook do músico;

- ✓ configuração e testes para o perfeito funcionamento do software em ambientes Mac OSX, desde a versão 10.6 (Snow Leopard), até a versão 10.10 (Yosemite);
- ✓ incorporação de partituras à música, utilizando opções de arquivos .PDF ou acesso direto via software externo (Encore, principalmente);
- ✓ incorporação da letra à música, utilizando arquivos .PDF;
- ✓ incorporação de 8 arquivos de áudio para cada música, com botões de seleção para cada arquivo;
- ✓ sincronismo dos botões ProgUp e ProgDown do software aos botões do protótipo apresentado no item 6.2;
- ✓ desenvolvimento de uma pedaleira controladora MIDI, utilizando, como referência, o protótipo apresentado no item 6.2, para que comporte, até, 8 equipamentos conectados simultaneamente;
- ✓ melhorias no aspecto visual do software;
- ✓ incorporação de quatro novos pedais ao hardware, com as funções de *NextMusic*, *PrevMusic*, *NextBlock* e *PrevBlock*;
- ✓ estudos para utilização do software em dispositivos móveis (preferencialmente, tablets com iOS e Android), com suporte à pedaleira controladora MIDI a ser desenvolvida;
- ✓ inclusão do tempo de execução de cada música;
- ✓ estudos para implementação de algoritmos de inteligência artificial para a criação de *setlists* “inteligentes”.

REFERÊNCIAS

8DIO.COM. **Instrumento Virtual 8Dio Solo Violin.** Disponível em: < http://8dio.com/instrument/solo-violin-designer-virtual-instrument-vst/>. Acesso em: 23 ago. 2015.

ABNT. **NBR ISO/IEC 9126-1**. 2003. Disponível em: <http://luizcamargo.com.br/arquivos/NBR%20ISO_IEC%209126-1.pdf>. Acesso em: 26 abr. 2015.

ARDUINO.CC. **Official Homepage of Arduíno.** Disponível em: < http://www.arduino.cc>. Acesso em: 02 mai. 2015.

BARBOSA, Álvaro M. **Edição Digital de Som:** Uma abordagem aos fundamentos da escultura sonora ambientada para criadores. 1999. Disponível em: <http://www.abarbosa.org/docs/edicao_digital_som.pdf>. Acesso em: 03 mai. 2015.

BOOTSTRAP. **Official Homepage.** Disponível em: < http://getbootstrap.com>. Acesso em: 11 out. 2015.

CALIMARO, Instrumentos Musicais. **Pedaleira controladora FBV Shortboard MKII Line 6.** Disponível em: <https://www.calimaro.com.br/pedaleira-controladora-fbv-shortboard-mkii-line-6.html>. Acesso em: 02 mai. 2015.

DI RENNA, Roberto B. et al. **Introdução ao Kit de Desenvolvimento Arduíno**. 2013. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/arduino/Tut_Arduino.pdf>. Acesso em: 28 mar. 2015.

DUCKETT, Jon. **Introdução à Programação Web com HTML, XHTML e CSS.** Rio de Janeiro: Editora Ciência Moderna, 2010.

FIELDS, Duane K. KOLB, Mark A. **Desenvolvendo na Web com JavaServer Pages.** Rio de Janeiro: Editora Campus, 2000.

FILIPEFLOP.COM. **Official Homepage of Filipeflop: Arduíno e Componentes Eletrônicos.** Disponível em: < http://www.filipeflop.com/pd-6b5b3-arduino-mega-2560-r3-cabo-usb.html>. Acesso em: 11 mai. 2015.

GEAR4MUSIC.PT **Official Homepage of Gear4Music: ROLI Seaboard Grand Stage.** Disponível em: < http://www.gear4music.pt/pt/Teclados-e-pianos/ROLI-Seaboard-GRAND-Stage/1BGX#full-des>. Acesso em: 02 nov. 2015.

GHEZZI, Carlo, JAZAYERI, Mehdi. **Conceitos de Linguagens de Programação.** Rio de Janeiro: Editora Ciência Moderna, 2008.

GLOBAL, Novation Music. **Teclado Controlador Novation Impulse 61 teclas.** Disponível em: < http://global.novationmusic.com/keys/impulse#>. Acesso em: 23 ago. 2015.

GOHN, Daniel M. **A tecnologia na música.** INTERCOM – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação – Artigo apresentado no XXIV Congresso Brasileiro da Comunicação. Campo Grande/MS, 2001. Disponível em: <http://www.intercom.org.br/papers/nacionais/2001/papers/NP6GOHN.pdf>. Acesso em: 03 mai. 2015.

GOLÇALVES, Edson. **Dominando JavaServer Faces e Facelets Utilizando Spring 2.5, Hibernate e JPA.** Rio de Janeiro: Editora Ciência Moderna, 2008.

GRANNELL, Craig. **O Guia Essencial de Web Design com CSS e HTML.** Rio de Janeiro: Editora Ciência Moderna, 2009.

GUEDES, Gilleanes T.A. **UML2:** Guia Prático. 2015. Disponível em: <http://www.novateceditora.com.br/livros/uml2/capitulo9788575221457.pdf>. Acesso em: 12 set. 2015.

HEMRAJANI, Anil. **Desenvolvimento Ágil em Java com Spring, Hibernate e Eclipse.** São Paulo: Pearson Education do Brasil, 2007.

HEUSER, Carlos A. **Projeto de Banco de Dados**. 1998. Disponível em: <https://jalvesnicacio.files.wordpress.com/2010/03/carlos_alberto_heuser-projeto_de_banco_de_dados.pdf>. Acesso em: 26 abr. 2015.

IMAGE-LINE, FLStudio. **Instrumento Virtual FLStudio 12.** Disponível em: < https://www.image-line.com/flstudio/>. Acesso em: 23 ago. 2015.

LEMAY, Laura. CADENHEAD, Rogers. **Aprenda em 21 dias –** Java 1.2. Rio de Janeiro: Editora Campus, 1999.

MA, Música e Áudio. **Tj-13M: Bass Pedal controlador MIDI fabricado no Brasil (Review)**. Disponível em: < http://musicapps.com.br/2013/09/tj-13m-bass-pedal-controlador-midi-fabricado-no-brasil-review>. Acesso em: 02 mai. 2015.

MCROBERTS, Michael. **Arduíno Básico**. 2010. Disponível em: <https://novatec.com.br/livros/arduino/capitulo9788575222744.pdf>. Acesso em: 02 mai. 2015.

MELO, Ana C. **desenvolvendo Aplicações com UML 2.2: do conceito à aplicação**. 3ª Ed., Rio de Janeiro: Brasport, 2010. Disponível em: <https://books.google.com.br/books?id=BPVHsG17bAYC&pg=PA72&dq=Prot%C3%B3tipos+de+tela&hl=pt-BR&sa=X&ei=d4NDVbuWF4yWNt7RgKgC&ved=0CE0Q6AEwBQ#v=onepage&q=Prot%C3%B3tipos%20de%20tela&f=false>. Acesso em 12 set 2015.

MENA, Fernanda. Entrevista com Luiz Schiavon in **Cover Teclado.** Ed. Jazz Music Ltda., [1997?], Ed. 12.

MULTILÓGICA SHOP. **Open Source Hardware.** Disponível em: < https://multilogica-shop.com/shield-midi>. Acesso em: 02 mai. 2015.

PADRINI, Marcus. **MA – Música e Áudio**. Disponível em: <http://musicapps.com.br/2013/09/tj-13m-bass-pedal-controlador-midi-fabricado-no-brasil-review/>. Acesso em: 02 mai. 2015.

PEREIRA NETO, Álvaro. **Série Bancos de Dados –** PostgreSQL. São Paulo: Editora Érica Ltda, 2003.

PRADO, Fábio. **MIDI**. 2006. Disponível em: <http://www.maestrofabioprado.com.br/livro/livro.pdf>. Acesso em: 28 mar 2015.

PRODANOV, Cleber C., FREITAS, Ernani C. **Metodologia do Trabalho Científico:** Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. 2ª Ed., 2013. Disponível em: <http://www.feevale.com.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf>. Acesso em: 07 mai 2015.

RATTON, Miguel B. **MIDI:** Guia Básico de Referência**.** Rio de Janeiro: H.Sheldon, 1997.

RATTON, Miguel B. **MIDI Total:** Fundamentos e Aplicações**.** Rio de Janeiro: Editora Música & Tecnologia Ltda., 2005.

RATTON, Miguel B. **Novas tecnologias aplicadas à música**. 2015. Disponível em: <http://www.music-center.com.br/ftp/Novas_Tecnologias_Aplicadas_a_Musica_MRatton.pdf>. Acesso em: 03 mai. 2015.

RODRIGUES, Willian C. **Metodologia Científica**. Paracambi: FAETEC/IST, 2007. Disponível em: <http://unisc.br/portal/upload/com_arquivo/metodologia_cientifica.pdf>. Acesso em 07 mai 2015.

SEBESTA, Robert W. **Conceitos de Linguagens de Programação, 5ªed**. 2006. Disponível em: < http://www.pdf-archive.com/2013/09/12/conceitos-de-linguagens-de-programac-o/conceitos-de-linguagens-de-programac-o.pdf>. Acesso em: 25 abr. 2015.

SEBESTA, Robert W. **Concepts of Programming Languages, 10thed**. 2012. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0CFIQFjAH&url=http%3A%2F%2Fwww.kau.edu.sa%2FGetFile.aspx%3Fid%3D203014%26fn%3DBook.pdf&ei=Szw9VezLE4WkNvWIgPAE&usg=AFQjCNH1hhgzchNY14SNeUvRmy7kSfN1oA&bvm=bv.91665533,d.eXY>. Acesso em: 25 abr. 2015.

SOMMERVILLE, Ian. **Engenharia de Software, 6ªed.** São Paulo: Addison-Wesley, 2003.

SOUNDSONLINE, EastWest. **Instrumento Virtual Symphonic Choirs.** Disponível em: < http://www.soundsonline.com/Symphonic-Choirs>. Acesso em: 23 ago. 2015.

TECLACENTER, Instrumentos Musicais. **Pedaleira controladora Roland FC-7.** Disponível em: <http://sejalivre.org/saiba-tudo-sobre-arduino>. Acesso em: 02 mai. 2015.

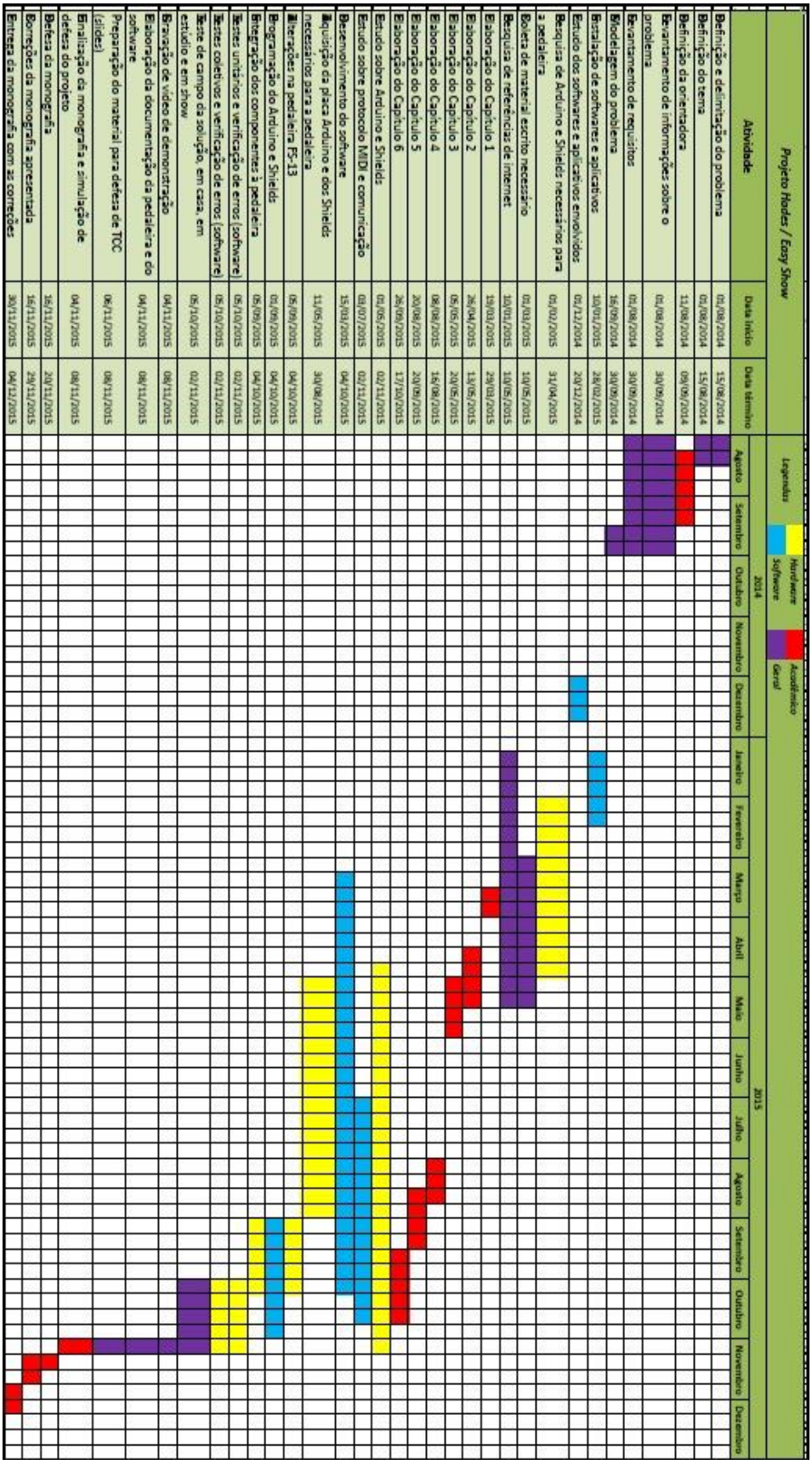
TOONTRACK, Upgrade to EZDrummer2. **Instrumento Virtual EZDrummer2.** Disponível em: < https://www.toontrack.com/product/ezdrummer-2/>. Acesso em: 23 ago. 2015.

VIEIRA, Vinícius. **Saiba tudo sobre Arduíno.** 2015. Disponível em: <http://sejalivre.org/saiba-tudo-sobre-arduino>. Acesso em: 02 mai. 2015.

WAZLAWICK, Raul S. **Análise e Projeto de Sistemas de Informação Orientados a Objetos, 5ªed**. Rio de Janeiro/RJ: Elsevier, 2011. Disponível em: <https://books.google.com.br/books?id=Cf6tE2zISf0C&printsec=frontcover&dq=An%C3%A1lise+e+Projeto+de+Sistemas+de+Informa%C3%A7%C3%A3o+Orientados+a+Objetos&hl=pt-BR&sa=X&ei=uVAZVbLsHovFgwTIhYLABw&ved=0CCoQ6AEwAA#v=onepage&q=An%C3%A1lise%20e%20Projeto%20de%20Sistemas%20de%20Informa%C3%A7%C3%A3o%20Orientados%20a%20Objetos&f=false >. Acesso em 12 set 2015.

APÊNDICES

APÊNDICE A – CRONOGRAMA DE ATIVIDADES



APÊNDICE B – QUESTIONÁRIO DE AVALIAÇÃO DO SOFTWARE

Questionário de Avaliação do Software

Para fins de avaliar o software desenvolvido, o presente questionário de avaliação foi elaborado. Este questionário possui um total de 14 perguntas, sendo 11 delas com 6 opções cada, onde será permitida somente uma opção por questão. Essas opções possuem os seguintes valores: muito bom (5 pontos), bom (4 pontos), regular (3 pontos), ruim (2 pontos), péssimo (1 ponto) e não analisado (0 pontos). Ainda, há duas questões nas quais devem ser indicados um ponto forte e um ponto fraco do sistema. A última questão é uma questão aberta, na qual podem ser indicadas críticas ou sugestões.

Em relação às questões de 1 a 11, será feita a somatória dos pontos das mesmas, cuja a média da somatória dará uma nota conceito ao software. As demais questões serão utilizadas como referência para posteriores melhorias no sistema.

1. Em relação ao visual, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

2. Em relação às funcionalidades de gerenciamento de bandas, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

3. Em relação às funcionalidades de gerenciamento de equipamentos, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

4. Em relação às funcionalidades de gerenciamento de programações, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

5. Em relação às funcionalidades de gerenciamento de músicas, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

6. Em relação às funcionalidades de gerenciamento de blocos de músicas, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

7. Em relação às funcionalidades de gerenciamento de *setlists*, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

8. Em relação às funcionalidades de gerenciamento de shows, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

9. Em relação às funcionalidades de execução dos shows, o software é:
   ( ) muito bom ( ) ruim
   ( ) bom ( ) péssimo
   ( ) regular ( ) não analisado

10. Em relação à dinâmica das trocas de programações, blocos e músicas durante a execução do show, o software é:
    ( ) muito bom ( ) ruim
    ( ) bom ( ) péssimo
    ( ) regular ( ) não analisado

11. Em relação à exibição/ocultação da letra da música, esse item pode ser considerado:

( ) muito bom ( ) ruim

( ) bom ( ) péssimo

( ) regular ( ) não analisado

12. Cite um ponto forte do sistema apresentado:

_____

_____.

13. Cite um ponto fraco do sistema apresentado:

_____

_____.

14. Faça as suas considerações, críticas ou sugestões sobre o software desenvolvido:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____.