UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

GUILHERME AGOSTINI

SISTEMA WEB PARA ESTUDO DE TEORIA MUSICAL

BENTO GONÇALVES

2022

GUILHERME AGOSTINI

SISTEMA WEB PARA ESTUDO DE TEORIA MUSICAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias, Campus Universitário da Região dos Vinhedos, da Universidade de Caxias do Sul.

Orientador: Prof. Dr. Daniel Luis Notari

BENTO GONÇALVES

2022

GUILHERME AGOSTINI

SISTEMA WEB PARA ESTUDO DE TEORIA MUSICAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias, Campus Universitário da Região dos Vinhedos, da Universidade de Caxias do Sul.

Aprovado em 29/11/2022

BANCA EXAMINADORA

Prof. Dr. Daniel Luis Notari
Universidade de Caxias do Sul - UCS

Prof. Msc. Gabriele Dani
Universidade de Caxias do Sul - UCS

Prof. Dr. Leonardo Dagnino Chiwiacowsky
Universidade de Caxias do Sul - UCS

AGRADECIMENTOS

Agradeço a DEUS pela minha vida e por me guiar no caminho, pois, sem ele, hoje, eu não estaria onde estou. À minha família, pelo incentivo e apoio, pois sempre estiveram juntos comigo neste trajeto acadêmico. Aos meus colegas e amigos, por me darem suporte. Aos professores Ricardo Maicá, por esclarecer minhas dúvidas, e Marcello Caminha, pelos ensinamentos de música.

Agradeço ao meu orientador, Profº Daniel Luis Notari, por me guiar neste trabalho. Gratidão a todos que estiveram comigo nesta caminhada e que me apoiaram e compartilharam conhecimento para que eu pudesse alcançar este objetivo.

"A música exprime a mais alta filosofia numa linguagem que a razão não compreende."

Arthur Schopenhauer

RESUMO

Atualmente, o aprendizado musical com auxílio da tecnologia vem se tornando um aliado para o estudo, e expandindo cada vez mais as novas ferramentas tecnológicas e metodológicas. Porém, os alunos possuem dificuldade de aprender devido à falta de metodologia ou pela dificuldade de sair de casa. Este trabalho apresenta a implementação de uma aplicação web para o estudo de teoria musical. O objetivo é fornecer um sistema web que proporcione o estudo de música para os alunos e que desenvolva a compreensão do conceito musical. Foram selecionados os quatro conceitos musicais básicos e fundamentais, de modo que os músicos iniciantes possam evoluir para a etapa posterior. Para o desenvolvimento deste projeto, é apresentada a teoria musical básica de uma forma explicativa e simplificada, a elaboração da modelagem de software através do uso da metodologia ICONIX para a construção do projeto e a arquitetura utilizada para desenvolvimento de aplicação web. Para a realização do teste automatizado, foi utilizado o JUnit. O resultado obtido neste projeto foi satisfatório e contribuiu no auxílio aos usuários para o aprendizado musical, além disso permite acrescentar novas funcionalidades futuras para ser um software web de teoria musical completo.

Palavras-chave: Aplicação web. Modelagem de software. Arquitetura da aplicação. Teoria Musical.

ABSTRACT

Nowadays, the musical learning with the aid of technology has become an ally for the study, and expanding more and more the new technological and methodological tools. However, students have difficulty learning due to lack of methodology or difficulty leaving home. This work presents the implementation a web application for the study of musical theory. The goal is to provide a web system that provides the study of music for students and that develops the understanding of the musical concept. We selected the four basic and fundamental musical concepts, so that the beginner musicians can evolve to the later stage. For the development of this project, the basic musical theory is presented in an explanatory and simplified way, the elaboration of software modeling through the use of the ICONIX methodology for project construction and architecture used for web application development. Junit was used to perform the automated test. The result achieved in this project was satisfactory and contributed in the aid to the users for musical learning, besides allows to add new future functionalities to be a complete music theory web software.

Keywords: Web Application. Software Modelling. Application architecture. Theory Musical.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

API *Application Programming Interface*

XP *Extreme Programing*

RUP *Rational Unified Process*

MVC *Model View Controller*

HTML *Hyper Text Markup Language*

JPA *Java Persistence API*

JAR *Java Archive*

IDE *Integrated Development Environment*

LISTA DE SÍMBOLOS

C	Dó
D	Ré
E	Mi
F	Fá
G	Sol
A	Lá
B	Si
T	Tom
ST	Semitom
$\sharp$	Sustenido
$\flat$	Bemol
$G\sharp$	Sol sustenido
$G\flat$	Sol bemol
$C\sharp$	Dó sustenido
$D\flat$	Ré bemol
$D\sharp$	Ré sustenido
$E\flat$	Mi bemol
$F\sharp$	Fá sustenido
$A\flat$	Lá bemol
$A\sharp$	Lá sustenido
$B\flat$	Si bemol
m	menor

SUMÁRIO

1 INTRODUÇÃO

A expansão da tecnologia tem favorecido o avanço dos estudos de música, de vários métodos diferentes de ensino, teorias e técnicas musicais, desfrutando do dispositivo tecnológico como um auxílio com abertura plausível para o aperfeiçoamento do aprendizado (CONSTANTE; ARAÚJO; SCHIAVONI, 2019). A busca de procedimentos não tradicionais combinados com os tradicionais eleva o nível de experiência com a expectativa de melhor desempenho da atividade musical (MASKE, 2000; MOTA, 2019).

De acordo com Dávila e Porto (2021), a procura tem se intensificado nos últimos anos e podem ser encontrados diversos aplicativos de treinamento de música (m.a.estro)[1], curso online (Manual do Violão Gaúcho)[2], plataforma de curso como *Udemy* e até os canais de cursos gratuitos do *Youtube*. Ainda segundo o autor, tem contribuído a atração dos novos músicos e compositores, devido à inspiração do aprendizado para tocar violão e para que os instrutores possam recorrer à aplicação como amparo e planejamento de aulas.

O uso de fundamentos teóricos musicais incentiva os usuários, propiciando uma a nova experiência de praticar o estudo, fornecendo auxílio aos professores músicos e aos alunos aprendizes. A computação na área de música vem crescendo em vários contextos musicais, como harmonia musical, notação, percepção musical e outros (ROMANO; SOBRINHO, 2016).

A contribuição deste trabalho é apresentar os fundamentos da harmonia musical utilizando software para a manipulação das combinações de notas por meio do aprendizado e do estudo. O programa não deve substituir o instrutor musical, é apenas uma ferramenta de auxílio para conhecimento (MUSIC, 2007; ROMANO; SOBRINHO, 2016).

1.1 PROBLEMA DE PESQUISA

Segundo Souza e Pedro (2015), a grande dificuldade do aprendizado está na falta de metodologia do estudo de teoria musical, deixando parte da população que mantém contato com o instrumento musical, partituras e teorias sem o domínio básico. Conforme Santana e Rovaron (2020), “Tendo como motivo para esta mudança o baixo custo para adquirir o conhecimento e também, pela facilidade de local e horário, onde geralmente são estudos auto direcionados que o usuário utiliza a aplicação, em momentos livres ao decorrer do seu dia a dia.”

Segundo Castro Jr. (2019), identificou-se a evolução na área musical relacionada à tecnologia:

> No âmbito da música, as novas tecnologias abarcam as tecnologias digitais, que começam a adentrar os espaços na evolução dos instrumentos musicais,

[1] <https://play.google.com/store/apps/details?id=com.pirilampomestre.maestro&hl=pt_BR&gl=US>

[2] <https://manual.marcellocaminha.com/3984985/mvg-new/mvginscricao>

> de recursos eletrônicos e aparelhos digitais. Portanto, os espaços de educação musical começaram a receber interferências da tecnologia nesta conjuntura.

Entre os problemas citados pelos autores, ressaltam os seguintes: falta de metodologia, falta de flexibilidade de horário, deslocamento e problema financeiro. Contudo, salienta-se que é importante aproveitar as tecnologias digitais para melhorar a experiência dos usuários.

1.2 OBJETIVO GERAL

O sistema web a ser desenvolvido tem por finalidade de auxiliar no aprendizado de conceitos da teoria musical.

No intuito de se chegar a este objetivo, os objetivos específicos deste trabalho são:

- compreender a teoria musical básica;
- projetar um sistema web sobre o conteúdo de teoria musical básico;
- desenvolver sistema web;
- testar a validação do sistema web.

1.3 ESTRUTURA DO TRABALHO

O documento apresentado está organizado desta forma: o capítulo 1 apresenta a introdução, problema de pesquisa e objetivo. O capítulo 2 elucida os conceitos referentes à teoria musical e apresenta os trabalhos relacionados. No capítulo 3, descreve-se a proposta de solução do projeto web, apresentando modelagem de software e arquitetura do projeto. No capítulo 4, descreve o desenvolvimento do sistema. No capítulo 5, expõe os testes e apresenta os resultados. Por fim, no capítulo 6, apresenta-se a conclusão.

2 REFERENCIAL TEÓRICO

Neste Capítulo, são apresentados os conceitos da teoria musical, abordando os conteúdos teóricos da música. Na continuidade, serão apresentados os trabalhos relacionados e como é o ensino de música.

2.1 CONTEXTO GERAL DAS NOTAS MUSICAIS

De acordo com Kroger (2012), "Uma nota é um símbolo que representa um som musical", ou seja, é um nome que se dá para identificar o som musical. Para a facilidade de compreensão e estudo, Guido de Arezzo (992 - 1050), considerado como o pai das notas musicais e fundamentador do sistema de notação musical, determinou a construção de escala musical. As notas naturais, que também chamamos de **notas diatônicas**, foram batizadas como **dó**, **ré**,**mi**, **fá**, **sol**, **lá** e **si** representadas por cifras de **C**, **D**, **E**, **F**, **G**, **A** e **B**, extraídas de um hino em latim a São João Batista, composto por Paulo Diácono (séc. VIII) (JESUS, 2017).

O conjunto de notas naturais possui intervalos que são a medida de distância entre duas notas sucessivas. O intervalo é designado de tom, representado pelo símbolo **T**, e semitom, representado pelo símbolo **ST**. Conforme a Figura 1, a sequência de tons de intervalo da escala maior natural é definido desta forma: **T**, **T**, **ST**, **T**, **T**, **T**, **ST** (JESUS, 2017).

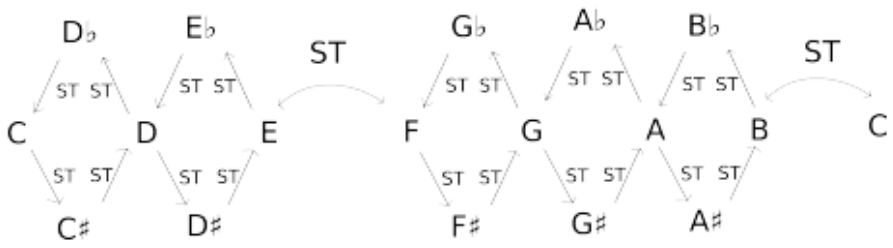
Figura 1 – Escalas Maiores

Intervalos da escala	*Tom*	*Tom*	*Semi-Tom*	*Tom*	*Tom*	*Tom*	*Semi-Tom*	
Escala Dó maior	C	D	E	F	G	A	B	C
Escala Ré maior	D	E	F#	G	A	B	C#	D
Escala Mi maior	E	F#	G#	A	B	C#	D#	E
Escala Fá maior	F	G	A	Bb	C	D	E	F
Escala Sol maior	G	A	B	C	D	E	F#	G
Escala Lá maior	A	B	C#	D	E	F#	G#	A
Escala Si maior	B	C#	D#	E	F#	G#	A#	B

Fonte: Adaptado de Jesus (2017).

As notas acidentais ou enarmônicas se caracterizam pela mesma nota porém com grafias diferentes. Pode ser aumentado um semitom, com nomenclatura de '♯' (sustenido), ou diminuído um semitom, com nomenclatura de '♭' (bemol). Por exemplo, a nota **G**♯ (sol sustenido) é um semitom aumentado em relação à nota **G** (sol maior). Da mesma forma que o **G**♭ (sol bemol), que é um semitom diminuído em relação à nota **G** (sol maior) (KROGER, 2012). O intervalo entre as notas **E** (mi maior) e **F** (fá maior) e também **B** (si maior) e **C** (dó maior) não contém notas de acidentais, pois entre eles existe um semitom (JESUS, 2017). A Figura 2 exemplifica os intervalos musicais incluindo as notas de acidentes, formando, assim, o conjunto completo de 12 notas.

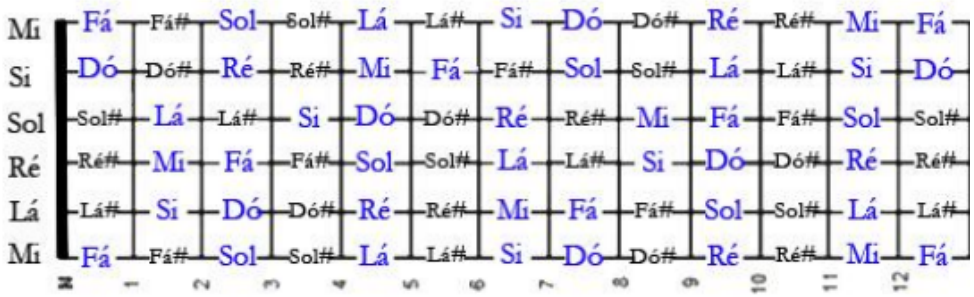
Figura 2 – Intervalo de notas musicais



Fonte: Adaptado de Jesus (2017).

É importante o violonista memorizar as notas no braço do violão, a fim de trabalhar diversas formas para ganhar eficiência nos estudos. Isso facilita localizar as notas para posicionar os dedos, além de descobrir as notas para formação de acorde. Conforme a Figura 3, a ordem das notas da corda solta começa **Mi**, **Si**, **Sol**, **Ré**, **Lá**, **Mi** (de cima para baixo) e a numeração da esquerda para a direita é a sequência de casa com as notas posicionadas ao longo do braço do violão (SOUZA, 2016b).

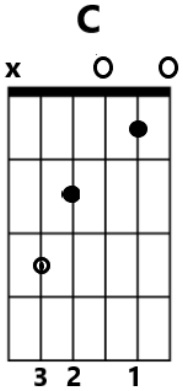
Figura 3 – Notas no braço do violão



Fonte: Adaptado de Souza (2016b).

Diagrama de acorde é uma figura de montagem de acorde. De acordo com a Figura 4, é representado o acorde **C** (dó maior) no braço do violão, onde o "**x**" indica que a corda não deve ser tocada, o círculo com fundo branco representa corda solta, os círculos com fundo preto que estão em cima da linha são a posição de dedos para pressionar cordas, o círculo e um ponto no meio representa a nota principal do acorde e a numeração que está no inferior do diagrama representa a posição dos dedos (JESUS, 2017).

Figura 4 – Diagrama de Acorde C



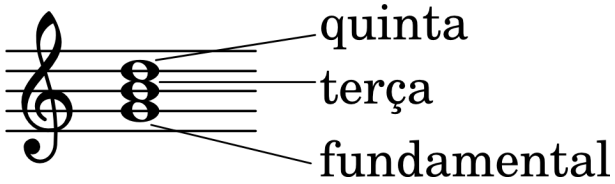
Fonte: O autor (2022), adaptado de Jesus (2017).

2.2 FORMAÇÃO DE ACORDES DE TRÍADE E TÉTRADE

Acordes são formados por um agrupamentos de sons, que são combinações de notas tocadas sincronicamente as quais chamamos de “linguagem harmônica” (MONZO, 2016). Esta expõe o detalhe de como os acordes são formados, descrevendo os tipos que mais advêm das configurações de estrutura de composição de notas.

Tríade é formado por 3 notas e está constituída por 2 intervalos de terça, que estão entre nota fundamental e terça, e entre terça e quinta, conforme a Figura 5, que representa os sons **I** graus (nota fundamental), **III** graus (terça maior ou menor) e **V** graus (quinta diminuta ou aumentada) (KOSTKA; DOROTHY, 2015).

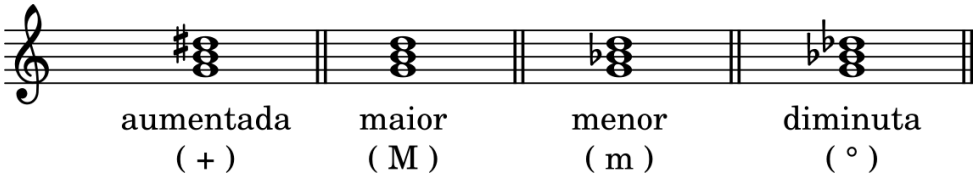
Figura 5 – Formação de tríade



Fonte: Adaptado de Kostka e DOROTHY (2015).

De acordo com a Figura 6, podem ser classificados quatro tipos de tríades: **III** graus define os modos maior e menor e o **V** graus define aumentada ou diminuta.

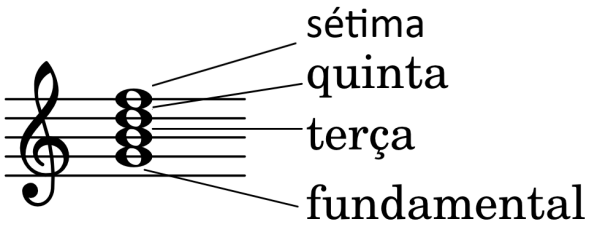
Figura 6 – Tipos de Tríades



Fonte: Adaptado de Kostka e DOROTHY (2015).

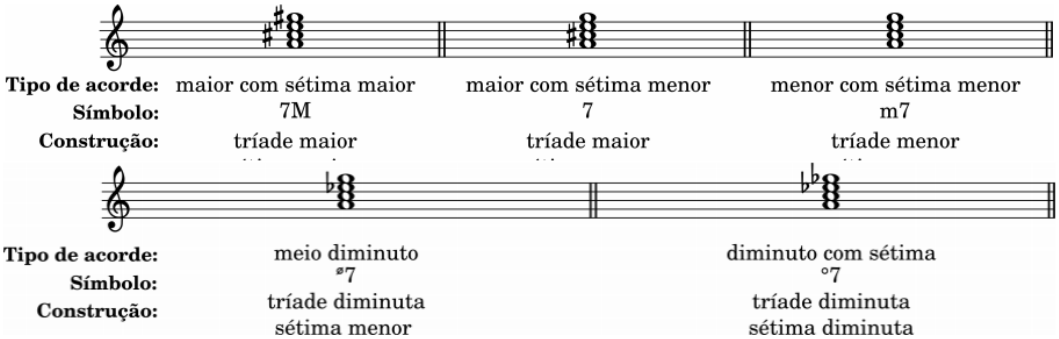
Além das 3 notas da tríade, é incluída uma quarta nota (**VII** graus) após a quinta, de acordo com a Figura 7, formando o tétrade, que também é chamado de *acordes com sétima*, devido ao intervalo entre a nota tônica e a adicionada. Existem cinco tipos de tétrades, conforme a Figura 8 (KOSTKA; DOROTHY, 2015).

Figura 7 – Formação de Tétrade



Fonte: O autor (2022) adaptado de Kostka e DOROTHY (2015).

Figura 8 – Tipos de Tétrade



Fonte: Adaptado de Kostka e DOROTHY (2015).

Os modos podem ser configurados na terça, que definem o intervalo em relação à nota fundamental. Possuem dois tipos de modos: terça maior, que é formado por dois tons e a terça menor, que é formada por um tom e um semitom (um tom e meio) (GONÇALVES; FILHO, 2015). O acorde de tríade do modo maior como **A** (lá maior) é constituída por notas de **A** (lá maior), **C**♯ (dó sustenido) e **E** (mi maior). A tríade do modo menor é acrescentado "**m**" após a cifra como, por exemplo, **Am** (lá menor), que é formado por notas de **A** (lá maior), **C** (dó maior) e **E** (mi maior) (CARDOSO, 2015).

2.3 TRANSPOSIÇÃO

A transposição é um processo usado para subir ou diminuir o tom da música de modo que combine a tonalidade do canto. Para transpor a tonalidade de uma música para outra tonalidade, por exemplo, o tom de uma música é **C** (dó maior), deve-se subir 3 tons e 1 semitom (3 e

½ tons) e o resultado será **G** (sol maior), ou seja, 3 e ½ tons é a distância calculada que separa a tonalidade de **G** (sol maior) em relação ao tom original, conforme Tabela 1 (LIMA *et al.*, 2006).

Tabela 1 – Transposta de 3 e ½ tons de distância

Distância	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Tonalidade	C	C♯	D	D♯	E	F	F♯	G	G♯	A	A♯	B

Fonte: O autor (2022).

Os acordes podem ser aplicados na mesma lógica, porém deve ser respeitada a sua função de música, ou seja, se o acorde original for o *acordes com sétima* ("**7**"), o acorde transposto também deverá ser o *acordes com sétima*, e assim por diante. Supondo o campo harmônico de **C** e utilizando a distância de 3 tons e 1 semitom, os outros acordes da sequência do campo harmônico deverão ser transpostos da mesma forma. A Tabela 2 exemplifica como ficará uma sequência em **C** (dó maior) transposta para **G** (sol maior) (LIMA *et al.*, 2006).

Tabela 2 – Transposta de acordes

Tonalidade Original	C	Am	Dm	G7
Tonalidade Transposta	G	Em	Am	D7

Fonte: O autor (2022).

2.4 TRABALHOS RELACIONADOS

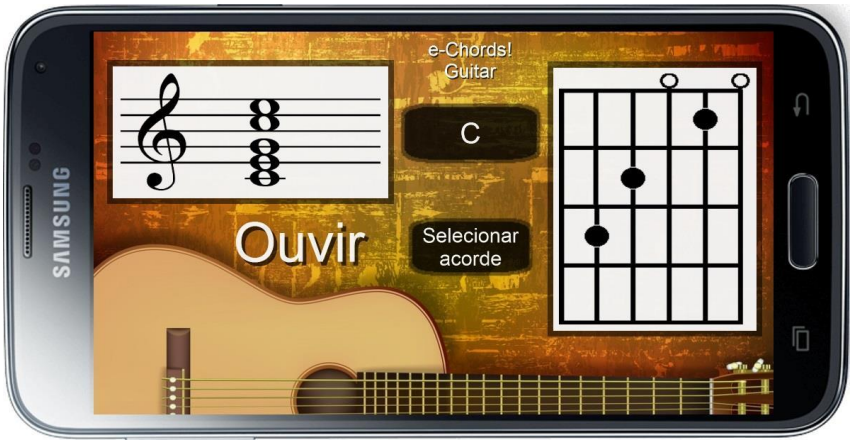
Atualmente existem diversos softwares/aplicativos que utilizam várias formas de ensino de teoria musical. Essas aplicações apresentadas possuem diversos métodos de ensino, destacando-se a utilização de sons, tablatura ou partitura e até mesmo exercícios.

2.4.1 e-Chords! Guitar

O e-Chords! Guitar é um aplicativo de dispositivo móvel para a aprendizagem musical, que aproxima os usuários. A aplicação dispõe o dicionário de acordes com diagrama de digitação (desenho de braço de violão), partitura (escrita musical) e sonoridade (áudio). As figuras musicais foram editadas através de editores gráficos e áudios gravado por conta de características sonoras. Os diagramas de acordes foram buscados na referência do autor Korsakov (1886, p.1-158) e Schoenberg (1911, p.1-580) (NETO, 2019). A Figura 9 apresenta a tela principal do aplicativo.

A interação com aplicação é bem simples e intuitiva: caso queira selecionar acorde, poderá ser usado o botão “Selecionar acorde”, que irá apresentar a lista de acordes e, caso se deseja ouvir o som do acorde, poderá ser usado o botão “Ouvir” e este será reproduzido. Foi

Figura 9 – Apresentação do aplicativo



Fonte: Adaptado de Neto (2019).

programado em linguagem Lua por suas características de simplicidade de programação (NETO, 2019).

2.4.2 Teoria.com

É um site[1] que funciona desde 1997, dedicado ao ensino e prática musical. Tem sido muito utilizado por professores e alunos, contribuindo para o treinamento de ouvido e para a avaliação de teoria e percepção musical. A tela do site é exibida conforme a Figura 10 (MOTA, 2019).

O referido site é dividido em seções:

- **Tutoriais**: representa a abstração da teoria musical;
- **Referência**: detalha uma referência à teoria da música;
- **Artigos**: complementa com artigos musicais e análise;
- **Exercícios**: para praticar e desenvolver a habilidade musical teórica, incluindo a explicação nos exercícios. Essa seção é composta pelas seguintes teorias musicais: treino de leitura musical, intervalos, escalas, acordes, funções harmônicas e formas musicais;

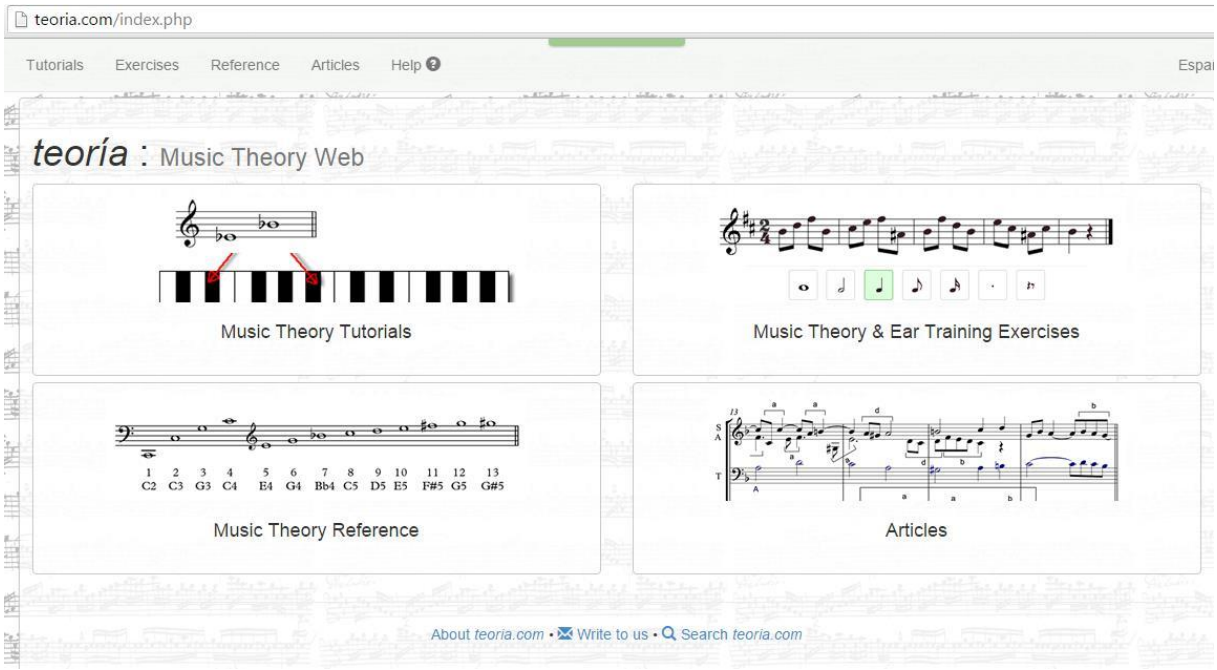
O site é gratuito e está em constante atualização. O uso dessa tecnologia exige que os iniciantes tenham uma base sólida, com a orientação do instrutor musical (MOTA, 2019).

2.4.3 No tom

Foi desenvolvido para sistema operacional Android da Google versão 4.4 (KitKat) ou superior e foi desenhado por design visual, em 2009. O aplicativo tem como objetivo introduzir

[1] <https://www.teoria.com/>

Figura 10 – Apresentação do site



Fonte: Adaptado de Mota (2019).

o estudo teórico musical básico. Possui três funções: afinador, metrônomo e aulas (SANTANA; ROVARON, 2020).

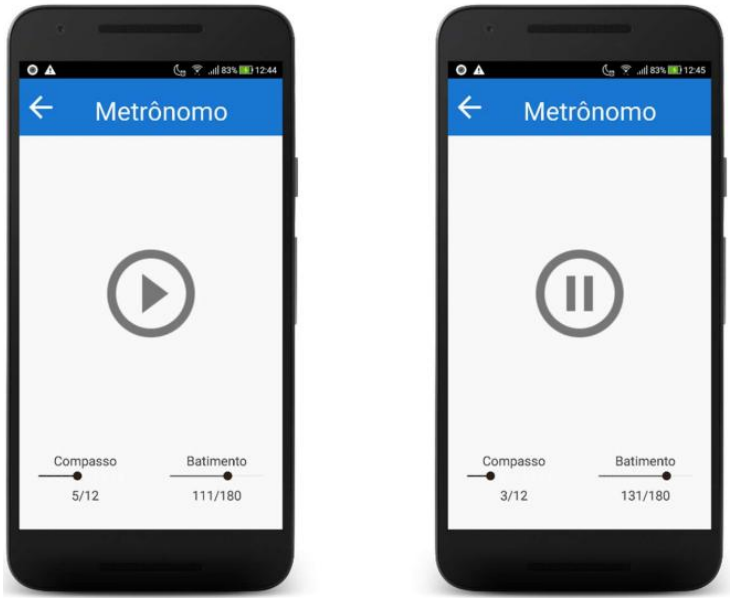
Seguem descritas as funções do aplicativo No Tom:

- **Afinador**: Serve para públicos iniciantes para auxiliar na afinação de violão. Na tela irá aparecer uma agulha central, que serve para indicar uma nota fundamental a ser afinada, que poderá ocorrer em duas situações: uma situação é quando a nota tocada é baixa ou alta; então, a agulha irá ajudar o usuário a passar a nota para outra, indicando a nota correta. A segunda situação é quando a nota indicada é tocada igual. Então, gerará uma mensagem informando a nota tocada. A Figura 11 mostra a tela de afinação.
- **Metrônomo**: Desenvolve a habilidade rítmica com marcação de tempo e ritmo da música. Na tela, o ritmo é iniciado com compasso binário e valor de batimento 60. O usuário poderá selecionar o compasso desejado em até 12 andamentos, além de alterar o batimento, que poderá ser variado entre 60 e 180 por minuto. O som emitido é "tick-tack", que representa a marcação de cada tempo. A Figura 12 apresenta a tela do metrônomo.

Figura 11 – Apresentação da tela de Afinador: 1° Nota baixa; 2° Nota Afinada; 3° Nota alta

Fonte: Adaptado de Santana e Rovaron (2020).

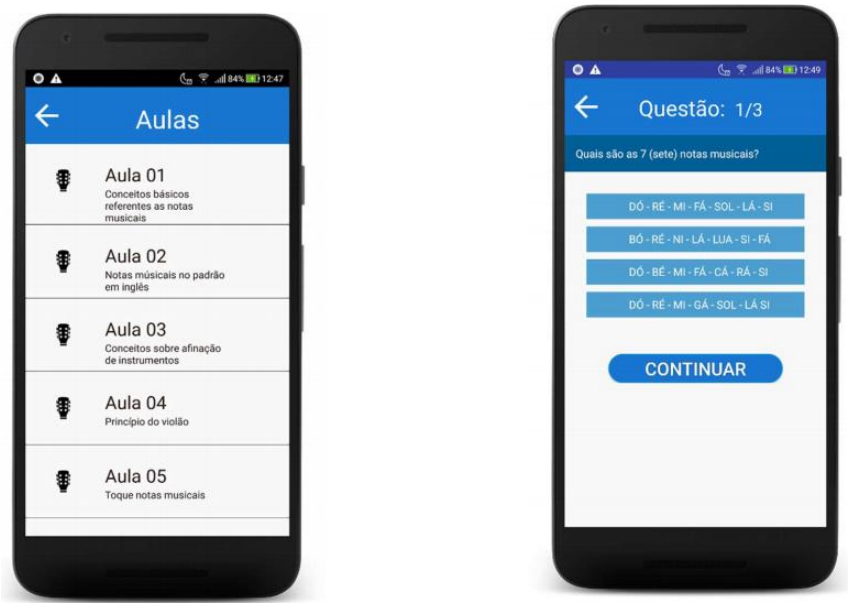
Figura 12 – Apresentação da tela do Metrônomo: 1° Início padrão; 2° Batimento alterado



Fonte: Adaptado de Santana e Rovaron (2020).

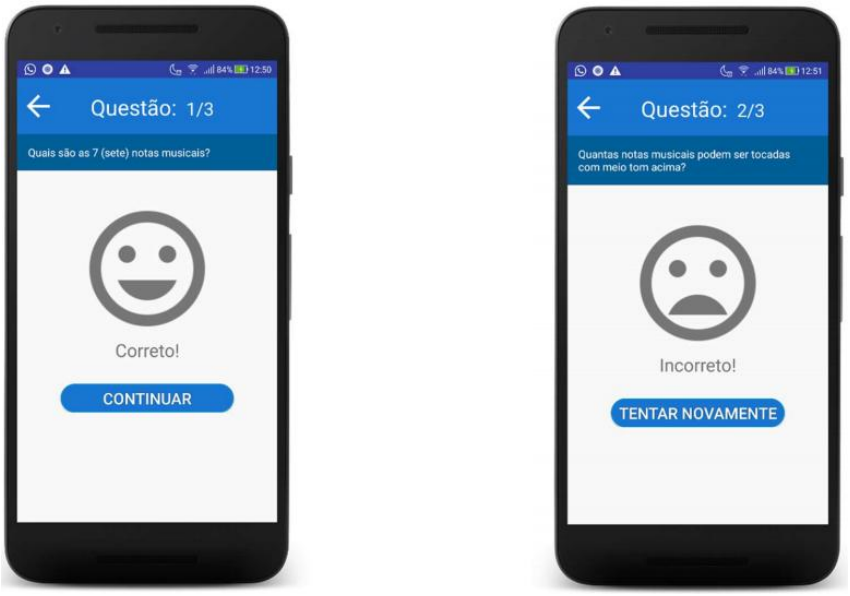
- **Aulas**: aulas são divididas em seis seções, de acordo com a dificuldade do conteúdo pré-ordenado. As aulas 1 e 4 contêm questões em forma de quiz, com o tema teoria musical. O usuário terá acertos e erros e também poderá desistir ou recomeçar a aula. A aula 5 mostra a prática musical e o usuário poderá tocar uma nota musical. Por último, na aula 6, o usuário poderá tocar uma nota musical simples com grau de dificuldade maior, incluindo o tempo para a execução de ritmo da música. As telas das Figuras 13 a 15 do aplicativo mostram a função Aulas.

Figura 13 – Apresentação da tela de Aulas: 1° Menu Aulas; 2° Questões



Fonte: Adaptado de Santana e Rovaron (2020).

Figura 14 – Apresentação da tela de Aulas: 1° Resposta certa; 2° Resposta errada

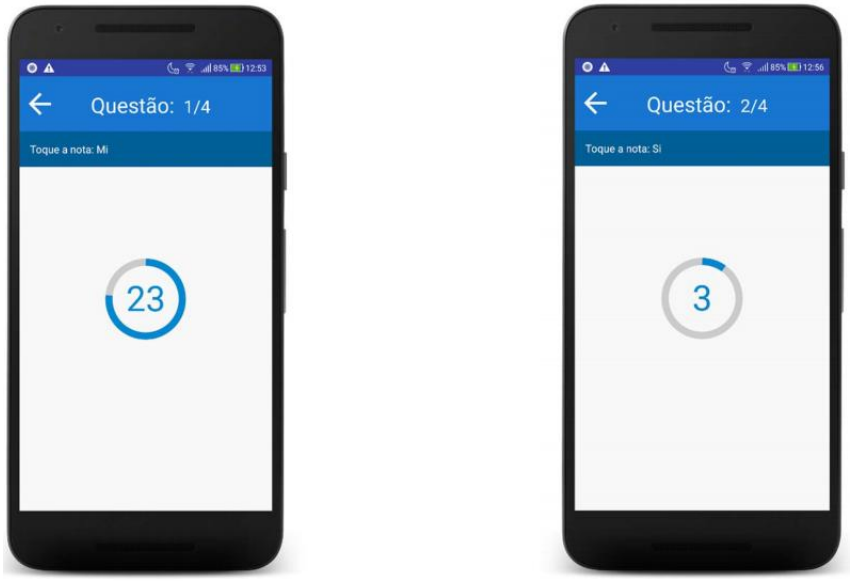


Fonte: Adaptado de Santana e Rovaron (2020).

2.4.4 Violearn

O Violearn é um software de guia manual, pois contém conteúdos teóricos, explicações como tocar violão, audiovisual, conceito musical, entre outros. A arquitetura é baseada por *Application Programming Interface* (API) que é requisitada através da comunicação com banco de dados, que retorna a resposta e, após, segue a regra de negócio da aplicação desenvolvida que envia os dados para o aplicativo, exibindo o resultado para o usuário. Foi modelado através de diagramas de Casos de Uso, Classes e de Entidade-Relacionamento baseado nas especificações

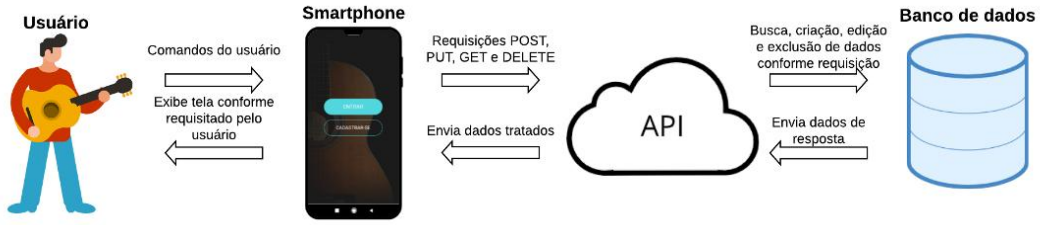
Figura 15 – Apresentação da tela de Metrônomo: 1° Nota a ser tocada; 2° Nota a ser tocada com tempo limite



Fonte: Adaptado de Santana e Rovaron (2020).

dos requisitos. Conforme a Figura 16, é uma arquitetura de acordo com a comunicação entre as requisições e respostas (DÁVILA; PORTO, 2021).

Figura 16 – Arquitetura de solução



Fonte: Adaptado de Dávila e Porto (2021).

Na Figura 17, (a) na tela inicial o usuário irá entrar na conta ou fará cadastro. Na próxima tela (b), será feita a escolha de módulo para iniciar o estudo e só poderá seguir os módulos seguintes se completar os anteriores. Após ser selecionado o módulo (c), o estudante terá as aulas em ordem pré-definida e irá desbloqueando as próximas na medida em que as anteriores forem completadas. Dentro das aulas (d), terá vídeo-aula e, no botão “Concluir”, termina e recebe a pontuação de acordo com a exibição no canto superior direito (DÁVILA; PORTO, 2021).

Por ora, foi desenvolvido o Módulo 1, que aborda conteúdo sobre a posição do violão, elementos do violão, uso de palheta, introdução de formação de acordes e levadas rítmicas. Possui conceitos de sistema de notação como tablatura, para a compreensão de acordes (DÁVILA; PORTO, 2021).

Figura 17 – As principais telas do Violearn

(a) (b) (c) (d)

Fonte: Adaptado de Dávila e Porto (2021).

2.4.5 Ouvido Perfeito

O Ouvido Perfeito[2] é um aplicativo de forma mais divertida e que, ao mesmo tempo, disponibiliza treinamento de ouvido e conteúdo teórico. Possui estilo de *layout* de jogos, com conquistas, desempenho e desafios, de modo atrair o aluno, que aprende com diversão (SILVA, 2018).

O aplicativo destaca muitos conceitos teóricos e práticos, conforme a Figura 18.

Figura 18 – As principais telas do Ouvido Perfeito

Fonte: Adaptado de Silva (2018).

Conforme a Figura 19, no menu principal, existe a opção “Perfil”, no canto inferior direito, que mostra o quadro de líderes, a pontuação dos últimos 3 dias e os desafios conquistados (SILVA, 2018).

Destaca-se também que, a tela Tocando Ritmo apresenta os exercícios de forma ordenada. Na aba Personalização, o usuário poderá escolher as questões ou tipo de níveis de difi-

[2] <https://play.google.com/store/apps/details?id=com.evilduck.musiciankit>

Figura 19 – As telas de Perfil

Fonte: Adaptado de Silva (2018).

culdades. De acordo com a Figura 20, possui telas de introdução de ritmo contendo conteúdo teórico e figuras musicais (SILVA, 2018).

Figura 20 – As telas de Tocando Ritmo

Fonte: Adaptado de Silva (2018).

2.4.6 Como é o ensino de música

A habilidade musical é adquirida por meio de treino e desenvolvimento da capacidade instintiva, com as quais se transforma a música. As pessoas que desenvolvem a aptidão pelo meio de prática, performance, improvisação, composição, condução, arranjo e instrução musical apresentam os atributos de criatividade, inteligência, desenvolvimento social e compreensão. Inclui-se, esta habilidade pode ser adquirida através da aprendizagem informal de exercícios, reconhecimento de harmonias e melodias, diversas músicas, entre outros aspectos que contri-

buem para o aprendizado (SANTIN, 2021). Existem diversas formas de aperfeiçoamento musical. Dentre elas, o exemplo da Tabela 3, com o método de ensino-aprendizado de forma planejada.

Tabela 3 – Tabela de Organização de Aulas

Módulos	**Conteúdo**
Habilidades auriculares	Desenvolvimento precisão rítmica; Entonação; Como uma música soa; Improvisação;
Habilidades cognitivas	Leitura de música; Transposição; Escalas musicais; Harmonia musical; Estrutura musical; Memorização da música; Composição; Estudo de estilos musicais
Habilidades técnicas	Agilidade técnica; Articulação;

Fonte: Santin (2021).

Aos professores de música, o método favorece a amplificação de planejamento de aulas e novas atividades musicais, sem deixar de lado a importância do estudo teórico e da percepção musical. Os conceitos básicos de aspectos teóricos da música são de suma importância para o primeiro passo da aprendizagem e para, posteriormente, durante o desenvolvimento, elevar suas dificuldades, conforme o conhecimento adquirido (BORGES; RICHIT, 2021).

2.5 CONSIDERAÇÕES DO CAPÍTULO

No decorrer desta pesquisa, ao abordar os elementos presentes das aplicações, percebe-se que há muitas informações sobre teoria musical. Cada aplicação apresenta diferentes metodologias de ensino. Para adequar ao objetivo dos usuários, alguns aplicativos têm a finalidade de melhorar a percepção musical e outros têm o propósito de aperfeiçoar a abstração musical.

É possível que as ferramentas contribuam positivamente no aprendizado musical, facilitando o dia a dia para os músicos iniciantes treinarem sem necessidade de sair de casa. Além disso, a tecnologia proporciona a experiência de sua utilização como uma nova forma de estudo.

Considera-se que as aplicações podem direcionar os instrutores de música como uma possibilidade de ensinamento, gerando um atalho para tomada de decisões e suas escolhas. Para os alunos iniciantes, é importante buscar um bom desempenho de estudo e prática, e a motivação é fundamental para o aprendizado de música com a utilização dessa ferramenta.

Dentre os trabalhos apresentados, destacam-se alguns pontos positivos e negativos de cada um. Os tutoriais dos conceitos musicais (aulas) são apresentados nas aplicações Teoria.com, No Tom, Violearn e Ouvido Perfeito. No caso e-Chords! Guitar, é um aplicativo incompleto, porém destaca-se pesquisa de acorde (selecionar acorde). O questionário é apresentado na aplicação No Tom e o sistema de cadastro o Violearn. O aplicativo Ouvido Perfeito é considerado o mais completo dos estudados, porém não é 100% gratuito e apresenta pontuação e desafio, que não serão contemplados neste trabalho.

3 PROPOSTA DE SOLUÇÃO

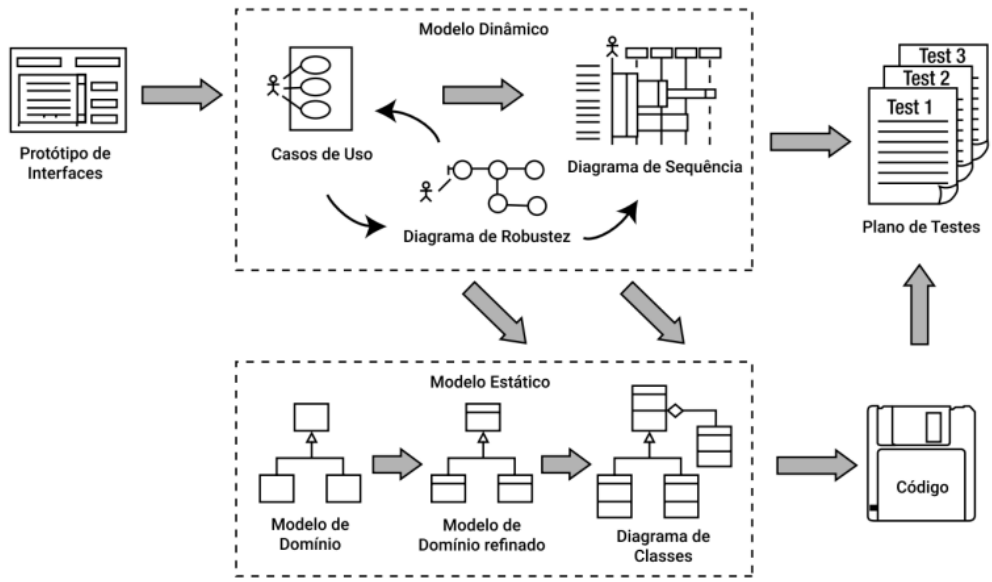
Neste capítulo, será apresentada a solução de um sistema web de ensino-aprendizado, abordando o processo de projeto de software utilizado, conforme descrito no objetivo deste trabalho. Serão descritos os requisitos funcionais e diagramas necessários para a implementação deste projeto. Na sequência, serão apresentadas as ferramentas utilizadas neste desenvolvimento.

3.1 ICONIX

A metodologia ICONIX é utilizada para o processo de desenvolvimento, a fim de apresentar casos de uso, protótipo de tela, diagramas de classe e diagrama de sequência. É classificada como intermediária em relação à complexidade do *Rational Unified Process* (RUP) e à simplicidade do *Extreme Programing* (XP). A sua finalidade é tornar o projeto mais simplificado e eficiente (SILVA, 2016).

Os artefatos da metodologia ICONIX são divididos em dois modelos: o modelo estático e o modelo dinâmico. O modelo estático é formado por diagrama de classes e modelo de domínio, sem interação com usuário. O modelo dinâmico é elaborado por diagrama de casos de uso e o diagrama de sequência, permitindo a interação do usuário com o sistema. De acordo com a Figura 21, é apresentado o panorama geral, seus principais artefatos e sua metodologia (BEGOSSI, 2021).

Figura 21 – Panorama geral da metodologia ICONIX



Fonte: Adaptado de Begossi (2021).

3.2 MODELAGEM DO SOFTWARE

Esta seção descreve os requisitos funcionais, os diagramas de caso de uso, de classes e sequência. Cada um deles será explicado detalhadamente, ou seja, como será a aplicação e como se comporta o sistema na interação com o usuário. Foram utilizadas ferramentas *Astah Uml*[1] para gerar os diagramas e *Balsamiq Wireframes*[2] para o protótipo.

3.2.1 Requisitos Não-Funcionais

Os requisitos não funcionais descrevem "o quê" o sistema deve fazer. Refere-se que o software tem que atender os atributos de qualidade como segurança, desempenho, capacidade de armazenamento, entre outros (SOUSA, 2021). Os seguintes requisitos não funcionais, deverão ser atendidos:

- o sistema deve gerar três notas para acorde tríade;
- o sistema deve gerar quatro notas para acorde tétrade;
- o sistema deve transpor a nota/acorde;
- o sistema deve ser acoplado, ou seja, integrar outras bibliotecas para o desenvolvimento;
- usuário do sistema web precisa ter acesso internet de boa qualidade.

3.2.2 Requisitos Funcionais

Descreve as necessidades que devem ser atendidas definidas pelo software por meio de funções, solucionando o problema para os usuários (VAZQUEZ; SIMÕES, 2016).

A Tabela 4 apresenta os requisitos do sistema com a devida descrição.

[1] <https://astah.net/>
[2] <https://balsamiq.com/>

Tabela 4 – Requisitos Funcionais

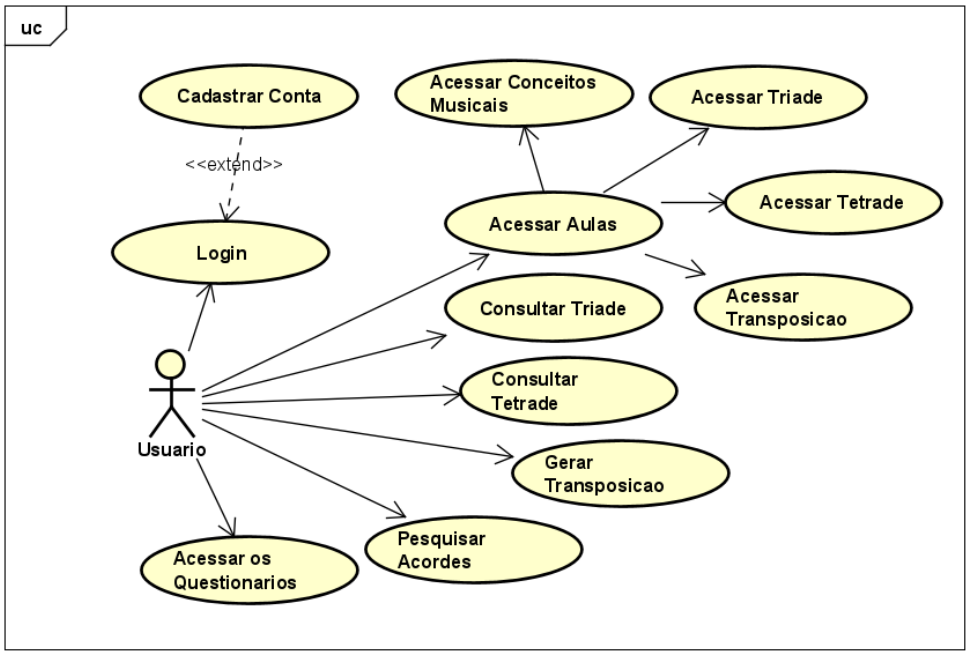
Requisitos	Descrição
REF01	O usuário poderá cadastrar conta no sistema.
REF02	O usuário poderá efetuar login no sistema.
REF03	O usuário poderá acessar as aulas.
REF04	O usuário poderá acessar Conceitos Musicais.
REF05	O usuário poderá acessar Tríade.
REF06	O usuário poderá acessar Tétrade.
REF07	O usuário poderá acessar Transposição.
REF08	O usuário poderá consultar as notas de tríade de um acorde.
REF09	O usuário poderá consultar as notas de tétrade de um acorde.
REF10	O usuário informa notas/acordes para transpor a quantidade de tons desejado e o sistema retorna os transpostos.
REF11	O usuário poderá pesquisar acorde desejado e o sistema retorna diagrama de acordo com o pesquisado.
REF12	O usuário poderá acessar Questionários sobre conceitos musicais.

Fonte: O autor (2022).

3.2.3 Diagrama de Caso de Uso

É um processo interativo que descreve como um usuário interage com o sistema para atingir o objetivo específico. O Caso de Uso define o ponto de vista do usuário ou como um software que ajuda a estabelecer a funcionalidade e as características do sistema (PRESSMAN; MAXIM, 2021). A Figura 22 exibe o diagrama de casos de uso que representa o funcionamento desse protótipo.

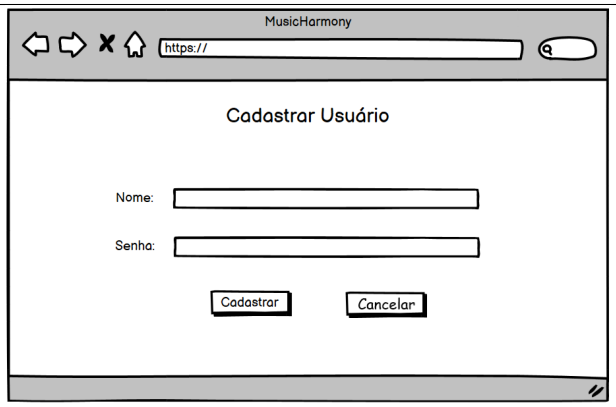
Figura 22 – Diagrama de casos de uso do sistema



Fonte: O autor (2022).

A Tabela 5 detalha sobre cadastro da conta, onde o usuário deverá ser cadastrado para logar e poder acessar o sistema.

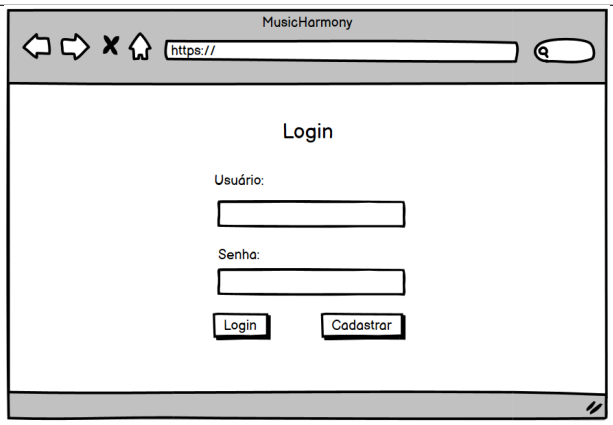
Tabela 5 – Cadastrar conta

Caso de Uso:	Cadastrar Conta
Requisito:	REF01
Descrição:	Para cadastrar a conta no sistema, deve informar o nome e senha.
Fluxo:	1. O usuário acessa o sistema e clica o botão “Cadastrar” para realizar o cadastro. 2. O usuário informa os dados necessários. 3. Acesso liberado para o usuário.
Casos de testes:	( ) Cadastro de usuário com dados validados ( ) Cadastro de usuário com dados inválidos
Protótipo:	

Fonte: O autor (2022).

Após cadastrar a conta no sistema, a Tabela 6 detalha o login onde o usuário poderá acessar o software.

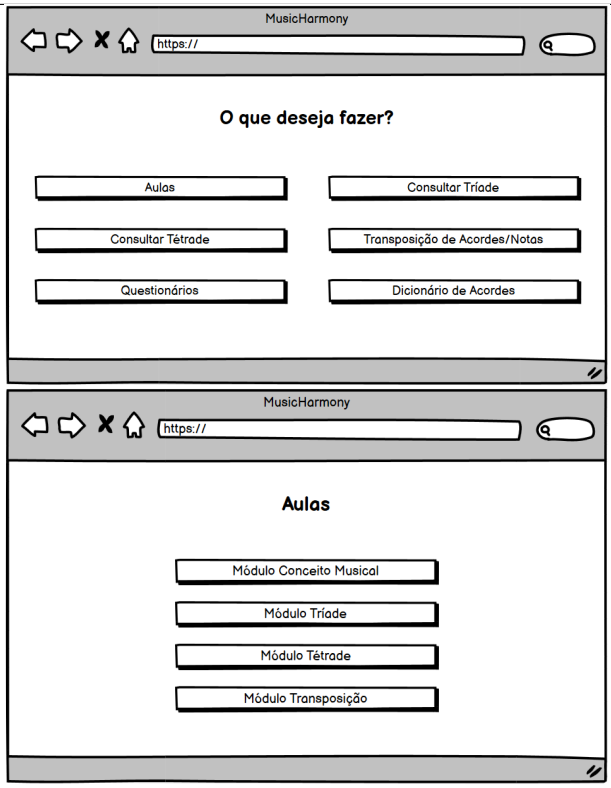
Tabela 6 – Login

Caso de Uso:	Login
Requisito:	REF02
Descrição:	Para logar no sistema, deve informar o e-mail e a senha
Fluxo:	1. O usuário acessa o sistema. 2. O usuário informa o usuário e a senha. 3. Acesso liberado para o usuário.
Casos de testes:	( ) Login com usuário e senha válidos ( ) Login com senha inválida ( ) Login com usuário inválido
Protótipo:	

Fonte: O autor (2022).

A Tabela 7 detalha o acesso às aulas, onde o usuário poderá escolher um dos módulos para acessar o conteúdo.

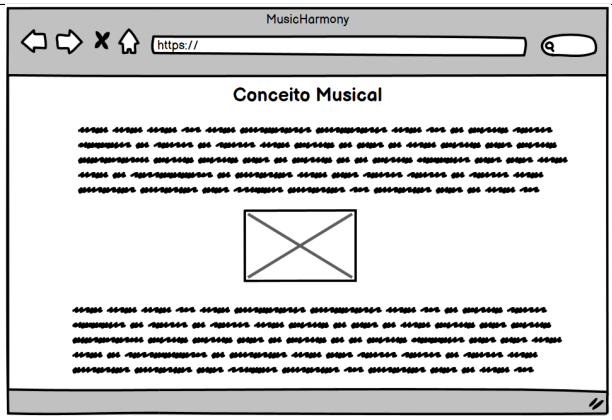
Tabela 7 – Acessar Aulas

Caso de Uso:	Acessar Aulas.
Requisito:	REF03.
Descrição:	Na tela principal, para acessar as aulas basta apenas clicar o botão “Aulas”.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Aulas”. 3. O usuário poderá escolher os módulos disponíveis.
Casos de testes:	( ) Acessar clicando botão “Aulas”. ( ) Acessar um dos módulos desejados.
Protótipo:	

Fonte: O autor (2022).

A Tabela 8 mostra como ter acesso ao conteúdo do módulo Conceito Musical, que trata dos conceitos e fundamentos básicos sobre notas musicais, as escalas, notas de acidentes, etc.

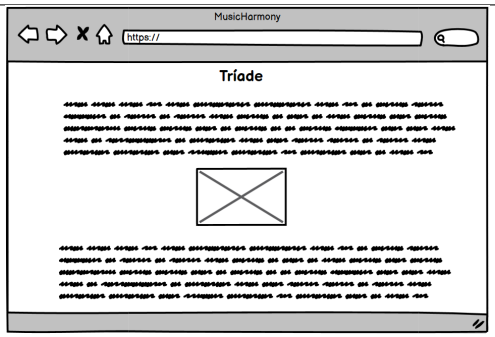
Tabela 8 – Acessar Conceito Musical

Caso de Uso:	Acessar Conceito Musical.
Requisito:	REF04.
Descrição:	O usuário poderá clicar o botão “Módulo Conceito Musical”. Dentro do módulo, terá explicação sobre conceitos musicais com figuras ou exemplos.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Aulas”. 3. O usuário poderá escolher o botão “Módulo Conceito Musical”.
Casos de testes:	( ) Acessar clicando botão “Aulas”. ( ) Acessar clicando o botão “Módulo Conceito Musical”.
Protótipo:	

Fonte: O autor (2022).

A Tabela 9 mostra como ter acesso ao conteúdo do Módulo Tríade, que aborda a formação de acorde de três notas.

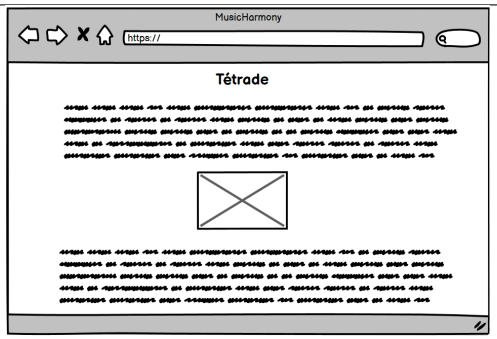
Tabela 9 – Acessar Tríade

Caso de Uso:	Acessar Tríade.
Requisito:	REF05.
Descrição:	O usuário poderá clicar o botão “Módulo Tríade”. Dentro do módulo terá explicação sobre tríade com figuras ou exemplos.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Aulas”. 3. O usuário poderá escolher o botão “Módulo Tríade”.
Casos de testes:	( ) Acessar clicando botão “Aulas”. ( ) Acessar clicando o botão “Módulo Tríade”.
Protótipo:	

Fonte: O autor (2022).

A Tabela 10 mostra como acessar o conteúdo do módulo Tétrade, que aborda a formação de acorde de quatro notas.

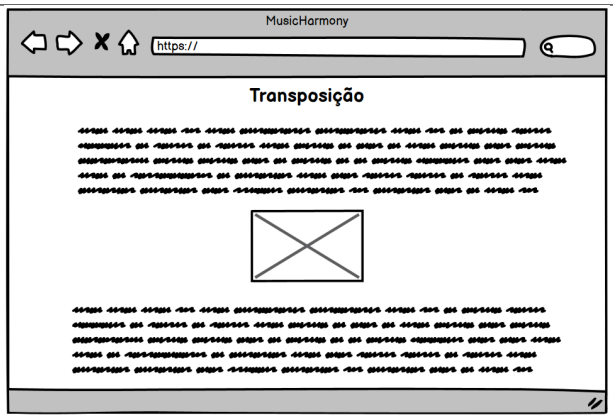
Tabela 10 – Acessar Tétrade

Caso de Uso:	Acessar Tétrade.
Requisito:	REF06.
Descrição:	O usuário poderá clicar o botão “Módulo Tétrade”. Dentro do módulo, terá explicação sobre tétrade com figuras ou exemplos.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Aulas”. 3. O usuário poderá escolher o botão “Módulo Tétrade”.
Casos de testes:	( ) Acessar clicando botão “Aulas”. ( ) Acessar clicando o botão “Módulo Tétrade”.
Protótipo:	

Fonte: O autor (2022).

A Tabela 11 mostra como ter acesso ao conteúdo do Módulo Transposição, que trata de como transpor uma nota ou um acorde.

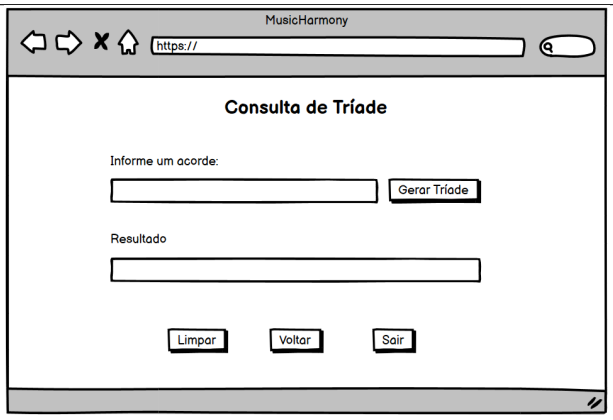
Tabela 11 – Acessar Transposição

Caso de Uso:	Acessar Transposição.
Requisito:	REF07.
Descrição:	O usuário poderá clicar o botão “Módulo Transposição”. Dentro do módulo, terá explicação sobre transposição com figuras ou exemplos.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Aulas”. 3. O usuário poderá escolher o botão “Módulo Transposição”.
Casos de testes:	( ) Acessar clicando botão “Aulas”. ( ) Acessar clicando o botão “Módulo Transposição”.
Protótipo:	

Fonte: O autor (2022).

Depois das aulas, a Tabela 12 detalha sobre a consulta de tríade.

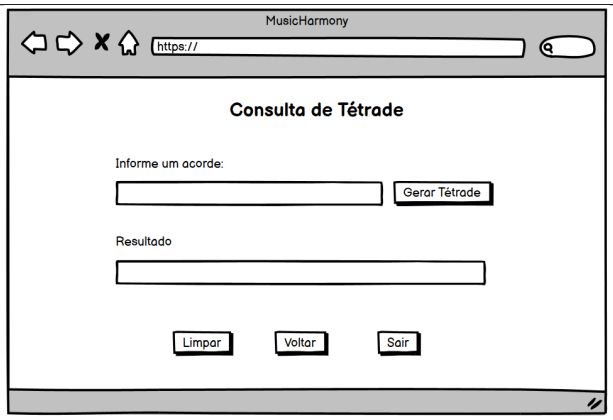
Tabela 12 – Consultar Tríade

Caso de Uso:	Consultar Tríade.
Requisito:	REF08.
Descrição:	O usuário poderá clicar o botão “Consultar Tríade”. Poderá ser informado o acorde e o sistema faz cálculo retornando as três notas que compõem o acorde informado.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Consultar Tríade”. 3. O usuário poderá informar o acorde. 4. O sistema retorna as 3 notas.
Casos de testes:	( ) Acessar clicando botão “Consultar Tríade”. ( ) Informar um acorde. ( ) Informar um acorde inválido. ( ) Clicar botão “Gerar Tríade”. ( ) Se o acorde for inválido, o sistema retornará a mensagem “Acorde inválido”. ( ) Se o acorde for válido, o sistema retornará resultado. ( ) Limpar os campos.
Protótipo:	

Fonte: O autor (2022).

A Tabela 13 detalha sobre a consulta de tétrade.

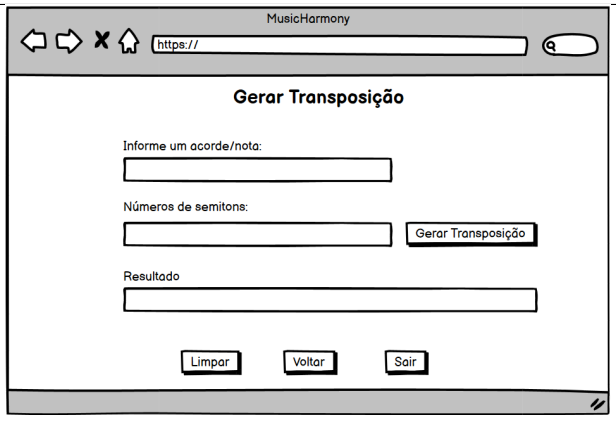
Tabela 13 – Consultar Tétrade

Caso de Uso:	Consultar Tétrade.
Requisito:	REF09.
Descrição:	O usuário poderá clicar o botão “Consultar Tétrade”. Poderá ser informado o acorde e o sistema faz cálculo retornando as quatro notas que compõem o acorde informado.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Consultar Tétrade”. 3. O usuário poderá informar o acorde. 4. O sistema retorna as quatro notas.
Casos de testes:	( ) Acessar clicando botão “Consultar Tétrade”. ( ) Informar um acorde. ( ) Informar um acorde inválido. ( ) Clicar botão “Gerar Tétrade”. ( ) Se o acorde for inválido, o sistema retornará a mensagem “Acorde inválido”. ( ) Se o acorde for válido, o sistema retornará resultado. ( ) Limpar os campos.
Protótipo:	

Fonte: O autor (2022).

A Tabela 14 detalha sobre a transposição, onde o usuário poderá fazer a consulta retornando a nota/acorde transpostos.

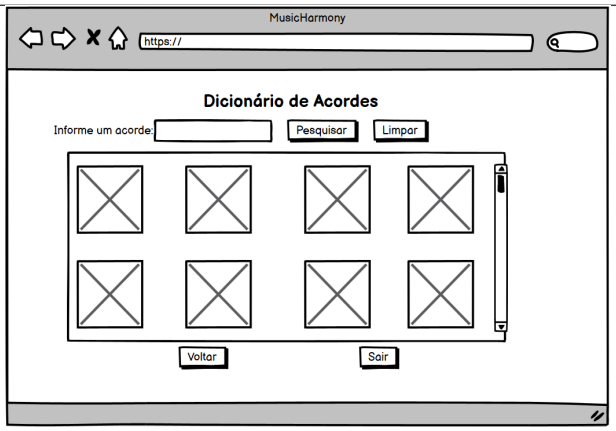
Tabela 14 – Gerar Transposição

Caso de Uso:	Gerar Transposição.
Requisito:	REF10.
Descrição:	O usuário poderá clicar o botão “Transposição de Acordes/-Notas”. Poderá ser informado o acorde/nota e números de semitons. O sistema faz cálculo de transposição retornando o transposto.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Transposição de Acordes/Notas”. 3. O usuário poderá informar o acorde/nota. 4. O usuário poderá informar os números de semitons. 4. O sistema retorna acorde/nota transposto.
Casos de testes:	( ) Acessar clicando botão “Transposição de Acordes/Notas”. ( ) Informar um(a) acorde/nota. ( ) Informar um(a) acorde/nota inválido. ( ) Informar números de semitons. ( ) Informar números de semitons inválidos. ( ) Clicar botão “Gerar Transposição”. ( ) Se o(a) acorde/nota for inválido(a), o sistema retornará a mensagem “Acorde/Nota inválido(a)”. ( ) Se o(a) acorde/nota for válido, o sistema retornará resultado. ( ) Limpar os campos.
Protótipo:	

Fonte: O autor (2022).

A Tabela 15 detalha sobre dicionário de acordes.

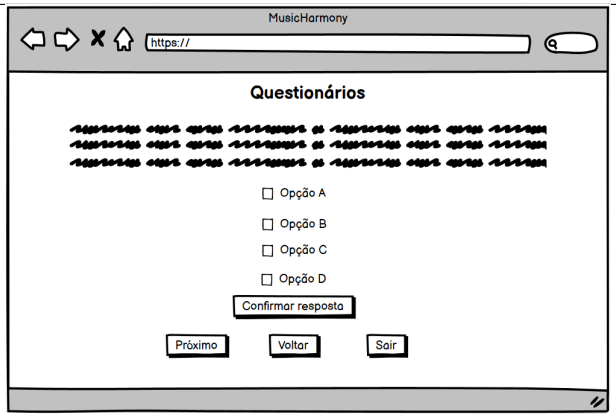
Tabela 15 – Pesquisar Acordes

Caso de Uso:	Pesquisar Acordes.
Requisito:	REF11.
Descrição:	O usuário poderá clicar o botão “Dicionário de Acordes”. Poderá ser pesquisando o acorde ou usando a rolagem para procurar.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Dicionário de Acordes”. 3. O usuário poderá informar um acorde. 4. O usuário poderá também usar a rolagem. 4. O sistema retorna acorde pesquisado.
Casos de testes:	( ) Acessar clicando botão “Dicionário de Acordes”. ( ) Informar um acorde. ( ) Usar a rolagem. ( ) Clicar botão “Pesquisar”. ( ) Se o acorde for inválido, o sistema retornará a mensagem “Acorde inválido ou inexistente”. ( ) Se o acorde for válido, o sistema retornará resultado. ( ) Limpar o campo.
Protótipo:	

Fonte: O autor (2022).

Por fim, a Tabela 16 detalha sobre questionário, onde o usuário poderá responder questões sobre conceitos musicais em geral.

Tabela 16 – Acessar Questionários

Caso de Uso:	Acessar Questionários.
Requisito:	REF12.
Descrição:	O usuário poderá clicar o botão “Questionários”. Poderá ter acesso a várias questões sobre conceitos musicais clicando o botão “Próximo”. Nas questões, terá o enunciado e as opções a serem assinaladas. Então, poderá clicar botão “Confirmar Resposta”. Se a resposta for incorreta, o sistema retorna a mensagem “Você Errou”, caso contrário, retornará a mensagem “Você Acertou”.
Fluxo:	1. O usuário acessa o sistema. 2. O usuário clica o botão “Questionários”. 3. O usuário poderá escolher as opções clicando checkbox. 4. O usuário clica o botão “Confirmar resposta”. 4. O sistema retorna a mensagem.
Casos de testes:	( ) Acessar clicando botão “Questionários”. ( ) Selecionar uma das opções. ( ) Clicar botão “Confirmar resposta”. ( ) Se a resposta for incorreta, o sistema retorna a mensagem “Você errou”. ( ) Se a resposta for correta, o sistema retorna a mensagem “Você acertou”. ( ) Clicar o botão “Próximo”.
Protótipo:	

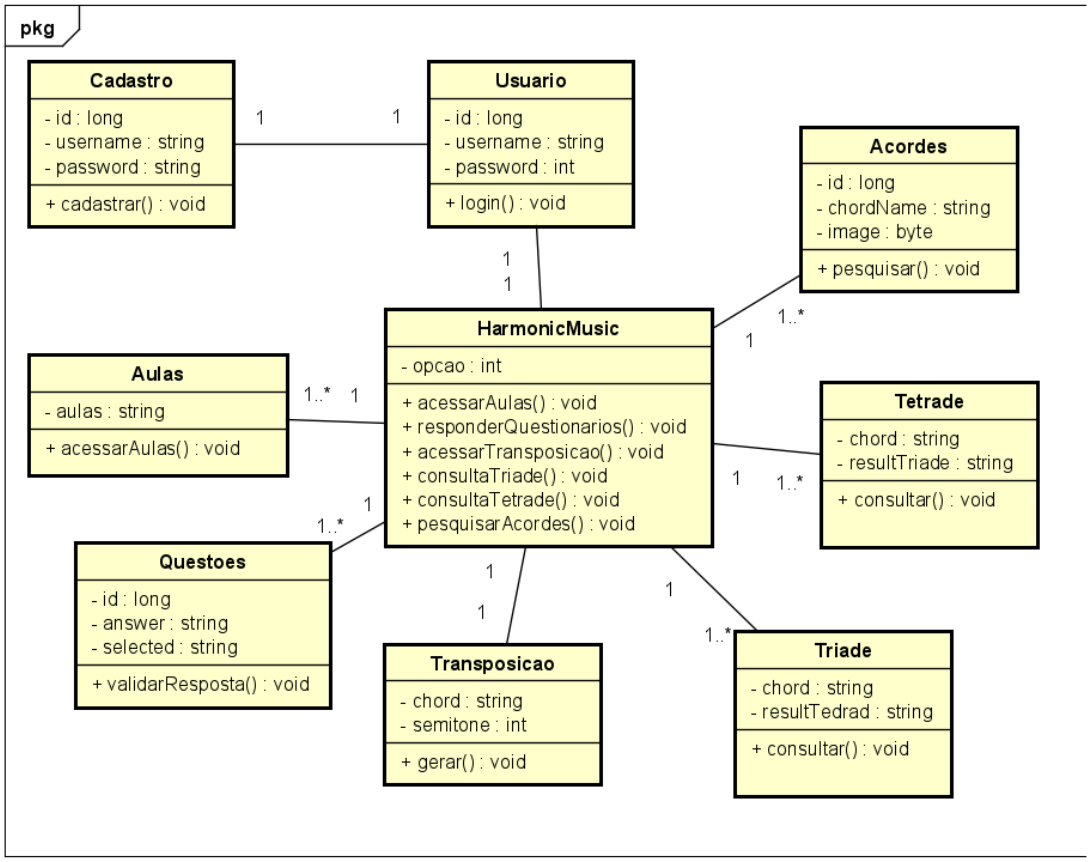
Fonte: O autor (2022).

3.2.4 Diagrama de Classes

O diagrama de classes é uma criação de modelo para visualizar a estrutura das tabelas de banco de dados e seus relacionamentos. Além do mais, fornece uma melhor observação do sistema que executa os processos através de suas entidades que expõem os seus atributos e operações (OLIVEIRA, 2020).

A seguir, a Figura 23 representa o modelo de diagrama de classe para o sistema.

Figura 23 – Diagrama de classes



Fonte: O autor (2022).

3.2.5 Diagrama de Sequência

O diagrama de sequência utiliza-se para designar, no decorrer da execução de uma tarefa, a comunicação entre os objetos, ou seja, a interação entre as entidades através de trocas de mensagens (PRESSMAN; MAXIM, 2021). As mensagens podem ser representadas por: Mensagem Síncrona, caracterizada pela chamada de métodos com retorno de resposta em que o envio é feito por uma uma seta de linha contínua e o retorno por uma seta com linha tracejada; Mensagem Assíncrona, representada por uma mensagem de linha única contínua (ARAÚJO, 2019).

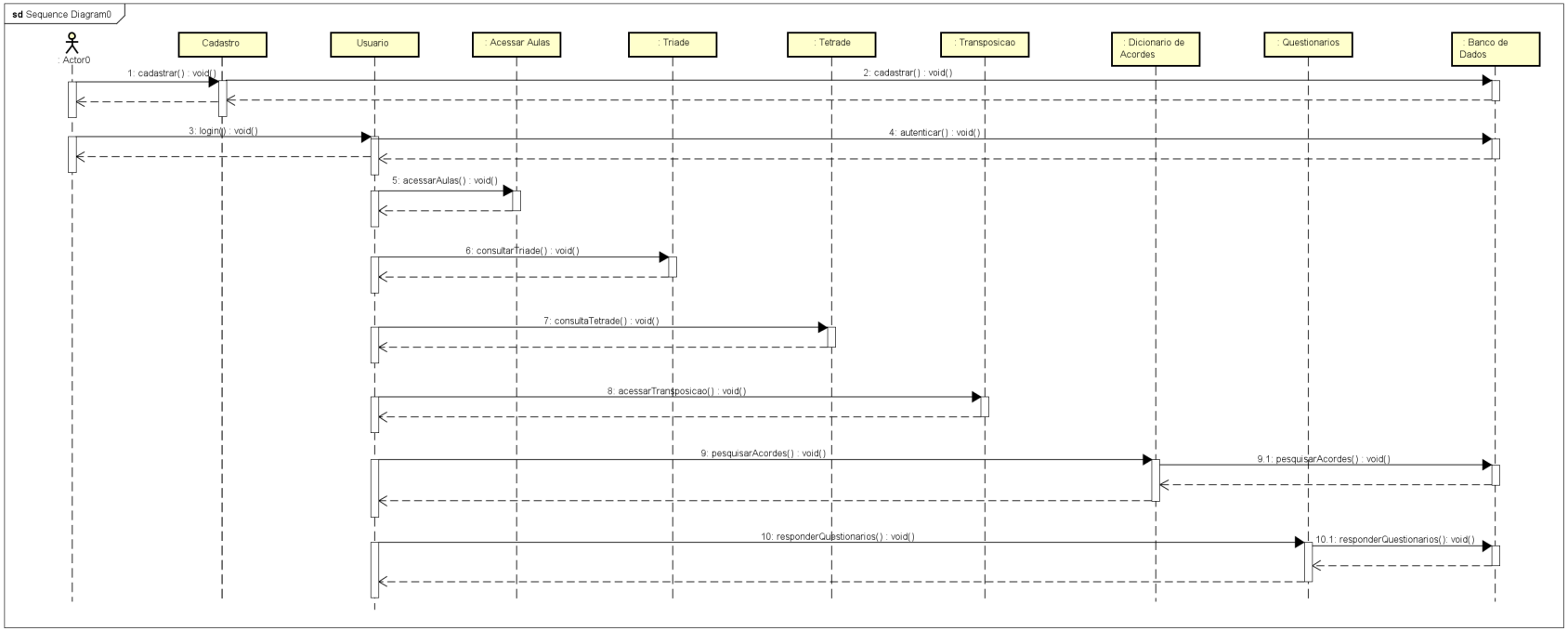
A Figura 24 apresenta o diagrama de sequência que representa o sistema em geral do protótipo.

3.2.6 Diagrama de Atividade

Um diagrama de atividade caracteriza um fluxo de interação em um cenário específico, baseado em caso de uso (PRESSMAN; MAXIM, 2021). O diagrama é semelhante a um fluxograma. É determinado entre uma atividade e outra que especifica o comportamento do sistema (ROSA, 2020).

A Figura 25 representa o diagrama de atividade de como seria o fluxo do sistema, desde

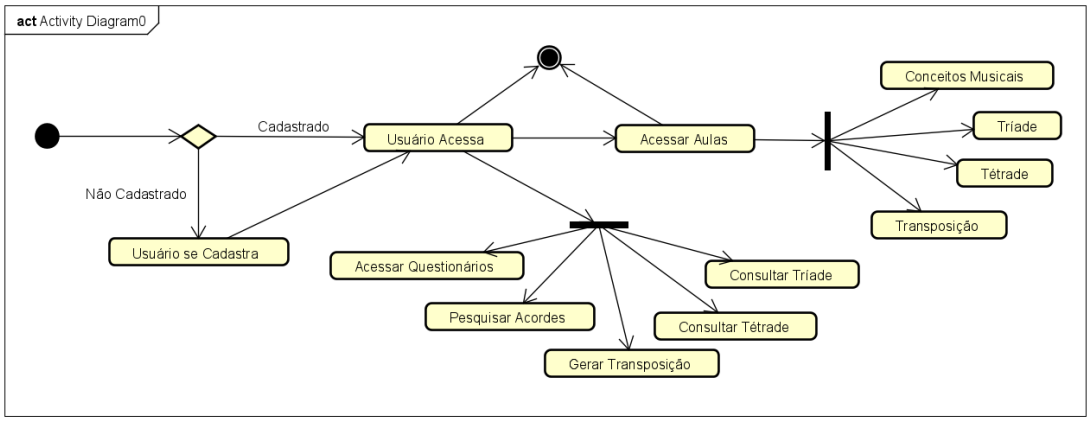
Figura 24 – Diagrama de sequencia



Fonte: O autor (2022).

o cadastro, o acesso até o fim.

Figura 25 – Diagrama de atividade

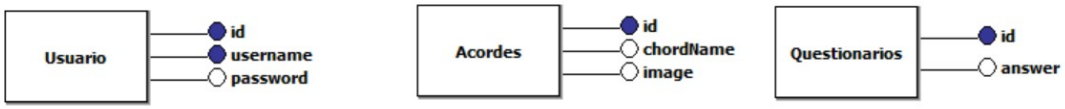


Fonte: O autor (2022).

3.2.7 Modelo de Banco de Dados

Foi utilizado para modelar o banco de dados o software brModelo[3], totalmente gratuito e de código aberto. Tem a função de abstrair diagramas de dados e os conceitos relacionados à modelagem conceitual e lógica (CANDIDO; MELLO, 2017). Conforme a Figura 26, este será o modelo de banco de dados do sistema. Como não existe a ligação entre as entidades e dada a simplicidade das informações que serão guardadas na base de dados, não terão relacionamentos entre as tabelas.

Figura 26 – Modelo de Banco de Dados



Fonte: O autor (2022).

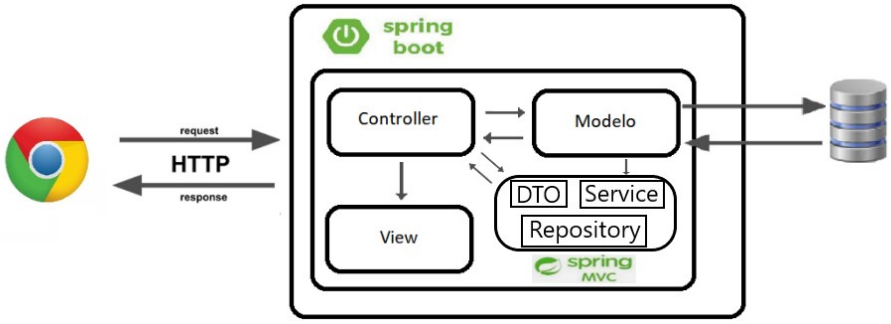
3.2.8 Arquitetura de Projeto

A arquitetura a ser utilizada para a troca de dados está apresentada na Figura 27. A linguagem a ser implementada é *Java*, utilizando como Framerwork o modelo *Spring Model View Controller (MVC)*[4], onde a camada *View* será implementada por meio de linguagem *Hyper Text Markup Language* (HTML) ou por meio de gerador de código dinâmico como *Thymeleaf*[5] e *Boostrap*[6]. A camada *Controller* é encarregada de tratar as requisições recebidas pelo sistema. Quanto à camada *Model*, define e gerencia as informações do domínio (FOGEL; BARROS, 2011).

[3] <http://www.sis4.com/brModelo/>
[4] <https://spring.io/>
[5] <https://www.thymeleaf.org/>
[6] <https://getbootstrap.com/>

Além disso, existem outros componentes: *Service, Repository* e *DTO*, que são responsáveis pela implementação de regras de negócio e persistência de dados (SOUSA *et al.*, 2017).

Figura 27 – Ilustração da arquitetura de projeto



Fonte: O autor (2022).

O banco de dados a ser implementado e utilizado especificamente para pesquisa de acordes será *PostgreSQL*[7]. De acordo com Carvalho (2017), é um ótimo gerenciador de banco de dados objeto-relacional de código aberto e gratuito.

Para a busca de dados com a aplicação, será implementada a API *JPA Hibernate*, onde *Hibernate*[8] é uma implementação da especificação *Java Persistence API* (JPA) da linguagem java, enquanto *JPA* tem o objetivo de simplificar o acesso ao banco de dados, que auxilia a padronização da interação com o banco de dados (PRADO, 2017; SOUZA, 2016a).

Spring Boot serve para embutir servidor web no ar sem depender de um servidor de aplicação com maior facilidade. Agiliza e simplifica o desenvolvimento com aplicações do Spring Framework. Além disso, também é utilizado para autenticação do sistema como *Spring Security* (BOAGLIO, 2017).

No conceito atual do *Spring Boot*, o controle é feito sobre suas regras de negócios empacotadas em um *Java Archive* (JAR) e no controle total providenciando o servidor web, conforme a Figura 28 (BOAGLIO, 2017).

[7] <https://www.postgresql.org/>
[8] <https://hibernate.org/>

Figura 28 – Spring Boot

Fonte: Adaptado de Boaglio (2017).

Sobretudo, o Spring Boot configura tudo deixando pronto para desenvolver. Utiliza-se com a base de *Maven*, onde possui o arquivo *pom.xml* para adicionar as dependências (GABRIEL; WILLIAM, 2020).

O *IntelliJ IDEA* foi criado pela empresa líder em desenvolvimento de software *JetBrains*[9] e é um ambiente de programação em linguagem Java, sempre visando à produtividade. Nos dias de hoje, é o *Integrated Development Environment* (IDE) mais utilizado por muitos desenvolvedores por possuir vários plugins e recursos, além de interface de boa utilidade (BRITZKE, 2017).

A etapa de controle de qualidade é vista como fundamental para o teste de software, uma vez que ela assegura que a aplicação web contemple as funcionalidades esperadas com funcionamento correto (PATRICK; OLIVEIRA, 2017). Para automatizar os testes, será utilizado *framework JUnit*[10], baseado em Java, com o objetivo de construir os testes unitários, além de executar os mesmos em modo console. O *JUnit* verifica se o sistema funciona de forma esperada, descomplica a produção e mostra os seus resultados (ALMEIDA *et al.*, 2019).

Para validação das funcionalidades deste projeto, as seguintes especificações serão avaliadas:

- **Consulta tríade:** testar se retorna as três notas corretamente;
- **Consulta tétrade:** testar se retorna as quatro notas corretamente;
- **Transposição:** testar se retorna nota/acorde corretamente;

[9] <https://www.jetbrains.com>
[10] <https://junit.org/junit5/>

- **Validação de acorde/nota da transposição:** testar se retorna nota/acorde inválido com valor booleano “false”.

3.3 CONSIDERAÇÕES DO CAPÍTULO

No decorrer do estudo da proposta da solução, foi encontrada uma forma de solucionar o problema da estrutura da implementação do projeto em web. Durante o desenvolvimento da proposta, constatou-se que foi possível simplificar o esboço e as tecnologias utilizadas para implementação. Não será necessário utilizar o banco de dados para o método algorítmico de tríades, tétrades e transposição, será apenas para cadastro e pesquisa de acordes.

Para o funcionamento bem sucedido do projeto, efetuou-se a modelagem de software abordando as descrições dos requisitos funcionais, os diagramas da ICONIX, os protótipos de telas e a solução da arquitetura de projeto para o desenvolvimento.

4 IMPLEMENTAÇÃO DA PROPOSTA DE SOLUÇÃO

Este capítulo, visa à explanação da implementação da proposta da solução, com alguns detalhamentos importantes, conforme o Capítulo 3, expondo a estrutura e as telas deste projeto via sistema web.

Serão apresentadas as bibliotecas utilizadas e *frameworks* dentro do projeto, além de algumas explicações da parte relevante do código com as suas principais as funcionalidades. Enfim, a conclusão a respeito da implementação.

4.1 ESTRUTURA DO PROJETO

A Figura 29 apresenta a estrutura da aplicação *web*, predominante na linguagem Java, com sua característica utilizada, o Spring MVC. Para facilitar a compreensão, a figura está enumerada, a fim de identificar os componentes para a elucidação.

A estrutura foi gerada automaticamente por meio de site oficial da *spring*[1], onde o desenvolvedor poderá nomear o artefato e incluir alguns *frameworks*. O diretório enumerado 1 é a raiz do projeto, que é onde estarão presentes todas as classes Java, assim como respectivos pacotes criados nele.

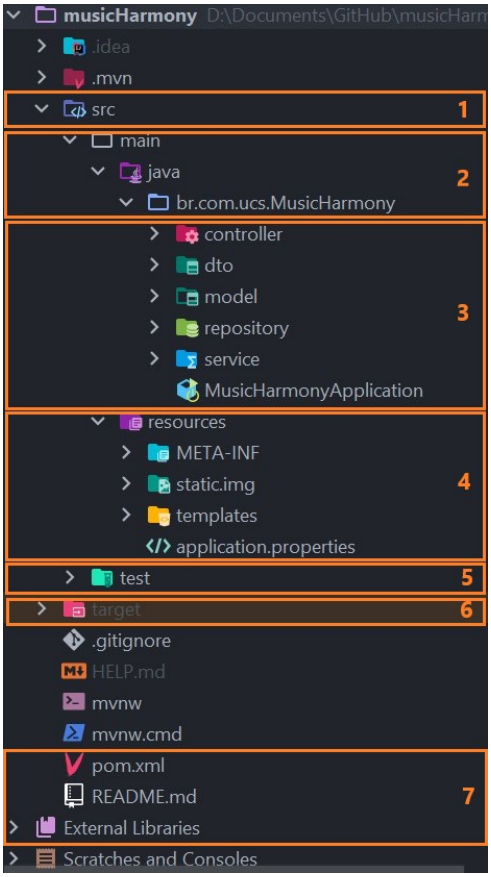
As pastas *main*, java e *br.com.ucs.MusicHarmony* são diretórios de caminho. Conforme enumerado 3, além de ter gerado automaticamente os componentes *controller* e *model*, foram criados, manualmente, *dto*, *repository* e *service*, de acordo com a necessidade do desenvolvimento de suas funcionalidades. Além disso, a classe principal *MusicHarmonyApplication*.

O enumerado 4, diretório *resources*, contém a implementação *front end* através do arquivo *HTML*. Possui componentes *static.img* (imagens), *templates* (view, em virtude do conceito Spring MVC) e a configuração da conexão do banco de dados. Para configurar a conexão do banco de dados, a pasta *META-INF* tem essa funcionalidade, porém não será utilizada. Será aplicado no arquivo *application.properties* pela razão do princípio da implementação *JPA Hibernate*.

O diretório *test*, enumerado 5, é utilizado para testar algumas funcionalidades da aplicação web, que será explicado no capítulo 5. Conforme enumerado 6, a pasta *target* contém o arquivo *class* compilado, gerado quando executada a aplicação. Por fim, enumerado 7, o arquivo *maven pom.xml* e suas dependências, quando executado, gera bibliotecas que estão dentro do diretório *External Libraries*. O arquivo *README.md* contém informações do projeto.

[1] <https://start.spring.io/>

Figura 29 – Spring Boot



Fonte: O autor (2022).

4.2 CONFIGURAÇÃO DO PROJETO

O projeto foi configurado através do arquivo *application.properties*, para a conexão com o banco de dados *PostgreSql*, de acordo com a Figura 30. Ele contém informações como *url*, o qual será conectado ao nome do usuário, senha do banco, *Hibernate* para criação das tabelas no banco de dados (LOPES, 2018).

Figura 30 – Configuração do application.properties do projeto

```
spring.datasource.driverClassName=org.postgresql.Driver
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.url=jdbc:postgresql://localhost:5432/musicHarmonyDB?serverTimezone=UTC
spring.datasource.username=postgres
spring.datasource.password=guimusic
```

Fonte: O autor (2022).

Outra configuração importante, é o arquivo *pom.xml*, onde as dependências são baixadas automaticamente pelo *maven*, conforme as Figuras 31, 32 e 33. Ele contém, por exemplo, as dependências do *Spring Boot, Thymeleaf, web*, entre outros.

Figura 31 – Configuração do maven

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.2</version>
        <relativePath/>
    </parent>
    <groupId>br.com.ucs</groupId>
    <artifactId>MusicHarmony</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>MusicHarmony</name>
    <description>MusicHarmony</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
```

Fonte: O autor (2022).

Figura 32 – Configuração do maven

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.5.0</version>
</dependency>
```

```xml
<dependency>
    <groupId>javax.persistence</groupId>
    <artifactId>javax.persistence-api</artifactId>
    <version>2.2</version>
</dependency>
<dependency>
    <groupId>org.hibernate.javax.persistence</groupId>
    <artifactId>hibernate-jpa-2.1-api</artifactId>
    <version>1.0.2.Final</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.9.0</version>
    <scope>test</scope>
</dependency>
```

Fonte: O autor (2022).

Figura 33 – Configuração do maven

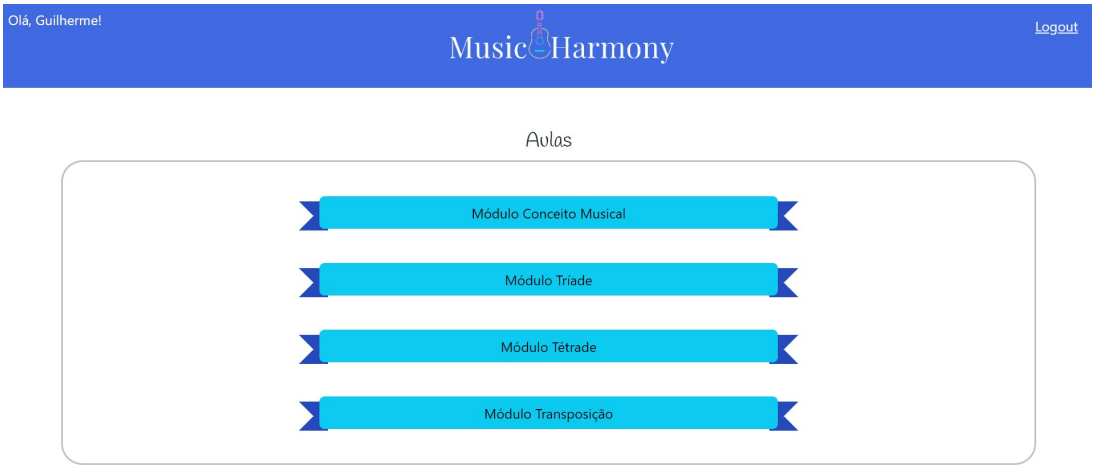
```
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Fonte: O autor (2022).

4.3 AULA

As aulas serão encontrada na tela “Aulas” conforme a Figura 34, que contém módulos de aulas que o usuário irá escolher para fim de estudo.

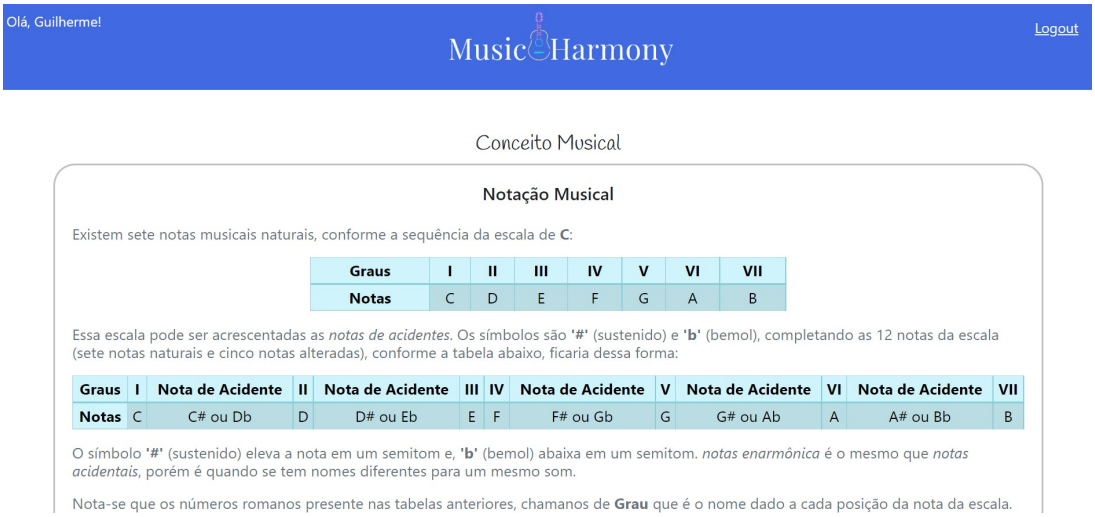
Figura 34 – Tela inicial Aulas



Fonte: O autor (2022).

Ao escolher um módulo, por exemplo, módulo Conceito Musical, de acordo com a Figura 35, irá exibir o texto explicativo sobre a teoria musical. Outros módulos terão o mesmo modelo.

Figura 35 – Tela do Módulo Conceito Musical

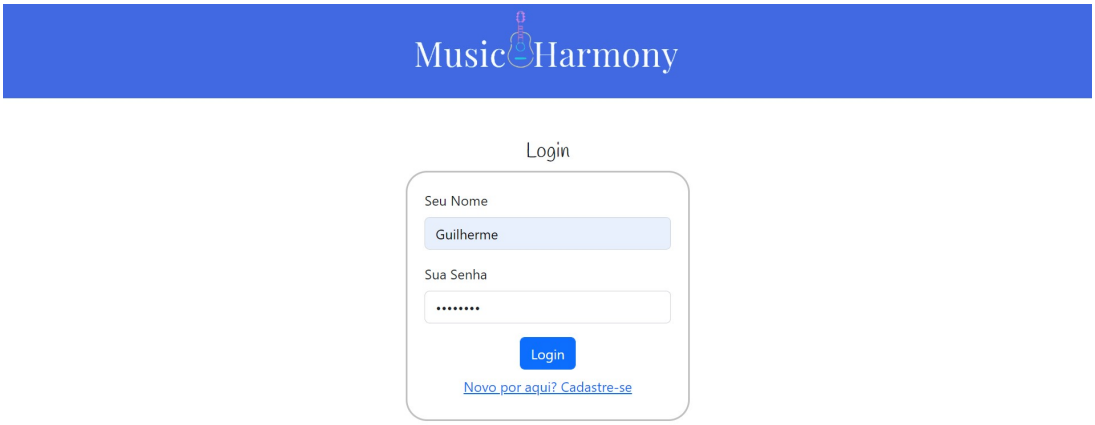


Fonte: O autor (2022).

4.4 FUNCIONALIDADES DAS TELAS LOGIN E CADASTRO

Conforme a Figura 36, esta é a tela de login onde o usuário poderá informar o nome e a senha para acesso ao sistema.

Figura 36 – Spring Boot



Fonte: O autor (2022).

Caso o usuário não tenha os dados cadastrados, poderá ir para tela de cadastro, de acordo com a Figura 37, informar todos os dados e clicar o botão "Confirmar". Após confirmar, voltará para tela de login automaticamente.

Figura 37 – Spring Boot

Fonte: O autor (2022).

A tabela de banco de dados do usuário é criado automaticamente pelo *Hibernate*, através de anotações do *JPA*, implementado na classe de modelo "*User*", que faz o mapeamento de uma classe à tabela do banco de dados relacional, conforme Algoritmo 1. As seguintes anotações utilizadas são explicadas uma a uma (LOLI, 2020):

- ***@Entity:*** objeto da classe “persistível” no banco de dados;
- ***@Table:*** especifica o nome da tabela;
- ***@Id:*** chave primária;
- ***@GeneratedValue:*** chave primária a ser populada pelo banco, no caso usamos *GenerationType.IDENTITY* como autoincremento;
- ***@Column:*** especifica o nome da coluna da tabela.

Algoritmo 1 – Classe de modelo User

```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id", nullable=false, unique=true)
    private Long id;
    @Column(name="username", nullable=false, unique=true)
    private String username;
    @Column(name="password", nullable=false)
    private String password;
    getter e setter
    .
    .
    .
    }
```

Fonte: O Autor (2020)

Para cadastro de usuário, conforme Algoritmo 2, o método *toSave* faz validação se existe o usuário no banco de dados, informando a mensagem de alerta que existe o cadastrado. Caso contrário, salva o usuário. O *request.getUsername()* faz a busca de valor do "nome" informado pelo usuário e compara com o *user.getUsername()* que está no banco.

As anotações do *Spring*, *@PostMapping* e *@GetMapping*, são tipo de chamada HTTP, que podem ser utilizadas ao acessar o *entrypoint* (ponto de entrada). No caso do Algoritmo 2, utiliza-se *@PostMapping*, o método "POST" para submeter uma entidade para funcionalidade específica (GARCIA; PAGANI, 2021).

Algoritmo 2 – Método toSave da Classe de controller FormController

```
    @PostMapping("/formulario")
    public String toSave(Model model, RequestRegistration request,
    BindingResult registrationError){
        User user = usuarioRepository.findByUsername(request.getUsername());
        if (user != null && (request.getUsername().equals(user.getUsername()
        ) || (request.getPassword().equals(user.getPassword())))){
            model.addAttribute("registrationError", registrationError);
        } else {
            User usuario = request.toUsuario();
            usuarioRepository.save(usuario);
            return "redirect:/login";
        }
        return null;
    }
```

Fonte: O Autor (2020)

O *@GetMapping* serve para solicitar a representação da funcionalidade, como por exemplo, o Algoritmo 3, que solicita o redirecionamento para página de "cadastro/formulario" (GAR-

CIA; PAGANI, 2021).

Algoritmo 3 – Método form da Classe de controller FormController

```
1
2       @GetMapping("formulario")
3       public String form(){
4           return "cadastro/formulario";
5       }
```

Fonte: O Autor (2020)

Para que isso funcione, é necessária a requisição do usuário da classe *RequestRegistration*. Além disso, salva os dados registrados, o método *toUsuario()*.

O atributo *usuarioRepository* com a anotação *@Autowired* que é uma injeção [2] baseada em campo sem necessidade de criar construtor, tem como propósito a busca de dados do usuário do banco de dados. Nessa funcionalidade, é definido o tipo da interface *UserRepository* que estende *JpaRepository*, habilitado a cumprir todas as operações do banco de dados como consulta (LOPES, 2018).

Feito cadastro, o login tem a lógica semelhante com o cadastro, porém com uma diferença: em vez de salvar, apenas valida os dados digitados comparando-os ao que está cadastrado no banco de dados. Além disso, busca a chave da sessão do usuário logado, o que será explicado na seção seguinte.

4.5 SESSÃO DO USUÁRIO LOGADO E LOGOUT

Após ser logado no sistema, a chave da sessão é gerada e armazenada no *cookies* do navegador. Conforme o Algoritmo 4, enquanto estiver logado, o usuário poderá navegar em qualquer página do sistema. Caso contrário, o sistema não permite acessar diretamente a uma página específica, resultando o redirecionamento para página de login (SILVEIRA; COSENTINO, 2009).

Algoritmo 4 – Método existsUsers da classe ExistsSessionService

```
1  @Service
2  public class ExistsSessionService {
3      @GetMapping
4      public Boolean existsUsers(HttpServletRequest request) {
5          HttpSession session = request.getSession();
6          return session.getAttribute("userIsLogged") == null ||
7                  session.getAttribute("userIsLogged").equals("");
8      }
9  }
```

[2] injeção: é uma técnica para evitar o alto nível de acoplamento de código, ou seja, diminui as dependências desnecessárias entre as classes.

Fonte: O Autor (2020)

O Spring MVC possibilita utilizar o objeto *HttpSession* como parâmetro e nele guarda a chave da sessão (SILVEIRA; COSENTINO, 2009), conforme o Algoritmo 5.

Algoritmo 5 – Chave da sessão salva através do objeto HttpSession

```
1  HttpSession session = requestSession.getSession();
2  session.setAttribute("userIsLogged", user);
```

Fonte: O Autor (2020)

A chave da sessão é capturada, conforme *log* console da Figura 38.

Figura 38 – Chave da sessão do usuário logado

```
Usuario Logado: F9D7CC611D07D741C22E2758405CC70B
```

Fonte: O autor (2022).

Enquanto *logout*, a chave da sessão será removida. Conforme o Algoritmo 6, é utilizado o método *invalidate()*.

Algoritmo 6 – Chave da sessão sendo excluída através do método invalidate

```
1  @Service
2  public class LogoutService {
3      public void invalidationSession(HttpServletRequest request) {
4          HttpSession session = request.getSession();
5          session.invalidate();
6      }
7  }
```

Fonte: O Autor (2020)

A remoção é executada quando o usuário faz logout, conforme a Figura 39.

Figura 39 – Chave da sessao removida

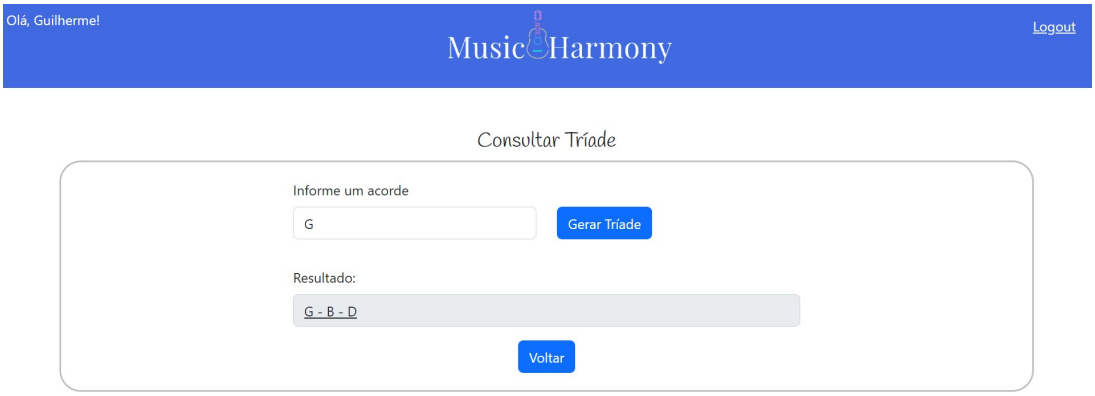
```
Excluindo a sessão usuario: F9D7CC611D07D741C22E2758405CC70B
```

Fonte: O autor (2022).

4.6 TRÍADE, TÉTRADE E TRANSPOSIÇÃO

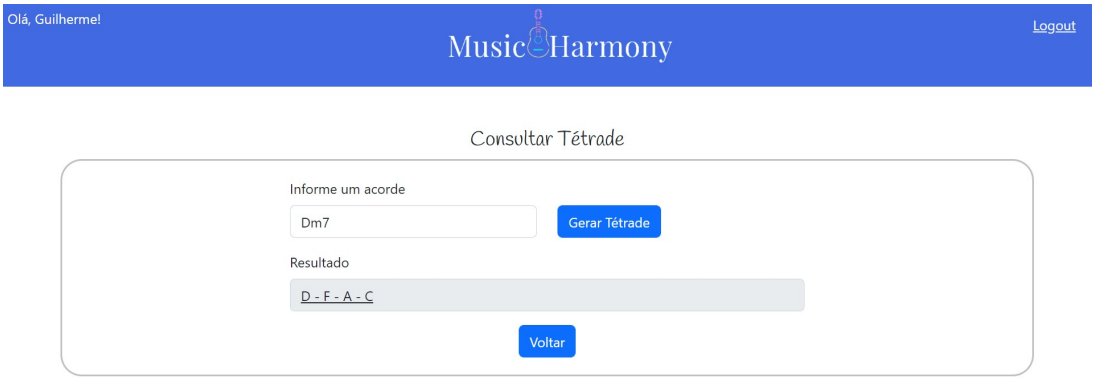
De acordo com as Figuras 40 e 41, são telas onde o usuário poderá consultar as notas que formam o acorde.

Figura 40 – Tela de Tríade



Fonte: O autor (2022).

Figura 41 – Tela de Tétrade



Fonte: O autor (2022).

Para a lógica servir para as duas classes (*TriadService* e *TetradService*) do componente *service*, foi criada a classe de modelo *Scale*, conforme exemplifica o Algoritmo 7. Contém sete escalas do tipo interface de lista *List<>* em *String* e, no construtor, é iniciado por padrão com suas listas de notas atribuídas conforme a escala. Como a lista não deve ser alterada, é utilizado a palavra-chave *final* (DEITEL; PAUL, 2016).

Algoritmo 7 – Classe de modelo Scale

```
1  public class Scales {
2      private final List<String> scaleDo;
3      // outras escalas
4      .
5      .
6      .
7      public Scales(){
8          this.scaleDo = new ArrayList<>(){{add("C");add("C#");add("D");
9          add("Eb");add("E");add("F");add("F#");add("Gb");add("G");add("G#");
10         add("A");add("Bb");add("B");}};
11        // outras escalas
12        .
13        .
14        .
15     }
```

Fonte: O Autor (2020)

Na classe *TriadService*, o atributo objeto *scales* do tipo *Scales*, é uma instância da classe *Scales()* com operador *new*. Para buscar os dados da lista, utiliza-se a estrutura de seleção *switch* para determinar quais são as notas que a compõem, de acordo com o atributo em *String chord* informado pelo usuário. Conforme o Algoritmo 8 da classe *TriadService* como um exemplo, existem séries de testes listado em um rótulo separado *case* (DEITEL; PAUL, 2016). A classe *TetradService*, possui lógica semelhante. A diferença é que é acrescentado mais uma nota.

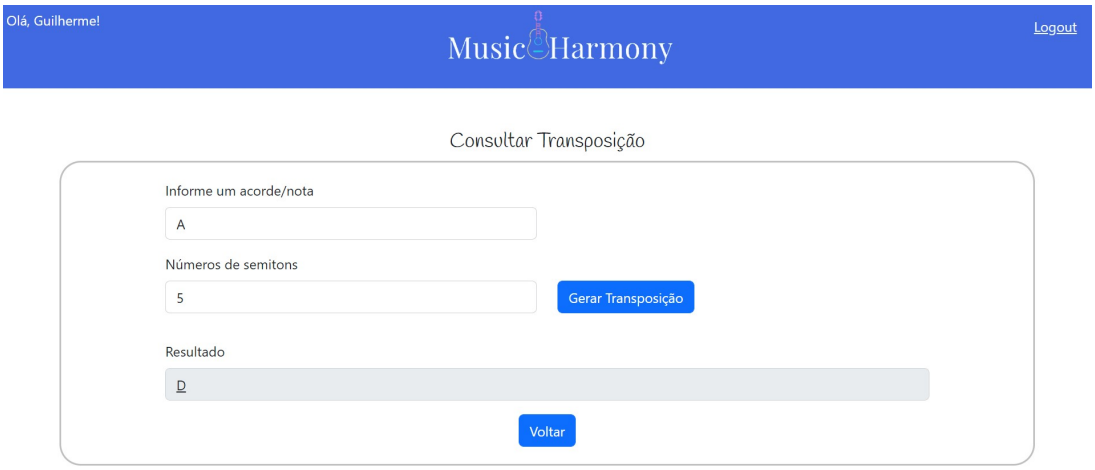
Algoritmo 8 – Classe TriadService

```
1  @Service
2  public class TriadService {
3      public String chordTriad(String chord){
4          String res;
5          Scales scales = new Scales();
6          switch (chord) {
7          //=====TIPO: TRIADE MAIOR======//
8              case "C" -> res = scales.getScaleDo().get(0) + "-" +
9              scales.getScaleDo().get(4) + "-" +
10             scales.getScaleDo().get(8);
11
12         // outras escalas e tipos de triades
13             .
14             .
15             .
16             default -> res = "Acorde invalido";
17         }
18         return res;
19     }
20 }
```

Fonte: O Autor (2020)

Quanto à transposição, conforme a tela da Figura 42, o usuário consulta a nota ou o acorde transposto.

Figura 42 – Tela de Transposição



Fonte: O autor (2022).

A validação de nota/acorde foi utilizado com auxílio do site regex101[3], para formar uma expressão válida de acordo com o padronização das notas musicais. Java possui o pacote regex nativo, java.util.regex, lançado desde a versão Java 1.4.0. É uma ferramenta poderosa e simples, que fornece funcionalidade inovadora com uma API organizada. O método a ser utilizado, é boolean matches(), designa se o regex do matcher coincide exatamente à região do String (FRIEDL, 2006). O Algoritmo 9 exibe a classe em que o atributo chord passa por seis validações.

Algoritmo 9 – Método validation da classe ValidationChordService

```
1  @Service
2  public class ValidationChordService {
3      public boolean validation(String chord){
4          if (chord.matches("[H-Z]") || chord.matches("[0-9]*") ||
5          chordNote.matches("[a-z]") || chord.matches("#")) {
6              return false;
7          }
8          if(chord.matches("((?![EB]#))([A-G]#?$)")){
9              return true;
10         }
11         return chord.length() < 2;
12     }
13 }
```

Fonte: O Autor (2020)

[3] <https://regex101.com/>

Após a validação, o cálculo de transposição é baseado em conceito de grupos cíclicos da teoria de conjuntos (RAHN, 1987), conforme o Algoritmo 10.

Algoritmo 10 – Método getTransposed da classe TranspositionService

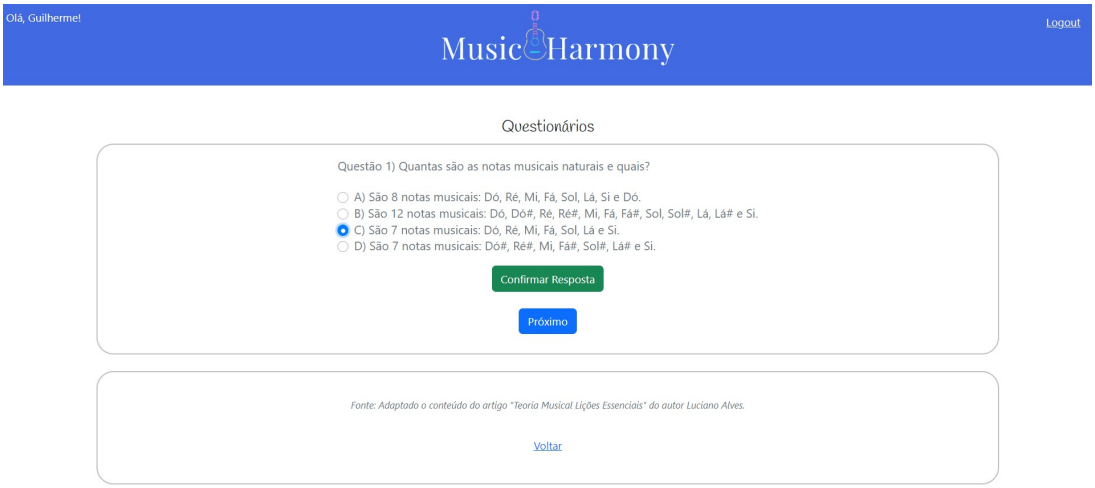
```
1   private int getTransposed(int semitone, LinkedList<String> scale,
2   int aux) {
3       int transposed;
4       int sum = aux + semitone;
5
6       if (sum >= 0) {
7           transposed = sum % scale.size();
8       } else {
9           transposed = abs(scale.size() + sum) % scale.size();
10      }
11      return transposed;
12  }
```

Fonte: O Autor (2020)

4.7 QUESTIONÁRIOS E DICIONÁRIO DE ACORDES

O sistema possui 12 questões referentes ao conteúdo da teoria musical. Cada questão terá quatro alternativas, como por exemplo a questão apresentada na Figura 43.

Figura 43 – Tela de Questionário

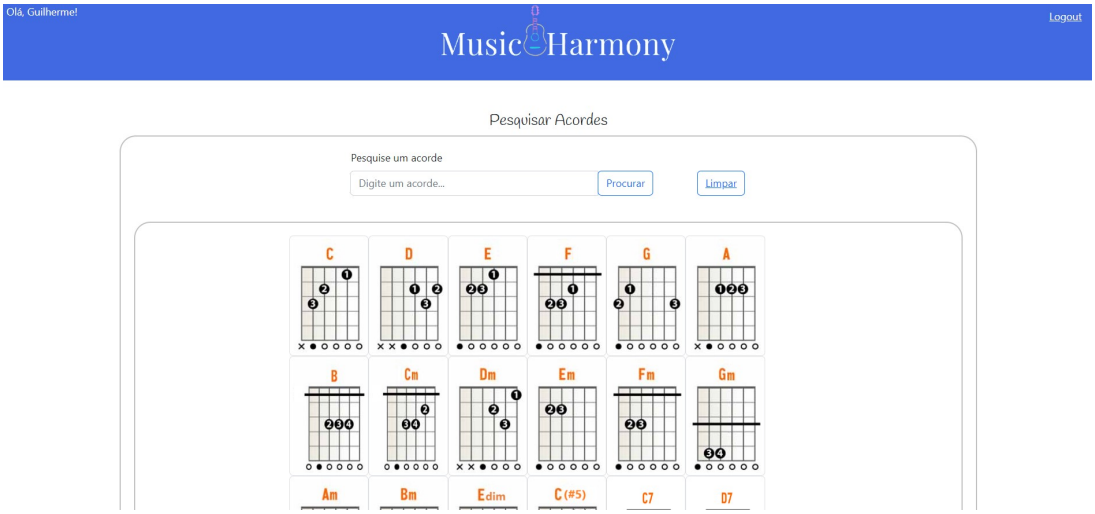


Fonte: O autor (2022).

O usuário poderá escolher uma alternativa. Caso a resposta esteja correta, receberá a mensagem “Você acertou!”. Do contrário, receberá a mensagem “Você errou!”.

De acordo com a Figura 44, ao acessar a página do dicionário de acordes, aparecerá todas imagens contidas do banco de dados.

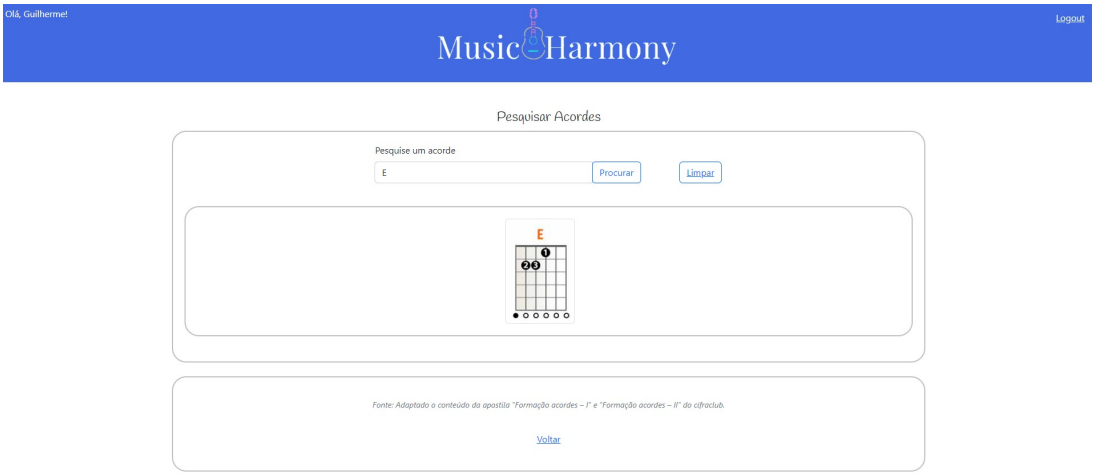
Figura 44 – Tela de Dicionário de Acordes



Fonte: O autor (2022).

Ao pesquisar um acorde, aparecerá somente uma imagem de diagrama de acorde, de acordo com a Figura 45.

Figura 45 – Tela de Dicionário de Acordes, pesquisando um acorde



Fonte: O autor (2022).

Caso o usuário digite um acorde incorreto ou inexistente, aparecerá a mensagem “Acorde não encontrado ou inválido” e retornará todas as imagens de diagrama de acorde. O botão “Limpar”, limpa o texto do campo digitado. Além disso, a imagem pesquisada também.

5 TESTE JUNIT DA APLICAÇÃO

Diante da aplicação web desenvolvida, este capítulo abordará os testes desenvolvidos conforme na seção 3.2.8 e o resultado de cada validação. O processo de verificação consiste em analisar se os códigos foram desenvolvidos corretamente de acordo com o requisito da documentação (OLIVEIRA, 2019). Dispõe a anotação *@Test*, que indica o método a ser testado. Será explicado como os resultados são obtidos através da implementação *JUnit*.

5.1 TESTE TRÍADE E TÉTRADE

No teste tríade, foi testada a classe *TriadService*, que concentra a lógica do conceito musical de acorde tríade. Nela, é criada a classe de teste *TriadServiceTest*, para fim de averiguar a funcionalidade. Foram testados cinco cenários diferentes, incluindo acorde incorreto e nulo.

Conforme o exemplo do primeiro cenário do Algoritmo 11, o método *ChordsBigger-Correct()*, avalia se os acordes maiores estão de acordo com as 3 notas esperadas.

O método a ser utilizado para averiguação é *assertEquals()*, que recebe o primeiro parâmetro como o valor esperado pelo desenvolvedor e, no segundo parâmetro, o resultado sendo testado. Caso esteja de acordo com o esperado, o teste passará.

Algoritmo 11 – Classe de teste TriadServiceTest

```
1  class TriadServiceTest {
2      private final TriadService triad = new TriadService();
3      @Test
4      public void ChordsBiggerCorrect() {
5          String chordC = "C";
6          // Outros acordes...
7          .
8          .
9          .
10         String resC = triad.chordTriad(chordC);
11         // Outros resultados...
12         .
13         .
14         .
15         assertEquals("C␣-␣E␣-␣G", resC);
16     }
17     .
18     .
19     .
20     // Outros cenarios...
21 }
```

Fonte: O Autor (2020)

Os cenários de testes abordados da tríade são:

- ***chordsBiggerCorrect():*** testa os acordes maiores;
- ***chordsMinorsCorrect():*** testa os acordes menores;
- ***chordsDiminutiveCorrect():*** testa os acordes diminutos;
- ***chordsAugmentedCorrect():*** testa os acordes aumentados.
- ***chordsInvalidate():*** testa os acordes inválidos.

No teste Tétrade é testada a classe *TetradService*. O teste é similar ao teste tríade, porém, a diferença está na nomenclatura de acorde e retorna 4 notas que formam o acorde de tétrade.

Os cenários de testes abordados da tétrade são:

- ***chordsMajorWithSeventhMajorCorrect():*** testa os acordes de sétima maior;
- ***chordsMajorWithSeventhMinorCorrect():*** testa os acordes de sétima menores;
- ***chordsMinorWithSeventhMinorCorrect():*** testa os acordes menores com sétima;
- ***chordsHalfDiminutiveWithSeventhMinorCorrect():*** testa os acordes meio diminuta com sétima.
- ***chordsInvalidate():*** testa os acordes inválidos.

Conforme o Algoritmo 12, o teste é semelhante (teste tríade e tétrade), basta apenas atribuir um acorde inválido ou nulo. Se retornar o resultado “Acorde inválido” e esteja de acordo com a expectativa, o teste é validado.

Algoritmo 12 – Método ChordsInvalidate, da classe TetradServiceTest

```
1  @Test
2  public void ChordsInvalidate() {
3      String chordInvalidate = "Cm7";
4      String chordInvalidate1 = "";
5
6      String resInvalidade = tetrad.chordTetrad(chordInvalidate);
7      String resInvalidade1 = tetrad.chordTetrad(chordInvalidate1);
8
9      assertEquals("Acorde␣invalido", resInvalidade);
10     assertEquals("Acorde␣invalido", resInvalidade1);
11 }
```

Fonte: O Autor (2020)

5.2 TESTE TRANSPOSIÇÃO

No teste de transposição, é testada a classe *TranspositionService*, onde é calculada a transposição. De acordo com o Algoritmo 13, deve informar o número de semitom e o acorde ou nota de origem. Após, retorna ao teste validado para ver se passou, de acordo com a expectativa.

Algoritmo 13 – Classe de teste TranspositionServiceTest

```
1  class TranspositionServiceTest {
2      private final TranspositionService transp = new TranspositionService();
3      @Test
4      public void transpositionSemitonePositivo() {
5          String chordTransp = transp.transposition(2, "C");
6          assertEquals("D", chordTransp);
7      }
8      .
9      .
10     .
11     // Outros cenarios de teste...
12 }
```

Fonte: O Autor (2020)

Os cenários de testes criados são:

- ***transpositionSemitonePositivo():*** testa número de semitom positivo dentro da mesma oitava;
- ***transpositionSemitoneNegative():*** testa número de semitom negativo dentro da mesma oitava;
- ***transpositionSemitoneOctaveUp():*** testa número de semitom positivo uma oitava acima;
- ***transpositionSemitoneOctaveBelow():*** testa número de semitom negativo uma oitava abaixo.

Quanto à invalidação do acorde/nota informada, é testada a classe *ValidationChordService*, que retorna o valor booleano. Para teste booleano, *Junit* dispõe o método *assertFalse()* que verifica se o valor do objeto é *"false"* e passará no teste. Conforme o Algoritmo 14, quando o acorde/nota é inválido, deve retorna *"false"*. Outro método utilizado é *assertTrue()*, que verifica se o valor do objeto é *true*, para validar a cifra precedido por "♯".

Algoritmo 14 – Método transpositionInvalidate, da classe TranspositionServiceTest

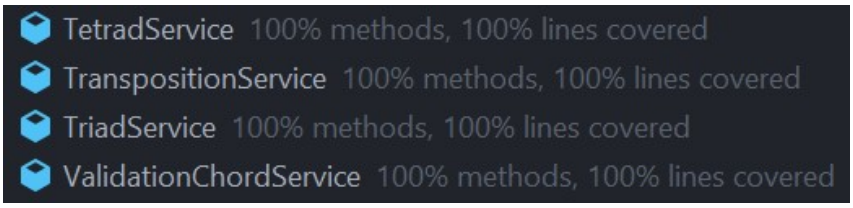
```
1  @Test
2  public void transpositionInvalidate() {
3      ValidationChordService valid = new ValidationChordService();
4      boolean val1 = valid.validation("C1");
5      assertFalse(val1);
6      boolean val11 = valid.validation("C#");
7      assertTrue(val11);
8      .
9      .
10     .
11     // Outras invalidacoes...
12     }
```

Fonte: O Autor (2020)

5.3 RESULTADOS OBTIDOS DO TESTE

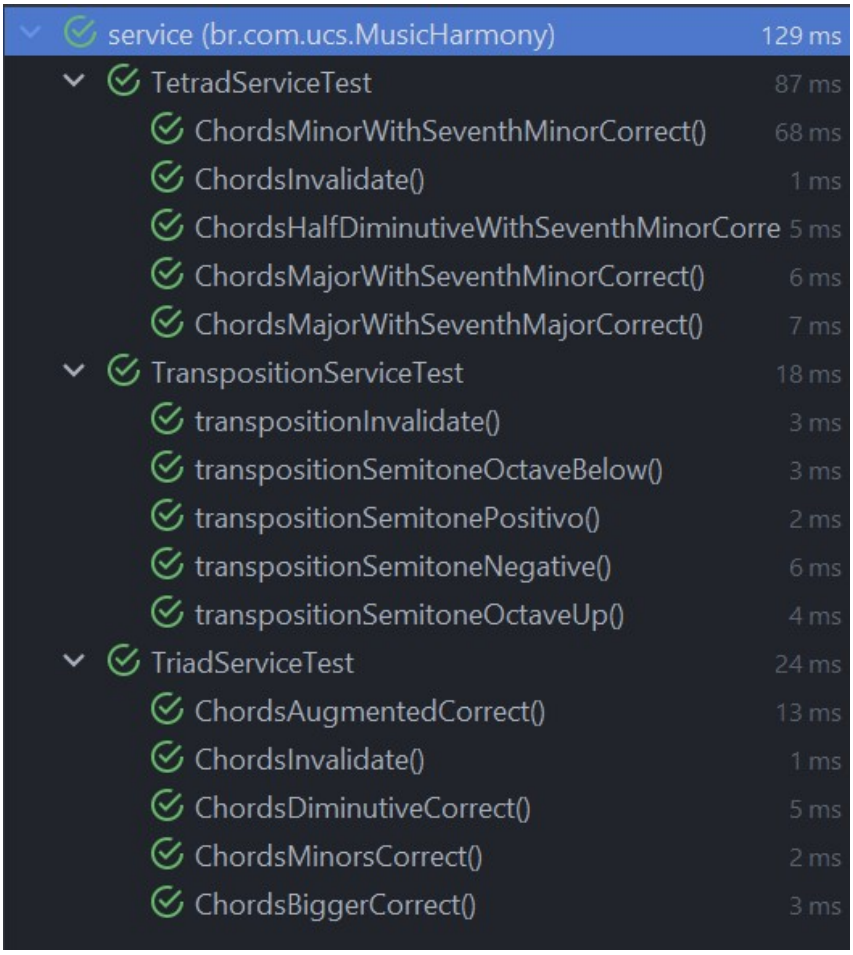
Analisando os resultados obtidos pelo teste, verificou-se que foram validados e abordados 100% dos métodos e 100% das linhas das classes abordadas, conforme as Figuras 46 e 47 , evidenciando os testes com resultados satisfatórios.

Figura 46 – Teste coverage



Fonte: O autor (2022).

Figura 47 – Testes executados e validados



Fonte: O autor (2022).

6 CONCLUSÕES

O objetivo principal deste trabalho foi apresentado, a proposta de desenvolvimento de uma aplicação web para auxiliar os músicos iniciantes a consultar além das aulas apresentadas, identificar as notas de acorde tríade, identificar as notas de acorde tétrade, identificar o acorde/nota transposta e pesquisa de diagrama de acorde. Pode-se notar que não é utilizado com "sons" e símbolos musicais, como a partitura ou tablatura, pois a característica desta aplicação é apenas visual em escrita.

Nos estudos de testes dos cenários específicos citados, foi crucial a verificação para garantir a qualidade da aplicação. A ferramenta serviu, como por exemplo, para realizar algumas modificações ou adequações conforme a necessidade do teste. Todavia, se o teste acusar o erro ou se o código não estiver funcionando corretamente, é necessário que o desenvolvedor saiba lidar com um erro. Portanto, conclui-se a importância de entregar o projeto com qualidade, testado e no prazo.

Para a solução da proposta desenvolvida através da modelagem de software, foi necessário o uso de alguns *frameworks* da linguagem Java, desenvolvimento de cálculo para transposição, a validação *Regular expression (regex)* para validar String, a lógica de cadastro e *login*, a utilização do *JPA Hibernate* para mapeamento do banco de dados relacional e o desenvolvimento de tríade e tétrade utilizando *List*.

Para a pesquisa de acordes, as imagens de diagrama de acorde foram adquiridas através de e-books de tutorial do *cifra club*[1], que é um site confiável e bastante utilizado tanto por músicos iniciantes quanto profissionais.

Logo, a aplicação atendeu de acordo com os requisitos propostos, possibilitando acrescentar novas funcionalidades e novos conceitos da teoria musical. No entanto, acredita-se que é possível acrescentar teoria musical intermediária e avançada, porém, pode ocorrer uma complexidade maior no desenvolvimento, necessitando de mais estudos.

A partir das análises, testes e outros, foram observadas as pendências para trabalhos futuros:

- **Melhoria "delay" da página:** ao clicar um determinado botão, por exemplo, o botão "Voltar", a página sofre um *delay*. Isso é causado por um código *front-end "href="/home""* ou do *back-end "redirect:..."*. No entanto, existe uma solução que é a linguagem *front-end javascript* que pode substituir o código problemático;
- ***Design Patterns*:** também conhecido como padrão de projeto, é uma solução para resolver problemas de códigos repetidos, além de organizar melhor para uma boa legibilidade;

[1] <https://www.cifraclub.com.br/>

- **Teste *login* e questionário:** não foi possível desenvolver devido à complexidade do método desenvolvido. Uma das soluções do *JUnit* para este caso é a utilização da biblioteca *Mockito*, que simula o comportamento de outra classe.

REFERÊNCIAS

ALMEIDA, I. M. B. d. *et al.* Testes de software junit, httpunit e jwebunit: Ferramentas de automatização de testes. **PESQUISA & EDUCAÇÃO A DISTÂNCIA**, n. 17, 2019.

ARAÚJO, D. J. F. d. **Verificação de Refinamento em Diagramas de Sequência com Estruturas de Controle**. Dissertação (B.S. thesis) — Brasil, 2019.

BEGOSSI, A. M. Protótipo de aplicativo para o auxílio de crianças com dislexia. 2021.

BOAGLIO, F. **Spring Boot: Acelere o desenvolvimento de microsserviços**. [S.l.]: Casa do Código, 2017.

BORGES, M. A. de S.; RICHIT, A. Saberes disciplinares para o ensino de música desenvolvidos por professores dos anos iniciais. **SCIAS-Arte/Educação**, v. 10, n. 2, p. 5–29, 2021.

BRITZKE, J. d. S. Controle financeiro wisecash. Universidade Tecnológica Federal do Paraná, 2017.

CANDIDO, C. H.; MELLO, R. dos S. Ferramenta de modelagem de banco de dados relacionais brmodelo v3. **XIII Escola Regional de Banco de Dados (ERBD)**, 2017.

CARDOSO, E. F. **Tétrades: relato de uma experiência de ensino da improvisação aplicada aos teclados percussivos**. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2015.

CARVALHO, V. **PostgreSQL: Banco de dados para aplicações web modernas**. [S.l.]: Editora Casa do Código, 2017.

CASTRO Jr., F. C. d. O uso de tablets nas aulas de música do ensino médio: um estudo com quatro professores de escolas da rede privada de brasília. 2019.

CONSTANTE, R. T.; ARAÚJO, J. T.; SCHIAVONI, F. L. Harmonia: uma ferramenta de aprendizagem musical. In: **XXIX Congresso da Anppom-Pelotas/RS**. [S.l.: s.n.], 2019. p. 80.

DÁVILA, G. M.; PORTO, J. B. Violearn: Um protótipo para ensino de teoria e prática musical por violão. In: SBC. **Anais do XXXII Simpósio Brasileiro de Informática na Educação**. [S.l.], 2021. p. 462–473.

DEITEL, H.; PAUL. **Java: como programar**. [S.l.]: Pearson Education do Brasil, 2017, 2016.

FOGEL, R.; BARROS, M. de O. Sistema de concessão de financiamento para a apresentação de artigos acadêmicos em spring mvc. 2011.

FRIEDL, J. E. **Mastering regular expressions**. [S.l.]: "O'Reilly Media, Inc.", 2006.

GABRIEL, P. d. S.; WILLIAM, G. S. A utilização do vue. js com uma api rest em spring boot. **Revista Interface Tecnológica**, v. 17, n. 2, p. 155–167, 2020.

GARCIA, G. B.; PAGANI, C. Do monolítico ao microserviço. 2021.

GONÇALVES, L. S.; FILHO, W. D. Fundamentos da harmonia. Ação Educacional Claretiana, 2015 – Batatais (SP), 2015.

JESUS, T. S. d. Chordix: um aplicativo para auto-acompanhamento musical através de cifras. Departamento de Ciência da Computação, 2017.

KOSTKA, S.; DOROTHY, P. **Harmonia Tonal: com uma Introdução a música do Século XX**. [S.l.]: New York: Mcgraw Hill, 2015.

KROGER, P. Music for geeks and nerds. **CreateSpace Independent Publishing Platform**, n. 1478345381, 2012.

LIMA, S. F. d. O. *et al.* Um sistema para transposição automática de seqüências midi baseada em alcance vocal. Universidade Federal de Uberlândia, 2006.

LOLI, S. B. **Code smells no contexto de mapeamento objeto-relacional em projetos Java: um catálogo e uma ferramenta de detecção**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2020.

LOPES, J. M. C. Sistema web para o auxílio na criação de frangos caipiras. 2018.

MASKE, R. C. **PROTÓTIPO DE UM SISTEMA PARA AUXILIAR NA COMPOSIÇÃO MUSICAL UTILIZANDO REGRAS DE HARMONIA**. Tese (Doutorado) — UNIVERSIDADE REGIONAL DE BLUMENAU, 2000.

MONZO, D. S. V. Improvisação musical e um improvisador: a música sem fronteiras de luiz eça. 2016.

MOTA, C. B. O uso de softwares na educação musical. **Revista Educação em foco-Edição**, n. 11, 2019.

MUSIC, E. A. D. So7 ware musica e sigestoes de aplic4c4o. 2007.

NETO, J. G. da S. A aprendizagem musical móvel: usando a tecnologia mobile na criação de aplicativos de apoio à aprendizagem e ensino do instrumento. **Revista Eletrônica de Música da UFAL**, n. 4, 2019.

OLIVEIRA, A. R. d. Projeto arquitetural de uma ferramenta para geração automatizada de casos de teste. Universidade Federal Rural do Semi-Árido, 2019.

OLIVEIRA, R. D. de. Análise do uso da cor no diagrama de classes da linguagem unificada de modelagem (uml)| analysis of the use of color in the unified modeling language class diagram (uml). **InfoDesign-Revista Brasileira de Design da Informação**, v. 17, n. 1, p. 116–130, 2020.

PATRICK, R. M.; OLIVEIRA, M. R. de. Velocidade do selenium webdriver & junit. **Revista Interface Tecnológica**, v. 14, n. 2, p. 40–51, 2017.

PRADO, T. F. d. Aplicação web para coletas de informações de vistorias imóveis empresariais e residenciais. Universidade Tecnológica Federal do Paraná, 2017.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software-9**. [S.l.]: McGraw Hill Brasil, 2021.

RAHN, J. **Basic atonal theory**. [S.l.]: New York, N.Y.: Schirmer Books; London: Collier Macmillan Publishers, 1987.

ROMANO, S. M. V.; SOBRINHO, E. M. S. B. Softwares online, com aprendizagem colaborativa, aplicados a educação musical. **Revista Processando o Saber**, v. 8, p. 43–63, 2016.

ROSA, H. Desenvolvimento de um aplicativo mobile para atividades voltadas ao pensamento computacional e computação desplugada. 2020.

SANTANA, J.; ROVARON, R. Desenvolvimento de um aplicativo para dispositivo móvel para estudo da teoria musical e abordagem prática para tocar violão development of a mobile application for studying music theory and practical approach to playing guitar. 2020.

SANTIN, G. Tecnologia digital na educação musical: a visão dos professores sobre a aplicabilidade de softwares como mediadores no processo de ensino de música. 2021.

SILVA, R. I. S. d. **Ensino da música através de aplicativos: um relato de experiência**. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2018.

SILVA, R. M. d. Sistema web para gerenciamentos de prontuários eletrônicos e gestão clínica. **Sistemas de Informação-Pedra Branca**, 2016.

SILVEIRA, P.; COSENTINO, R. Fj-21 java para desenvolvimento web. **Caelum Ensino e Soluções em Java**, 2009.

SOUSA, A. M. C. d. Ontologia para especificação de requisitos em sistemas embarcados. Universidade Federal de Pernambuco, 2021.

SOUSA, E. P. de *et al.* Arquitetura de aplicações spring mvc: Uma análise baseada no acoplamento lógico. 2017.

SOUZA, C. V. A. de; PEDRO, M. V. Compondo no musescore. **Interlúdio-Revista do Departamento de Educação Musical do Colégio Pedro II**, v. 3, n. 4, p. 16–19, 2015.

SOUZA, D. P. V. de. Desenvolvimento de uma aplicação financeira pessoal para web. Universidade Federal de Minas Gerais, 2016.

SOUZA, J. B. C. d. Estratégias para o aprendizado de obras com scordaturas não usuais: Um estudo com violinistas. 2016.

VAZQUEZ, C. E.; SIMÕES, G. S. **Engenharia de Requisitos: software orientado ao negócio**. [S.l.]: Brasport, 2016.