UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

IGOR ANDRÉ GOBETTI

SISTEMA BASEADO EM HEURÍSTICAS PARA GERAÇÃO PROCEDURAL DE MELODIAS

PATO BRANCO

2022

IGOR ANDRÉ GOBETTI

SISTEMA BASEADO EM HEURÍSTICAS PARA GERAÇÃO PROCEDURAL DE MELODIAS

System based on heuristics for procedural generation of melodies

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Curso de Tecnologia da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Érick Oliveira Rodrigues

Coorientador: Prof. Me. Vinicius Pegorini

PATO BRANCO

2022

IGOR ANDRÉ GOBETTI

SISTEMA BASEADO EM HEURÍSTICAS PARA GERAÇÃO PROCEDURAL DE MELODIAS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Curso de Tecnologia da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 07/dezembro/2022

Érick Oliveira Rodrigues
Doutorado
Universidade Tecnológica Federal do Paraná (UTFPR)

Jefferson Tales Oliva
Doutorado
Universidade Tecnológica Federal do Paraná (UTFPR)

Rúbia Eliza de Oliveira Schultz Ascari
Doutorado
Universidade Tecnológica Federal do Paraná (UTFPR)

PATO BRANCO
2022

RESUMO

Esta pesquisa tem como finalidade, abordar a geração procedural de melodias e a importância da música e seus benefícios e por meio disso desenvolver um sistema web que gere melodias por meio de características informadas pelo usuário como parâmetros. Tendo em vista que a melodia deve ser minimamente agradável, o sistema irá coletar dados sobre o quão satisfatório ficou a melodia gerada para o ouvido humano. Para compor este trabalho, inicialmente houve uma pesquisa teórica a respeito de todos os itens abordados. A revisão bibliográfica e documental foi realizada por meio de livros e artigos que serviram como base para a construção do referencial teórico e trabalhos relacionados. Este trabalho contém diferenciais perante os trabalhos relacionados, que é o *feedback* do usuário referente à melodia gerada e a opção de consultar as 10 melodias mais bem avaliadas.

Palavras-chave: música; melodia; geração procedural; geração procedural de melodias; sistema web.

ABSTRACT

This research paper aims to approach the procedural melody generation, the importance of music and its benefits; and through that, develop a web system that generates melodies by using characteristics informed by the user as parameters. In view that the melody ought be pleasant, the system will colect data about how satisfying became the generated melody for human ears. Initially were made a theoretical research around all the topics approched to compose this study. The bibliographical and documentary review was accomplished by books and articles that served as a basis for building a theoretical framework and related papers. This research paper/study includes advantages, mainly the user feedback referring to the generated melody and the option to confer the best rated melodies.

Keywords: music; melody; procedural generation; procedural melody generation; web system.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE SIGLAS

bpm	batimentos por minuto
CSIRAC	*Commonwealth Scientific and Industrial Research Automatic Computer*
Hz	*hertz*
IDE	*Integrated Development Environment*
MIDI	*Musical Instrument Digital Interface*
MP3	*MPEG-1/2 Audio Layer 3*
PCG	*Procedural Content Generation*
URL	*Uniform Resource Locator*
WAV	*WAVEform audio format*

SUMÁRIO

1 INTRODUÇÃO

O termo “música” tem diversos significados perante cada cultura. Segundo Brown (2006), a comunicação sonora fora do domínio da linguagem falada existe em todas as sociedades humanas conhecidas e estudadas. O campo da música computadorizada foi concebido durante a década de 1950, onde a arquitetura da máquina *Commonwealth Scientific and Industrial Research Automatic Computer* (CSIRAC) foi manipulada para tocar uma seleção de melodias populares (DOORNBUSCH, 2004). Conforme Johnson *et al.* (2015), desde esse avanço, a computação se sobressaiu, alterando a maneira de consumo e produção de trilhas sonoras. No entanto, ao que tudo indica, a futura evolução computacional proporcionará uma mudança positiva no meio da música, considerando o avanço da inteligência artificial e a constante evolução tecnológica.

Diante das técnicas utilizadas para a geração de conteúdo de forma computacional existe a geração procedural, a qual é uma abordagem que cria conteúdos de forma automática, podendo ser totalmente ou parcialmente randômica, considerando que regras podem ser aplicadas a partir de conjuntos de algoritmos previamente desenvolvidos. Essa prática vem sendo adotada para a criação de diversos tipos de materiais, tais como: áudio, vídeo, jogos, mapas de jogos, etc (MADEIRA, 2018). Tal prática contribui para que possa ser criado partes de um produto (*software*) sem que precise intervenção humana, seja para economizar tempo e/ou recursos. O recurso que será produzido a partir deste trabalho é a melodia, que deve ser cativante para os ouvidos humanos. Segundo Juslin e Lindström (2010) a música é considerada parte das emoções, sendo uma experiência sobre a qual é despendido muito tempo pelos elevados níveis de prazer que é despertado (PEREIRA *et al.*, 2011).

Esse trabalho envolve a construção de um *software* que gera melodias de forma procedural adotando diferentes regras de geração, disponibilizando mais opções de configuração que os equivalentes disponíveis no mercado, tendo a opção de informar o seu nome, contribuir com o *feedback* sobre o quão agradável ficou a melodia gerada e também consultar as 10 melodias mais bem avaliadas. Além disso, existe a opção para ser informado através de um campo de *feedback*, uma nota avaliativa, que será utilizada para destacar os melhores padrões de geração.

1.1 Objetivo

Desenvolver um sistema *web* à fim de obter melodias através da geração procedural.

1.1.1 Objetivos específicos

- Construir um sistema *web* para possibilitar a geração de melodias.

- Obter parâmetros informados pelo usuário para gerar uma ou mais melodias.
- Obter o *feedback* do usuário, para ter a ciência do quão agradável ficou a melodia.
- Disponibilizar o *download* da melodia no formato *Musical Instrument Digital Interface* (MIDI)

1.2 Justificativa

Dentre as motivações para desenvolver esse trabalho, tem-se a grande usabilidade das aplicações *web* (SILVEIRA *et al.*, 2015). O presente trabalho contém uma avaliação de aceitação da melodia pelo usuário ouvinte (*feedback*), ou seja, é possível se basear na avaliação do usuário para que futuras gerações de melodias se tornem mais agradáveis, como também a opção de consultar as 10 melodias mais bem avaliadas pelos usuários. Não foram encontrados trabalhos que utilizam da mesma metodologia na literatura científica.

De acordo com Scirea *et al.* (2015), a geração procedural musical é um campo muito explorado, isso engloba desde a criação simples de efeitos sonoros até uma amostra mais complexa que procura evitar repetir criações semelhantes, para permitir serem criadas harmonias e melodias com maior aceitação por parte dos ouvintes.

Procurando contemplar os usuários de uma forma satisfatória, a proposta apresenta um visual entregando uma usabilidade facilitada e que possa gerar melodias diferentes, ou seja, uma melodia pode ser mais agradável do que a outra de acordo com a avaliação subjetiva exigida no final da melodia gerada. Sendo assim, poderá também servir como o início de uma análise estatística de como gerar proceduralmente melodias cada vez mais atraentes. O sistema desenvolvido também contemplará a possibilidade de ser feita a análise estatística das melodias geradas a partir de diferentes heurísticas.

Nessa proposta, o resultado da melodia gerada pode ser muito satisfatório, ou seja, é possível ter uma melodia rica, que soe perfeitamente aos ouvidos humanos a ponto de a melodia gerada poder ser utilizada em jogos, ou aplicada como base para a criação de uma música ou até mesmo a disseminação através do compartilhamento do arquivo .MIDI pelo *Whatsapp*.

1.3 Estrutura do trabalho

Este texto está organizado em capítulos. O Capítulo 2 contém o referencial teórico sobre a importância da música e seus benefícios, elementos básicos da música, geração procedural de conteúdo e trabalhos relacionados. O Capítulo 4 apresenta os materiais e métodos utilizados no desenvolvimento do sistema proposto. Os resultados obtidos até o momento são apresentados no Capítulo 5. Por último estão as referências utilizadas no sistema proposto.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados argumentos e informações para o conhecimento básico do assunto, sendo eles: o impacto da música nas pessoas, os elementos da música, e os conceitos iniciais de música, tais como: intervalos musicais, escalas maiores e menores, tipos de escalas e, logo após, geração procedural.

2.1 A importância da música e seus benefícios

Há uma necessidade notória de o ser humano se comunicar desde o nascimento, onde a primeira relação de comunicação se dá entre a mãe e a criança no ventre, através de afeto estabelecendo assim uma comunicação.

A música é “a arte de exprimir ideias por meio de sons”, ela pode ser considerada como um instrumento de desenvolvimento no processo de ensino e aprendizagem (DOURADO, 2004). Ao saber ouvir e no fazer musical desenvolve a capacidade de cognição do educando, fortalece mudanças no seu potencial perceptivo, além do que o exercício da música e o canto em conjunto possibilitam conectar a parte do cérebro que funciona criativa e intuitivamente, proporcionando novas formas de sentir, pensar e de expressar (SEKEFF, 2007).

O estudo da música vem sendo cada dia mais analisado, especialmente na sua capacidade de influenciar no desenvolvimento intelectual do ser humano. Gardner (1994), em seu estudo sobre Inteligências Múltiplas, observa que existem sete possíveis habilidades, e que cada ser está conectado com elas de formas distintas, sendo elas: inteligência musical, corporal-cinestesia, matemática, linguística, espacial, interpessoal e intrapessoal. Quando se trata da inteligência musical, é possível caracterizar através da facilidade para identificar sons e ritmos e também o gosto pelo canto ou instrumentos musicais (GAMA, 1998). Ao ser desenvolvida, a música libera na pessoa muito de seu potencial de inovação, além de diminuir o estresse e a possível tensão da rotina.

Em referência ao estudo de inteligências múltiplas, é uma forma de enxergar a questão das sete habilidades como um todo, porém ainda existem pessoas que contestam essa abordagem na literatura.

2.2 Elementos básicos da Música

Para se entender sobre o que é a música, é necessário compreender todos os elementos que a constituem. Segundo Shneiderman, Preece e Pirolli (2011), para fazer surgir qualquer sentimento, ou descrever através da música qualquer caso ou situação, torna-se indispensável a presença de três elementos, sendo eles: melodia, ritmo e harmonia. O significado do termo

“música” vem sendo descrito como a arte de combinar os sons simultânea e sucessivamente, com ordem, equilíbrio e proporção dentro do tempo (MED, 1996).

De acordo com Andrade (2015), Pitágoras foi quem caracterizou a harmonia nas áreas teóricas e práticas, que este elemento musical surge na música através dos intervalos harmônicos, esclarecendo o entendimento sobre teoria musical. Em teoria, os intervalos harmônicos foram classificados por Pitágoras (século VI a.C.), o qual por meio do monocórdio (instrumento de uma só corda) determinou a relação proporcional entre os sons. Por meio da vibração concordante entre as cordas (frequências), ele obteve a série dos sons harmônicos (ANDRADE, 2015).

Segundo Priolli (2021), além da harmonia, existe um elemento não menos importante, o ritmo. O ritmo é o movimento dos sons regulados pela sua maior ou menor duração, o ritmo consegue ser estudado através das reações do ser humano diante dele.

Considerando um tempo que tenha aproximadamente um ritmo igual ao da pulsação cardíaca normal, que pode ser considerado como uma calmaria, é como se o corpo humano entrasse em sintonia, de forma a pensar consigo só: “ah, está bem, estamos ambos em uníssono". De fato, se você levar a mão ao coração enquanto estiver ouvindo uma música assim, verificará que o coração tende rapidamente a corrigir qualquer discrepância do seu tempo, até atingir perfeita afinação com a música (TAME, 1984). Por meio disso, é possível perceber que o autor propõe que uma música mais agitada, por exemplo, pode exercer uma sensação de mais euforia, destoando de uma música mais lenta, que traz a sensação de paz, fazendo com que a aceleração do coração fique mais lenta.

Ainda falando sobre um dos elementos da música, segundo Med (1996), a melodia é um conjunto de sons emitidos de maneira sucessiva. De acordo com Britannica (2021), quando se fala de melodia é possível dizer que é uma série de diferentes notas, ou sons. As notas podem ser cantadas ou tocadas sucessivamente, formando um conjunto resultando em uma melodia podendo ser em um tom agudo ou grave. A representação das notas musicais comumente são palavras aceitas em vários idiomas, as notas musicais na escala MAIOR se chamam: DÓ, RÉ, MI, MI, SOL, MI, SI, também podem ser representadas por letras : C(DÓ), D(RÉ), E(MI), F(MI), G(SOL), A(MI), B(SI).

Conforme Med (1996), as melodias são formadas por frases musicais e possuem regras, como a linguagem que é falada pelo ser humano. Um bom exemplo para entender o que é melodia é a própria voz, ou o canto, que é reproduzido na grande maioria das vezes somente uma nota por vez através do som que sai da voz, e o conjunto das diferentes notas cantadas e/ou faladas formam a melodia. Para se entender melhor as notas musicais, destaca-se na sequência, algumas informações sobre Intervalos Musicais.

2.2.1 Intervalos Musicais

De acordo com Sadie (1994), todos os conjuntos musicais audíveis que podemos utilizar para fazer música, compõe-se de aproximadamente 97 sons com frequências variando entre 27,5 *hertz* (Hz) a 7.040 Hz. O que chamamos de intervalos musicais é justamente a distância entre um som e outro. Matematicamente, o intervalo é medido pela razão entre as frequências. Ao invés de se trabalhar com números de frequências (27,5 Hz), usamos na música nomes próprios da linguagem musical, a exemplo disso pode-se considerar a escala de DÓ Maior: DÓ, RÉ, MI, MI, SOL, MI, SI, DÓ, que substituem os números de Hz das frequências utilizadas na Física/Acústica e na Matemática.

Conforme Karolyi (2002), a regra básica da classificação dos intervalos musicais se dá por meio de saltos, denominados como tom e semitom, sendo que um tom significa um salto inteiro de uma nota para outra e um semitom representa "meio tom", ou seja, dois semitons equivale a um tom.

Quanto aos intervalos, a denominação é a seguinte: do intervalo de DÓ (C) para RÉ (D) equivale à segunda maior, de DÓ (C) para MI (E) um intervalo de terça maior, de DÓ (C) para a nota SOL (G) o intervalo é uma quinta justa, de DÓ (C) para MI (A) é uma sexta maior (DÓ, RÉ, MI, MI, SOL, MI), e assim sucessivamente, considerando sempre o intervalo inicial sendo a própria nota musical, e que de DÓ para DÓ temos uma oitava justa, que é o dobro da frequência da própria nota (132,0 Hz e 264,0 Hz respectivamente). Os intervalos podem ser: Justos (quartas(4ªas), quintas(5ªas) e oitavas(8ªas)), maiores e menores (terceiras (3ªas), sextas (6ªas) e sétimas (7ªas)) aumentados e diminutos (uníssono, segundas(2ªas), terceiras(3ªas), quartas(4ªas), quintas(5ªas) e oitavas(8ªas)) (KAROLYI, 2002). A distância entre uma nota e outra pode ser classificada por nome (maior, menor, justa, aumentada e diminuta) ou pela quantidade de medida de distância (tom(ns)). Para entender a Tabela 1, precisa-se entender o conceito de notas sustenidas, que denomina-se pelo símbolo "#", como exemplo tem-se o salto de DÓ + 1 semitom pode-se denominar pelo símbolo C#, nesse caso o nome da nota é DÓ Sustenido, no caso do RÉ + 1 semitom usa-se o símbolo D# denominado por RÉ Sustenido, e assim sucessivamente. Esse conceito dos saltos de semitons é entendido por escalas cromáticas. Conforme a Tabela 1, é possível ver a representação desses conceitos.

Tabela 1 – Tabela de intervalos, notas e seus respectivos saltos (em Tons)

Classificação dos Invervalos	Nota	Distância (salto em quantidade de Tons)
Fundamental	C	0
2ª menor	C#	1/2
2ª maior	D	1
3ª menor	D#	1 e 1/2
3ª maior	E	2
4ª justa	F	2 e 1/2
4ª aumentada ou 5ª diminuta	F#	3
5ª justa	G	3 e 1/2
5ª aumentada ou 6ª menor	G#	4
6ª maior	A	4 e 1/2
7ª menor	A#	5
7ª maior	B	5 e 1/2
8ª justa	C	6

Fonte: Adaptado de Sadie (1994).

A Tabela 1 também pode ser lida da seguinte forma:

- C - Fundamental
- C até C# – 2ª menor
- C até o D – 2ª maior
- C até D# – 3ª menor
- C até E – 3ª maior
- C até F – 4ª justa
- C até F# – 4ª aumentada ou 5ª diminuta
- C até G – 5ª justa
- C até G# – 5ª aumentada ou 6ª menor
- C até A – 6ª maior
- C até A# – 7ª menor
- C até B – 7ª maior

Os intervalos são a chave mestra para se penetrar na linguagem musical no que se refere à sua melodia e à sua harmonia (GRILLO, 2013). Logo, os leigos que não se importam com esse conhecimento de intervalos musicais, desafinam com muita facilidade. Portanto desafinar

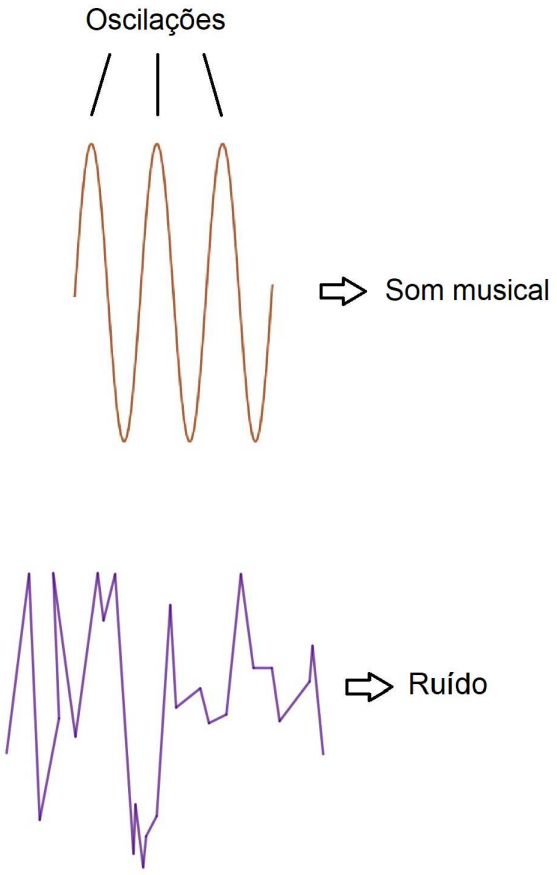
é, estar fora de intervalo da escala musical, como exemplo temos os corais, que de forma geral executam a música mais por repetição do que por estudo dos intervalos, esses, estão mais sujeitos a erros (desafinação).

Para tanto, deduz-se que para manter a afinação é necessário que a nota esteja dentro da escala musical, como trata a subseção abaixo.

2.2.2 Escalas Musicais

Conforme Ratton (2006), a partir do ponto de vista acústico, várias características físicas compõe os sons utilizados na produção de música, dentre elas: oscilações (frequências) e a presença de harmônicos. Quando se fala de oscilações bem estabelecidas, é considerado que um som musical quase sempre ocorre de forma sustentada, de forma que a característica de oscilação continua por alguns ciclos, ao contrário de ruídos (sons não musicais). A Figura 1 demonstra um exemplo de como pode ser representado um som musical e um ruído.

Figura 1 – Sons e ruídos



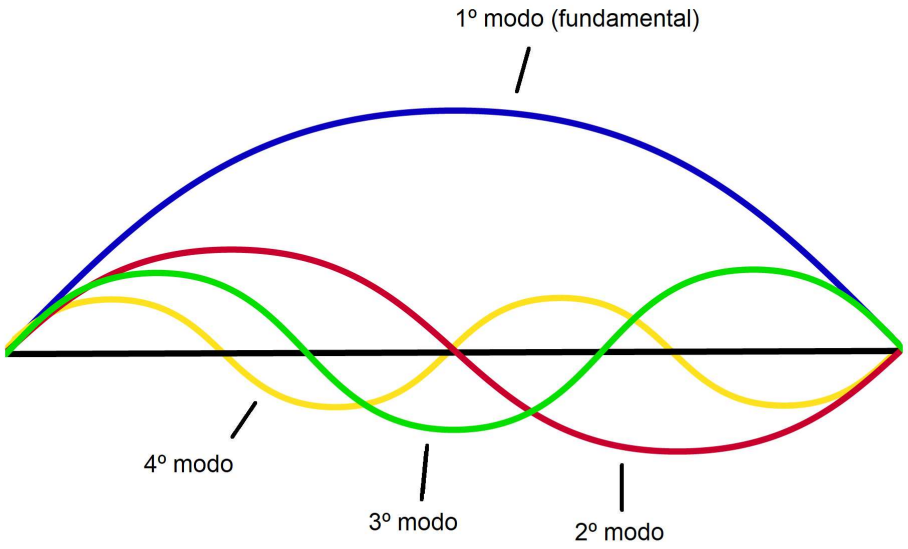
Fonte: Adaptado de Ratton (2006).

Quando se fala da presença de harmônicos, vale salientar que grande parte dos sons musicais não acontece somente da forma mais simples de vibração (modo fundamental), e sim

em conjunto com os modos harmônicos, que equivale ao corpo vibrante variando em ambas as frequências (múltiplas inteiras e frequência do modo fundamental) (RATTON, 2006).

A Figura 2 exibe modos de vibração de uma corda (deve-se atentar para o fato de que na realidade a vibração da corda ocorre conforme a soma – superposição – de todos os modos presentes). Os harmônicos presentes em um som são componentes extremamente importantes no processo musical, tanto na formação das escalas musicais, como na harmonia musical. Por causa dessas características naturais, sons com alturas (frequências) diferentes, quando postos a ocorrer ao mesmo tempo, podem criar sensações auditivas esteticamente diferentes (RATTON, 2006).

Figura 2 – Modos de vibração de uma corda



Fonte: Adaptado de Ratton (2006).

Em uma análise inicial, Ratton (2006) afirma que sons que conservam uma relação inteira diante dos valores das frequências fundamentais, proporcionará uma sensação agradável, ou pelo menos audível, pois os seus harmônicos estarão em acordo (consonância). Quando a frequência fundamental de um som equivale à metade ou ao dobro da frequência fundamental de outro, considera-se que o primeiro está oitavando o segundo, pode ser denominado como oitava acima, isto é, o mesmo que dizer como sendo um intervalo musical de 2/1. O nome oitava, têm referência à sequência das oito notas da escala maior de DÓ: DÓ, RÉ, MI, MI, SOL, MI, SI, DÓ, à que se chama igualmente "uma oitava".

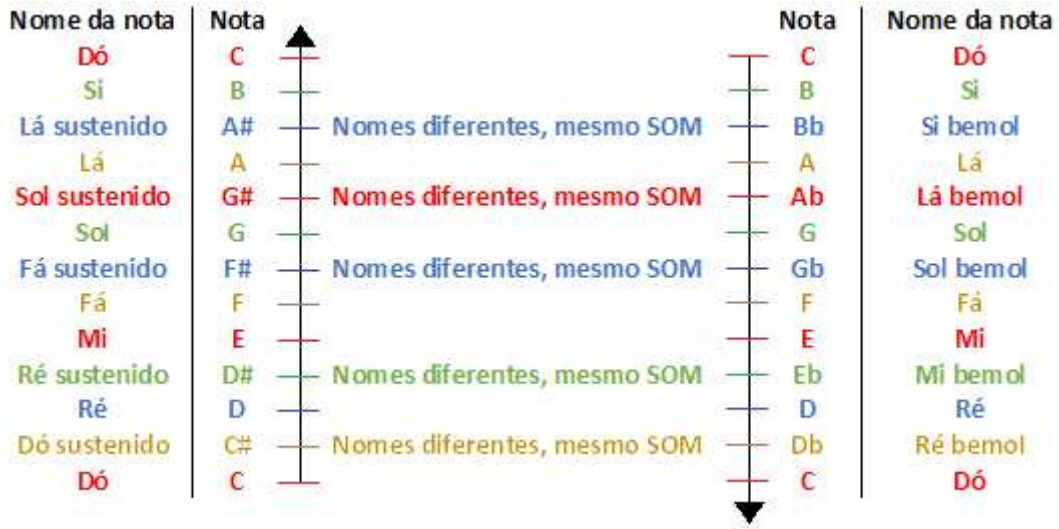
2.2.3 Tipos de Escalas

Escala serve para organizar um conjunto de sons, geralmente limitados em um intervalo de oitava, de maneira a obedecer a uma determinada regra, que depende do tipo de escala. De forma mais simples podemos dizer que a escala é uma sequência de notas musicais comportadas geralmente dentro de uma oitava. As escalas têm uma grande importância na música, toda execução de uma composição musical é realizada com base em uma escala (RATTON, 2006).

Dentre as diversas opções de escalas musicais, nesse trabalho serão utilizados dois tipos delas: escala menor (1 semitom) e escala maior (1 tom). Escalas menores costumam reproduzir sons tristes, já as escalas maiores tem um tom de diversão.

Para entender melhor a Figura 3 precisa-se de um melhor entendimento sobre notas sustenidas (quando sobe a frequência), o conceito que pode ser empregado é que quando considerada a nota SOL(G), sustenido significa o som mais próximo a essa nota na direção dos agudos (somando a frequência) formando o G# ou SOL SUSTENIDO. Já as notas denominadas bemóis (quando desce a frequência), que são representadas pela letra "b" logo após o símbolo da nota, por exemplo a nota G (SOL) quando considerada bemol, é denominada por Gb (SOL bemol). Para as notas bemóis, quando analisada a nota SI (B), e quiser obter a nota mais próxima na direção dos graves, o resultado é a nota Bb (Si bemol) (RATTON, 2006). Exemplos de notas sustenidas e bemóis podem ser encontrados na Figura 3.

Figura 3 – Representação das notas bemóis e sustenidas



Fonte: Autoria própria (2022).

Após compreendido as denotações das notas e seus saltos de tons e semitons podendo ser ascendentes (sustenidos) e descendentes (bemóis), a seguir pode ser observado na Tabela 2, as respectivas notas existentes com seus intervalos possíveis sendo representadas também pelas suas factíveis frequências entre tons e semitons. Neste caso, subindo as frequências (as notas de forma ascendentes ou seja, notas sustenidas), considerando que o do-

bro da frequência equivalente à uma nota pode-se obter a mesma nota, ou seja, o resultado é o mesmo, porém com uma frequência diferente, podendo ter um som dinâmico para os ouvidos.

Tabela 2 – Notas Musicais e suas respectivas frequências (Hz) em seus intervalos

Nota musical	1ª Oitava	2ª Oitava	3ª Oitava	4ª Oitava	5ª Oitava	6ª Oitava	7ª Oitava	8ª Oitava	9ª Oitava
C(DÓ)	33	66	132	264	528	1056	2112	4224	8448
C#	34,94	69,89	139,79	279,6	559,15	1118,3	2236,6	4473,2	8946,4
D(RÉ)	37,02	74,05	148,1	296,2	592,42	1184,8	2369,7	4739,3	9478,7
D#	39,23	78,47	156,95	313,90	627,79	1255,60	2511,20	5022,30	10045
E(MI)	41,58	83,16	166,32	332,60	665,28	1330,60	2661,10	5322,20	10644
F(MI)	44,05	88,11	176,22	352,40	704,88	1409,80	2819,50	5639	11278
F#	46,66	93,32	186,65	373,30	746,59	1493,20	2986,4	5972,70	11945
G(SOL)	49,43	98,86	197,74	395,50	790,94	1581,90	3163,80	6327,60	12655
G#	52,37	104,74	209,48	419	837,94	1675,90	3351,70	6703,50	13407
A(MI)	55,50	111,01	222,02	444	888,10	1776,20	3552,40	7104,80	14210
A#	58,80	117,61	235,22	470,40	940,90	1881,80	3763,60	7527,20	15054
B(Si)	62,30	124,61	249,22	498,40	996,86	1993,70	3987,50	7974,90	15950
C(DÓ)	66	132	264	528	1056	2112	4224	8448	16896

Fonte: Adaptado de Ratton (2006).

Na Tabela 2 é possível analisar as notas maiores e sustenidas (somando as frequências). Para melhor entendimento sobre as notas bemóis, basta considerar a Tabela 2 da maior frequência para a menor, ou seja, ao invés de somar as frequências basta subtrair e substituir a nota conforme a Figura 3.

Para uma escala menor concebe-se a seguinte sequência: tom, semitom, tom, tom, semitom, tom, tom. No caso da escala menor de RÉ, tem-se: D - E - F - G - A - Bb (A#) - C. Já na escala maior que se resume na sequência tom, tom, semitom, tom, tom, tom, semitom, concebe-se o exemplo: RÉ - MI - MI# - SOL - MI - SI - DÓ#, da escala de RÉ maior.

2.2.4 Impacto da melodia em pessoas

Conforme Benfica (2021), quando o homem entra em sintonia com a música, ele se beneficia alterando positivamente o corpo e a mente. Daí surgem várias teorias em relação à música e o homem, quando existe a conciliação desses elementos dentro de determinadas frequências sonoras, alguns efeitos são notados, devido ao fato das ondas sonoras interferirem na estrutura humana, ofertando privilégios ao lado vital.

Diante das características da música, dispõe algumas propriedades que servem como auxílio no desenvolvimento de áreas medicinais, como por exemplo em terapias, que foi sugerida por estudos como um tipo de ramo da medicina, a musicoterapia (BENFICA, 2021).

A musicoterapia é o âmbito de recuperação do ato musical como fenômeno grupal, coletivo, gerador, e veiculador de estados emocionais Fregtman (1996). De acordo com o autor, a musicoterapia engloba várias circunstâncias, seja algo grupal ou um fenômeno que espalha comoção. Considerando isso, a usabilidade da musicoterapia na sociedade tem sido uma boa

prática para fins terapêuticos, instigando a saúde nas áreas psicológicas e emocionais onde a melodia tem um papel fundamental para controlar ou estabilizar emocionalmente uma pessoa.

Na relação terapêutica, os sons permitem que se erga uma ponte de comunicação pRÉ-verbal, de expressão arcaica e concreta, relacionada com o “aqui e agora”, comparando as dissonâncias entre o dito, o expresso e o vivido (FREGTMAN, 1996).

O autor descreve que os sons permitem uma conexão, de forma terapêutica, de expressão entre o que é dito e o que é vivido, e que as ondas musicais proporcionam sentimentos e podem ser intensificados através de estímulos que influenciam nos estados de emoção do ser humano.

Segundo Campadello (1995), de todos os estímulos presentes, a música consegue ser o mais puro. Pois ela consegue despertar os mais sinceros sentimentos, modificando o humor, superando a ansiedade e dominando a depressão e também tem a capacidade de fazer com que se esqueça por alguns instantes o contato com a realidade, fazendo com que os problemas pareçam inexistentes.

Através dos conceitos da terapia musical, o homem, por meio das canções, consegue se religar com as suas emoções, sendo influenciado pelo fluxo da imaginação e por fim, sentir todo o ambiente. Como afirma Fregtman (1996), a Musicoterapia é essencialmente uma terapia ativa.

Já que o conceito de música faz a diferença no dia a dia e na vida das pessoas, podemos considerar a evolução tecnológica do momento atual e apropriar-se disso para obter artefatos que beneficiem a sociedade de maneira prática e pouco custosa. Para isso inclusive que se caracteriza a geração procedural de conteúdo.

2.3 Geração Procedural

Técnicas procedurais são frequentemente utilizadas na criação de mapas aleatórios de jogos, tais técnicas podem ser mais abrangentes na criação de enredos, personagens, texturas, entre outros. Existem casos em que é possível gerar um sistema completo totalmente aleatório, como planetas, animais, vegetação, entre outros (MENEZES, 2018).

A geração procedural de conteúdo PCG é normalmente considerada uma característica dos jogos digitais. No entanto, quando interpretado de forma ampla, PCG significa simplesmente qualquer tipo de conteúdo que é gerado computacionalmente seguindo as regras que são determinadas pelo desenvolvedor da aplicação (SMITH, 2015). Já que a geração procedural de conteúdo é fortemente utilizada em jogos, a Figura 4 apresenta alguns exemplos citados por Madeira (2018), de jogos que utilizaram técnicas de geração procedural de conteúdo. Os jogos exemplificados utilizaram a PCG para criar os seguintes conteúdos: o jogo *Elite* utilizou geração procedural de conteúdo para gerar 8 galáxias sendo cada uma delas com 256 planetas, enquanto o jogo *Rogue* teve níveis gerados a cada partida iniciada, já no jogo *Diablo*, a PCG foi utilizada para gerar mapas e alocar itens e inimigos no mapa, no *Spore* para criar animação de

novos personagens (*non-player character*), no *Minecraft* para gerar o mundo inteiro juntamente com o seu respectivo conteúdo e no *Spelunky* para gerar automaticamente as variações dos níveis do jogo.

Figura 4 – Exemplos de jogos que fizeram uso de PCG no processo de desenvolvimento

Fonte: Adaptado de Madeira (2018).

Normalmente, esse conteúdo seria produzido por um humano, mas, visando diminuir custos financeiros e de tempo, essa prática tem sido considerada relevante nos últimos tempos, já que o conteúdo gerado pode ter a qualidade controlada diante das diversas técnicas ou métodos.

As diferentes técnicas da PCG comumente se encontram na matemática (TOGELIUS *et al.*, 2011). Diante disso, supõe-se que não existe a melhor técnica utilizada quando previamente se espera um determinado conteúdo, já que as diferentes técnicas disponíveis podem gerar diferentes conteúdos, sendo que a cada execução um conteúdo pode ser melhor do que o outro.

2.3.1 Técnicas existentes

Dentre as técnicas de geração procedural de conteúdo mais comuns tem-se a PCG baseada em gramática, a PCG baseada em busca (utilizada no presente trabalho) e a PCG baseada em geometria fractal.

2.3.1.1 Baseada em Gramática

Gramática é uma prática comum à geração procedural de conteúdo, de acordo com Shaker, Togelius e Nelson (2016), ela consiste em um conjunto de regras de produção para reescrever *strings*, ou seja, transformando uma *string* em outra. Alguns exemplos de regra pode ser exemplificado da seguinte forma:

1 - A $\rightarrow$ AB

2 – B $\rightarrow$ b

O uso de gramática é simples, neste exemplo acima é possível compreender da seguinte forma: a *string* inicial é “A”, na primeira etapa da reescrita o “A” será substituído por “AB” (de acordo com a regra 1), logo a *string* resultante é “AB”, na segunda etapa da reescrita, a *string* “A” será novamente transformado em “AB” e a *string* “B” será transformada em “b” (de acordo com a regra 2). Logo a *string* resultante a partir da segunda etapa será ABb, na terceira etapa produz a *string* “ABbb” e assim sucessivamente, sempre substituindo as *strings* resultantes encontradas do lado esquerdo pelo lado direito da regra, sendo que, o critério de parada é definido pelo próprio desenvolvedor, conforme o resultado que deseja-se obter.

As gramáticas formais foram originalmente introduzidas na década de 1950, como forma de modelar a linguagem natural. No entanto, há aplicação generalizada na computação, uma vez que muitos problemas sobre, podem ser convertidos em termos de geração e compreensão de *strings* em uma linguagem formal (SHAKER; TOGELIUS; NELSON, 2016).

Existem duas distinções principais que são relevantes para o aplicativo de gramáticas na geração procedural de conteúdo, que é se as gramáticas são determinísticas e a ordem em que são expandidas. Gramáticas determinísticas têm exatamente uma regra que se aplica a cada símbolo ou sequência de símbolos, de modo que, para uma determinada *string*, é completamente inequívoco quais regras devem ser utilizadas para reescrevê-los. Já na ordem em que são expandidas, várias regras podem ser aplicadas a uma determinada *string*, produzindo diferentes resultados possíveis de uma determinada etapa de reescrita. Uma maneira de escolher qual regra usar é simplesmente escolher aleatoriamente. Em tal caso, a gramática pode até incluir probabilidades de escolha de cada regra. Outra maneira é usar alguns parâmetros para decidir como expandir a gramática, a escolha do criador da regra específica (SHAKER; TOGELIUS; NELSON, 2016).

2.3.1.2 Baseada em busca

A abordagem baseada em busca, tem sido intensamente investigada na pesquisa acadêmica de PCG. Segundo Shaker, Togelius e Nelson (2016), a PCG baseada em busca é composta pelo uso de métodos evolutivos computacionais e métodos semelhantes para gerar algum conteúdo. Desse modo, costuma-se ser mais assertiva, devido às seguintes características que a compõe:

- Algoritmo de busca - Este é o “motor” de um método baseado em busca, frequentemente algoritmos evolutivos relativamente simples funcionam bem o suficiente. Embora às vezes hajam benefícios substanciais em usar algoritmos mais sofisticados que levam, por exemplo, restrições em consideração, ou que são especializados para um determinado tipo de conteúdo.

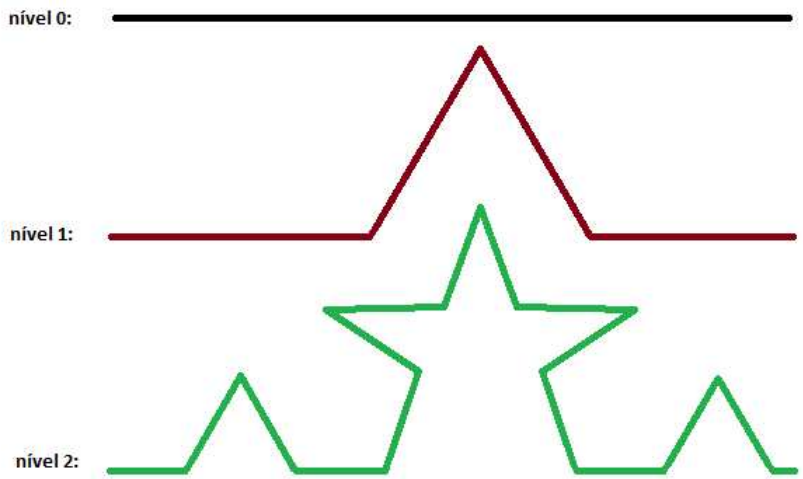
- A representação de conteúdo - Esta é a representação dos artefatos a serem gerados, por exemplo, níveis, missões, etc., no caso de jogos. A representação do conteúdo poderia ser qualquer coisa, desde uma matriz de números reais a um gráfico ou uma *string*. O conteúdo representado define (e, portanto, também limita) qual conteúdo pode ser gerado, e determina se o algoritmo de busca vai ser eficaz o suficiente para atender.

- Função(ões) de avaliação - Uma função de avaliação é uma função de um artefato (uma peça individual de conteúdo) para um número que indica a qualidade do artefato. A saída de uma função de avaliação pode indicar, por exemplo, a jogabilidade de um nível, a complexidade de uma busca ou a estética de um personagem. Construindo uma função de avaliação que mede de forma confiável, o aspecto da qualidade de um jogo que é destinado a medir está frequentemente entre as tarefas mais difíceis no desenvolvimento de um sistema que contém a PCG baseada em busca.

2.3.1.3 Baseada em geometria fractal

De acordo com Montalvão (2010), um fractal pode ser definido como uma forma geométrica fragmentada, que pode ser subdividida em partes, onde cada parte menor do objeto ou do processo coincide a um todo, ou seja, uma forma que se repete em si mesma, gerando similaridade que contenha partes iguais com outras partes que já foram geradas. Um exemplo pode ser representado como a Figura 5, que basicamente se resume na curva de Koch triangular, para um fator fractal a = 3, a curva é criada pela modificação de um segmento de reta por outros três segmentos com um terço do comprimento original.

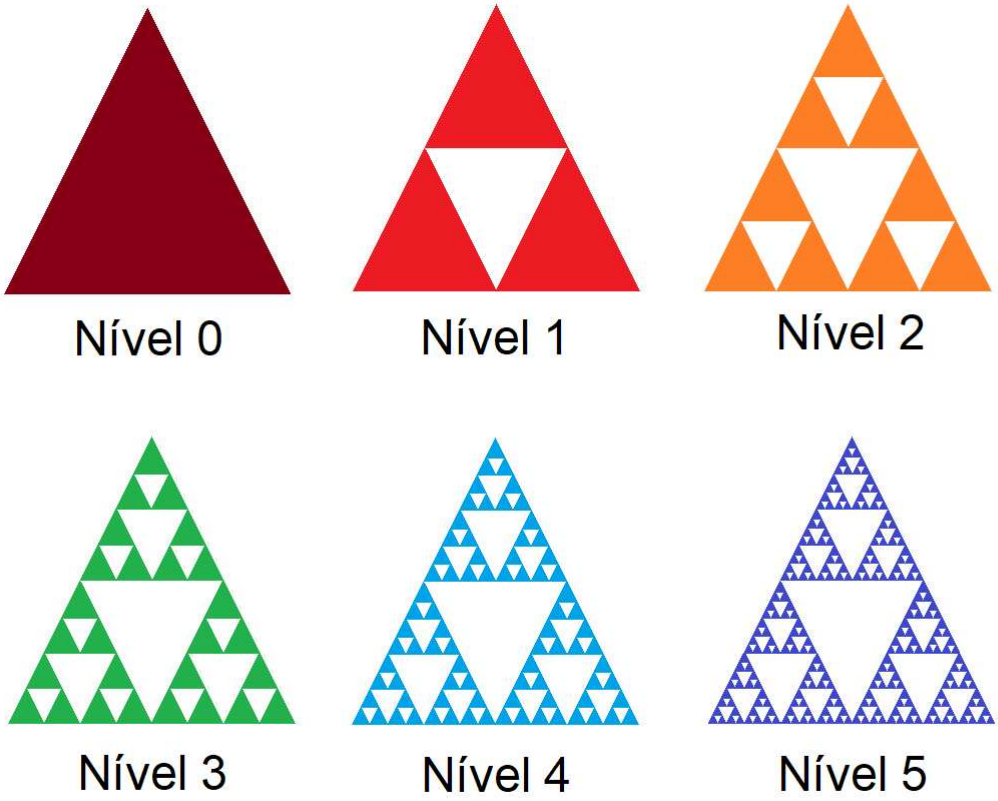
Figura 5 – Curva de Koch



Fonte: Adaptado de Montalvão (2010).

O fractal é gerado através de fórmulas matemáticas, a partir de funções na maioria das vezes simples, que quando são aplicadas de forma iterativa, produzem formas geométricas através de padrões que se repetem infinitamente. Uma das principais categorias de fractais são os geométricos, que tem uma regra fixa de substituição geométrica, é o caso do triângulo de Sierpinski. Como exemplo, se o triângulo maior for dividido em quatro outros triângulos concordantes entre si e entre o triângulo original, cujos vértices são os pontos médios do triângulo de origem, então os subconjuntos do fractal são três cópias escalonadas de triângulos derivados da iterada anterior. Esta separação do fractal em cópias escalonadas pode ser continuada recursivamente nos outros triângulos produzidos, conforme a Figura 6.

Figura 6 – Triângulo de Sierpinski



Fonte: Adaptado de (MONTALVÃO, 2010).

2.3.2 Geração procedural na música

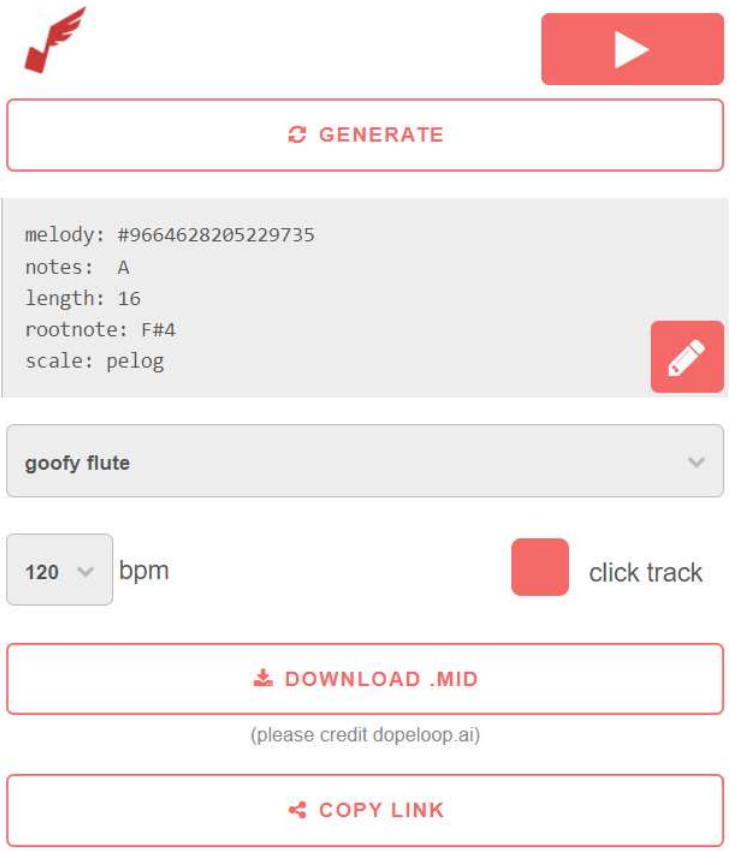
Como citado acima, um dos conteúdos que é possível se obter com a geração procedural de conteúdo, é a melodia. Mesmo não possuindo muito conteúdo que envolvam informações explícitas, é possível gerar melodias através de um processo de criação algorítmica, que é aquele no qual um conjunto de parâmetros é passado para um algoritmo que retornará automaticamente uma resposta. Uma melodia gerada computacionalmente pode ser agradável ou não, devido a aleatoriedade da geração. O mesmo não se pode dizer quando uma melodia é gerada por meio de um mestre ou profissional do meio. Porém, se utiliza desta técnica no meio da música com vários intuitos, seja para economizar tempo e/ou recursos, já que a geração procedural de conteúdo é rápida e automática (não depende de mãos humanas).

3 TRABALHOS RELACIONADOS

No presente momento existem alguns trabalhos relacionados, cada um com suas particularidades. A seguir são apresentados alguns exemplos de soluções existentes no mercado.

O DopeLoop (2022) é um sistema *web* gerador de melodias, sendo que essas são geradas de forma procedural. Uma das grandes vantagens desse sistema, é que o mesmo disponibiliza as melodias isentas de *royalties*, ou seja, pode ser feito o *download* do arquivo MIDI para o compartilhamento e/ou uso livre da melodia gerada. Esse sistema recebe algumas parametrizações, tais como: comprimento da melodia, nota raíz (nota principal da melodia), tipo de escala (maior, menor, pentatônica, etc.), instrumento musical e batimentos por minuto (bpm). Após informados os parâmetros, é possível gerar a melodia e ouvi-las no próprio *player* disponibilizado na interface do sistema. Na Figura 7, é apresentada a tela do sistema.

Figura 7 – Imagem da interface do *Dope Loop Melody Generator*



Fonte: (DOPELOOP, 2022).

O Dope Loope Melody Generator também possui a versão para dispositivos móveis, porém, essa é uma versão paga do sistema. É possível compartilhar o próprio link do site para a divulgação da melodia gerada.

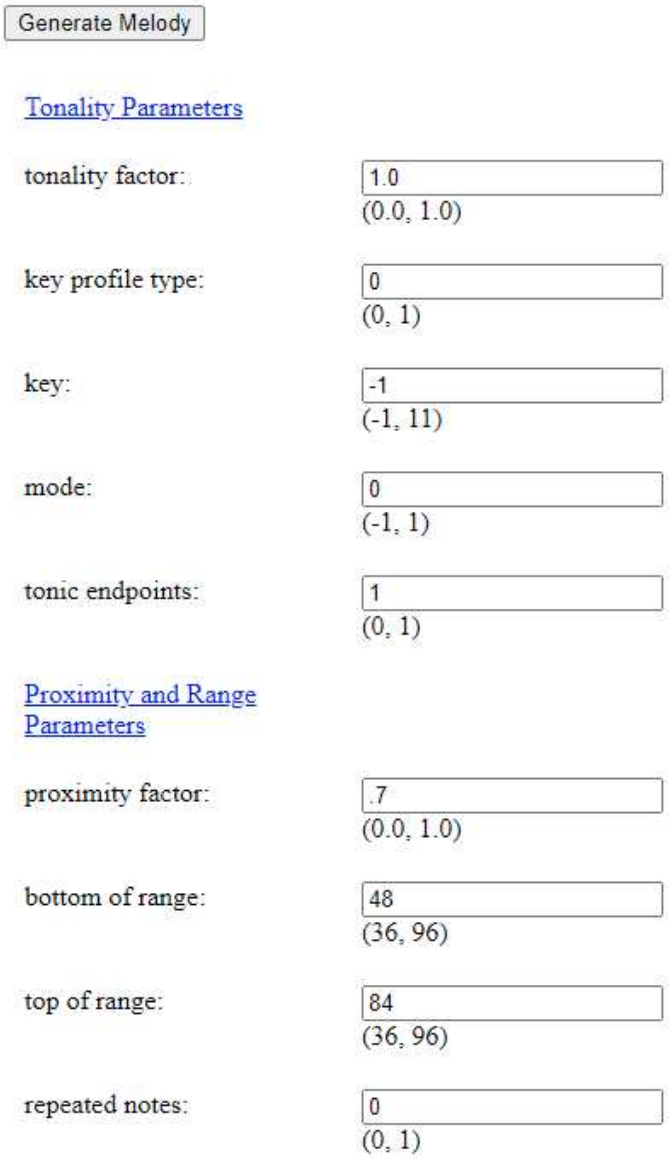
O Melisma Stochastic Melody Generator é outro sistema *web* que pode ser citado, ele permite gerar melodias originais usando processos estocásticos (ou seja, aleatórios). A partir

deste sistema, é possível gerar uma melodia usando parâmetros de sua escolha, a melodia será devolvida na forma de um arquivo MIDI, que será reproduzido em seu computador, porém, não é possível executar na própria página *web* (STOCHASTIC, 2022).

É possível notar que o Melisma não foi projetado para produzir grandes obras de arte ou sucessos número um, em vez disso, pretende ser uma forma de testar certas restrições e princípios básicos que podem operar na forma como as melodias são construídas e percebidas. Esse sistema citado, incorpora alguns dos princípios explorados no livro de David Temperley, *The Cognition of Basic Musical Structures*.

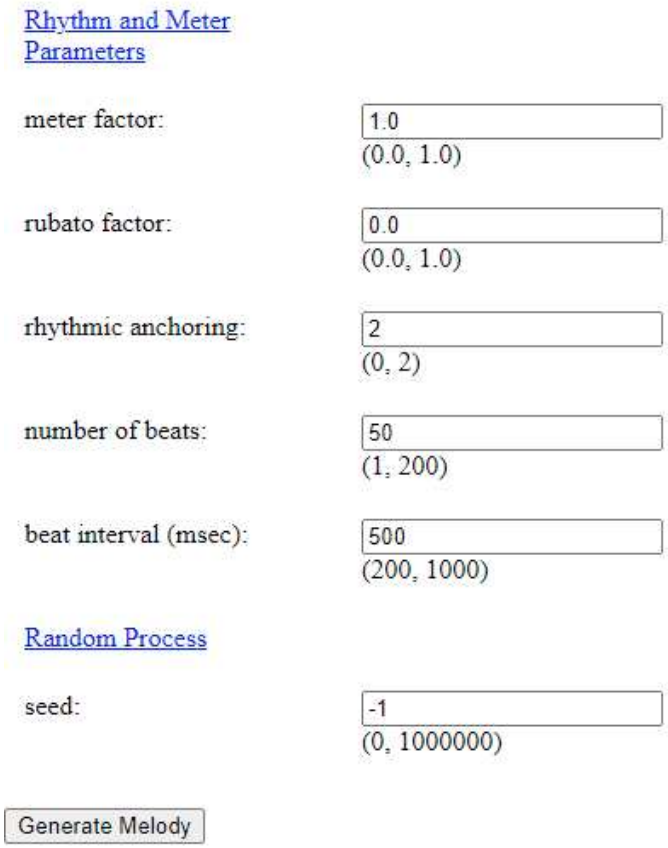
Para gerar melodias com o Melisma Stochastic, basta digitar os parâmetros desejados nos campos conforme a Figura 8 e a Figura 9.

Figura 8 – Interface parte 1 - parâmetros do *Melisma Stochastic*



Fonte: (STOCHASTIC, 2022).

Figura 9 – Interface parte 2 - parâmetros do *Melisma Stochastic*



Fonte: (STOCHASTIC, 2022).

Logo abaixo, na Tabela 3 encontram-se os parâmetros que podem ser informados na interface do sistema Melisma Stochastic e seus respectivos significados.

Tabela 3 – Parâmetros do sistema Melisma Stochastic

Nome Parâmetro	Descrição
tonality factor	Quão tonal deve ser a melodia? (0,0 = nada, 1,0 = muito)
key profile type	Se for tonal, a melodia deve usar todos os tons da escala igualmente ou enfatizar mais as notas tônicas? (1 = sim, 0 = não)
key	Em que tom deve estar? (0 = C, 1 = C#, 2 = D, etc. Caso -1, o programa escolhe aleatoriamente)
mode	Em que modo deve estar? (0 = maior, 1 = menor, -1 = o programa escolhe aleatoriamente)
tonic endpoints	Deve começar e terminar no tom tônico? (1 = sim, 0 = não)
proximity factor	Deve haver preferência por intervalos pequenos? (1,0 = forte preferência, 0,0 = nenhuma preferência)
bottom of range	Qual deve ser a parte inferior do intervalo? (36 = 2 oitavas abaixo do dó central, 96 = 3 oitavas acima)
top of range	Qual deve ser o topo da faixa? (36 = 2 oitavas abaixo do dó central, 96 = 3 oitavas acima)
repeated notes	Notas imediatamente repetidas devem ser permitidas? (1 = sim, 0 = não)
meter factor	Quão métrica deve ser a melodia, ou seja, quanta pressão deve haver para que as notas coincidam com batidas fortes? (1,0 = muito, 0,0 = nada)
rubato factor	Quanto rubato deve haver, ou seja, o andamento deve ser perfeitamente regular ou muito irregular? (0,0 = muito regular, 1,0 = muito irregular)
rhythmic anchoring	As notas em tempos muito fracos (colcheia) devem ser restritas para ocorrer apenas quando uma nota ocorre em um tempo forte (0 = sem restrição, 1 = somente quando há uma nota no tempo forte anterior ou seguinte, 2 = somente quando há uma nota no tempo forte seguinte.)
number of beats	Qual deve ser a duração da melodia em termos do número de batidas "tactus" de nível básico? (o número indica o número de batidas)
beat interval (msec)	Qual deve ser o andamento básico das tactus beats (O número indica o intervalo de tempo em milissegundos)
seed	Qualquer número inteiro positivo inserido aqui gerará a mesma melodia toda vez que for inserido. Se deixado em -1, o programa escolherá sua própria semente a cada execução.

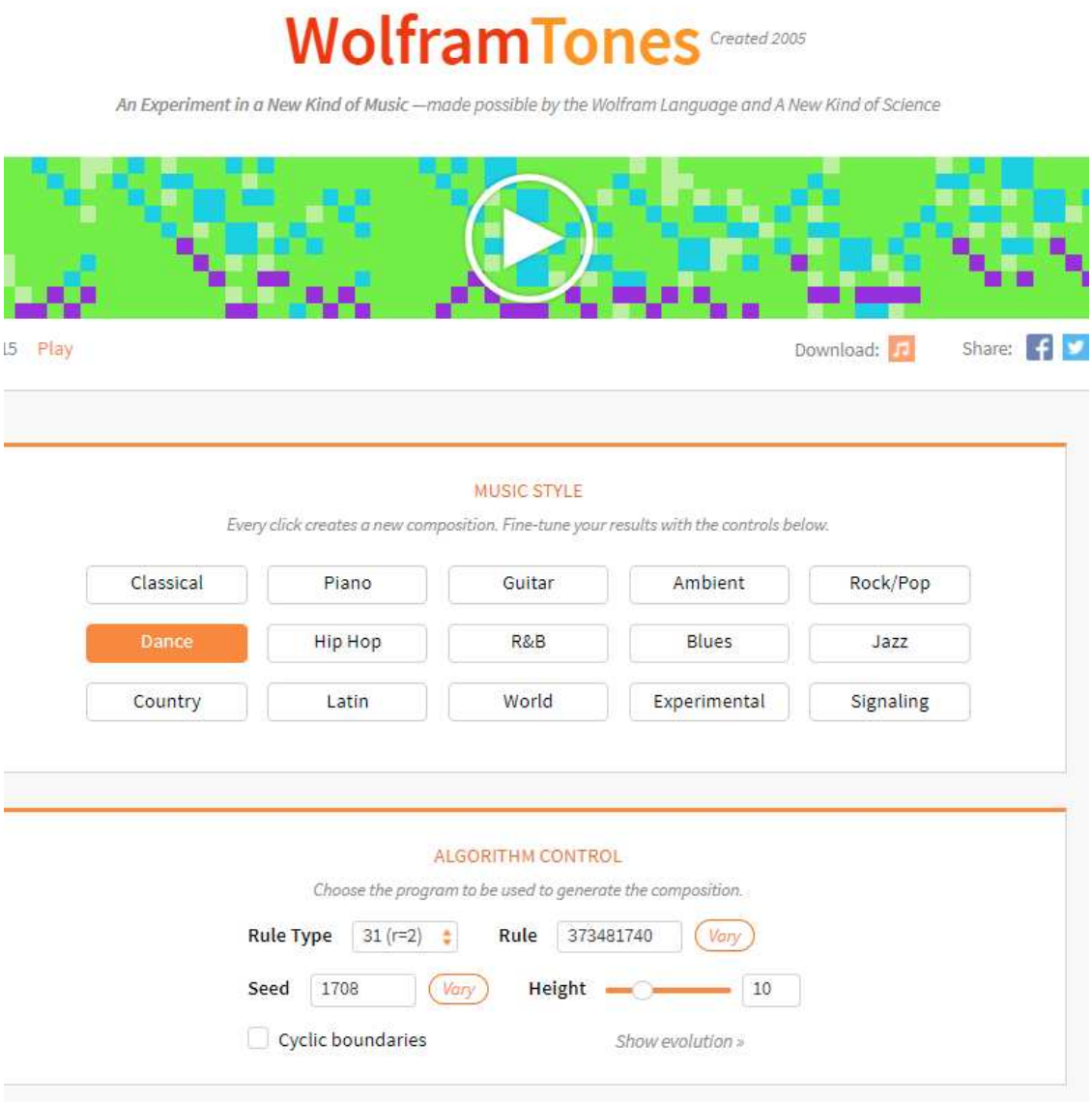
Fonte: Autoria própria (2022).

Logo após informados os parâmetros, basta clicar no botão "Gerar Melodia"e a melodia será gerada e disponibilizado o arquivo para *download*. Embora algumas melodias geradas por esse sistema sejam agradáveis, esse sistema não possui a funcionalidade de *feedback* do usuário. Outra desvantagem é que a melodia gerada só pode ser reproduzida se efetuado o *download* (disponibilizado no formato MIDI) da mesma, ou seja, não existe um *player* no sistema web para ouvir a melodia.

Com o WolframTones (2022) é possível gerar uma música completa ou somente a melodia. Existe a possibilidade de passar vários parâmetros para gerar a música, tais como: estilo da música, controle de algoritmo (pode passar a “semente”, que serve para fazer o cálculo da nota que vai ser gerada), instrumentos e ritmo que deseja na música (o máximo é 5 instrumentos simultâneo), tipo de escala da música e controle de tempo (o máximo é 15 segundos). A geração da melodia desse sistema se baseia em regras criadas pelo(a) próprio(a) desenvolvedor(a), assim como utiliza uma linguagem de programação própria, que é a *Wolfram Language*. Esse sistema também disponibiliza a música gerada para ser feito o *download* em formatos como: *MPEG-1/2 Audio Layer 3* (MP3) e *WAVEform audio format* (WAV).

O *WolframTones* funciona com os sistemas operacionais Windows, Macintosh, Linux, entre outros, pois é um sistema distribuído, ou seja, sistema *web*. Conforme a Figura 10, é possível observar o exemplo de parte da página do sistema.

Figura 10 – Exemplo parte 1 - interface do *WolframTones*

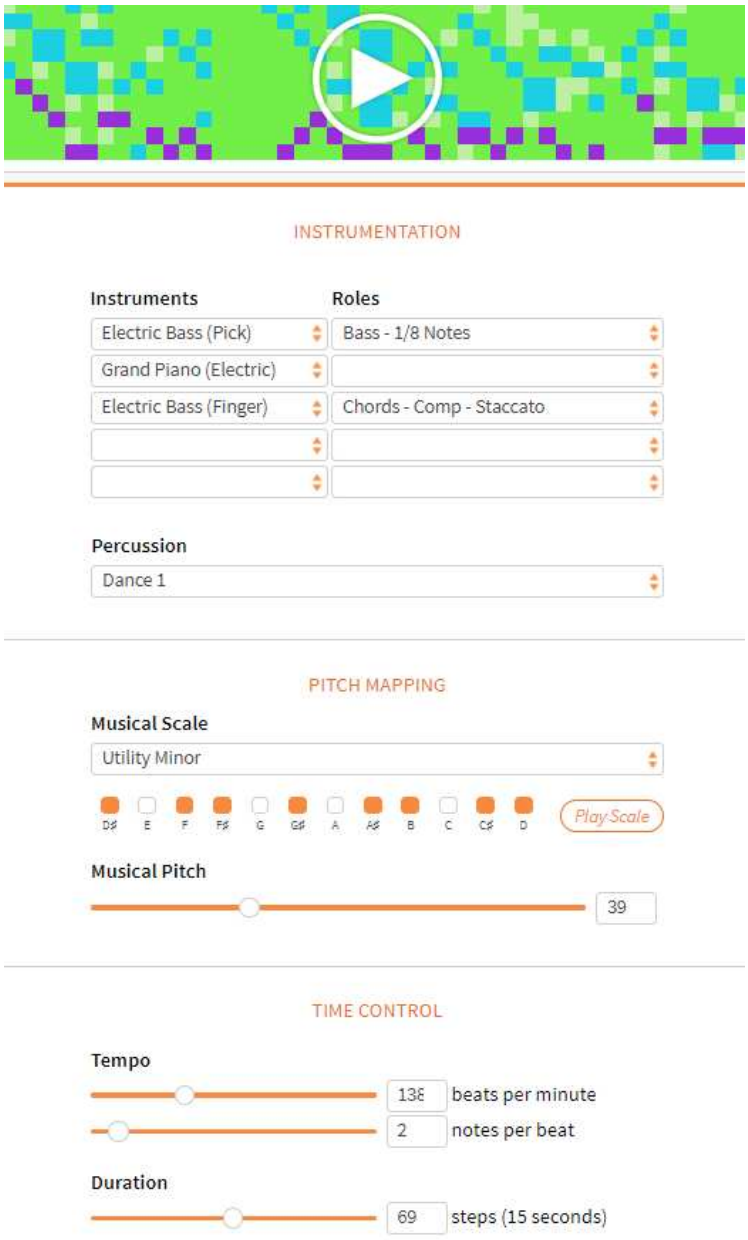


Fonte: (WOLFRAMTONES, 2022).

Nesse sistema, alguns instrumentos podem mudar mais do que outros no resultado da melodia gerada. Instrumentos mais silenciosos, como sopros, são normalmente os primeiros a deixarem a música/melodia mais suave. O WolframTones soa melhor com bons fones de ouvido ou alto-falantes e um subsistema de som de ponta, pois assim, é possível explorar melhor as frequências presentes na música ou melodia. Mais um recurso interessante desse sistema é a possibilidade do compartilhamento por meio das redes sociais (*Facebook* e *Twitter*) da música ou melodia gerada.

Segundo a documentação, o sistema WolframTones utiliza aprendizado de máquina para a geração de melodias. A melodia pode ser ouvida através do botão *play*, que fica lozalizado no topo da página. Conforme a Figura 10 e a Figura 11, é possível analisar a interface do sistema.

Figura 11 – Exemplo parte 2 - interface do *WolframTones*

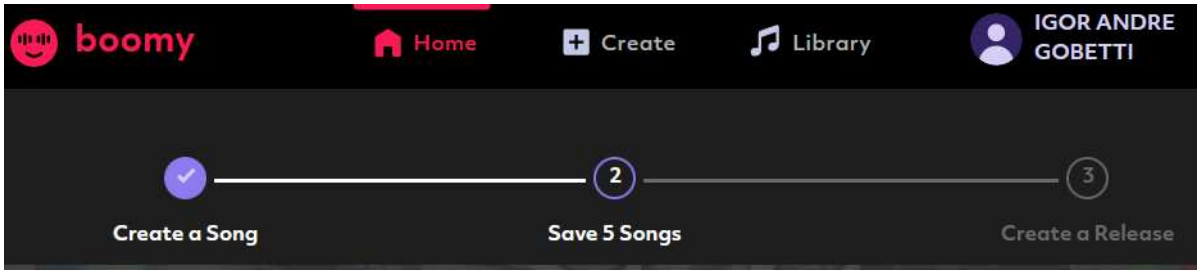


Fonte: (WOLFRAMTONES, 2022).

É possível perceber que o sistema WolframTones não possui a funcionalidade de avaliar a melodia gerada, tampouco um banco de dados das melodias geradas. No entanto, uma vez gerada a melodia, caso o usuário não tenha feito o *download* da mesma, não será possível "resgatar" a melodia gerada.

Já quando se fala em geração procedural de melodias utilizando inteligência artificial, vale a pena citar sistemas como o Boomy (2022), já que esse sistema também pode gerar músicas de diversos estilos e disponibiliza diversas funcionalidades até mesmo a edição da melodia gerada dentro da própria interface. Na Figura 12, é possível observar a tela inicial do sistema.

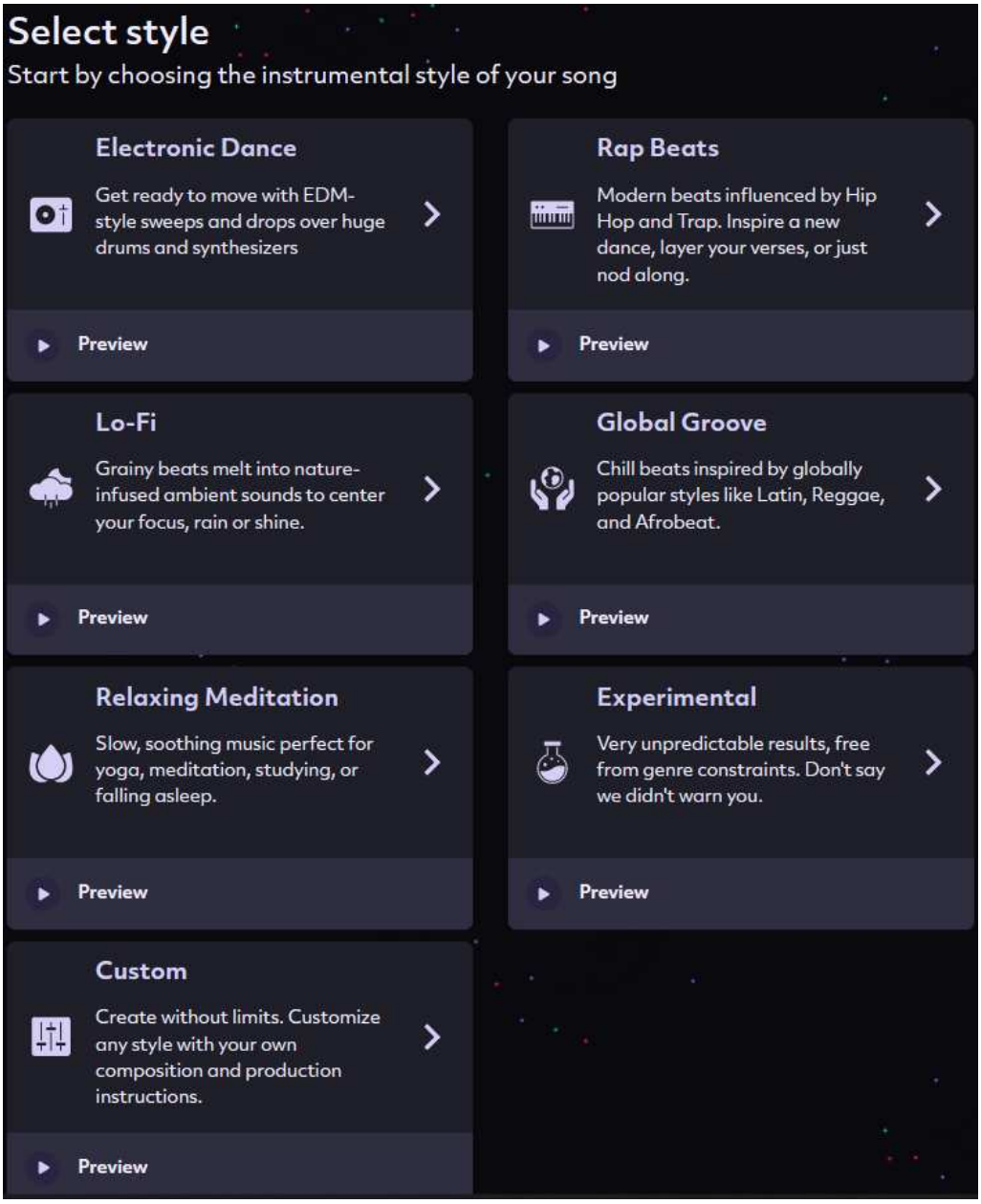
Figura 12 – Página inicial do *Boomy*



Fonte: (BOOMY, 2022).

Logo após entrar na página para gerar a melodia, o sistema apresenta possíveis estilos de melodias com parâmetros pré-programados. Tais estilos se resumem em: *Eletronic Dance* (Dança Eletrônica), *Rap Beats* (Batidas de Rap), *Lo-Fi* (Som ambiente), *Global Groove* (Ritmo Global), *Relaxing Meditation* (Meditação Relaxante), *Experimental* (Experimental) e *Custom* (Customizado). Um exemplo disso é possível verificar na Figura 13.

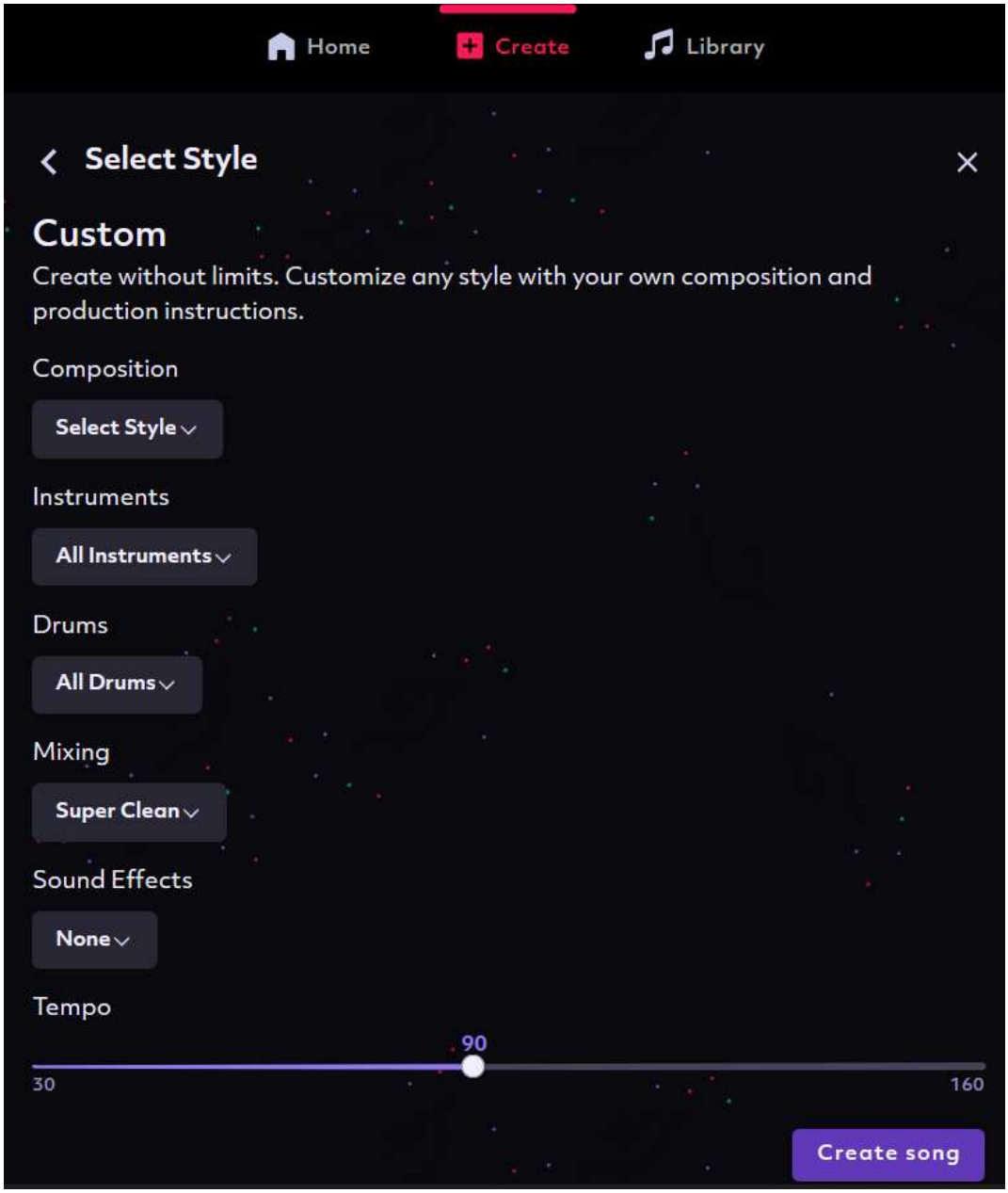
Figura 13 – Interface de melodias com parâmetros pré-definidos do sistema *Boomy*



Fonte: (BOOMY, 2022).

O sistema Boomy também pode gerar uma melodia ou uma música completa, já que disponibiliza a adição de ritmo de bateria e até mesmo a inclusão de voz. A música ou a melodia é gerada de acordo com os parâmetros informados pelo usuário, tais como: *Composition* (Estilo), *Instruments* (quais instrumentos o usuário deseja que esteja presente, sendo que pode ser 1 ou mais), *Drums* (Ritmo), *Mixing* (Tipo da composição), *Sound Effects* (Efeitos) e Tempo (Tempo de duração). A Figura 14, apresenta a interface do sistema com os parâmetros a serem informados.

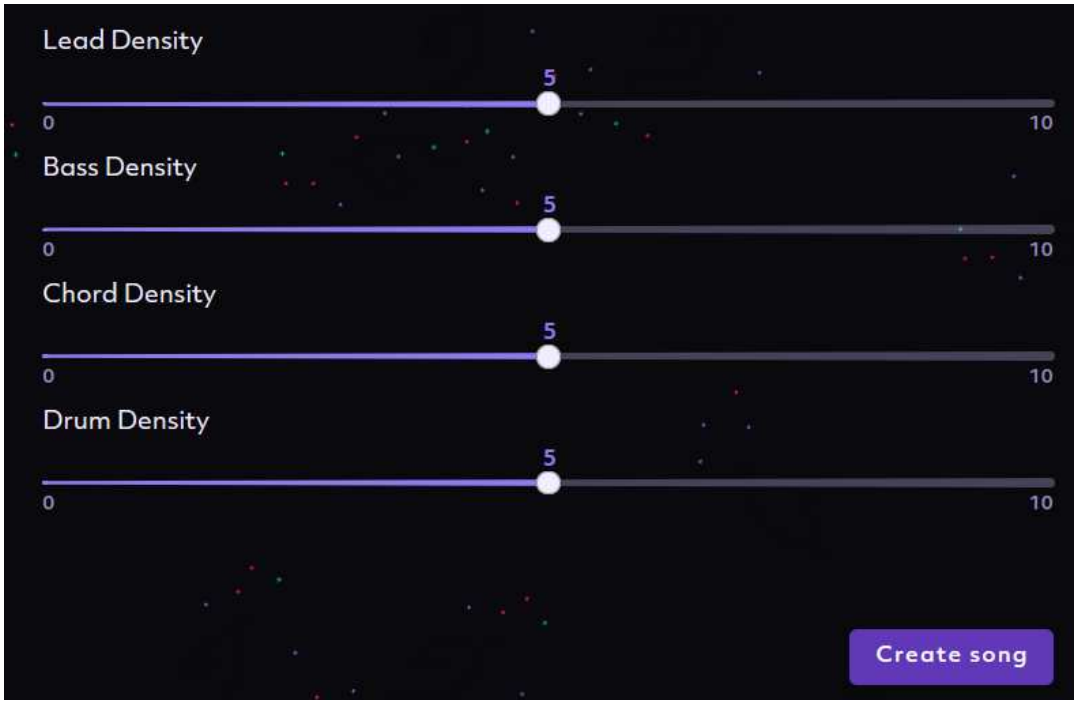
Figura 14 – Página de parâmetros à serem passados no sistema *Boomy*



Fonte: (BOOMY, 2022).

Após informados os parâmetros para a geração, tais como: Instrumentos, tipo de ritmo de bateria, efeitos da música/melodia e tempo, é possível configurar também a intensidade dos diferentes tipos de instrumentos na música ou melodia que vai ser gerada, tais como intensidade do baixo, bateria e até mesmo instrumentos de cordas. Na Figura 15, é possível ver a interface dessa funcionalidade. Após definido isso, é possível criar o som, clicando no botão *Create song*.

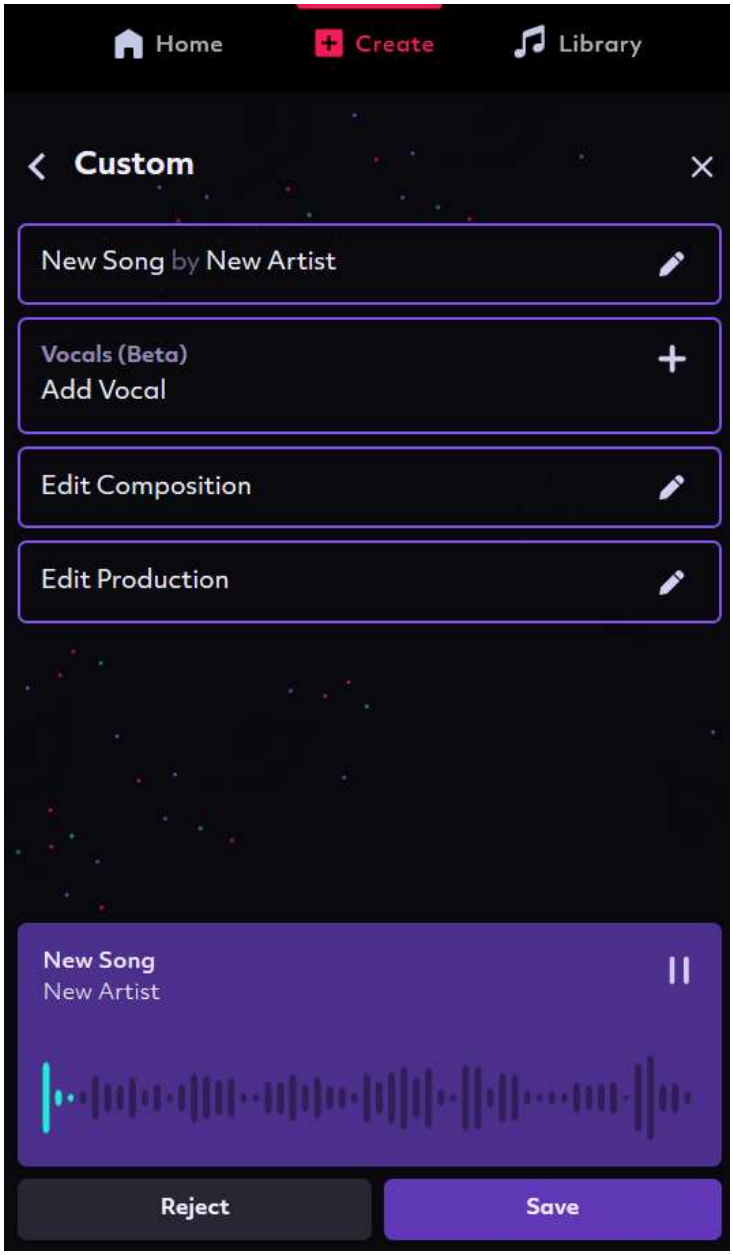
Figura 15 – Imagem da possível configuração para a geração do som no sistema *Boomy*



Fonte: (BOOMY, 2022).

Na sequência, a música ou a melodia gerada pode ser reproduzida por intermédio do *player* disponível na própria página. O sistema Boomy possui um diferencial diante dos trabalhos relacionados apresentados anteriormente, que é a edição do som já gerado, possibilitando informar o nome do usuário, adicionar voz (canto, sendo que o cantor pode ser o próprio usuário, o mesmo pode gravar sua voz cantando em cima da melodia ou música gerada), editar algum parâmetro informado anteriormente ou até mesmo inverter partes da melodia ou da música que foi gerada. Também é possível consultar melodias geradas aplicando filtros, essa funcionalidade é disponibilizada por meio do menu *Library*, conforme mostra a Figura 16.

Figura 16 – Imagem da interface com o som gerado pelo sistema *Boomy*



Fonte: (BOOMY, 2022).

O presente trabalho tem como diferencial diante dos trabalhos relacionados expostos aqui neste capítulo algumas características, tais como: avaliação do quão agradável a melodia ficou, por meio do *feedback* do usuário, que será obtido após a geração da melodia, interface desenvolvida utilizando ferramentas atuais permitindo melhor interatividade e a possibilidade de obter a melodia gerada no formato MIDI através do *download*.

4 MATERIAIS E MÉTODO

A seguir estão os materiais e método utilizado para a implementação do sistema obtido como resultado da realização deste trabalho.

4.1 Materiais

A Tabela 4 apresenta a lista de ferramentas e tecnologias utilizadas para o desenvolvimento deste trabalho.

Tabela 4 – Lista de ferramentas e tecnologias

Ferramenta / Tecnologia	**Versão**	**Finalidade**
Visual Paradigm	16.3	Modelagem do sistema
Intellij IDEA Ultimate	2021.2	IDE de Desenvolvimento
JFugue	5.0	Biblioteca para programação musical
HTML	5	Linguagem de Marcação de Hipertexto
CSS	3	Folha de estilo
Bootstrap	5.1	*Framework* para desenvolvimento de componentes de interface
JavaScript	1.5	Linguagem de programação
Jquery	3.6.0	Biblioteca de funções Javascript
Spring Boot	2.5.2	*Framework* plataforma Java
Lombok	1.18.20	*Framework* Java
Java	1.8	Linguagem de Programação
Postgre	5.2	Sistema Gerenciador de Banco de Dados
SQL	–	Linguagem de Banco de Dados
Brave	1.26.74	Navegador Web

Fonte: Autoria própria (2022).

Dos materiais utilizados para análise e desenvolvimento do trabalho proposto, com o *Visual Paradigm* foi possível desenvolver os diagramas entidade-relacionamento e diagrama de casos de uso, O *Intellij IDEA Ultimate* foi a IDE de desenvolvimento utilizada para codificar tanto o *front-end* como o *back-end* , O *JFugue* foi a biblioteca utilizada para programação musical com a linguagem Java, já O HTML, CSS, *Bootstrap*, *Javascript* e *Jquery* foram utilizados para o desenvolvimento da interface do sistema, a estilização e o comportamento de seus componentes, o *Spring Boot* e o *Lombok* foram os *frameworks* utilizadas para o desenvolvimento juntamente com a liguagem Java, já o *Postgre* e o SQL foram os responsáveis por manipulação e armazenamento dos dados e por fim o *Brave* que foi o navegador utilizado para a execução do sistema *Melody Generator*.

4.2 Método

O método utilizado neste trabalho foi particionado em 5 etapas, sendo elas: Análise de Requisitos, Diagrama Entidade-Relacionamento, Diagrama de Casos de Uso, Desenvolvimento do projeto e Testes.

4.2.1 Análise de requisitos

Nesta etapa do trabalho, são identificados os principais requisitos para desenvolver o projeto, ou seja, tudo o que é preciso para que o trabalho tenha o andamento almejado e antija os objetivos. A análise foi executada com base nos trabalhos relacionados, identificando pontos positivos e negativos de cada um, sendo que nenhum deles apresentou uma interface interativa, muito menos uma avaliação (*feedback*) por parte do usuário. Após isso, foram escolhidas as tecnologias que seriam utilizadas durante o desenvolvimento do trabalho, logo, foram escolhidas tecnologias de código aberto em função da gratuidade da utilização e considerando que já estão estabelecidas no mercado e são renomadas na atualidade.

De forma detalhada, nas tabelas a seguir serão apresentados os requisitos funcionais e não-funcionais do sistema.

Na Tabela 5 é possível notar que o requisito “Receber parâmetros para gerar a melodia” obriga o usuário informar todos os parâmetros que contenham no formulário para que a melodia possa ser gerada, para isso, o requisito não funcional "RNF1.1", fará a validação dos campos. Ainda conforme o requisito "RNF1.2", o usuário não precisará fazer login para poder gerar melodia(s), ou seja, é possível gerar uma ou mais melodias sem precisar possuir um cadastro no site.

Tabela 5 – Requisito receber parâmetros para gerar melodia

REQUISITOS FUNCIONAIS	
RF1 - Receber parâmetros para gerar melodia	
Descrição	O sistema deverá receber obrigatoriamente os parâmetros para que uma melodia possa ser gerada. São eles: Nome, Instrumento, Nota Escala, Tipo Escala, Batimentos por minuto(bpm), Quantidade de intervalo de oitavas e Quantidade de repetições.
REQUISITOS NÃO FUNCIONAIS (regras de negócio)	
RNF1.1 -Validação dos campos de parâmetros	Os campos do formulário de parâmetros deverão ser validados a fim de que não sejam nulos e que tenham sido preenchidos com valores esperados.
RNF1.2 - Acesso à geração de melodias	Não precisará possuir cadastro no sistema para gerar melodia.

Fonte: Autoria própria (2022).

Na Tabela 6, é apresentado o requisito "gerar melodia", que deve garantir que o sistema faça a geração da melodia de acordo com os parâmetros informados, respeitando o tempo máximo de espera para o processamento da geração.

Tabela 6 – Requisito gerar melodia

REQUISITOS FUNCIONAIS	
RF2 - Gerar Melodia	
Descrição	O sistema deverá gerar a melodia de acordo com os parâmetros informados pelo usuário.
REQUISITOS NÃO FUNCIONAIS (regras de negócio)	
RNF2.1 - Tempo máximo para a geração da melodia	Deve-se garantir que a geração da melodia seja executada em no máximo 30 segundos, através de um limitador do campo "Quantidade de repetições".

Fonte: Autoria própria (2022).

O requisito apresentado na Tabela 7, o usuário poderá ouvir a melodia gerada por meio do *player* disponibilizado dentro da própria página do sistema.

Tabela 7 – Requisito ouvir online a melodia gerada

REQUISITOS FUNCIONAIS	
RF3 - Ouvir online a melodia gerada	
Descrição	O sistema deverá permitir ao usuário a execução online da melodia que foi gerada, contendo um *player* de áudio na página, podendo pausar a melodia.

Fonte: Autoria própria (2022).

A Tabela 8, apresenta o requisito para permitir o *download* da melodia gerada, o usuário poderá executar a melodia em qualquer *player* de sua preferência, desde que tenha compatibilidade com o formato MIDI.

Tabela 8 – Requisito permitir o *download* no formato MIDI da melodia gerada

REQUISITOS FUNCIONAIS	
RF4 - Permitir o download no formato MIDI da melodia gerada	
Descrição	O sistema deverá disponibilizar o arquivo no formato MIDI para que o usuário faça o download da melodia gerada.
REQUISITOS NÃO FUNCIONAIS (regras de negócio)	
RNF4.1 - Integridade do arquivo	O arquivo no formato MIDI deve ser disponibilizado íntegro para a execução no player da preferência do usuário, não poderá estar corrompido.

Fonte: Autoria própria (2022).

É possível ver conforme a Tabela 9, o requisito que garante que o sistema receberá o *feedback* do usuário em relação à melodia gerada, juntamente com a nota de avaliação será possível sugerir melhoria(s) ao sistema.

Tabela 9 – Requisito receber *feedback* da melodia gerada

REQUISITOS FUNCIONAIS	
RF5 - Receber *feedback* da melodia gerada	
Descrição	O sistema deverá receber *feedback* da melodia gerada por meio de um formulário com a nota de avaliação e um campo descritivo para sugestão de melhoria.
REQUISITOS NÃO FUNCIONAIS (regras de negócio)	
RNF5.1 - Validação de campos.	Validar se foi informada a nota de avaliação da melodia.

Fonte: Autoria própria (2022).

E por último, na Tabela 10, o requisito garante que o sistema disponibilizará um menu que conterá as 10 melodias mais bem avaliadas pelos usuários, permitindo também a execução dessas melodias no *player* da própria página, ou o *download* de qualquer uma dessas melodias no presente menu.

Tabela 10 – Requisito disponibilizar menu com as 10 melodias mais bem avaliadas

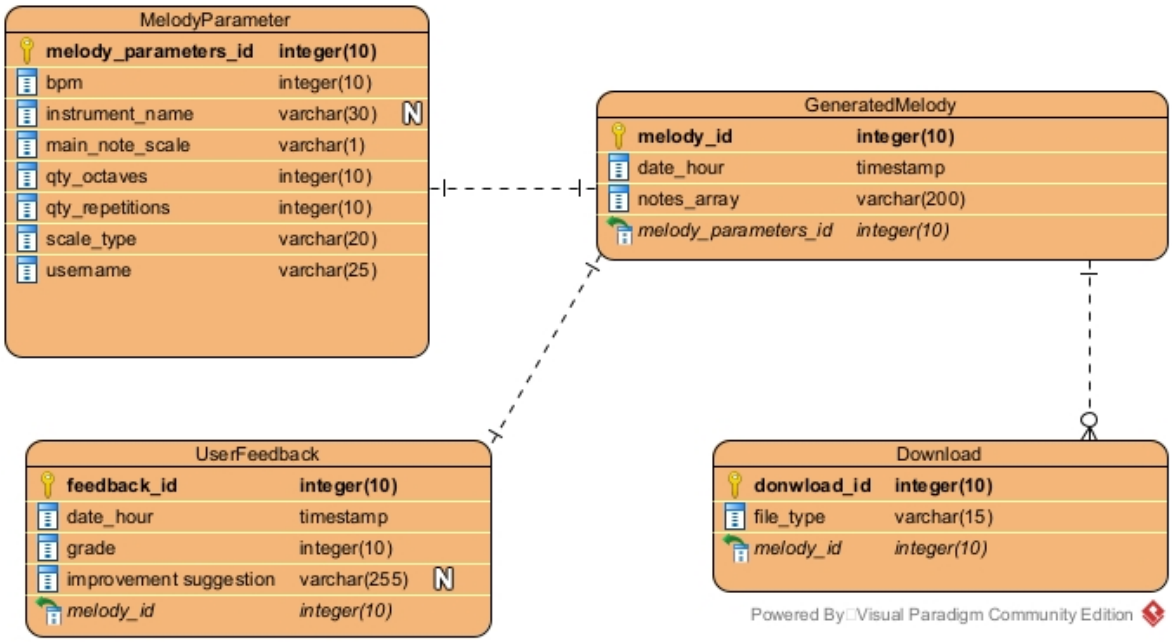
REQUISITOS FUNCIONAIS	
RF6 - Disponibilizar menu com as 10 melodias mais bem avaliadas	
Descrição	O sistema deverá disponibilizar um menu contendo as 10 melodias mais bem avaliadas pelos usuários, permitindo serem executadas e baixadas no formato MIDI.
REQUISITOS NÃO FUNCIONAIS (regras de negócio)	
RNF6.1 - Filtragem de dados	Permitir filtrar as melodias mais bem avaliadas pela nota da escala.

Fonte: Autoria própria (2022).

4.2.2 Diagrama Entidade-Relacionamento

A Figura 17 apresenta o diagrama de entidades e relacionamentos, referente ao banco de dados da aplicação.

Figura 17 – Diagrama Entidade-Relacionamento



Fonte: Autoria própria (2022).

Conforme representado na Figura 17, a tabela MelodyParameter possui relacionamento 1:1 com a tabela GeneratedMelody, esses relacionamentos foram definidos para que um conjunto de parâmetros da tabela MelodyParameter permita gerar somente 1 melodia e que uma melodia pertença a um único e exclusivo conjunto de parâmetros. A tabela GeneratedMelody também possui relacionamento 1:1 com a entidade UserFeedback, permitindo ao usuário avaliar somente 1 vez a melodia gerada. E, por fim, a tabela Download se relaciona com a cardinalidade N:1 em relação à tabela GeneratedMelody, definindo que uma melodia gerada pode ter vários eventos de *download*.

Nas tabelas a seguir encontram-se as descrições das tabelas do banco de dados. A Tabela 11 representa os campos da entidade MelodyParameter.

Tabela 11 – Campos da tabela MelodyParameter

Nome do campo	Tipo	Nulo	É chave primária	É chave estrangeira
melody_parameters_id	Inteiro	Não	Sim	Não
bpm	Inteiro	Não	Não	Não
instrument_name	Texto	Não	Não	Não
main_note_scale	Inteiro	Não	Não	Não
qty_octaves	Inteiro	Não	Não	Não
qty_repetitions	Inteiro	Não	Não	Não
scale_type	Texto	Não	Não	Não
username	Texto	Não	Não	Não

Fonte: Autoria própria (2022).

Logo abaixo, é possível entender o que cada parâmetro da tabela MelodyParameter significa:

- melody_parameters_id - é um atributo do tipo inteiro e chave primária da entidade, ele significa um identificador único e é gerado automaticamente, garantindo que cada registro de conjunto de parâmetros terá um número de identificador único;
- bpm - é um campo do tipo inteiro, bpm significa batimentos por minuto e define a velocidade da melodia, os valores aceitos são de 20 a 1000, esse campo não pode ser nulo;
- instrument_name - esse parâmetro é do tipo texto e recebe o nome do instrumento que o usuário deseja que a melodia execute, os possíveis valores para esse campo são: Piano, Guitarra, Violino, Viola, Trombone, Trompete e Baixo, não é permitido valores nulos para esse campo;
- main_note_scale - esse atributo do tipo texto se refere à nota principal que a melodia deve ser gerada, ou seja, recebe a nota base da escala da melodia, aceitando as notas Dó (C), Ré (D), Mi (E), Fá (F), Sol (G), Lá (A) ou Si (B), main_note_scale não pode ser nulo;
- qty_octaves - é um atributo do tipo inteiro e significa a quantidade de oitavas que deseja permitir para que a melodia seja gerada, os valores possíveis são de 1 a 8, esse é um campo que não pode ser nulo;
- qty_repetitions - esse atributo é do tipo inteiro e tem referência à quantidade de repetições das notas da escala que deseja permitir, podendo ser informado valor de 1 a 100, esse atributo não pode ser nulo;
- scale_type - esse campo é do tipo Texto e recebe o tipo da escala que o usuário deseja gerar a melodia, podendo ser maior ou menor, esse campo não pode ser nulo;

- username - esse parâmetro recebe o nome do usuário que informou os parâmetros para gerar a melodia, não pode ser nulo.

A seguir na Tabela 12, estão os campos da entidade GeneratedMelody.

Tabela 12 – Campos da tabela GeneratedMelody

Nome do campo	Tipo	Nulo	É chave primária	É chave estrangeira	Considerações
melody_id	Inteiro	Não	Sim	Não	
date_hour	Data	Não	Não	Não	
notes_array	Texto	Não	Não	Não	
melody_parameters_id	Inteiro	Não	Não	Sim	Oriundo da tabela MelodyParameter

Fonte: Autoria própria (2022).

Na sequência, podem ser compreendidos os atributos da Tabela 12:

- melody_id - é um campo do tipo inteiro e chave primária da entidade, o número pertencente à este campo é um identificador único gerado automaticamente, cada melodia gerada possuirá um número correspondente;
- date_hour - é um campo do tipo data, esse campo recebe a data e hora no exato momento em que a melodia foi gerada, não aceita valor nulo;
- notes_array - esse parâmetro é do tipo texto e é preenchido pelo sistema, o campo recebe a sequência de notas referente à melodia gerada, esse campo não pode ser nulo;
- melody_parameters_id - esse atributo do tipo inteiro recebe o código identificador (campo melody_parameters_id da tabela MelodyParameter), esse campo não será nulo, pois é uma chave estrangeira da tabela.

A Tabela 13, contém os campos pertencentes à entidade UserFeedback.

Tabela 13 – Campos da tabela UserFeedback

Nome do campo	Tipo	Nulo	É chave primária	É chave estrangeira	Considerações
feedback_id	Inteiro	Não	Sim	Não	
date_hour	Data	Não	Não	Não	
grade	Inteiro	Não	Não	Não	
improvement_suggestion	Texto	Sim	Não	Não	
melody_id	Inteiro	Não	Não	Sim	Oriundo da tabela GeneratedMelody

Fonte: Autoria própria (2022).

Para melhor compreender, abaixo estão descritos os campos da entidade UserFeedback:

- feedback_id - é um campo do tipo inteiro e chave primária da entidade, o número pertencente à este campo é um identificador único gerado automaticamente, cada feedback da melodia gerada possuirá um número exclusivo correspondente;
- date_hour - é um campo do tipo data, esse campo recebe a data e hora no exato momento em que a melodia foi avaliada, não aceita valor nulo;
- grade - esse parâmetro é do tipo inteiro, ele receberá a nota do quão agradável ficou a melodia gerada, valores entre 1 e 5 são permitidos, esse campo não pode ser nulo.
- melody_id - esse atributo do tipo inteiro recebe o código identificador da melodia (campo melody_id da tabela GeneratedMelody), esse campo não será nulo, pois é uma chave estrangeira da tabela.

E por fim, na Tabela 14, estão os dados da tabela Download.

Tabela 14 – Campos da tabela Download

Nome do campo	Tipo	Nulo	É chave primária	É chave estrangeira	Considerações
download_id	Inteiro	Não	Sim	Não	
file_type	Texto	Não	Não	Não	
melody_id	Inteiro	Não	Não	Sim	Oriundo da tabela GeneratedMelody

Fonte: Autoria própria (2022).

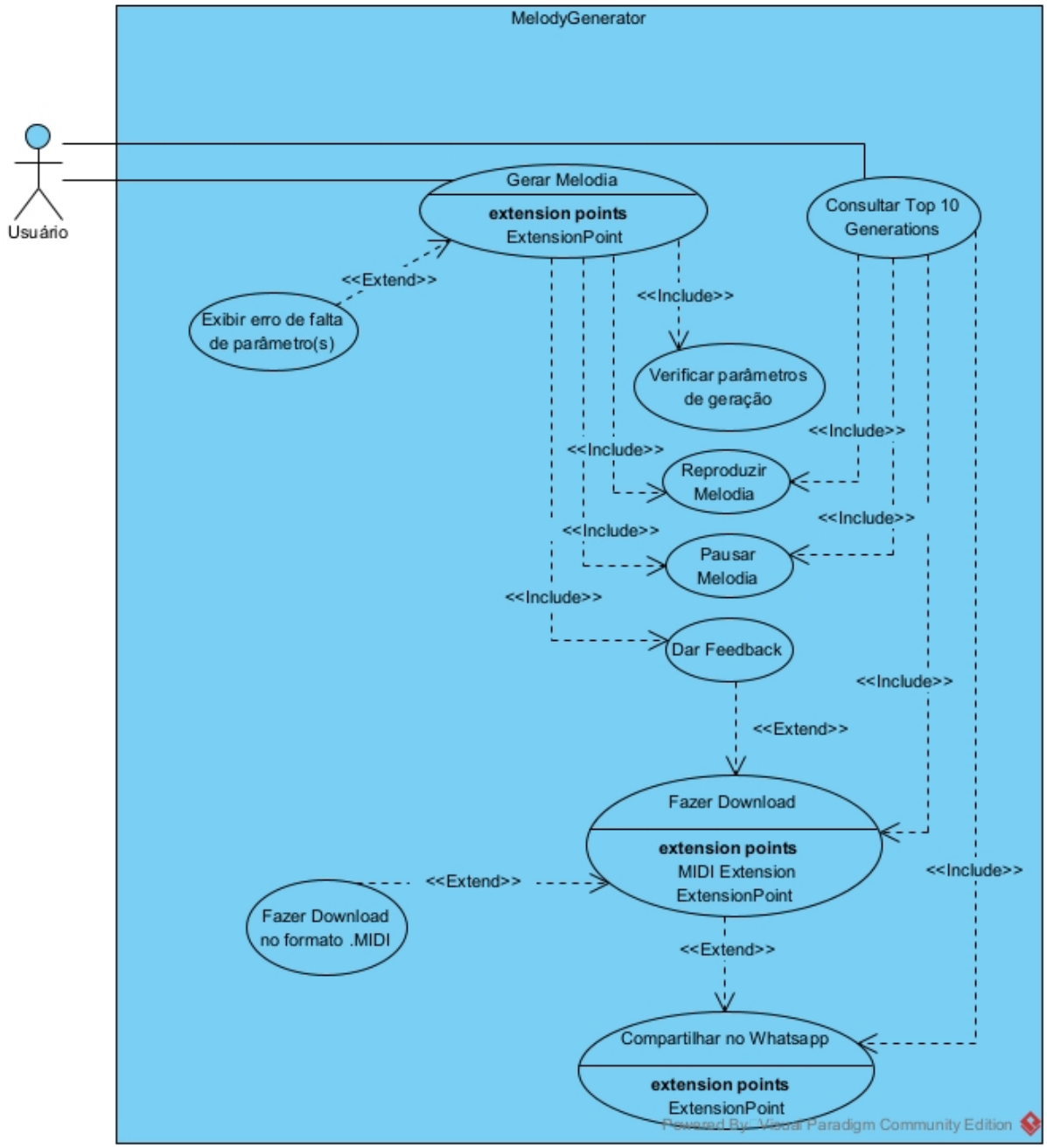
- download_id - é um campo do tipo inteiro e chave primária da entidade, o número pertencente à este campo é um identificador único gerado automaticamente, cada download da melodia gerada possuirá um número exclusivo correspondente;
- file_type - é um campo do tipo texto, esse campo recebe o tipo do arquivo disponibilizado para download, o valor padrão para este trabalho neste campo é MIDI, esse campo não aceita valor nulo;
- melody_id - esse atributo do tipo inteiro recebe o código identificador da melodia (campo melody_id da tabela GeneratedMelody), esse campo não será nulo, pois é uma chave estrangeira da tabela.

Nesse subcapítulo, foram apresentadas as entidades referente ao banco de dados da aplicação, bem como seus respectivos atributos e as chaves estrangeiras, descrevendo também o significado de cada atributo e os valores aceitos para cada campo.

4.2.3 Diagrama de Casos de Uso

O diagrama de casos de uso apresentado na Figura 18 contém as funcionalidades essenciais do sistema organizadas contendo inclusive o seu ator, que no caso é o próprio usuário do sistema.

Figura 18 – Diagrama de Casos de Uso



Fonte: Autoria própria (2022).

Abaixo, podem ser observados os respectivos casos de uso, bem como sua descrição e suas exceções. Existem dois tipos de sequência típica de eventos do Fluxo Principal, são elas: *IN* que significa uma ação de entrada no sistema(requisição) e *OUT* a ação de saída do sistema(resposta).

Na Tabela 15, é possível compreender o caso de uso Gerar Melodia.

Tabela 15 – Caso de uso Gerar Melodia

Caso de uso:	Gerar melodia
Ator:	Usuário
Pré-condições:	Os parâmetros: Nome, Instrumento, Nota escala, Tipo escala, Batimentos por minuto (BPM), Quantidade de intervalo de oitavas e Quantidade de repetições, devem estar preenchidos e validados
Pós-condições:	Melodia gerada de acordo com os parâmetros informados.
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator clica no botão "Gerar Melodia", na tela inicial. 2. [OUT] O sistema apresenta a tela de parâmetros. 3. [IN] O ator informa os parâmetros. 4. [OUT] O sistema valida os parâmetros informados e gera a melodia.	
Tratamento de Exceções	
Exceção 1a: Falta de parâmetro(s) 1.1. O sistema informa o ator que faltam parâmetros para gerar a melodia. Exceção 2a: Parâmetro(s) incorreto(s) 2.1. O sistema informa o ator que existe(m) parâmetro(s) incorreto(s).	

Fonte: Autoria própria (2022).

Na apresentação da Tabela 16, é possível observar a descrição do caso de uso Play.

Tabela 16 – Caso de uso Play

Caso de uso:	Reproduzir melodia
Ator:	Usuário
Pré-condições:	Existir uma melodia gerada
Pós-condições:	Reprodução da melodia gerada
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator clica no botão "play"no player da interface. 2. [OUT] O sistema reproduz a melodia gerada.	

Fonte: Autoria própria (2022).

Na Tabela 17, é apresentado o caso de uso Pause.

Tabela 17 – Caso de uso Pausar

Caso de uso:	Pausar melodia
Ator:	Usuário
Pré-condições:	Melodia estar sendo reproduzida no player
Pós-condições:	Pause da melodia que está em reprodução.
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator clica no botão "pause", no player da interface. 2. [OUT] O sistema pausa a melodia gerada.	

Fonte: Autoria própria (2022).

Conforme a Tabela 18, o caso de uso Top 10 Generations é apresentado.

Tabela 18 – Caso de uso Top 10 *Generations*

Caso de uso:	Top 10 Generations
Ator:	Usuário
Pré-condições:	Melodias que foram geradas possuírem alguma avaliação
Pós-condições:	Apresentação das 10 melodias mais bem avaliadas
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator clica no menu Top 10 na interface do sistema. 2. [OUT] O sistema apresenta a tela com as 10 melodias mais bem avaliadas, de acordo com a nota musical escolhida no filtro. 3. [IN] O ator solicita a reprodução de qualquer melodia do top 10. 4. [OUT] O sistema reproduz a melodia solicitada. 5. [IN] O ator solicita para que a melodia que está sendo reproduzida seja pausada. 6. [OUT] O sistema pausa a melodia que está sendo reproduzida. 7. [IN] O ator solicita o \ *download* da melodia. 8. [OUT] O sistema baixa a melodia no formato \ MIDI para o usuário.	

Fonte: Autoria própria (2022).

A Tabela 19 mostra o caso de uso *Feedback*.

Tabela 19 – Caso de uso *Feedback*

Caso de uso:	Avaliar melodia (*Feedback*)
Ator:	Usuário
Pré-condições:	Existir uma melodia gerada, usuário informar pelo menos o parâmetro nota, não sendo obrigatório informar a sugestão de melhoria do sistema.
Pós-condições:	Armazenamento da avaliação da melodia gerada e disponibilização do *download* da melodia.
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator informa a nota de avaliação e/ou sugestão para a melodia gerada. 2. [OUT] O sistema armazena no banco de dados a resposta do usuário. 3. [OUT] O sistema disponibiliza o botão para o *download* da melodia gerada.	

Fonte: Autoria própria (2022).

E por último, na Tabela 20, é apresentado o caso de uso *Download*.

Tabela 20 – Caso de uso Download

Caso de uso:	Fazer *Download*
Ator:	Usuário
Pré-condições:	A melodia gerada possuir uma avaliação
Pós-condições:	Disponibilização do *download* da melodia.
Sequência típica de eventos(Fluxo Principal)	
Esse caso de uso inicia quando: 1. [IN] O ator clica no botão *download*. 2. [OUT] O sistema baixa no dispositivo do usuário a melodia gerada no formato MIDI.	

Fonte: Autoria própria (2022).

4.2.4 Conceitos de Engenharia de Software utilizados

A Engenharia de Software se tornou importante no desenvolvimento desse projeto em função da prevenção das potenciais falhas e para resultar na alta performance dessa solução. A Análise de Requisitos foi utilizada para identificar dados necessários e indispensáveis para que fossem alcançados os objetivos do trabalho, foram coletados também as exigências necessárias para garantir a solução do problema proposto. Tais requisitos foram detalhados e relevantes para o projeto, pois foram eles que forneceram a referência para a validação do resultado final.

Já na parte do Diagrama Entidade-Relacionamento, foi possível obter as entidades e seus atributos envolvidos no trabalho bem como seus relacionamentos. Nessa etapa, foram descritas as tabelas que constituem o banco de dados, os seus respectivos atributos, bem como suas chaves primárias e estrangeiras.

E por último o Diagrama de Casos de Uso, que foi responsável por demonstrar as maneiras que o usuário pode interagir com o sistema. Assim, foi possível obter o cenário que o

sistema interage com o usuário, modelar o fluxo básico de eventos, bem como ter uma visão geral do escopo do sistema.

4.2.5 Testes

Os testes realizados no trabalho proposto foram informais, ou seja, sem definir um plano de testes e aconteceram simultaneamente com o desenvolvimento do projeto. Logo, alguns testes no momento da implementação foram importantes para a definição de algumas regras já descritas nesse trabalho, tal como, o tempo de duração das notas. O objetivo dos testes foi verificar possíveis erros de codificação, o cumprimento dos requisitos e o bom funcionamento do sistema como um todo. Logo, a fase de teste foi concluída com êxito pelo próprio autor do trabalho.

5 RESULTADOS

5.1 Escopo do sistema

O sistema gera proceduralmente uma melodia empregando geração procedural baseada em busca aplicando diferentes heurísticas, podendo ser agradável ao ouvido humano ou não. De acordo com critérios pré definidos e a parametrização por parte do usuário, será possível obter uma melodia gerada proceduralmente. De acordo com a parametrização, é possível obter diferentes tipos de melodias, podendo ser gerado uma melodia diferente da outra. O acesso ao sistema se dá por meio de uma *Uniform Resource Locator* (URL) (site *WEB*), podendo ser acessado de qualquer computador ou dispositivo que tenha acesso à internet. O *software* será responsivo, podendo também ser acessado por qualquer dispositivo móvel. Para que a melodia seja gerada, o usuário deve informar os seguintes parâmetros: Nome do usuário, Instrumento musical (piano, guitarra, violino, viola, trombone, trompete e baixo), Nota da escala (Dó (C), Ré (D), Mi (E), Fá (F), Sol (G), Lá (A), Si (B)), Tipo da escala (maior ou menor), a velocidade (bpm), quantidade de intervalo de oitavas e a quantidade de repetições do *Array* de *String* de notas geradas. Após isso, basta clicar no botão “Gerar” para ser gerada a melodia proceduralmente, através de um *player* disponibilizado dentro do site, é possível ouvir a melodia que foi gerada. Dentre as funcionalidades do sistema, é permitido também fazer uma avaliação da melodia que foi gerada deixando uma nota avaliativa e opcionalmente uma breve sugestão de melhoria de forma escrita, para que futuras gerações possam ser melhoradas, e logo após caso o usuário opte, poderá fazer o *download* da melodia no formato MIDI.

Por fim, há o menu das 10 melodias mais bem avaliadas pelos usuários, podendo ser aplicado um filtro da nota principal da escala, com isso, é possível ouvir as melodias ou fazer o *download* das mesmas.

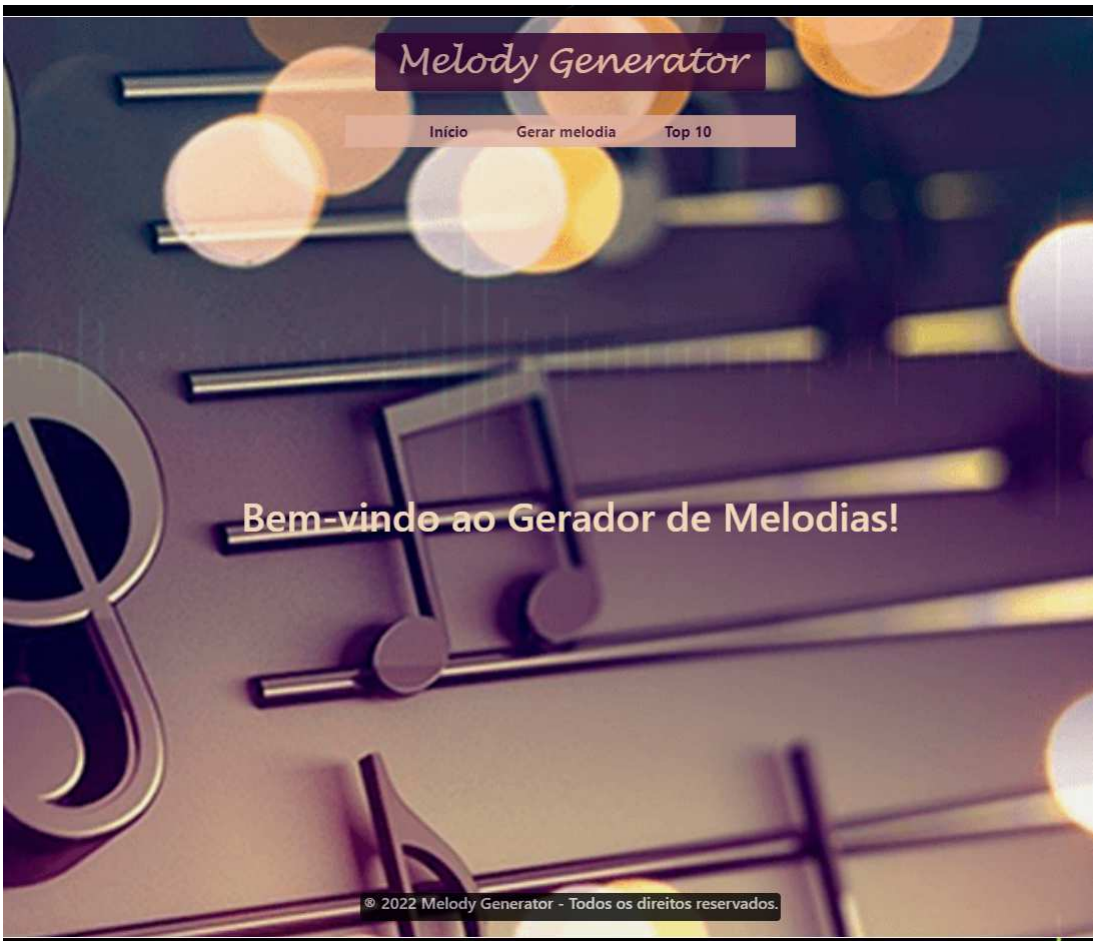
Já que usualmente melodias são utilizadas para jogos e é um dos elementos que pode compor uma música, a vantagem do sistema é o poder da disseminação da melodia gerada, podendo ser um caso de sucesso diante do livre compartilhamento do arquivo, já que este é disponibilizado pelo sistema.

5.2 APRESENTAÇÃO DO SISTEMA

Apresentado na Tabela 4, o sistema foi desenvolvido utilizando a *Integrated Development Environment (IDE) Intellij*, tanto para o *front-end* quanto para o *back-end*.

A Figura 19, exibe a página inicial do sistema, onde é possível acessar os seguintes menus: Início (exibe a tela inicial do sistema), Gerar melodia (direciona para a página de parâmetros para que uma melodia possa ser gerada) e Top10 (contém as 10 melodias mais bem avaliadas do sistema).

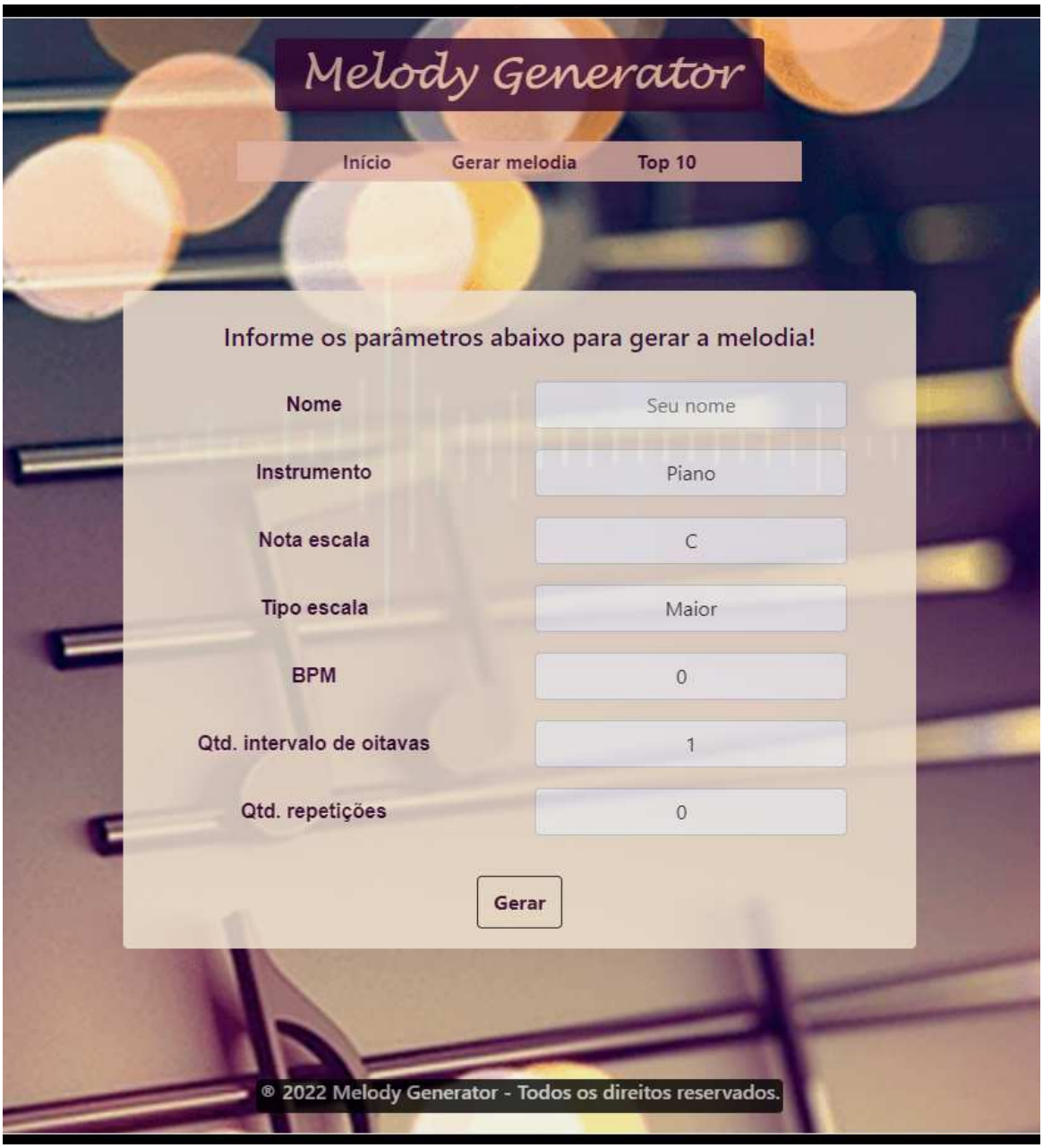
Figura 19 – Página inicial



Fonte: Autoria própria (2022).

Logo abaixo, conforme a Figura 20, quando clicar no menu "Gerar melodia", é apresentada a página de parametrização para que possa ser gerada uma melodia.

Figura 20 – Página Gerar Melodia



Fonte: Autoria própria (2022).

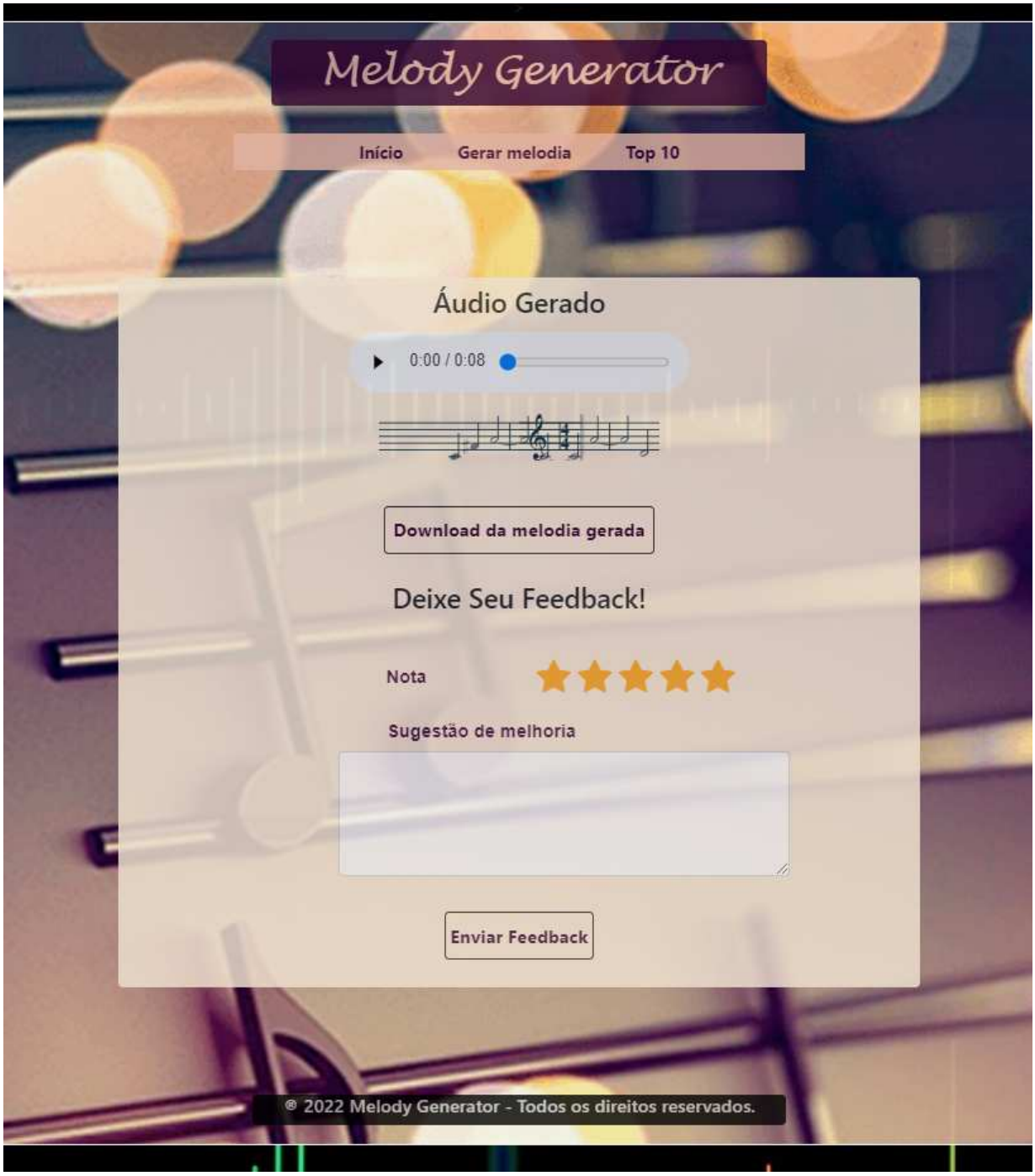
Apresentados na Figura 20, obrigatoriamente precisam ser informados todos os parâmetros, tais como:

- Nome - que consiste no nome do usuário;
- Instrumento - informar o instrumento por meio de uma caixa de seleção, que deseja que a melodia reproduza, sendo eles, Piano, Guitarra, Violino, Viola, Trombone Trompete e Baixo;

- Nota escala - informar a nota principal da escala que deseja que a melodia seja gerada, dentre as possíveis, pode selecionar entre: C (Dó), D (Ré), E (Mi), F (Fá), G (Sol), A (Lá) e B (Si);
- Tipo escala - selecionar o tipo de escala que deseja: Maior ou Menor;
- BPM (batimentos por minuto) - informar a velocidade da melodia, denotada pelo parâmetro bpm, os valores possíveis são entre 20 e 1000;
- Qtd. intervalo de oitavas - selecionar a quantidade de oitavas à serem consideradas para a geração, os valores vão de 1 à 8;
- Qtd. repetições - informar a quantidade de repetições que deseja da sequência de notas geradas.

Após informar todos os parâmetros solicitados, é possível gerar a melodia clicando no botão "Gerar", que redirecionará o usuário para a página da melodia gerada. É importante salientar que conforme a preferência do usuário da utilização dos parâmetros, uma melodia diferente pode ser gerada se alterado apenas 1 dos parâmetros. A representação da página se dá pela Figura 21.

Figura 21 – Página Melodia Gerada



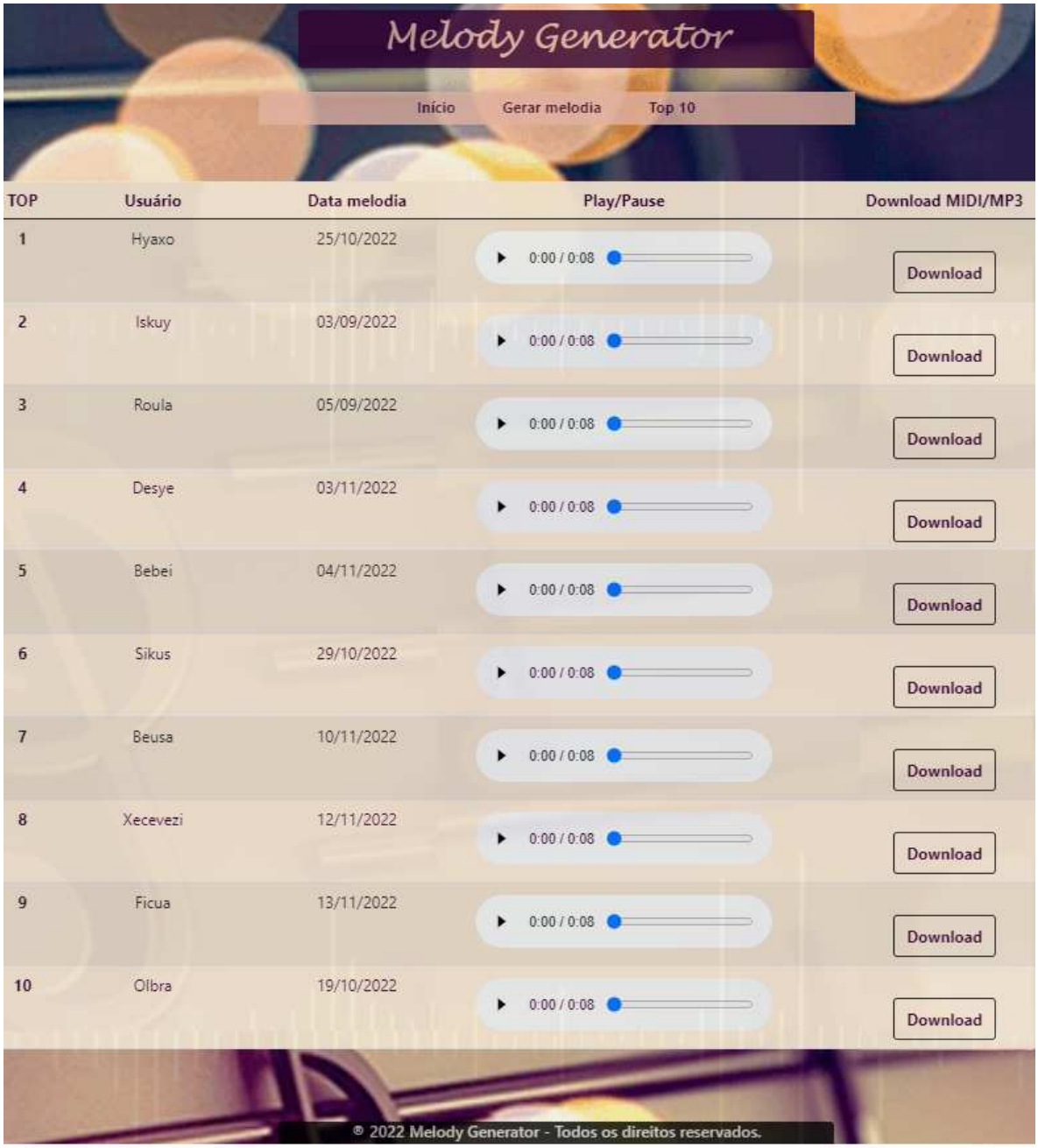
Fonte: Autoria própria (2022).

Na página da melodia gerada, é possível ouvir a melodia que foi gerada de acordo com os parâmetros selecionados na página anterior, sendo que o *player* de áudio da página contém as opções de iniciar ou pausar o áudio. Logo abaixo do *player*, é possível notar o botão "Download da melodia gerada", que se clicado disponibiliza o arquivo em formato MIDI para o usuário. O texto "Deixe seu Feedback" indica os campos que podem ser preenchidos para avaliar o quão agradável a melodia gerada ficou, permitindo informar uma nota de 1 à 5 para

a avaliação, bem como preencher o campo "Sugestão de melhoria" com algum ponto a ser considerado para as próximas gerações.

Enfim, na Figura 22, é apresentada a página referente ao menu Top 10, que é o resultado das 10 melodias mais bem avaliadas pelos usuários.

Figura 22 – Top 10



Fonte: Autoria própria (2022).

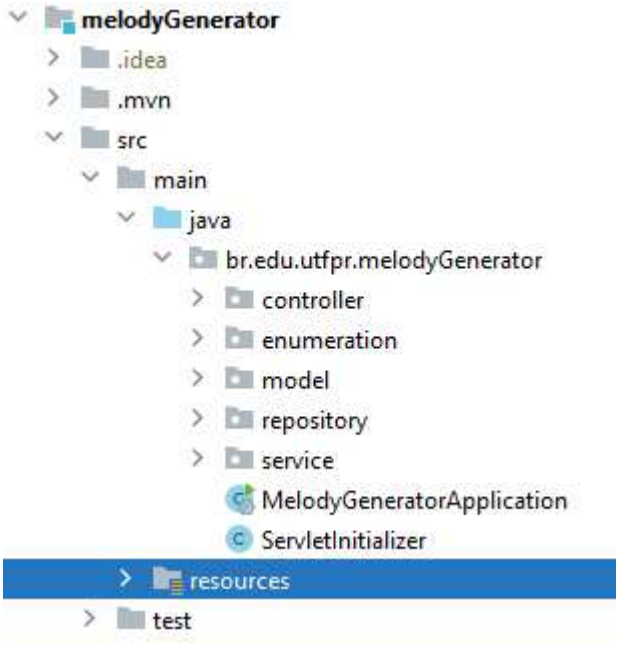
Nessa página, há a possibilidade de ser informado filtro "Nota Principal da melodia", que significa que o sistema irá buscar a melodia conforme a nota escolhida no filtro. É possível também ouvir a melodia na própria página, bem como fazer o download no formato MIDI da mesma.

Nesta seção, foi apresentado o sistema, bem como suas telas e suas respectivas funcionalidades.

5.3 IMPLEMENTAÇÃO DO SISTEMA

A implementação do sistema será apresentada a seguir destacando partes relevantes do código-fonte que foi desenvolvido, utilizando as tecnologias apresentadas na Tabela 4. O passo inicial, foi construir o banco de dados, utilizando o Sistema gerenciador PostgreSQL. Para o desenvolvimento do *back-end* do sistema foi utilizado a linguagem Java juntamente com o *framework* Spring. Conforme a Figura 23, é possível analisar a estrutura de pacotes do *back-end*.

Figura 23 – Disposição dos pacotes no back-end



Fonte: Autoria própria (2022).

Na Listagem 1, é apresentado o método "player", que é o método principal da classe MelodyParameterController.

Listagem 1 – Parte do código da classe MelodyParameterController

```
@Controller
@RequestMapping("generateMelody")
public class MelodyParameterController {
    @PostMapping
    public String player(@Valid MelodyParameter melodyParameter,
                         BindingResult result,
                         Model model,
                         RedirectAttributes attributes) {
        if (result.hasErrors()) {
            model.addAttribute("melodyParameter", melodyParameter);
            return "melodyParameter/form";
        }

        String scale = melodyParameter.getMain_note_scale() + " "
        + melodyParameter.getScale_type();

        RandomMelody randomMelody = new RandomMelody(
        new ArrayList<String>
                (Arrays.asList(RandomMelody.keys.get(scale))));

        randomMelody.setMainInstrumentName(melodyParameter
                                    .getInstrument_name());
        randomMelody.setOctaveInterval(melodyParameter
                                       .getQty_octaves());
        randomMelody.setBpm(melodyParameter.getBpm());

        generateMelodyService.createMelody(randomMelody);
        melodyParameterService.save(melodyParameter);

        GeneratedMelody generatedMelody = new GeneratedMelody();
        generatedMelody.setDate_hour(Timestamp
                                     .valueOf(LocalDateTime.now()));
        generatedMelody.setNotes_array(randomMelody.getMelody());
        generatedMelody.setMelody_id(melodyParameter
                                     .getMelody_parameters_id());
        generatedMelodyService.save(generatedMelody);

        melodyService.setRandomMelody(generateMelodyService
                                    .getRandomMelody());
        try {
            melodyService.exportMelody(System.
                                       getProperty("user.dir"),
            generatedMelody.getMelody_id().toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return "redirect:/generatedMelody";
    }
}
```

Fonte: Autoria própria (2022).

Na Listagem 1, a anotação @Controller indica que essa classe é responsável por receber requisições e enviar a resposta ao usuário com dados a serem exibidos pela *view*. Logo após, a anotação @RequestMapping indica que toda requisição feita para o *endpoint* "generateMelody", será tratada por essa classe. Já a anotação @PostMapping, descrita na linha 4 está indicando o verbo HTTP da requisição, ou seja, se enviada uma requisição *post* para o caminho da aplicação /generateMelody, será executado o método "player", que é responsável por receber os parâmetros informados pelo usuário através da *view* e processar a geração da melodia através do método "createMelody". Tal método está presente na classe "GenerateMelodyService" conforme representa a Listagem 2. Logo após, é possível notar o método "save" da classe MelodyParameterService, o qual é responsável por salvar no banco de dados a melodia que foi gerada. Na sequência, o método "exportMelody" salva a melodia gerada em uma pasta local do servidor, para logo após, ser disponibilizada para *download*. Os processos desses métodos citados serão descritos abaixo, bem como seus eventuais métodos ou classes descendentes.

Listagem 2 – Código da classe GenerateMelodyService

```
1  @Data
2  @AllArgsConstructor
3  @NoArgsConstructor
4  @Service
5  public class GenerateMelodyService {
6
7      RandomMelody randomMelody;
8
9      public RandomMelody createMelody(RandomMelody randomMelody){
10         randomMelody.setMelody("");
11
12         this.randomMelody = randomMelody;
13
14         final List<String> enumNames = Stream.of(Durations.values())
15                 .map(Durations::name)
16                 .collect(Collectors.toList());
17
18         float currentLength = 0; //melody length
19         while (currentLength < randomMelody.getLength()){
20
21             String note = randomMelody.getNotes().get((int)
22             (randomMelody.getNotes().size()*randomMelody
23             .getR().nextFloat()));
24
25             String durationName = enumNames.get((int)
26             (enumNames.size()*randomMelody.getR().nextFloat()));
27             float durationValue = Durations.valueOf(durationName)
28             .duration;
29
30             if (randomMelody.getOctaveInterval() > 1){
31                 int shift = (int)randomMelody.getOctaveInterval() /2;
32                 if (randomMelody.getR().nextFloat() <
33                             randomMelody.getOctaveShiftProbability())
34                     octave += -shift + randomMelody.getR()
35                             .nextInt(randomMelody.getOctaveInterval());
36             }
37
38             randomMelody.setMelody(randomMelody.getMelody()
39                             + note  + octave + durationName + " ");
40
41             currentLength += durationValue;
42         }
43
44         return randomMelody;
45     }
46 }
```

Fonte: Autoria própria (2022).

A Listagem 2 mostra o código da classe GenerateMelodyService, responsável por aplicar as regras para gerar a melodia. Na linha 1 a anotação @Data é utilizada para que os métodos *getters* e *setters* sejam gerados automaticamente, já a anotação @AllArgsConstructor serve para gerar um construtor com 1 parâmetro para cada atributo da classe, ao contrário da anotação @NoArgsConstructor que gera um construtor sem argumentos, e por último a anotação @Service para indicar que essa classe pertence à uma classe de serviço, ou seja, indicando que significa uma classe de serviço, possibilitando, a injeção de dependência em outras classes. Na sequência, é possível observar na linha 10 o início do método responsável por criar a melodia, logo no início do método a variável "enumNames" recebe os valores contidos no Enum Durations, um exemplo do código desse Enum pode se obter na Listagem 3.

Listagem 3 – Enum Durations

```
1
2 @AllArgsConstructor
3 public enum Durations {
4     w(1),
5     h(1f/2),
6     q(1f/4);
7
8     public final float duration;
9 }
```

Fonte: Autoria própria (2022).

Ainda conforme a Listagem 2, o sistema verifica o parâmetro de tamanho informado pelo usuário sendo esse o valor máximo de um loop para gerar as notas e suas respectivas durações dentro da melodia. A cada iteração do *loop*, é randomizada uma nota, desde que essa esteja dentro da escala pré definida conforme mostra o objeto "keys" da classe RandomMelody demonstrada na Listagem 4. A escala citada foi parametrizada da linha 17 à 19 da Listagem 1 de acordo com a nota principal informada pelo usuário como mostra o método "RandomMelody.keys.get(scale)" entre as linhas referenciadas. Logo após, ainda dentro do loop é obtido o parâmetro da quantidade de intervalos de oitavas que o usuário deseja na melodia. Ao fim de todas as iterações é possível obter um ArrayList do tipo *String* com as notas, durações e oitavas referente à melodia gerada.

Listagem 4 – Classe RandomMelody

```
1
2  @Data
3  @AllArgsConstructor
4  @NoArgsConstructor
5  public class RandomMelody {
6      private ArrayList<String> notes;
7      private String instrumentName = "";
8      private String melody;
9      private final Random r = new Random();
10     private float length;
11     private int bpm;
12     private int octaveInterval;
13     private float octaveShiftProbability;
14
15     public RandomMelody(final ArrayList<String> notes,
16                         final float length, final int bpm){
17         this.length = length;
18         this.bpm = bpm;
19         this.notes = (ArrayList<String>) notes.clone();
20     }
21
22 public static HashMap<String, String[]> keys =
23                     new HashMap<String, String[]>(){
24 {
25     put("C MAJOR", new String[]{"C","D","E","F","G","A","B"});
26     put("D MAJOR", new String[]{"D","E","F#","G","A","B","C#"});
27     put("E MAJOR", new String[]{"E","F#","G#","A","B","C#","D#"});
28     put("F MAJOR", new String[]{"F","G","A","Bb","C","D","E"});
29     put("G MAJOR", new String[]{"G","A","B","C","D","E","F#"});
30     put("A MAJOR", new String[]{"A","B","C","D","E","F#","G#"});
31     put("C MINOR", new String[]{"C","D","Eb","F","G","Ab","Bb"});
32     put("D MINOR", new String[]{"D","E","F","G","A","Bb","C"});
33     put("E MINOR", new String[]{"E","F#","G","A","B","C","D"});
34     put("F MINOR", new String[]{"F","G","Ab","Bb","C","Db","Eb"});
35     put("G MINOR", new String[]{"G","A","Bb","C","D","Eb","F"});
36     put("A MINOR", new String[]{"A","B","C","D","E","F","G"});
37     put("B MINOR", new String[]{"B","C#","D","E","F#","G","A"});
38     }
39 };
40 }
```

Fonte: Autoria própria (2022).

Na Listagem 4, é mostrado o código da Classe RandomMelody, que contém a parametrização do usuário referente a melodia.

Depois do método *createMelody()* apresentado na Listagem 1 e descrito acima, é possível observar o método *save()* da interface "CrudService", responsável por salvar no banco de dados a melodia que foi gerada. Um exemplo da interface pode ser visualizado na Listagem 5.

Listagem 5 – Interface CrudService

```
1
2  public interface CrudService <T, ID extends Serializable> {
3      List<T> findAll();
4
5      List<T> findAll(Sort sort);
6
7      Page<T> findAll(Pageable pageable);
8
9      T save(T entity);
10
11     T saveAndFlush(T entity);
12
13     Iterable<T> save(Iterable<T> iterable);
14
15     void flush();
16
17     T findOne(ID id);
18
19     boolean exists(ID id);
20
21     long count();
22
23     void delete(ID id);
24
25     void delete(T entity);
26
27     void delete(Iterable<T> iterable);
28
29     void deleteAll();
30 }
```

Fonte: Autoria própria (2022).

A Listagem 5 representa a interface que possui os métodos comumente utilizados nas classes *services*. Após isso, a Listagem 6, representa a classe abstrata CrudServiceImpl, que é responsável pela implementação padronizada de vários métodos, designando as operações de banco de dados para as classes *repository*.

Listagem 6 – Classe abstrata CrudServiceImpl

```
1
2  public abstract class CrudServiceImpl<T, ID extends Serializable>
3  implements CrudService<T, ID> {
4
5      protected abstract JpaRepository<T, ID> getRepository();
6
7      @Override
8      @Transactional(readOnly = true)
9      public List<T> findAll() {
10         return getRepository().findAll();
11     }
12
13      @Override
14     @Transactional //(readOnly = false)
15     public T save(T entity) {
16         return getRepository().save(entity);
17     }
18
19     @Override
20     @Transactional(readOnly = true)
21     public T findOne(ID id) {
22         return getRepository().getOne(id);
23     }
24
25     @Override
26     @Transactional(readOnly = true)
27     public boolean exists(ID id) {
28         return getRepository().existsById(id);
29     }
30
31     @Override
32     @Transactional(readOnly = true)
33     public long count() {
34         return getRepository().count();
35     }
36
37     @Override
38     @Transactional
39     public void delete(ID id) {
40         getRepository().deleteById(id);
41     }
42
43     @Override
44     @Transactional
45     public void delete(T entity) {
46         getRepository().delete(entity);
47     }
48 }
```

Fonte: Autoria própria (2022).

Já o repository é obtido a partir do método getRepository(), que é implementado nas classes que estendem a CrudServiceImpl. O método save(), possui a anotação @Transactional, responsável por fazer o controle transacional.

O exemplo de código da Listagem 7, apresenta a classe *model* do sistema, onde pertencem as entidades da aplicação.

Listagem 7 – Classe MelodyParameter

```
1
2  @Entity(name = "melodyparameter")
3  @Data
4  @AllArgsConstructor
5  @NoArgsConstructor
6  @EqualsAndHashCode(of = "melody_parameters_id")
7  public class MelodyParameter implements Serializable {
8      private static final long serialVersionUID = 1L;
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Long melody_parameters_id;
13
14     @Column(nullable = false)
15     private String instrument_name;
16
17     @Column(nullable = false)
18     private int qty_octaves;
19
20     @Column(nullable = false)
21     private int bpm;
22
23     @Column(length = 10, nullable = false)
24     private String scale_type;
25
26     @Column(length = 2, nullable = false)
27     private String main_note_scale;
28
29     @Column(nullable = false)
30     private int qty_repetitions;
31
32     @Column(nullable = false)
33     private String username;
34 }
```

Fonte: Autoria própria (2022).

Conforme o código descrito acima, a anotação @Entity é usada para informar que a classe também é uma entidade do banco de dados, informando também entre parênteses, o nome da entidade. A classe MelodyParameter, contém todos os atributos que serão salvos no banco de dados, indicando o seu tipo e se aceita valores nulos. No caso da linha 10, a anotação @Id, significa que o campo melody_parameters_id é um campo chave na tabela do banco de

dados. A anotação @GeneratedValue contida na linha 11, informa para que seja gerada uma chave única para cada registro salvo no banco de dados.

Por fim, depois de gerada a melodia, dando sequência na explicação da Listagem 1, a melodia gerada é exportada para que seja possível a disponibilização da mesma para o *download*.

5.4 Disponibilização do Código-fonte e algumas melodias geradas

O código-fonte pode ser obtido por meio do link do *github*: https://github.com/GobettiIgor/melodyGenerator.git, bem como alguns exemplos de melodias que foram geradas estão presentes no caminho do *github*: "melodyGenerator/src/main/resources/temp/".

6 CONCLUSÃO

Este trabalho apresentou uma proposta e a implementação de um sistema *web* responsivo para gerar melodias proceduralmente, ou seja, ser possível a geração de melodias baseando-se em diferentes heurísticas. O projeto, a modelagem, o desenvolvimento e a implementação do sistema foram processos que fizeram parte deste trabalho para se obter o resultado aqui apresentado. A IDE utilizada foi o *Intellij*, possibilitando um desenvolvimento facilitado, considerando que a interface é bem interativa. Aplicando as possibilidades das dinâmicas da biblioteca *JFugue*, foi possível notar que melodias agradáveis podem sim ser geradas proceduralmente, como também pode ser encontrado um resultado não satisfatório, já que a geração é parcialmente randômica. Ainda se tratando da biblioteca *JFugue*, é possível ir além do que foi desenvolvido nesse trabalho, pois há uma grande variedade de recursos da biblioteca que não foram utilizados no *Melody Generator*.

Neste trabalho, foi aplicado o conceito cliente/servidor, que consiste em uma estrutura de aplicação distribuída, sendo que, o processamento das informações é dividido em módulos ou processos distintos, tornando o sistema mais leve e rápido, devido à essa distribuição de cargas de trabalho.

As dificuldades encontradas durante o desenvolvimento do sistema, foram: a primeira vez da utilização da biblioteca *JFugue*, que foi superada através da consulta e do estudo da documentação e o desconhecimento de como reproduzir uma melodia por meio do *front-end*, que também foi superada através da pesquisa documental.

O sistema desenvolvido apresentou um significado relevante para a geração de melodias de forma computacional, considerando que a melodia é gerada por intermédio da parametrização, ou seja, de acordo com o parâmetros desejados e definidos pelos usuários. Também, por meio do *feedback* dos usuários, além de poder melhorar as futuras gerações, é possível classificar as 10 melodias mais bem avaliadas de acordo com o critério de nota de avaliação. No caso de cogitar a divulgação da melodia gerada, é possível obter o arquivo no formato MIDI para *download*, seja para o compartilhamento em redes sociais, o emprego da melodia em jogos, músicas e/ou até mesmo a livre utilização do usuário. Outro fator importante desse trabalho, é que, segundo a percepção do autor, foi possível notar que as melodias geradas com o parâmetro do tipo da escala "Maior" ficam mais agradáveis aos ouvidos humanos.

As telas que foram desenvolvidas, apresentaram um resultado satisfatório em relação à resposta de interatividade, já que as tecnologias que foram utilizadas já estão consolidadas no mercado e possuem código aberto, ou seja, livre de licenças. O conjunto das tecnologias/ferramentas utilizadas conforme a Tabela 4, permitiram o resultado eficaz deste sistema, atingindo os objetivos almejados.

A percepção que norteia, é que os recursos podem ir além do que foi desenvolvido nessa proposta. Sendo assim, fica a possibilidade de implementações para trabalhos futuros, há alguns recursos que podem ser implementados, a fim de melhorar o sistema *Melody Generator*,

tais como a implementação da geração de ritmos e vozes podendo gerar uma música completa de fato, ao invés de somente uma simples melodia. Considerando esses pontos, as sugestões dos trabalhos futuros são: a análise dos *feedbacks* para a melhoria das futuras gerações, a conversão das melodias geradas em mais formatos, tal como o MP3, a possível geração de uma música completa incluindo melodia, ritmo e voz, a possibilidade de compartilhar a melodia gerada pelas redes sociais e também a possibilidade de fazer uma análise de imersão, ou seja, o quanto aquele usuário imergiu com a melodia gerada.

REFERÊNCIAS

ANDRADE, M. de. **Pequena história da música**. [*S.l.*]: Nova Fronteira, 2015.

BENFICA, A. L. F. A influÊncia da mÚsica na mente e no corpo. 2021. Disponível em: https://escola.britannica.com.br/artigo/m%C3%BAsica/481991. Acesso em: 12 aug. 2021.

BOOMY. **Boomy**. 2022. Url: https://boomy.com/.

BRITANNICA, E. Música. 2021. Disponível em: https://escola.britannica.com.br/artigo/musica/481991. Acesso em: 12 aug. 2021.

BROWN, U. V. S. **Music and Manipulation. On the Social Uses and Social Control of Music**. Berghahn Books, 2006. ISBN 9781845450984. Disponível em: https://www.amazon.com.br/Music-Manipulation-Social-Uses-Control/dp/1845450981.

CAMPADELLO, P. **Musicoterapia na autocura**. Maltese, 1995. ISBN 9788571804968. Disponível em: https://www.amazon.com.br/musicoterapia-na-autocura-pier-campadello/dp/8571804966.

DOORNBUSCH, P. Computer sound synthesis in 1951: The music of csirac. **Computer Music Journal**, MIT Press, v. 28, n. 1, p. 10–25, 2004.

DOPELOOP. **Dope Loope Melody Generator**. 2022. Url: https://dopeloop.ai/melody-generator/?s=5964495031391033.

DOURADO, H. **Dicionário de termos e expressões da música**. Editora 34, 2004. ISBN 9788573262940. Disponível em: https://books.google.com.br/books?id=cL6zQ9vAUwkC.

FREGTMAN, C. D. **Corpo, Música E Terapia**. APGIQ, 1996. ISBN 9788585886021. Disponível em: https://www.amazon.com.br/Teoria-M%C3%BAsica-Bohumil-Med/dp/8570920393.

GAMA, M. A teoria das inteligências múltiplas e suas implicações para educação. **Página integrante do site Psy_coterapeutas on line www. homemdemello. com. br/psicologia/intelmult. html**, 1998.

GARDNER, H. **Estruturas da Mente - A teoria das inteligências múltiplas**. penso, 1994. ISBN 9788573073461. Disponível em: https://www.amazon.com.br/Estruturas-Mente-Howard-Gardner/dp/8573073462.

GRILLO, L. R. P. M. L. **A FISICA NA MUSICA**. EDUERJ, 2013. ISBN 9788575112809. Disponível em: https://www.travessa.com.br/a-fisica-na-musica-1-ed-2013/artigo/df009799-c26f-4fe5-9e75-5cdf11fb21eb.

JOHNSON, C. *et al.* **Evolutionary and Biologically Inspired Music, Sound, Art and Design**. [*S.l.*]: Springer, 2015.

JUSLIN, P. N.; LINDSTRÖM, E. Musical expression of emotions: Modelling listeners' judgements of composed and performed features. **Music Analysis**, Wiley Online Library, v. 29, n. 1-3, p. 334–364, 2010.

KAROLYI, O. **Introducao a Música**. Martins Fontes - Selo Martins, 2002. ISBN 9788533615434. Disponível em: https://www.amazon.com.br/Introducao-Musica-Otto-Karolyi/dp/8533615434.

MADEIRA, W. A. e Eduardo Aranha e C. Geração procedural de conteúdo aplicada a jogos digitais educacionais. **Jornada de Atualização em Informática na Educação**, v. 7, n. 1, p. 1–20, 2018. ISSN 23167734. Disponível em: https://www.br-ie.org/pub/index.php/pie/article/view/7857.

MED, B. **Teoria da música**. [*S.l.*]: Brasília: Musimed, 1996. v. 996.

MENEZES, P. H. G. de. **Geração Procedural de Conteúdo para Jogos de Realidade Aumentada**. 2018.

MONTALVÃO, E. d. S. R. **Uma proposta de FSS fractal com geometria simplificada**. 2010. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2010.

PEREIRA, C. S. *et al.* Music and emotions in the brain: familiarity matters. **PloS one**, Public Library of Science, v. 6, n. 11, p. e27241, 2011.

PRIOLLI, M. L. de M. **Princípios Básicos da Música para Juventude**. Irmãos Vitale, 2021. ISBN 9786557580073. Disponível em: https://www.amazon.com.br/Princ%C3%ADpios-B%C3%A1sicos-M%C3%BAsica-para-Juventude/dp/6557580078/ref=asc_df_6557580078/?tag=googleshopp00-20&linkCode=df0&hvadid=379739168605&hvpos=&hvnetw=g&hvrand=18129906570971242724&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1031887&hvtargid=pla-1363460203869&psc=1.

RATTON, M. **Escalas musicais-quando a matemática rege a música**. 2006.

SADIE, S. **Dicionário Grove de Música**. Concisa, 1994. ISBN 9788571103016. Disponível em: https://www.skoob.com.br/dicionario-grove-de-musica-136482ed151753.html.

SCIREA, M. *et al.* Smug: Scientific music generator. *In*: **ICCC**. [*S.l.*: *s.n.*], 2015. p. 204–211.

SEKEFF, M. de L. **Da música: Seus usos e recursos**. Editora Unesp, 2007. ISBN 9788571397682. Disponível em: https://www.amazon.com.br/m%C3%BAsica-Maria-Lourdes-Sekeff/dp/8571397686.

SHAKER, N.; TOGELIUS, J.; NELSON, M. J. **Procedural content generation in games**. [*S.l.*]: Springer, 2016.

SHNEIDERMAN, B.; PREECE, J.; PIROLLI, P. Realizing the value of social media requires innovative computing research. **Communications of the ACM**, ACM New York, NY, USA, v. 54, n. 9, p. 34–37, 2011.

SILVEIRA, D. S. *et al.* Uef-web: framework de apoio à engenharia de usabilidade para aplicações web. Universidade Federal de Sergipe, 2015.

SMITH, G. An analog history of procedural content generation. *In*: **FDG**. [*S.l.*: *s.n.*], 2015.

STOCHASTIC, M. **Melisma Stochastic Melody Generator**. 2022. Url : https://www.link.cs.cmu.edu/melody-generator/.

TAME, D. **O poder Oculto da Música**. Cultrix, 1984. ISBN 9788531603020. Disponível em: https://www.amazon.com.br/Poder-Oculto-M%C3%BAsica-David-Tame/dp/8531603021.

TOGELIUS, J. *et al.* What is procedural content generation? mario on the borderline. *In*: **Proceedings of the 2nd international workshop on procedural content generation in games**. [*S.l.*: *s.n.*], 2011. p. 1–6.

WOLFRAMTONES. **WolframTones**. 2022. Url: https://tones.wolfram.com/generate/.