UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Júlio César de Sousa

T.R.E.N.S: Uma Arquitetura Web Colaborativa com base em LMMS e Computação na Borda para a Cultura Underground

São João Del-Rei

2026

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Júlio César de Sousa

T.R.E.N.S: Uma Arquitetura Web Colaborativa com base em LMMS e Computação na Borda para a Cultura Underground

Dissertação apresentada como requisito para obtenção do título de Mestre em Ciência da Computação do curso de mestrado do Programa de Pós Graduação em Ciência da Computação (PPGCC) da Universidade Federal De São João Del-Rei (UFSJ).

Orientador: Dr. Flávio Luiz Schiavoni

Universidade Federal De São João Del-Rei

Mestrado em Ciência da Computação

São João Del-Rei

2026

Júlio César de Sousa

T.R.E.N.S: Uma Arquitetura Web Colaborativa com base em LMMS e Computação na Borda para a Cultura Underground

Dissertação apresentada como requisito para obtenção do título de Mestre em Ciência da Computação do curso de mestrado do Programa de Pós Graduação em Ciência da Computação (PPGCC) da Universidade Federal De São João Del-Rei (UFSJ).

Trabalho apresentado em São João Del-Rei, 10 de Abril de 2026:

Dr. Flávio Luiz Schiavoni
Orientador

Dr. Daniel Ludovico Guidoni
Convidado 1

Dr. Edimilson Batista dos Santos
Convidado 2

Dr. Rômulo Augusto Vieira Costa
Convidado 3

São João Del-Rei
2026

Agradecimentos

Aos amigos que dividiram comigo o beat frenético do RAP e o peso dos desafios acadêmicos, meu muito obrigado, do fundo da alma. Vocês não foram apenas espectadores, foram o combustível de cada conquista. Cada rima que escrevi e cada verso que entoei carregam um pedaço das nossas vivências. A saudade de vocês ecoa em cada nota dessa trajetória que hoje se encerra no papel, mas continua viva na nossa história.

Minha segunda casa, meu porto seguro em São João del-Rei. Quando o mundo lá fora parecia pesado demais, a Xilau me estendeu a mão. Vocês foram família quando eu mais precisei, oferecendo o calor e a compreensão que me permitiram mudar de vida. Cada risada no sofá, cada perrengue superado no coletivo... tudo isso virou o alicerce de quem eu sou hoje. Sou eternamente grato pela receptividade e por terem feito de mim um de vocês.

Aos amigos do dia a dia, que tornaram os corredores e as madrugadas de estudo menos solitários, esse passo não é só em busca de um título, é o retrato de anos de amadurecimento ao lado de vocês. Um salve especial à galera do ALICE. Vocês abriram meus olhos para uma nova dimensão da produção musical. Conviver com tantos artistas gigantes nesse laboratório moldou meu percurso de um jeito que eu nunca imaginei. Ao Flavio, obrigado por me abraçar, acreditar no meu potencial e viabilizar essa pesquisa. Você ajudou a realizar um sonho que parecia impossível.

Por fim, dedico todo o meu amor ao meu núcleo familiar e à minha namorada espetacular. Vocês foram o meu chão quando tudo parecia instável. O suporte incondicional de vocês me guiou por esse curso desafiador e me deu forças para não desistir. Obrigado por serem meu cais, minha fonte de amor e minha maior motivação. Amo vocês profundamente.

Não posso fechar esse ciclo sem reconhecer os departamentos pelos quais eu passei, DCOMP, PPGCC e as instituições que acreditaram na ciência e na educação. Meu muito obrigado à FAPEMIG, CAPES, CNPq, PROPE, PROAE, PROEX. Cada auxílio, cada bolsa e cada suporte institucional foram os degraus que me trouxeram até aqui.

Este é apenas o começo de uma caminhada que agora transborda ainda mais possibilidades. Espero, de coração, conseguir retribuir à altura tudo o que recebi de cada um de vocês.

A jornada continua.

"Eu faço RAP, me liberto, eu não fico mudo.
Porque nós, é tiro no escuro
Sem nenhum holofote
É Hip Hop, ChinaTown, direto do Vetor Norte.".
(Bernardi MC)

Resumo

A produção musical digital contemporânea é fortemente marcada pela exclusão mercadológica imposta pelo alto valor das licenças de Estações de Trabalho de Áudio Digital (*Digital Audio Workstations* ou DAWs) proprietárias, o que cria barreiras estruturais severas para produtores independentes. Neste contexto, este trabalho tem como objetivo geral propor, desenvolver e validar o ecossistema T.R.E.N.S. (Tecnologias em Rede para Emancipação e Narrativas Sonoras), uma arquitetura *web* colaborativa baseada no *software* livre LMMS e na Computação na Borda. A pesquisa materializa uma plataforma de aprendizagem e criação através das ferramentas `mmpSearch` e `mmpCreator`, desenhada especificamente para a cultura *underground* e a prática colaborativa *Do It With Others* (DIWO). O estudo é guiado por duas hipóteses centrais. A primeira postula que a transferência da renderização de áudio para o navegador do usuário subverte a dependência de infraestruturas robustas, pois mitiga a latência e viabiliza a colaboração musical contínua mesmo em redes instáveis. A segunda afirma que a indexação e a estruturação semântica de projetos musicais abertos operam como uma Zona de Desenvolvimento Proximal (ZDP), viabilizando o letramento musical ao permitir que produtores iniciantes realizem a engenharia reversa cognitiva de arranjos complexos. A avaliação experimental, de caráter quantitativo e qualitativo, validou as premissas da arquitetura proposta. No *backend*, a estratégia de paralelismo de dados alcançou a ingestão e indexação massiva de projetos, provando resiliência técnica frente a gargalos de entrada e saída (I/O) que emulam as condições restritas de iniciativas independentes. No *frontend*, a arquitetura na borda garantiu latências operacionais entre 10 e 15 milissegundos, preservando o *groove* e a precisão micro temporal indispensáveis na performance do *Hip Hop*. Somado a isso, a mineração sociotécnica dos metadados extraídos comprovou a Estética da Escassez, revelando adaptações criativas da comunidade, como a priorização de sintetizadores nativos leves e o uso da escala de Lá Menor para composição em teclados comuns de computador, além de classificar automaticamente os projetos por níveis pedagógicos. Conclui-se que a plataforma atua com eficácia como um mecanismo de emancipação digital, transformando repositórios estáticos em ambientes vivos de letramento, autonomia criativa e resistência sociocultural.

Palavras-chave: Produção Musical; Computação na Borda; Sistemas Distribuídos; Letramento Musical Digital; Cultura *Underground*.

Abstract

Contemporary digital music production is heavily marked by the market exclusion imposed by the high cost of proprietary *Digital Audio Workstations* (DAWs) licenses, which creates severe structural barriers for independent producers. In this context, this work aims to propose, develop, and validate the T.R.E.N.S. (Network Technologies for Emancipation and Sonic Narratives) ecosystem, a collaborative *web* architecture based on the free *software* LMMS and *Edge Computing*. The research materializes a learning and creation platform through the `mmpSearch` and `mmpCreator` tools, specifically designed for the *underground* culture and the *Do It With Others* (DIWO) collaborative practice. The study is guided by two central hypotheses. The first postulates that transferring audio rendering to the user's browser subverts the dependence on robust infrastructures, as it mitigates latency and enables continuous musical collaboration even on unstable networks. The second asserts that the indexing and semantic structuring of open musical projects operate as a Zone of Proximal Development (ZPD), enabling musical literacy by allowing beginner producers to perform cognitive reverse engineering of complex arrangements. The experimental evaluation, of a quantitative and qualitative nature, validated the premises of the proposed architecture. In the *backend*, the data parallelism strategy achieved the massive ingestion and indexing of projects, proving technical resilience against *I/O* bottlenecks that emulate the restricted conditions of independent initiatives. In the *frontend*, the edge architecture guaranteed operational latencies between 10 and 15 milliseconds, preserving the *groove* and the microtemporal precision indispensable in *Hip Hop* performance. In addition, the sociotechnical mining of the extracted metadata proved the Aesthetics of Scarcity, revealing creative adaptations of the community, such as the prioritization of lightweight native synthesizers and the use of the A Minor scale for composition on common computer keyboards, besides automatically classifying projects by pedagogical levels. It is concluded that the platform acts effectively as a mechanism of digital emancipation, transforming static repositories into living environments of literacy, creative autonomy, and sociocultural resistance.

Keywords: Music Production; *Edge Computing*; Distributed Systems; Digital Music Literacy; *Underground* Culture.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACK	Acknowledgement, em português, Confirmação de recebimento.
ADSR	Attack, Decay, Sustain, Release. Parâmetros que definem a evolução temporal de um som (envelope).
ALICE	Arts Lab in Interfaces, Computers, and Everything Else (like Education, Exceptions, Experiences, Enterntainment, Environment, Entropy, Errors, and Etecetera ...).
ANPPOM	Associação Nacional de Pesquisa e Pós-Graduação em Música.
API	Application Programming Interface, em português, Interface de Programação de Aplicações.
BEAT	Faixa sonora instrumental, para que seja gravado e mixado uma canção de RAP.
BPM	Batidas por minuto.
CAP	Consistency, Availability, Partition Tolerance. Teorema que descreve as limitações de sistemas distribuídos.
CDE	Collaborative Development Environments, em português, Ambientes de Desenvolvimento Colaborativo.
CDN	Content Delivery Network, em português, Rede de Entrega de Conteúdo.
CLT	Consolidação das Leis do Trabalho.
CPU	Central Processing Unit, em português, Unidade Central de Processamento.
CSCW	Computer Supported Cooperative Work, em português, Trabalho Cooperativo Apoiado por Computador.
DAC	Digital Audio Converter, em português, Conversor de Áudio Digital. Pode ser considerado como um conversor de sinal digital para sinal analógico.
DAW	Digital Audio Workstation ou do Português, Estação de Trabalho de Áudio Digital.

DIY	"Do it yourself", ou "Faça você mesmo".
DJ	Disc Jockey - Tocador de discos.
DOM	Document Object Model, em português, Modelo de Objeto de Documento.
DSP	Digital Signal Processing, em português, Processamento Digital de Sinais.
EP	Extended Play. Faixas musicais maiores do que singles, porém menores do que um álbum, geralmente possuem de 4 à 7 faixas, com duração total de 30 minutos no máximo.
ETL	Extract, Transform, Load, em português, Extrair, Transformar e Carregar.
EWMA	Exponential Weighted Moving Average, em português, Média Móvel Exponencialmente Ponderada.
GIL	Global Interpreter Lock, bloqueio global do interpretador Python que limita a execução de threads.
HDD	Hard Disk Drive, em português, Disco Rígido.
HPC	High Performance Computing, em português, Computação de Alto Desempenho.
HTTP	Hypertext Transfer Protocol, protocolo de transferência de hipertexto.
I/O	Input/Output, em português, Entrada/Saída.
IA	Inteligência artificial.
IMD	Instrumentos Musicais Digitais.
IOS	Sistema operacional da Apple.
IP	Internet Protocol, em Português Protocolo da Internet.
JSON	JavaScript Object Notation, formato leve de intercâmbio de dados.
LDAP	Lightweight Directory Access Protocol, protocolo leve de acesso a diretório.
LMMS	Linux Multimedia Studio ou do Português, Estúdio Multimídia do Linux.

LP	Long-Play, disco de vinil de longa duração.
MC	Mestre de Cerimônia.
MIDI	Musical Instrument Digital Interface, em Português Interface Digital de Instrumento Musical.
MIR	Music Information Retrieval, em português, Recuperação de Informação Musical.
MMP	LMMS Project File, formato de arquivo de projeto nativo do software LMMS (baseado em XML).
MP3	MPEG Audio Layer III.
MPC	Music Production Controller.
PD	Pure Data.
PPGCC	Programa de Pós-Graduação em Ciência da Computação.
PWA	Progressive Web App, em português, Aplicação Web Progressiva.
RAM	Random Access Memory, memória de acesso aleatório.
RAP	Estilo de música derivado do Hip Hop, na verdade, um dos elementos do mesmo, o elemento da composição musical, expressão verbal, sua sigla no Inglês se dá por Rhythm and Poetry ou do Português, Ritmo e Poesia.
RTT	Round Trip Time, tempo de ida e volta de um sinal na rede.
SaaS	Software as a Service, em português, Software como Serviço.
SO	Sistema Operacional.
SSD	Solid State Drive, unidade de estado sólido de armazenamento.
TCC	Trabalho de conclusão de curso.
UDP	User Datagram Protocol, em Português Protocolo de Datagrama do Usuário.
UFSJ	Universidade Federal de São João del Rei.
UI	User Interface, em português, Interface de Usuário.
VST	Sigla para "Virtual Studio Technology", em português, Tecnologia de Estúdio Virtual.

Wasm	WebAssembly, formato de instrução binária para máquina virtual baseada em pilha.
WAV	Waveform Audio File Format, formato de arquivo de áudio padrão.
XML	Extensible Markup Language, em português "Linguagem de Marcação Extensível".
XSD	XML Schema Definition, definição de esquema XML.

Sumário

Prefácio

Toda pesquisa nasce de uma inquietação. Esta aqui nasce do incômodo com a exclusão, com o silêncio imposto às vozes que gritam nas margens da cidade, nos vãos da tecnologia e nos becos da arte. Com a falta da possibilidade de fazer, mesmo tendo a letra e tendo toda a vontade. Não é apenas um trabalho acadêmico moldado para ser aprovado, na verdade é um tributo, uma tentativa de agradecer ao *Hip Hop* por tudo que ele fez por mim.

A proposta se baseia em uma abordagem interdisciplinar, que transita entre os campos da Computação, Música, Educação e Gratidão. A partir desse encontro, surgem novas possibilidades de formação, criação e transformação social. Produzir música com *software* livre[1] é mais do que uma escolha técnica. É uma postura política frente à exclusão tecnológica e social, uma estratégia educativa para formação crítica, e uma ação cultural voltada para a valorização de narrativas marginais e autenticamente brasileiras.

A proposta não busca "levar" tecnologia para a periferia ou pro movimento cultural *underground*. Porque ela já está lá. Não busca dizer o que é certo ou o que é errado,"como fazer" ou "o que fazer". A ideia aqui é outra, queremos **reconhecer**, **fomentar**, **potencializar** e **respeitar**. Colocar a tecnologia a serviço de quem já cria, inventa e transforma.

Neste trabalho, apresentamos um espaço de aprendizagem, criação e distribuição[2] musical pensado a partir das práticas do universo *underground*, especialmente do Rap[3]. Um ambiente que compreenda as particularidades de *hardware* e conectividade de quem produz à margem, e por isso mesmo, cria no centro da potência criativa. Faz da arte a válvula de escape, o instrumento de sobrevivência.

> "O compromisso é como o povo, é o X da questão
> É o resgate do ladrão, a música do irmão
> Rap é o som e tenho o dom da imaginação
> ...
> Trilha sonora e facção
> VPN rapnacional.com" (SABOTAGE RAPPIN' HOOD, 2002)

[1] Podemos considerar como qualquer programa de computador que garante aos usuários quatro liberdades fundamentais: 1) executar para qualquer propósito; 2) estudar e modificar o código-fonte; 3) redistribuir cópias; e 4) sugerir melhorias, compartilhando as versões modificadas.

[2] Neste caso, estamos falando sobre a distribuição do conhecimento musical, para além da própria obra. Pois o compartilhamento de projetos, aliados a uma comunidade colaborativa traz pontos de vista e discussões que podem fomentar a criatividade, ampliando os horizontes criativos dos artistas.

[3] Do inglês *Rhythm And Poetry* ou, do português, ritmo e poesia.

Motivação Pessoal e Contexto da Pesquisa

Júlio César de Sousa (Figura 1), brasileiro nato, nascido na Zona Leste de Belo Horizonte - MG, no ano de 1996, atleticano fanático. MC (Mestre de Cerimônia), *beatmaker*, produtor musical, compositor, entusiasta da arte de rua. Cientista da Computação, formado pela UFSJ (Universidade Federal de São João del-Rei) em 2023.

Figura 1 – Foto de 2016, quando tive a honra de me apresentar na escola em que estudei da sexta à oitava série do ensino fundamental. Escola Estadual Cecília Dolabela Portela Azeredo, em Lagoa Santa - MG.

Fonte: O Autor (2016).

Meu primeiro contato com Rap veio por volta dos 10 anos de idade, em meados de 2006. Em um dos diversos engarrafamentos na rodovia da morte (BR-381), estávamos com dificuldade para sintonizar o rádio, no carro, mas em algum momento conseguimos pegar uma estação amadora. Era em AM, a qualidade não era das melhores, mas dava pra ouvir uma música quase falada, diferente do que eu já tinha escutado, com muitas palavras e assuntos que eu já tinha ouvido na escola, fiquei prestando atenção e aguardei até que falassem o nome da música na rádio, visto que minha família não ouvia esse gênero musical. Até que disseram: Racionais MC's (Figura 2) - Homem na Estrada (Racionais MCs, 1993).

Já tinha acesso à *internet* nesta época, então procurei por mais MP3[4] desse tal de Racionais e fui lendo as letras das músicas enquanto ouvia diversas vezes. Tentando compreender porque tinha tanta gente ruim, tanta situação ruim, eu ainda não entendia

[4] *MPEG Audio Layer* III. É um formato de arquivo digital popular para armazenamento de áudio e tamanho consideravelmente leve (em relação ao espaço em disco), se comparado com outros formatos de arquivos deste tipo.

Figura 2 – Racionais MC's. Da esquerda pra direita, Mano Brown, KL Jay, Edi Rock, Ice Blue. Em 2023, receberam o título de "Doutor Honoris Causa" pela Universidade Estadual de Campinas (Unicamp).

Fonte: <https://jornal.usp.br/wp-content/uploads/2023/07/20230731_racionais_1200x630.jpg>. Acesso em 12 Fev. 2026.

sobre classes sociais, desvios de dinheiro público ou racismo. Fui mergulhando cada vez mais neste universo, conhecendo mais grupos como 509-E, RZO, Realidade Cruel, Facção Central, SNJ, dentre tantas outras lendas do nosso Rap nacional. Quando fui para a sexta série, minha família resolveu se mudar para Lagoa Santa, na região metropolitana de Belo Horizonte - MG. Ao fazer novos amigos e estabelecer novos laços, na nova cidade, uma das coisas que me auxiliou na aproximação foi esse contato com o Rap, fiz diversos irmãos que levo comigo pra sempre enquanto ouvíamos música na hora do recreio com as caixinhas de som do PC (*Personal Computer*) que eu pegava em casa e levava pra escola.

Durante idas e vindas, este trabalho parte de uma vivência concreta, a trajetória de um artista independente no cenário do Rap nacional. Iniciado pelo embalo da revolta e rebeldia jovem, em 2014, o nascimento do grupo *ChinaTown MC's* (Figura 3), foi um divisor de águas na minha vida. O amadurecimento e compreensão sobre as responsabilidades da vida que eu pude absorver durante os 3 anos em que participei ativamente do grupo, foram cruciais para a minha formação como pessoa. Cada furada, cada aperto, cada sufoco, tudo isso valeu a pena para que hoje eu possa estar programando e escrevendo sobre.

> "ahhhh
> calangar valeu a pena
> bolso furado, é sem dilema
> 4 magrin lombrando a cena
> roubando a cena
> lombrando a cena". *Queria poder citar, mas infelizmente nunca foi lançada. Chama-se Palestina, ChinaTown MC's e é de 2015.*

Após mais uma mudança, agora saindo de Lagoa Santa - MG e toda movimentação noturna, por casas de *show*, festas e agitação caótica, diretamente para a Zona

Figura 3 – Foto de uma apresentação do ChinaTown MC's em 2015 no Cravo Vermelho, em Sabará - MG. Na foto, da esquerda para a direita, Dj Tuts, JotaChina, Villaz MC, Kash (do grupo Outlaws, também de Lagoa Santa - MG) e Bernardi MC.

Fonte: O Autor (2015).

Rural de Coronel Xavier Chaves, uma cidadezinha pacata no interior de Minas Gerais, com aproximadamente 3100 habitantes, próxima a histórica São João del Rei, onde eu resido atualmente - olha só, mais uma mudança. Entretanto, desde 2017 sigo com meu trabalho independente, onde tive que aprender na marra como ser, de fato, um artista independente. Um pouco mais dessa transição é discutida em (SOUSA, 2023), (SOUSA; SCHIAVONI, 2024), dentre outros[5]. Em resumo, não tinha condições financeiras de pedir que me produzissem e, muito menos, tinha os meios de fazer as gravações para encaminhá-las a alguém que pudesse produzir, pois no início nem internet tínhamos lá em casa. Era só [6,7] na TV e dados móveis, com o celular estático em um ponto específico ou subindo o pasto. Além disso, muito trabalho.

Todas as oportunidades que apareciam, eram aceitas e iam desde *freelancer* de garçom, limpeza e bilheteiro, servente, auxiliar de carpinteiro e pintor. Calouro da República Xilau[8] (Figura 5) a estagiário na Cacel[9]. Monitor de ISLD[10] a oficinas de produções

[5] Todos os artigos e eventos que participei estão disponíveis no link <https://buscatextual.cnpq.br/buscatextual/visualizacv.do;jsessionid=2FAD461AC3A94BA48F0A3A77D1941979.buscatextual_0>. Acesso em 20 Mar de 2026.

[6] Record

[7] A *Record* é conhecida por ser uma rede de televisão comercial aberta brasileira. Focada em assuntos voltados à religião e programas policiais, como o "Balanço Geral".

[8] A República Xilau, é considerada uma república tradicional e foi fundada em 2012. Segue até hoje em São João del Rei e ser desse meio requer muito trabalho, é preservar uma história e ser a família de pessoas que vão viver os ditos "melhores anos" de suas vidas junto com você. Vão aprender e amadurecer, compreender situações da vida, vão ser literalmente a sua família, neste período e após ele.

[9] Grupo de concessionária de veículos Volkswagen, Chevrolet e BYD do Sul de Minas Gerais.

[10] Introdução a Sistemas Lógicos e Digitais. Matéria ministrada pela Dra. Milene Barbosa, docente que

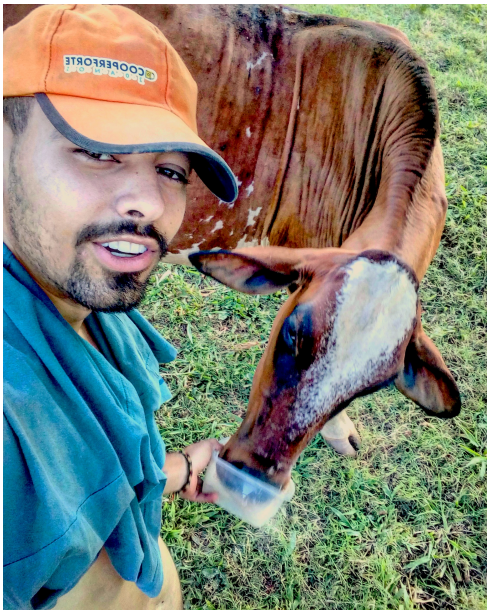

Figura 4 – *Selfie* tirada enquanto alimentava as vacas no pasto lá de casa.

Fonte: O Autor (2017).

musicais com *software* livre, elaboradas por mim. A esta altura, pesquisador e sonhador.

> "Sempre fui sonhador, é isso que me mantém vivo
> Quando pivete, meu sonho era ser jogador de futebol
> Vai vendo!
> Mas o sistema limita nossa vida de tal forma
> E tive que fazer minha escolha, sonhar ou sobreviver
> Os anos se passaram e eu fui me esquivando do circulo vicioso
> Porém o capitalismo me obrigou a ser bem sucedido
> Acredito que o sonho de todo pobre, é ser rico." (Racionais MC's; Afro-X, 2002)

Contudo, ainda em 2017, conheci um primo que não havia sido apresentado ainda e nos tornamos muito próximos, tão próximos que somos do mesmo laboratório de pesquisa. Emanuel "Rinba", obrigado! Conseguimos emplacar este artigo (SOUSA; SOUSA; SCHIAVONI, 2023) no 8º Congresso Internacional de Arte, Ciência e Tecnologia e Seminário de Artes Digitais 2023 (CIACT-SAD 08), em Belo Horizonte - MG (Figura 6). Foi meu braço direito durante todo o tempo, me apoiando nas produções musicais, opinando nos *beats*, videoclipes, você foi incrível! Este trabalho não seria o mesmo sem a sua força, obrigado.

Lançamos nosso próprio selo musical (InCubv Prxd.) junto com 2 primos e meu irmão do meio. Aprendemos a captar o áudio, comecei com um microfone de lapela, depois

tenho muita admiração e gratidão.

Figura 5 – Foto do 9º encontro de ex-moradores da República Xilau.

Fonte: O Autor (2021).

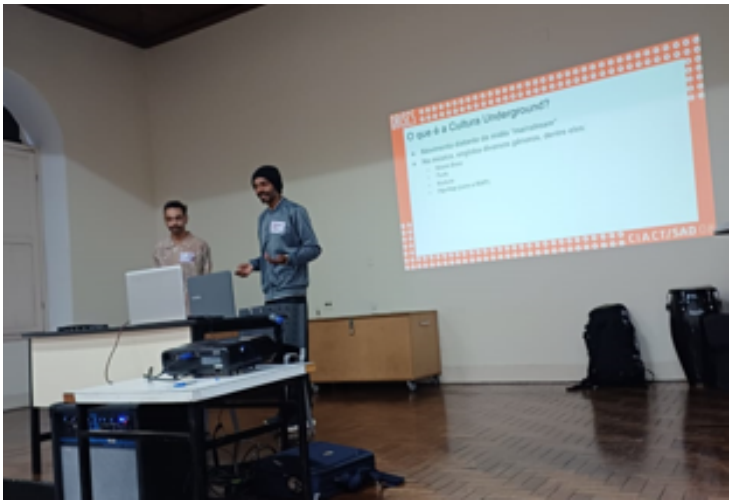

Figura 6 – Foto da apresentação de um artigo nosso no CIACT-SAD (Congresso Internacional de Arte, Ciência e Tecnologia e Seminário de Artes Digitais) 2023 em Belo Horizonte - MG.

Fonte: O Autor (2023).

comprei um de R$30,00[11] (Figura 7). Com ele gravei 7 *singles*[12] e 1 EP[13] (*Extended Play*) com 6 faixas entre 2017 e 2025, quando comprei um BM800 e uma interface TEYUN Q-12. Sim, nenhuma dessas faixas foi gravada com mesa de som ou interface de áudio,

[11] Que me acompanha até hoje.
[12] Disponíveis em: <https://audio.alice.ufsj.edu.br/@jotachina>
[13] Disponível em: <https://audio.alice.ufsj.edu.br/channels/pensamentosLentos>

foram gravadas com microfone ligado no P2 da placa de áudio *onboard*[14] do PC e *pop filter*[15] de pano, vide Figura 7.

Figura 7 – Foto de uma gravação feita em 2018 no meu *Homestudio* na roça.

Fonte: O Autor (2018).

Ainda durante o ano de 2017 comecei um curso técnico de Informática na Escola Estadual Coronel Xavier Chaves e conheci o Professor Ricardo Torga, que ministrou algumas disciplinas do curso. Tivemos uma forte aproximação e ele foi um dos pilares para que eu me inscrevesse para o ENEM (Exame Nacional do Ensino Médio) de 2018. Em 2019 entrei no curso de Ciência da Computação na UFSJ. No início eu não compreendia muito o que o curso englobava e fui fazendo contato com veteranos, que me receberam muito bem e me auxiliaram muito.

Mas no Inverno Cultural da UFSJ de 2019 eu estava trabalhando no Mutirão Cultural que estava localizado no Fortim dos Emboabas, nesta ocasião, Michel era o diretor dessa área. Em algumas conversas falei sobre o Rap, que tinha vontade de me apresentar na cidade e etc. Até que em um dia, em uma canja pós expediente, surge Flávio Luiz Schiavoni, de quem eu já tinha ouvido falar e todos diziam que eu deveria conhecê-lo. Michel o conhecia e nos apresentou, papo vai, papo vem, ao me despedir o Flávio disse que era pra que eu aguardasse que um dia teríamos um trabalho juntos.

Em 2021, durante a pandemia do Covid-19 recebo o convite para fazer uma Iniciação Científica. A partir dai a minha graduação mudou completamente o sentido e eu pude ver como eu poderia contribuir de uma maneira única e com enorme potencial criativo. Iniciei no meio da pesquisa acadêmica e comecei a escrever artigos, participar de eventos (Figura 8), conhecer pessoas que estavam discutindo coisas parecidas com o que eu tinha em mente. Fui em eventos de diversos nichos e tive trocas espetaculares. Obrigado pela oportunidade.

[14] Placas *onboard* são soldadas diretamente na placa mãe do PC, geralmente são mais "fracas" que as placas *offboards*, que são voltadas para alta performance.

[15] Acessório essencial durante a captação de áudio em estúdios, colocado entre a boca e o microfone para

Figura 8 – Foto no SPACL (Seminário de Pesquisas em Artes, Cultura e Linguagens) na UFJF (Universidade Federal de Juiz de Fora) em 2023. Da esquerda para a direita, Flávio, Eu, João Lara e Cleisson. Este foi o primeiro evento acadêmico que me apresentei pessoalmente.

Fonte: O Autor (2023).

Foquei minha pesquisa na aprendizagem musical do artista independente. Juntando o saber teórico, adquirido durante a pesquisa no laboratório ALICE, ao saber prático, fruto da experiência no Rap e na cultura *underground*, comecei a fomentar oficinas de produção musical utilizando *software* livre.

Durante essas oficinas, utilizando a Estação de Trabalho de Áudio Digital LMMS (*Linux MultiMedia Studio*), pude observar de perto a "Estética da Escassez" operando na prática. Vi artistas driblando a falta de computadores potentes e a ausência de internet banda larga para conseguir expressar suas ideias sonoras. Ali, a tecnologia deixou de ser apenas um objeto de estudo e se revelou como um gargalo real para a emancipação daqueles produtores.

No momento em que escrevo esta dissertação, vivo o que poderíamos chamar de melhor momento. Estou em um relacionamento saudável, com uma mulher incrível. Tenho contato com amizades que são importantes pra mim e após anos difíceis, possuo uma rede de apoio mais estruturada. Minha mãe está bem e saudável, encontro-me em constante evolução profissional e acadêmica. O desenrolar deste trabalho é o fechamento de (mais) um ciclo. É a abertura de um novo mundo de possibilidades. É a transformação do marginal em um Mestre em Ciência da Computação. É o resultado direto de toda a vivência. Trata-se de retribuir ao *Hip Hop* o que ele me proporcionou. Se antes o foco era a capacitação pontual através de oficinas, agora a intenção é prover a infraestrutura tecnológica distribuída que permita o ensino e a produção de música de forma acessível,

não capturar sons característicos de palavras com a letra p, f e s, por exemplo.

resiliente e coletiva. Espero que este não seja a minha última, nem a minha melhor e maior contribuição[16].

[16] Neste caso, é uma figura de linguagem, buscando representar que a próxima contribuição sempre vai ser "maior" e "melhor" que a anterior no fator social e na aprendizagem musical a partir da utilização de ferramentas de *software* livre.

1 Introdução

A produção musical tem soado cada vez mais digital e é um reflexo do uso, quase obrigatório, de Estações de Trabalho de Áudio Digital (*Digital Audio Workstations* – DAWs). Estas ferramentas foram uma verdadeira revolução na forma como a música é concebida, permitindo que arranjos complexos sejam criados inteiramente em ambientes computacionais, sem a necessidade de estúdios físicos de alto custo com uma banda e diversos instrumentos.

No entanto, a democratização do acesso à criação musical através de *software* esbarra em limitações estruturais e financeiras significativas quando observamos o cenário sob a ótica do *design* das plataformas e da colaboração em rede, de forma geral. O paradigma dominante das DAWs proprietárias impõe barreiras comerciais severas através do alto valor de suas licenças e da manutenção de formatos de projetos fechados, que impedem a interoperabilidade e dificultam a distribuição de conhecimento. Um projeto feito no *FL Studio* (Image-Line, 2026), por exemplo, é compatível somente com a mesma versão de *software*. A interoperabilidade entre essas DAWs é mínima, beirando a inexistência. Embora a restrição de *hardware* continue sendo um fator de exclusão[1], a barreira central no ambiente digital colaborativo é ditada pelo modelo de negócios altamente restritivo das ferramentas pagas.

Para dimensionar o impacto dessa restrição no cenário brasileiro, é necessário cruzar o custo de mercado desses *softwares* com a realidade de renda da população. Em 2026, considerando o salário mínimo nacional de R$ 1.621,00 (Brasil, 2025), a aquisição de uma ferramenta padrão da indústria, como a versão completa do *Ableton Live* (Ableton, 2026), exige um investimento que ultrapassa 242% da renda mensal básica (Tabela 1). Essa barreira financeira atua como um filtro excludente severo ao observarmos os dados do Censo Demográfico 2022 (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE), 2025), que revelam que aproximadamente 35,3% dos trabalhadores brasileiros recebem até um salário mínimo, e cerca de 69% ganham no máximo dois salários mínimos. Na prática, o acesso legalizado às principais plataformas de produção é matematicamente inviável para a maior parte da força de trabalho ocupada no país.

Como consequência direta dessa exclusão mercadológica, a pirataria digital emerge não apenas como uma infração legal, mas como a principal, e muitas vezes única, infraestrutura de acesso para produtores de baixa renda. A apropriação de *softwares* não licenciados (conhecidos popularmente como *cracks*) torna-se, paradoxalmente, o mecanismo viabilizador da economia criativa nas periferias, em um fenômeno que a literatura des-

[1] Especialmente no que diz respeito a equipamentos de captação e produção física, como um espaço com isolamento acústico, interfaces de áudio e microfones.

Tabela 1 – Comparativo do custo de licenças de DAWs proprietárias em relação ao Salário Mínimo brasileiro (2026)

Software (Versão Completa)	**Valor (USD)**	**Valor Est. (BRL)**	**% do Salário**	**Meses de Trabalho**
Ableton Live 12 Suite	$ 749,00	R$ 3.932,25	242,5%	2,42
Avid Pro Tools Studio (Perpétua)	$ 599,00	R$ 3.144,75	194,0%	1,94
Steinberg Cubase Pro 15	$ 579,00	R$ 3.039,75	187,5%	1,87
Image-Line FL Studio (*All Plugins*)	$ 499,00	R$ 2.619,75	161,6%	1,61
Apple Logic Pro	$ 199,99	R$ 1.049,95	64,7%	0,64

Notas: Conversão direta com Dólar PTAX médio a R$ 5,25 (Março/2026) (Banco Central do Brasil – BCB, 2026). Salário mínimo nacional fixado pelo Decreto nº 12.797/2025 em R$ 1.621,00 (Brasil, 2025). Valores não incluem IOF (4,38%) e possíveis impostos de importação sobre transações em cartão de crédito.

Fonte: (Ableton, 2026), (Avid Technology, 2026), (Steinberg Media Technologies, 2026), (Image-Line, 2026), (Apple Inc., 2026)

creve como uma "democratização informal" ou economia invisível (LEMOS; OUTROS, 2007). Essa dinâmica, no entanto, cobra o seu preço. A dependência de versões adulteradas expõe os computadores a graves vulnerabilidades de segurança (como *malwares*), instabilidade de sistema durante a produção, e isola o produtor das redes oficiais de suporte, colaboração em nuvem e atualizações.

A cultura musical urbana e periférica, notadamente o Rap e o *Funk*, tem demonstrado enorme capacidade de resistir e contornar essas limitações financeiras e tecnológicas. Impulsionados pela conectividade digital e pela apropriação subversiva de ferramentas de produção alternativas (ou não licenciadas), esses gêneros transicionaram da marginalidade para o centro da indústria fonográfica nacional (HERSCHMANN; FERNANDES, 2022). Uma análise do *Top 50 Brasil 2026* do *Spotify* (Figura 9) evidencia essa mudança de paradigma, onde estéticas sonoras fora do *mainstream*[2] dividem o topo das paradas com superproduções do Sertanejo e do *Pop*. Contudo, a crescente nas métricas de *streaming* não deve mascarar a precariedade de sua base: o *underground*[3]. Embora o talento seja abundante, a ascensão desses gêneros não reflete uma superação das barreiras estruturais. Na verdade, traz à tona o poder de resistência e perseverança da "cultura do improviso".

1.1 Movimento Cultural *Underground*

Falar de cultura *underground* é, antes de tudo, reconhecer uma forma de (r)existência que se constrói na contramão, é a manifestação da contracultura. É o que nasce quando a arte não cabe nos moldes do mercado. Quando não existe microfone, o grito ecoa, a expressão vira sobrevivência. O *underground* não é moda, não é rótulo. É movimento. É urgência, é arte feita com o que se tem, muitas vezes com quase (ou com) nada.

[2] Caracteriza-se pelo cenário musical de ampla aceitação, alta produção comercial, estruturas acessíveis e, frequentemente, com pouca experimentação artística. Por alguns, é considerado como "mais do mesmo", pois artistas encontram as "fórmulas" para as músicas repetitivas e mantém somente esta estratégia.

[3] No cenário musical podemos considerar como toda produção cultural que opera fora dos padrões comerciais, da grande mídia e da indústria da música convencional (*mainstream*)

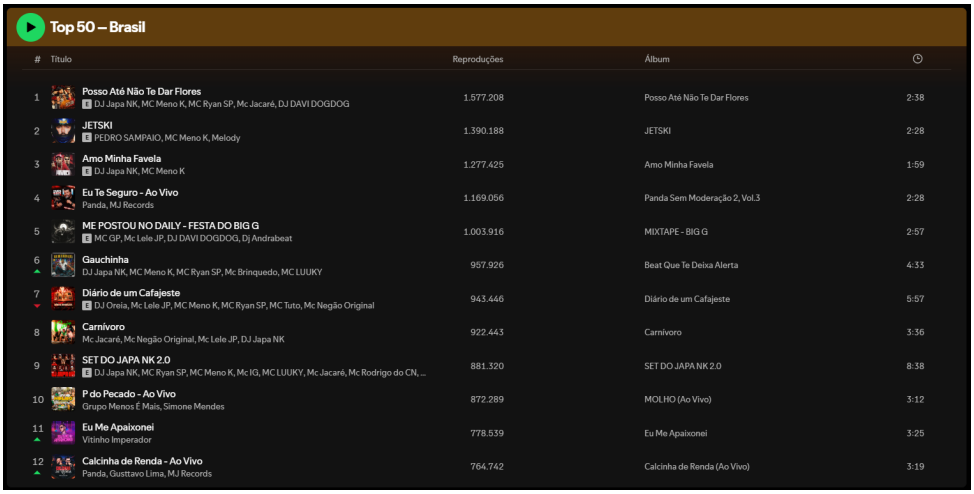

Figura 9 – Top 50 Spotify Brasil em Fevereiro de 2026

Fonte: O Autor (2026).

Para compreender a profundidade desse movimento, é preciso enxergá-lo como a manifestação contínua da contracultura. Como bem documentado por Goffman e Joy (2007), a contracultura não é um evento isolado, mas uma força motriz histórica que impulsiona a inovação e a liberdade contra sistemas restritivos ou monopolistas. Os autores traçam essa linhagem desde o mito de Prometeu, o rebelde original que desafiou os deuses para entregar o fogo e a técnica à humanidade, até desaguar na cultura digital e na ética *hacker*. No contexto deste trabalho, o "fogo" roubado é o acesso irrestrito à tecnologia de produção e compartilhamento.

No cenário contemporâneo, esse mesmo ímpeto transgressor ecoa nas margens da sociedade, nos becos sem *glamour*, nos muros pixados de denúncia, nas rimas feitas na mente enquanto se pega dois ônibus e três estações de metrô para chegar ao trampo:

> "Pegar o trem é arriscado
> Trabalhador não tem escolha
> Então enfrenta aquele trem lotado" (RZO, 1997).

É nesse lugar que a rua cria uma estética e uma ética próprias. A estética do marginalizado, às margens. Conforme é discutido na monografia de conclusão de curso (SOUSA, 2023), o termo marginal ganha um novo significado estético para o movimento *underground* brasileiro. Autores se empoderam em suas letras utilizando o termo e se autointitulando marginais. À margem do *mainstream*, do amparo estatal, da proteção da máquina corporativa de patrulha, excludente e enviesada em ideais não sociais.

Essa apropriação da marginalidade não é somente uma postura estética, na verdade é uma tecnologia social de comunicação e resistência. Como demonstra Rose (1994) em sua análise sobre a cultura *Hip Hop*, a estética das ruas nasce da apropriação e recon-

figuração de recursos descartados ou inacessíveis, criando "textos ocultos" de resistência que circulam à revelia dos centros de poder.

No contexto brasileiro, Herschmann (2000) observa que a periferia utiliza essa arte como uma rede alternativa de sociabilidade e informação, suprindo o vazio deixado pela ausência de infraestrutura estatal e pela exclusão da mídia hegemônica. Somado a isso, Pardue (2008) destaca como o próprio termo "marginal" deixa de ser um estigma criminal para se tornar uma posição estética e discursiva de empoderamento no *Hip Hop* paulista.

É exatamente nessa capacidade de forjar circuitos autônomos de troca de informação que a estética *underground* se encontra com a teoria das redes. A roda de rima, por exemplo, é a materialização física de um sistema onde a hierarquia é horizontalizada. Sob a ótica de sistemas, o *underground* opera como uma rede distribuída orgânica. Essa dinâmica encontra um paralelo exato no modelo de "produção colaborativa entre pares" (P2P), formalizado por Benkler (2006) em sua análise sobre a economia e a liberdade nas redes. Assim como uma arquitetura P2P (*Peer to Peer*, ou em português, Par a Par) genuína prescinde de servidores centrais para operar, a cultura de rua é estruturalmente descentralizada e baseada no compartilhamento direto nas bordas. Cada indivíduo na roda de rima atua simultaneamente como cliente e servidor (consumidor e produtor de cultura).

Consequentemente, o movimento torna-se altamente tolerante a falhas. A ausência ou interrupção de um nó (um artista ou equipamento) não derruba a rede, pois o conhecimento e a infraestrutura estão distribuídos coletivamente entre os pares. Na Figura 10, é possível observar que o movimento cultural *underground* opera de forma análoga à rede distribuída, onde não há um controle centralizado, garantindo autonomia e tolerância a falhas na comunicação.

O movimento contracultural *underground* sempre encontrou brechas na arquitetura do colapso urbano para erguer seus próprios templos de resistência. No universo musical das ruas, essa insurgência tem uma raiz tecnológica clara, os *Sound Systems* jamaicanos da década de 1950. Como documentado pelo antropólogo Stolzoff (2000), estes surgiram como imensas discotecas móveis, formadas por "paredões" de caixas de som construídas artesanalmente e amplificadores customizados por engenheiros locais para maximizar as frequências graves. Mais do que equipamentos de áudio, essas estruturas eram a principal plataforma de entretenimento, comunicação e engajamento sociopolítico para a classe trabalhadora de Kingston, que era financeiramente excluída dos clubes noturnos tradicionais e das raras transmissões de rádio locais.

Nessas festas de rua, a equipe técnica composta pelo *selector* (quem escolhia os discos) e pelo *deejay* ou *toaster* (quem animava o público no microfone, o precursor direto do MC), operava a própria infraestrutura cultural da comunidade.

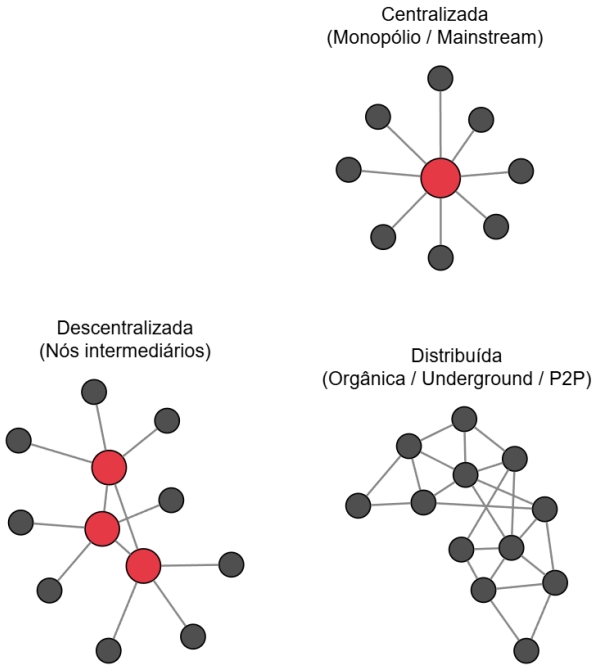

Figura 10 – Representação das topologias de rede.

Fonte: O Autor (2026).

Quando o DJ Kool Herc imigrou para o Bronx, em Nova Iorque, na década de 1970, ele trouxe consigo não apenas os graves potentes e a técnica dessa tradição caribenha. Na verdade, trouxe a filosofia fundamental do reaproveitamento do espaço público e da tecnologia para congregar os marginalizados. Ali, nos escombros de uma cidade negligenciada pelo Estado, cujas comunidades foram fraturadas por políticas urbanistas excludentes (CHANG, 2005), a cultura *Hip Hop* consolidou-se em seus quatro pilares fundamentais. Ou melhor, tecnologias de visibilidade, demarcação e sobrevivência para aqueles que o sistema havia decretado como invisíveis ou fora do comum:

O *DJ* (a manipulação do som): Em posse dos *Disk Jockeys*, o toca discos deixou de ser um mero aparelho de reprodução para se tornar um instrumento musical de assalto. Como argumenta Katz (2012), o DJ *underground* subverteu a tecnologia comercial, utilizando o *mixer* e o vinil para criar *loops* infinitos (os *breakbeats*), inaugurando uma nova engenharia sonora que reescreveu a forma de consumir e produzir música.

O *MC* (a palavra ritmada): O Mestre de Cerimônia atua como o *griot* contemporâneo, o cronista da selva de pedra. Conforme fundamenta Keyes (2004), a rima herda a tradição oral e barda africana, transmutada em uma arma de denúncia e sobrevivência no contexto urbano. É a voz ganhando ritmo para documentar uma realidade ignorada pela mídia hegemônica, “hackeando” a atenção pública e condu-

zindo a energia coletiva através da consciência de rua.

O *Breakdance* (a expressão corporal livre): As batalhas de *B-boys* e *B-girls* transmutaram a violência das gangues em disputas simbólicas e coreográficas. Em seu estudo sobre as fundações da dança *Hip Hop*, Schloss (2009) destaca que o *breaking* é uma rigorosa disciplina estética e de retomada do espaço público, onde o asfalto e o papelão se convertem em palcos de reinvenção física e espacial.

O *Graffiti* (a demarcação visual e política do território): A assinatura e os murais coloridos foram a resposta visual ao apagamento social. Conforme documentado por Castleman (1982), o movimento de pintar os vagões do metrô de Nova Iorque garantiu que os nomes dos jovens marginalizados circulassem por toda a metrópole, forçando a cidade inteira a ler e reconhecer sua existência.

Juntos, esses quatro elementos formaram uma rede analógica de produção colaborativa, antecipando a ética do "faça você mesmo" (DIY) que, décadas mais tarde, encontraria ressonância no ciberespaço e na cultura do código aberto. Nessa conjuntura, o coração pulsante do movimento sempre foi a invenção rítmica. Inicialmente, a batida era uma prática analógica e efêmera, nascida da técnica do isolamento do *break* (a seção instrumental de discos de *funk* e *soul*), prolongado infinitamente em dois toca-discos. Como pontua Rose (1994), a apropriação dessas máquinas reprodutoras para criar novos sons gerou "textos ocultos" de resistência. Com o avanço tecnológico, o *beat* evoluiu de uma performance ao vivo nos bailes para uma composição elaborada dentro de máquinas, como as clássicas MPCs[4].

Na cultura *Hip Hop*, a base instrumental sobre a qual o MC (Mestre de Cerimônia) declama seus versos é convencionalmente chamada de *beat* (batida). Contudo, reduzir o *beat* a um mero acompanhamento rítmico é um erro conceitual. Conforme fundamentado por Schloss (2014) em sua obra *Making Beats*, o *beat* é uma colagem sonora autônoma, uma escultura rítmica e harmônica que dita a atmosfera emocional da obra e atua como a assinatura estética de seu produtor. Ele é o alicerce onde a poética urbana se sustenta e toma forma.

Se no passado a estética do *underground* dependia de *hardwares* caros e inacessíveis, hoje a trincheira de produção é digital. E é exatamente quando transportamos essa prática de criação para os computadores que a ferramenta utilizada para esculpir essa base ganha um novo e profundo peso político. O *beat* feito num *software* livre carrega mais do que som. Carrega vivência, carrega história. Além disso, carrega a possibilidade de uma quebra histórica na estética proprietária e hermética dos *softwares* que dominam amplamente a

[4] Em Inglês *Music Production Controller*, ou em português, Controlador de Produção Musical. É uma estação de trabalho musical, lançada em 1988. Combina *sampler*, sequenciador e bateria eletrônica em um único dispositivo.

produção musical no mercado mundial. O que o *mainstream* chama de alternativo, para nós é cotidiano. A tecnologia, finalmente, democratizada e operando nas bordas.

> "Respeita doidão, aí, não fala assim
> Bolinho pra você é família pra mim" (Racionais MC's, 2002)

Derivado da fusão simbiótica entre o DJ e o MC, o Rap (frequentemente decodificado como *Rhythm and Poetry*, em Português, Ritmo e Poesia) materializou-se como o principal veículo de exportação e documentação da cultura *Hip Hop*. Como argumenta a pesquisadora Perry (2004) em sua obra *Prophets of the Hood*, o Rap transcende a classificação de mero gênero musical para operar como uma literatura oral complexa, onde a poética e a política se fundem para criar um fórum público de vozes sub-representadas. Não por acaso, Chuck D, líder do lendário grupo *Public Enemy*, imortalizou o Rap com a definição de que ele seria a "CNN[5] dos negros" (e dos marginalizados), a rede de notícias não oficial que transmitia o que a mídia hegemônica se recusava a cobrir.

No Rap, portanto, cada verso é um documento histórico. Mesmo com tanta precariedade, o *underground* ensina. Ensina a criar sem depender, a resistir sem pedir licença, a fazer com o que tem. É esse espírito de autonomia e "hacktivismo" de sobrevivência que guia as premissas deste trabalho, entender que a tecnologia pode (e deve) estar a serviço da cultura, e não o inverso.

A infraestrutura tecnológica proposta não pode replicar modelos de dependência e vigilância centralizada. Pelo contrário, ela deve ser arquitetada para garantir que a ferramenta de produção permaneça sob o controle absoluto da comunidade, onde a conectividade com a internet pode ser escassa ou nula, mas a criatividade é abundante. O objetivo é estabelecer uma plataforma livre de aluguéis de *software*, de licenças proibitivas ou da dependência algorítmica de grandes corporações.

Para que essa soberania seja efetiva no cenário contemporâneo, é imperativo dissecar as ferramentas centrais de composição musical de hoje. Se o toca discos e a MPC foram as armas do século XX, o estúdio virtual é a trincheira do século XXI. É essa transição do equipamento físico (e muitas vezes inacessível) para o ambiente computacional que nos leva à necessidade de compreender a arquitetura e o impacto dessas plataformas.

1.2 Produção Musical Digital e DAWs

A música eletrônica e o Rap utilizam predominantemente instrumentos digitais, onde ferramentas como sintetizadores, *samplers*[6] e, posteriormente, as DAWs[7], tornaram-

[5] CNN é um dos principais (e mais impactantes) canais de notícias do mundo.

[6] É um instrumento musical eletrônico que grava, armazena, manipula e reproduz amostras de som (*samples*), sejam trechos de músicas, instrumentos reais ou sons do cotidiano.

[7] Em inglês *Digital Audio Workstation*, ou em Português, Estação de Trabalho de Áudio Digital)

se onipresentes. A prática do *sampling*[8] e o uso de MPCs representam, em essência, uma sofisticada técnica de computação criativa. Trata-se da captura, armazenamento e manipulação de fragmentos sonoros pré-existentes para a construção de uma nova obra. Conforme analisa Katz (2010), o *sampler* democratizou radicalmente a composição ao eliminar a barreira do acesso a instrumentos caros e à educação musical formal. O letramento clássico foi substituído pela curadoria de discos e pela habilidade técnica de manipulação de áudio, transformando o ouvinte em compositor.

Com o avanço dos microcomputadores e a padronização de protocolos como o MIDI (*Musical Instrument Digital Interface*), consolidou-se a figura do produtor independente e os tão famosos *home studios*[9]. Segundo Burgess (2014), esse arquétipo representa o artista autossuficiente que compõe, grava, mixa e masteriza sua obra inteiramente em ambientes domésticos. Essa revolução foi impulsionada pelas DAWs, que, como aponta Théberge (1997), funcionam como um estúdio virtual completo dentro do computador, centralizando e barateando todas as etapas da cadeia de produção sonora.

O protocolo MIDI, padronizado no início da década de 1980, foi um dos maiores marcos dessa democratização digital. Diferente de uma gravação de áudio tradicional (que captura as ondas sonoras analógicas), o MIDI não armazena o som em si, mas sim instruções de dados[10]. Como detalhado por Huber (2007), o protocolo funciona como uma partitura digital invisível que informa a um dispositivo ou *software* exatamente o que fazer, qual nota tocar, com qual intensidade e por quanto tempo. Através do MIDI, o artista ganhou uma liberdade criativa sem precedentes, tornando possível gravar uma melodia, corrigir erros milimétricos de tempo matemático, alterar o tom e trocar instrumentos virtuais de forma totalmente não destrutiva.

A composição de um projeto musical nestas plataformas envolve a orquestração visual de diversos elementos computacionais. O produtor utiliza sequenciadores passo a passo (*Step Sequencers*, vide Figura 11), como o esqueleto rítmico da composição. Essa abordagem ancora-se no conceito de "música com caixinhas", discutido em (SOUSA, 2023), uma estética que se vale do esqueumorfismo, o princípio de *design* onde itens virtuais imitam suas contrapartes do mundo real. Essa interface traduz para a tela do *software* a experiência tátil e a lógica de programação de dispositivos físicos clássicos, como as lendárias caixas de ritmo (*drum machines*) dos anos 1980 (Figuras 12 e 13).

De forma complementar, para o desenho de padrões harmônicos e melódicos complexos, utilizam-se as interfaces de *Piano Roll* (vide Figura 14). Enquanto as "caixinhas"

[8] É considerada como uma técnica de reutilizar um trecho de uma gravação pré-existente (áudio, voz, batida) e incorporá-lo em uma nova criação musical.

[9] Estúdios improvisados que são montados em cômodos das casas dos produtores (geralmente seus próprios quartos). Em muitos casos, é apenas um microfone e fone ligados no computador do produtor e muita determinação durante as sessões.

[10] Parâmetros de intensidade, velocidade, altura, duração, tom, dentre outros.

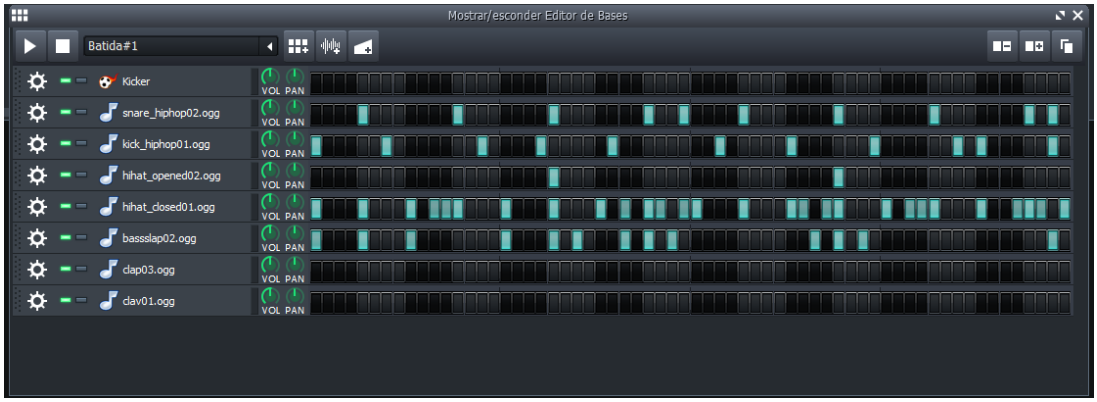

Figura 11 – Exemplo de utilização do *Step Sequencer* do LMMS.

Fonte: O Autor (2026).

Figura 12 – *Drum Machine* TR-808.

Fonte: <https://www.smithsonianmag.com/arts-culture/history-tr-808-drum-machine-180975205/>

ditam batidas cíclicas voltadas à percussão, o *Piano Roll* dita cirurgicamente quando, em qual nota (tom), com qual intensidade e por quanto tempo um som deve soar.

Estes eventos acionam instrumentos virtuais e o controle dinâmico da música é feito através de curvas de automação. Toda essa estrutura lógica precisa ser processada em tempo real pela CPU para gerar o som final. Contudo, a imensa maioria das DAWs que dominam a indústria fonográfica global opera sob lógicas de mercado herméticas, licenças com custos proibitivos e código proprietário. É exatamente neste ponto de estrangulamento mercadológico que o *software* livre se consolida como a principal ferramenta de

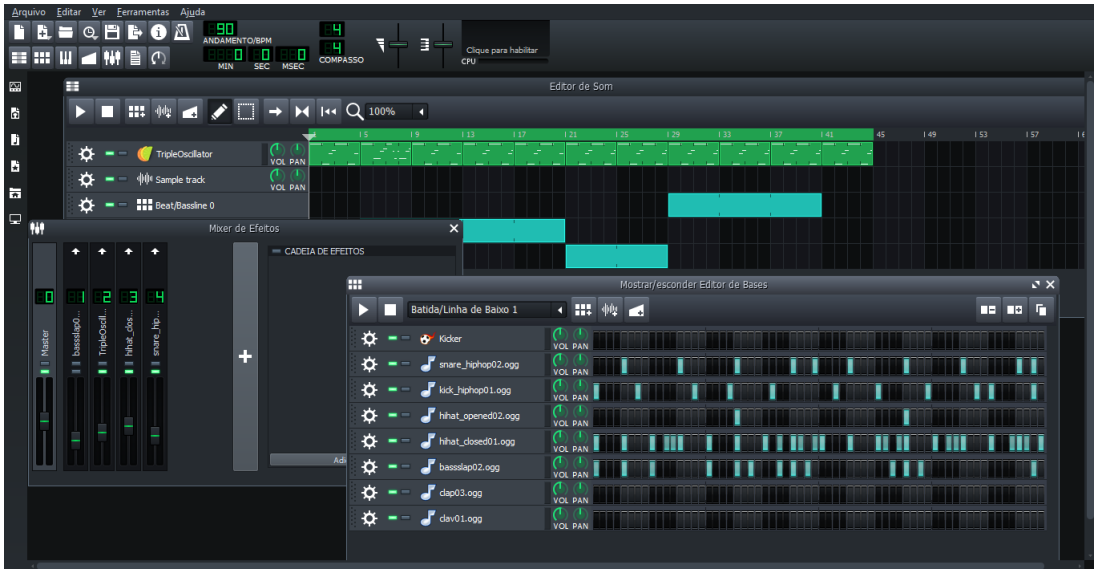

Figura 13 – Tela do LMMS.

Fonte: O Autor (2026).

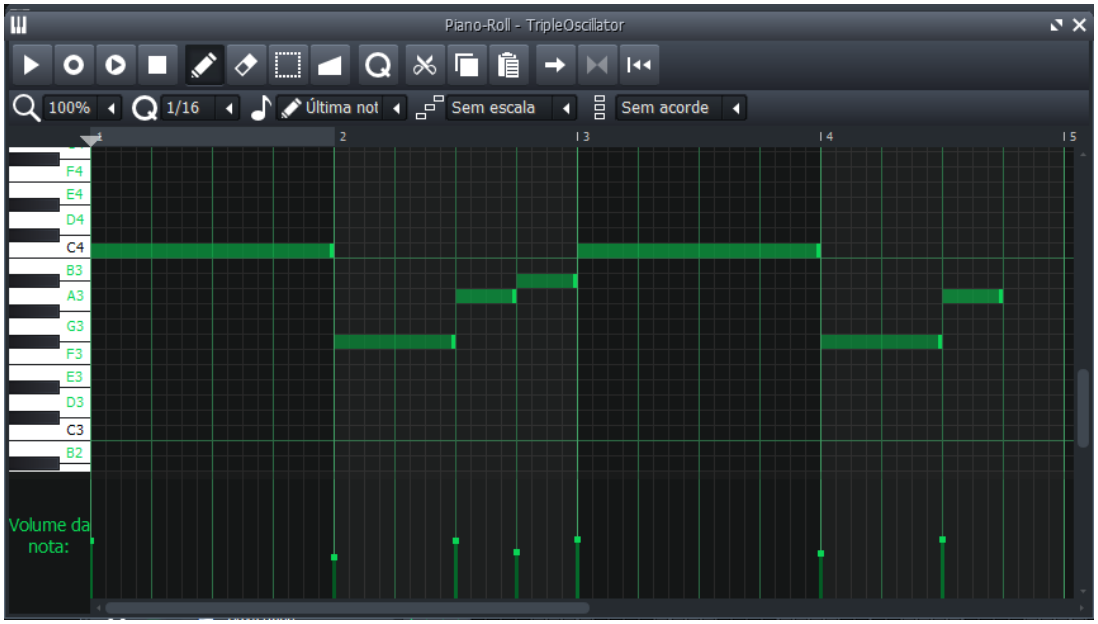

Figura 14 – Exemplo de utilização do *Piano Roll* do LMMS.

Fonte: O Autor (2026).

hacktivismo do *underground* contemporâneo.

O LMMS (*Linux MultiMedia Studio*) exemplifica perfeitamente essa alternativa, alinhando-se ideologicamente com a democratização do acesso, oferecendo soberania tecnológica, transparência de código e custo zero (LMMS Development Team, 2020). O LMMS é uma DAW de código aberto (*open-source*), multiplataforma e de interface esquemórfica e acessível.

Conforme apontam estudos sobre a aplicação de tecnologias abertas na América Latina, como os de Prias e Barrios (2019), ferramentas como o LMMS são amplamente adotadas em oficinas de letramento musical e projetos educacionais de resistência. Seu grande trunfo estrutural, além de não possuir amarras comerciais, é a otimização de processamento. Ele viabiliza a produção musical avançada sem exigir computadores de última geração para suas funções básicas, operando perfeitamente nas margens tecnológicas onde a infraestrutura é precária, mas a urgência de criação é latente.

1.3 O *Underground* como Sistema Distribuído Natural

Para que essa soberania seja efetiva no cenário contemporâneo, a infraestrutura tecnológica não pode replicar modelos de dependência e vigilância centralizada. A cultura *underground*, manifestada através do *Hip Hop*, desenvolveu-se nas margens dos grandes centros urbanos fomentando a produção cultural através de redes com distribuição descentralizada e resilientes.

No contexto da produção musical, a prática do "Faça Você Mesmo" (DIY) impulsionou uma migração forçada dos meios de produção para ambientes domésticos. Contudo, a própria essência coletiva do movimento, evidenciada nas rodas de rima e na troca de *beats*, exige uma transição conceitual do DIY para o DIWO (*Do It With Others* – Faça Com os Outros). Isso significa expandir o foco do indivíduo isolado para a inteligência coletiva e a cocriação, assemelhando-se cada vez mais do *peer-to-peer*, conforme as analogias que estamos desenvolvendo durante este capítulo.

Sob a ótica da Ciência da Computação, essa evolução sociocultural espelha perfeitamente a transição de um computador pessoal isolado para um nó ativo em um ecossistema distribuído. O ecossistema do Rap opera analogamente a uma rede de computação em malha, pois é resiliente a falhas (a falta de recursos não interrompe a arte), descentralizada (sem depender de uma grande gravadora) e heterogênea. A semântica da palavra "marginal", frequentemente pejorativa, ressignifica-se aqui, tendo o sujeito "marginal" como o cliente na Borda da arquitetura de sistemas.

1.4 Hipóteses da Pesquisa

O ecossistema T.R.E.N.S. (Tecnologias em Rede para Emancipação e Narrativas Sonoras) emerge como resposta estrutural. Esta plataforma foi concebida não apenas como um artefato de *software*, mas como uma infraestrutura de mobilidade digital, operando como uma metáfora urbana viva. Assim como os trens conectam as margens aos grandes centros, o sistema visa escoar a produção cultural independente através da superação de barreiras infraestruturais. A partir desta concepção, a pesquisa empírica e a avaliação de

usabilidade em cenários reais com artistas independentes são norteadas por duas hipóteses centrais.

A primeira hipótese, focada na usabilidade emancipatória e na prática DIWO (*Do It With Others*), postula que a transposição da estação de trabalho para o navegador, viabilizada pelo `mmpCreator`, subverte a lógica excludente das DAWs proprietárias. Acredita-se que essa abordagem, fundamentada em computação distribuída e computação na borda, dialoga diretamente com a Estética da Escassez, eliminando a fricção imposta pela dependência de infraestruturas robustas de *hardware*. Desta forma, a plataforma busca resgatar a essência coletiva e tolerante a falhas das rodas de rima e dos *Sound Systems* originais. Na prática, a utilização do `mmpCreator` deve viabilizar a fluidez da produção e a colaboração musical contínua, independentemente das oscilações de conectividade enfrentadas pelos artistas no cotidiano.

A segunda hipótese aborda o letramento musical e o que definimos como engenharia reversa cognitiva. A indexação e a estruturação semântica de projetos musicais abertos, conduzidas pelo motor `mmpSearch`, rompem a barreira do conhecimento tácito isolado, operando como um autêntico mecanismo de aprendizagem por pares no ecossistema digital. Ao permitir a inspeção visual e algorítmica de arranjos sem a necessidade de espelhamento técnico do ambiente local, esta ferramenta atua de forma análoga à Zona de Desenvolvimento Proximal (ZDP) proposta por Vygotsky. Isso potencializa o letramento musical ao permitir que produtores periféricos desconstruam e analisem a engenharia de *beats* mais complexos elaborados pela comunidade. Consequentemente, espera-se que a interação com o `mmpSearch` expanda a autonomia criativa do usuário e eleve o seu domínio técnico sobre a composição, utilizando a própria cultura como ferramenta de ensino.

1.5 Formalização do Problema e Objetivos

Apesar da gratuidade de ferramentas como o LMMS facilitar o primeiro contato com a produção musical, o fluxo de trabalho digital contemporâneo rompeu com uma característica essencial da cultura *underground*, a coletividade.

A democratização da aprendizagem musical digital, quando confrontada com a realidade de infraestruturas restritas, revela barreiras que vão além de questões socioeconômicas, configurando-se como desafios complexos de Engenharia de *Software*. Acredita-se que a viabilização de um ambiente de produção musical neste cenário esbarra em alguns limitadores técnicos críticos:

- **Déficit de Estruturação Semântica:** Existe uma vasta quantidade de conhecimento tácito disperso na *internet*, porém há uma escassez de dados estruturados que permitam a consulta algorítmica. Isso torna inviável a pesquisa ágil de refe-

rências técnicas específicas, dificultando processos de aprendizagem colaborativa e descoberta de soluções. A visualização de projetos em plataformas de colaboração é extremamente limitada, trazendo apenas informações básicas, sem o real conteúdo dos projetos (Figura 16).

- **Dependência de Ambiente Local e Quebra de Projetos:** O compartilhamento atual é limitado pela necessidade de espelhamento técnico. Para colaborar, é exigido que as partes possuam versões idênticas da DAW, além dos mesmos *plugins*[11] e *samples*[12]. A ausência de uma inspeção *online* e independente do ambiente local gera um "gargalo criativo", pois qualquer disparidade técnica impede a execução do projeto.

- **Interoperabilidade e Gestão de Alterações:** As plataformas atuais não oferecem ferramentas nativas de versionamento e comunicação de mudanças em tempo real. Isso obriga os produtores a uma coordenação manual exaustiva, onde cada alteração precisa ser comunicada e sincronizada previamente, tornando a colaboração assíncrona lenta e propensa a erros.

Do ponto de vista arquitetural, a superação desses obstáculos exige uma mudança de paradigma. Enquanto soluções de mercado pressupõem estações robustas ou banda larga constante, este trabalho propõe apresentar, documentar, formalizar e implementar uma plataforma de produção musical *online*, baseada em LMMS, o `mmpCreator`. Nesta arquitetura, espera-se que a carga de processamento do sinal de áudio e a renderização da interface são movidas para o cliente (navegador/dispositivo), visando eliminar a latência de rede durante o processo criativo.

A construção desta plataforma iniciou-se com o desenvolvimento do `mmpSearch`, um motor de busca projetado para facilitar a visualização, compreensão e busca de referências em projetos de composição, criação e produção musical feitos utilizando a DAW LMMS. Visando trazer uma nova usabilidade para as plataformas de compartilhamento, como a *Sharing Platform* do próprio LMMS (LSP)[13]. O objetivo é que produtores e MCs possam pesquisar em projetos musicais de forma tão ágil quanto buscam um texto, sanando dúvidas específicas sobre a composição sem a necessidade prévia de *download* (conforme ilustram as Figuras 15 e 16). Essa abordagem pode ter um grande potencial para a evolução de produtores iniciantes ou entusiastas em busca de referências práticas

[11] *Softwares* auxiliares, que atuam como instrumentos virtuais ou efeitos, que funcionam dentro do programa principal de produção musical, como a DAW por exemplo. Podem ser *plugins* VSTs (*Virtual Studio Technology* ou Tecnologia de estúdio virtual) ou LADSPA (*Linux Audio Developer's Simple Plugin API* ou API de *Plugin* Simples para Desenvolvedores de Áudio Linux), por exemplo.

[12] É considerado como um trecho de uma gravação sonora existente, como uma melodia, batida, vocal ou acorde. É utilizado para "recriar" músicas a partir de manipulações em timbre, tom, velocidade ou fazendo recortes e remanejos durante o tempo da amostra.

[13] LMMS | *Sharing Platform*. Disponível em: <https://lmms.io/lsp/>.

para a composição. Por exemplo, posso baixar um *sample* na plataforma após visualizá-lo em outras produções e ter uma noção de como ele soa junto com outros *samples* e *plugins*.

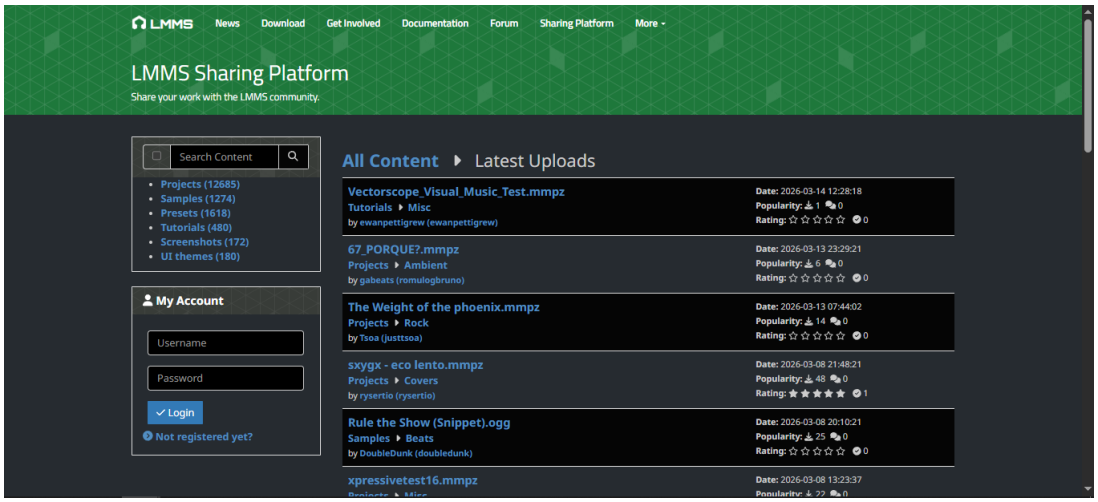

Figura 15 – Página de projetos compartilhados pela comunidade do LMMS.

Fonte: <https://lmms.io/lsp/>

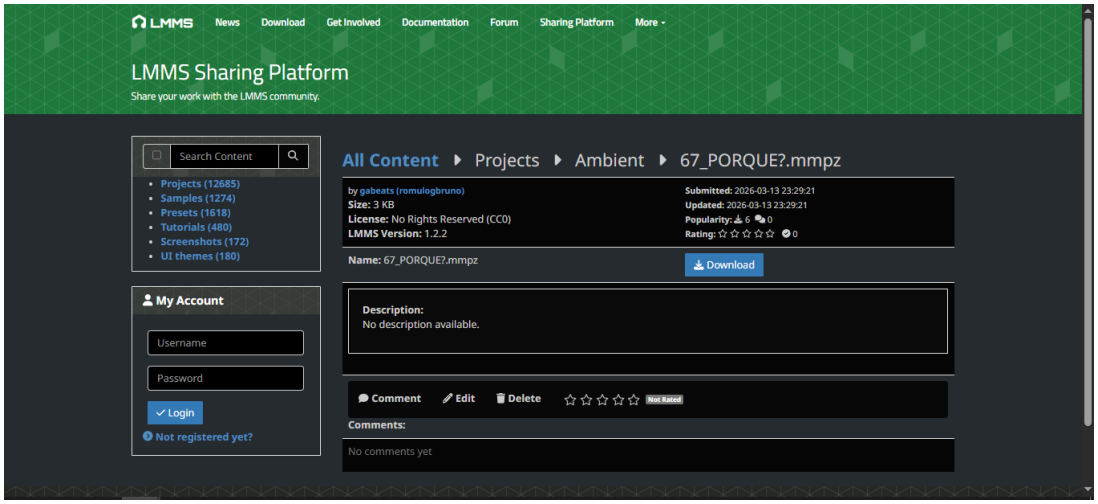

Figura 16 – Página de um projeto individual compartilhado pela comunidade do LMMS.

Fonte: <https://lmms.io/lsp/?action=show&file=24179>

Simultaneamente, o volume potencial de projetos gerados pela comunidade traz consigo o "pote de ouro" para esta abordagem. A viabilização de uma ferramenta como o `mmpSearch`, por exemplo, para a equipe do LMMS pode agir como um potencializador de produtores. A grande quantidade de projetos que a plataforma possui, somado a comunidade ativa transformaria a plataforma de compartilhamento do LMMS em uma enorme biblioteca voltada à aprendizagem de produção musical. Milhares de projetos de referência, criados por centenas (ou milhares) de produtores em países e culturas diferen-

tes. Funcionaria como um grande manual de consulta de criação de *beats*, revolucionando a forma em que o saber musical digital é transferido entre os participantes.

Por outro lado, a quantidade massiva de projetos impõe desafios de escalabilidade no servidor, caracterizando um cenário de *Big Data* aplicado à Musicologia Computacional. Para processar, catalogar e indexar milhares de arquivos XML (`MMP`) de forma eficiente, são necessárias estratégias de Paralelismo de Dados e ingestão[14] assíncrona.

Este trabalho tem como objetivo geral desenvolver e validar uma arquitetura de sistema distribuído voltada para a aprendizagem musical a partir de referências em projetos da própria comunidade e produção musical colaborativa, desenhada especificamente para operar em busca de favorecer o "aqui e agora", fortalecer a cultura do faça com os outros (DIWO). A cultura *underground* a favor da própria cultura. Visando ser fonte de consulta e de criação. O usuário deve ser capaz de baixar e enviar *samples* e projetos para a plataforma. Além disso, terá a possibilidade de produzir por cima de projetos já existentes e fazer o *download* dos resultados destes experimentos. O usuário deve ter a possibilidade de gravar seus próprios *samples* ou músicas na plataforma `mmpCreator`.

Para alcançar este propósito, definem-se os seguintes objetivos específicos:

- **Mapear e Formalizar:** Realizar a engenharia reversa do formato de arquivo MMP do LMMS, criando um esquema de validação (XSD) que permita a interpretação semântica de projetos musicais fora do software nativo.
- **Implementar a Ingestão Paralela:** Desenvolver um *pipeline* de processamento de dados (ETL) capaz de ingerir, analisar e classificar massivamente projetos musicais, utilizando estratégias de paralelismo para superar gargalos de I/O (*Input/Output* ou Entrada/Saída), garantindo que o processamento não fique ocioso enquanto aguarda a leitura e escrita dos grandes volumes de dados musicais.
- **Desenvolver a Interface na Borda:** Construir uma aplicação *web* (*Thick Client*[15]) que utilize a *Web Audio API*[16] para renderizar áudio no navegador do cliente, garantindo a soberania da produção mesmo em redes instáveis.
- **Validar a Arquitetura:** Avaliar o desempenho do sistema comparando métricas de tempo de ingestão (*Speedup*) e latência de interação em cenários reais de uso.

[14] O termo "Ingestão de Dados" é adotado nesta pesquisa em conformidade com a literatura de Engenharia de Dados e *Big Data*, referindo-se ao processo arquitetural de extração, transporte e carregamento de grandes volumes de dados para um repositório central.

[15] É um modelo de *software* onde a maior parte do processamento e da lógica de negócios ocorre diretamente na máquina do usuário, dependendo do servidor apenas para armazenamento de dados ou sincronização.

[16] É uma API nativa dos navegadores web que permite manipular, sintetizar e processar áudio diretamente no *JavaScript*.

1.6 Organização do Texto

O restante deste trabalho está estruturado, abaixo, para conduzir o leitor desde a fundamentação sociotécnica até a validação experimental da arquitetura proposta.

O **Capítulo 2** define as diretrizes metodológicas adotadas no desenvolvimento deste estudo e apresenta a arquitetura da plataforma sobre a infraestrutura "*Wonderland*" (CARNEIRO, 2024). Detalha-se a engenharia reversa do formato MMP para formalização de um esquema de dados (XSD) e especifica-se o protocolo proprietário de sincronização temporal e consistência eventual desenvolvido para viabilizar a colaboração em tempo real (CHIKOFSKY; CROSS, 1990; CRISTIAN, 1989; HOLT, 2004).

O **Capítulo 3** descreve a materialização técnica da solução em dois subsistemas. Explora-se a implementação do *backend* de ingestão paralela e mineração de dados (ETL em Python) para contornar gargalos de I/O, bem como o desenvolvimento do *frontend* (`mmpCreator`) utilizando a *Web Audio API* para síntese sonora determinística e escalonamento *Lookahead* no navegador (KIMBALL; ROSS, 2013; GORELICK; OZSVALD, 2020; WILSON, 2013).

O **Capítulo 4** reporta a avaliação experimental quantitativa e qualitativa. Apresentam-se métricas de *speedup* obtidas através do paralelismo de dados no servidor e a análise comparativa de latência (*End-to-End*) entre a abordagem proposta e modelos tradicionais de renderização em nuvem, validando a eficácia do modelo na borda (AMDAHL, 1967; SHI et al., 2016).

Por fim, o **Capítulo 5** sintetiza as contribuições técnico-científicas, discute as limitações encontradas, como o gargalo de I/O em discos mecânicos e aponta direções para trabalhos futuros, incluindo a integração com IA Generativa Simbólica e Federação de Repositórios (AMDAHL, 1967).

2 Diretrizes Metodológicas e Infraestrutura de Orquestração

Este capítulo detalha a abordagem metodológica utilizada para o desenvolvimento da plataforma e descreve a arquitetura de *software* da plataforma proposta. A solução foi desenhada para operar sobre a infraestrutura do laboratório ALICE ("*Wonderland*"), integrando conceitos de computação distribuída para atender aos requisitos de escalabilidade. A arquitetura privilegia o desacoplamento entre a persistência de dados e a interface do usuário. Para tal, o sistema foi dividido em dois subsistemas complementares. O **mmpSearch**, responsável pela ingestão, curadoria e recuperação da informação musical, e o **mmpCreator**, que atua como a estação de trabalho de áudio digital (DAW) baseada na borda. Esta separação garante que o processamento pesado de indexação seja realizado de forma assíncrona, enquanto a experiência de criação musical permanece fluida no navegador do usuário.

A infraestrutura que alicerça este trabalho não é um mero detalhe técnico, mas uma escolha política e arquitetural de *design*. O sistema opera sobre o *framework Wonderland* (CARNEIRO, 2024), uma plataforma de orquestração de serviços desenvolvida no Laboratório ALICE para solucionar problemas crônicos de fragmentação tecnológica e dependência de software proprietário em ambientes de pesquisa e criação artística.

Pode-se fazer uma analogia do *Wonderland* com um "sistema operacional distribuído" para o laboratório. Enquanto soluções de nuvem pública comercial (como AWS ou *Azure*) abstraem a infraestrutura a um custo financeiro proibitivo para comunidades independentes, criando o que Carneiro (CARNEIRO, 2024) identifica como barreiras de acesso e controle, o *Wonderland* materializa o conceito de nuvem privada acadêmica baseada em *Software Livre*. O *Wonderland* provê a persistência de dados, a segurança e a gestão de identidade necessárias para que a plataforma proposta neste trabalho foque exclusivamente em sua regra de negócio, a música e a colaboração. Neste arranjo arquitetural, o papel do NFS (*Network File System*) é absolutamente central. Ele atua como a espinha dorsal do armazenamento distribuído do laboratório, permitindo que os sistemas de arquivos sejam compartilhados de forma transparente através dos nós da rede. É graças a essa topologia de rede que os diretórios de usuários, especificamente a pasta `public_html`, são montados e servidos pelos servidores *web* da infraestrutura.

De forma prática, a adoção de geradores de sites estáticos (SSGs - *Static Site Generators*), como o Jekyll, beneficia-se intrinsecamente dessa arquitetura. Diferente de aplicações *web* tradicionais, que exigem processamento contínuo no servidor e oneram a infraestrutura com consultas a bancos de dados a cada requisição do usuário, o Jekyll

pré-compila o conteúdo, neste caso, os repositórios e metadados musicais extraídos, em artefatos leves e estáticos (HTML, CSS e JavaScript). Ao direcionar a saída dessa compilação (*build*) diretamente para os diretórios `public_html` sincronizados por uma automação no GIT [1] também provido por (CARNEIRO, 2024), a atualização do sistema ocorre de maneira fluida e imediata. O servidor atua estritamente como um distribuidor de arquivos já renderizados, garantindo uma publicação quase instantânea, com alta disponibilidade, tolerância a picos de acesso e baixíssimo custo computacional. A própria aplicação *mmpSearch*, essencial para este trabalho por permitir a busca e indexação dos arquivos do LMMS, se mantém acessível *online* para o público devido a essa montagem remota de volumes via NFS. Essa abordagem elimina a necessidade de esteiras complexas de implantação (*deploy*) ou custos com hospedagens em nuvens corporativas. Basta que os artefatos de *software* estejam no diretório correto mapeado pelo NFS para que a rede os distribua. Desta forma, o *Wonderland*, é muito mais que uma dissertação de mestrado, tornou-se um *cluster* acadêmico isolado. Sendo a ponte direta entre o desenvolvimento local e a distribuição global. Utilizando a infraestrutura institucional para cumprir seu papel social de habilitar, democratizar e sustentar a criação artística independente[2].

Quanto à sua natureza, este trabalho classifica-se como uma pesquisa aplicada. Segundo Wazlawick (2014), esta modalidade na Ciência da Computação caracteriza-se pela construção de artefatos inovadores para a solução de problemas práticos, guiada por rigor científico na validação. O ponto de inflexão tecnológica do estudo baseou-se na técnica de Engenharia Reversa. Conforme definido por Chikofsky e Cross (1990), este processo consiste na análise de um sistema para identificar seus componentes e relacionamentos, criando representações em alto nível de abstração. No contexto deste trabalho, a engenharia reversa foi aplicada aos arquivos de projeto do LMMS (`.mmp`), revelando que estes não são binários opacos[3], mas documentos XML estruturados passíveis de manipulação semântica.

Para sistematizar esse processo de descoberta e desenvolvimento, adotou-se um ciclo de vida Iterativo e Incremental (PRESSMAN, 2010), permitindo o refinamento contínuo da arquitetura a partir dos resultados de cada fase experimental. O desenvolvimento seguiu as três fases macroscópicas correlacionadas à Tabela 2.

[1] Sistema de controle de versão distribuído. É utilizado para rastrear alterações em arquivos de código fonte durante o desenvolvimento de *software*.

[2] Não é somente este o objetivo da utilização da infraestrutura institucional, mas no universo de interesse deste trabalho, o artista independente é o maior beneficiado.

[3] Assim como os projetos das grandes DAWs proprietárias que são amplamente utilizadas na produção musical contemporânea.

Tabela 2 – Fluxo de Engenharia Reversa e Desenvolvimento da Plataforma

Ação Técnica	**Descrição do Procedimento**	**Fase Correspondente**
1. Reconhecimento Estrutural	Abertura de arquivos `.mmp` em editores de texto para identificação de padrões de tags e hierarquia.	Fase 1: Análise de Domínio
2. Interpretação Semântica	Mapeamento manual do significado de nós críticos (ex: `<pattern>`, `<instrument>`) e atributos de áudio.	
3. Testes de Mutação	Alteração manual de valores no XML e observação do impacto visual na GUI do LMMS (causa-efeito).	
4. Formalização (XSD)	Criação de esquemas de validação (DTD/XSD) para garantir a integridade dos dados extraídos.	
5. Automação de Leitura	Desenvolvimento de *scripts* (*Parsers*) em Python para leitura programática em lote.	Fase 2: Prototipagem
6. Pipeline ETL	Implementação da extração de metadados e áudio para alimentação do *mmpSearch*.	
7. Recuperação (RI)	Desenvolvimento de algoritmos de busca baseados em texto e descritores de áudio.	
8. Validação Cruzada	Verificação automatizada dos arquivos via XSD durante o *upload*.	Fase 3: Validação
9. Interoperabilidade	Testes de carga e reprodução entre a plataforma *Web* e o *software* nativo.	

Fonte: O Autor (2026).

1. **Levantamento de Requisitos e Análise de Domínio:** Correspondente às etapas 1 a 4, onde a engenharia reversa permitiu mapear a estrutura hierárquica XML utilizada para representar o sequenciamento MIDI e automações.

2. **Prototipagem da Arquitetura Distribuída:** Correspondente às etapas 5 a 7, focada na implementação dos agentes de ingestão (*backend/ETL*) e das interfaces de borda (*mmpCreator*).

3. **Validação em Campo (Testes Práticos):** Correspondente às etapas 8 e 9, onde a ferramenta foi submetida a testes de usabilidade e desempenho em cenários reais com artistas independentes e entusiastas.

Visto que o alicerce[4] deste estudo é a experiência própria, o rigor metodológico descrito acima serve para traduzir a vivência prática ("a garra") em conhecimento científico replicável.

2.1 Concepção Arquitetural e Mineração de Dados Musicais

A proposta de uma plataforma colaborativa baseada em tecnologias *Web* visa transferir a complexidade do processamento para a borda da rede, garantindo que o usuário na "margem" precise apenas de um navegador para acessar a plataforma, permitindo que o foco permaneça na expressão da sua realidade. A intenção é priorizar o "aqui e agora", trazendo viabilidade para a composição de projetos, mesmo que não sejam as versões finais das obras. Mas a possibilidade de criar, basicamente, em qualquer lugar é uma ferramenta que eleva exponencialmente a criatividade e a colaboração entre os envolvidos. Trazendo um recorte pessoal, por volta de 2015/2016 o *ChinaTown MC's* fazia parte do coletivo Essência *Crew*[5] e encontrávamos bastante com 2 grupos, principalmente, o Modo Rua e o Dimensão 31. Nestes encontros, falávamos muito sobre Rap, cotidiano, vivências e fazíamos "ensaios" improvisados em lugares aleatórios da cidade, como praças, ruas, orla da lagoa, dentre tantos outros "palcos". Utilizávamos *beats* do *YouTube* ou de nossas próprias músicas. Nós colocávamos uma faixa, eles mandavam o som e vice-versa. Isso trazia novas métricas para nossas composições, outros pontos de vista sobre as letras. Era extremamente enriquecedor. Basicamente, um grupo de pesquisa sobre composição e interpretação musical em sua mais pura essência.

Uma ferramenta acessível, com a capacidade de composição de *beats* e captação de áudio seria uma grande aliada nestes momentos. Principalmente se pudéssemos buscar referências ou procurar projetos de outros produtores, visando diversificar cada vez mais o repertório e as técnicas sonoras. Indo além, seria um diferencial se fosse possível migrar essa gravação para uma DAW e finalizar alguns processos mais pesados, utilizando o teclado e *mouse* convencionais. Ou até mesmo compartilhar o projeto com outros produtores[6] e atuar em uma produção colaborativa em tempo real. No andamento da pesquisa feita durante a graduação (SOUSA, 2023) foi perceptível que a música faz parte do interior dos participantes das oficinas. Por mais que 55% dissessem nunca ter produzido algo, os resultados foram extremamente satisfatórios. Visto que 83,3% dos participantes conseguiram ter êxito na criação de, pelo menos, um *beat*. Levando em consideração que

[4] Claro que existe um embasamento teórico, com fundamentos em estudos consolidados. Mas a experiência própria e a dor do não poder (a dor do não conseguir por falta de dinheiro, falta de acesso, falta de uma possibilidade) é a que faz com que a gente "se vire" e hoje, alguns anos após o meu "ápice" como compositor e MC, estou me virando e tentando possibilitar a tal "oportunidade" para outros artistas independentes ao redor do mundo.

[5] Coletivo formado por artistas da Zona Oeste de Belo Horizonte, majoritariamente formado por grupos de Contagem, cidade da região metropolitana.

[6] Que podem (ou não) estar fisicamente no local.

25% nunca tinha tido contato com uma DAW, são ótimos números. A abordagem das oficinas e criação do material didático é validada e vista com um potencial enorme para a aprendizagem musical utilizando ferramentas computacionais. A princípio, a ideia de uma DAW gratuita e disponível multiplataforma já é bastante interessante. Entretanto, pode entrar no limbo das inúmeras que já existem. O que poderia ser um diferencial neste caso? Uma plataforma de pesquisa para o produtor musical. Assim como temos o *Google* para fazermos buscas gerais ou o *SearchCode*[7] para buscar referências de utilização de funções ou trechos de código, uma plataforma para pesquisa em projetos de produção musical em LMMS seria de grande auxílio para destravar a criatividade do produtor.

Mas essas soluções já existem e, conforme discorrido no Capítulo 1, são extremamente limitadas no quesito praticidade. Para visualizar o projeto, é necessário baixá-lo. Alguns projetos vem corrompidos, sem *samples* essenciais para a reprodução correta ou com *plugins* externos. Não é muito didático, tampouco funcional. A possibilidade de visualizar o projeto e destrinchar suas *patterns*[8] (padrões) de forma similar àquela vista na própria DAW (Figura 11) é algo extremamente positivo do ponto de vista pedagógico. A "engenharia reversa" ou o ato de desconstruir e analisar visualmente projetos e arranjos existentes, atua como um dos principais mecanismos de aprendizagem (BELL, 2018). Além disso, buscar projetos utilizando filtros de instrumentos, *plugins*, nomes de *patterns*, *samples* ou desenho rítmico é uma abordagem inovadora suportada pelos conceitos de Recuperação da Informação Musical (*Music Information Retrieval* – MIR). A plataforma eleva a experiência do usuário se contemplar os três níveis de especificidade em consultas musicais propostos por Casey et al. (2008):

1. **Nível 1 - Busca de Item Conhecido:** Sistemas de alta especificidade. Representa a necessidade de encontrar instâncias exatas. No sistema, isso é materializado por uma busca textual direta por nomes de *patterns* específicos, títulos de projetos ou arquivos de *samples* referenciados na árvore XML (ex: instâncias de `<audiofileprocessor>`).

2. **Nível 2 - Busca Categórica e Semântica:** Sistemas de baixa especificidade. Envolve a recuperação de metadados que agrupam músicas por atributos compartilhados. Na interface a ser desenvolvida, este nível pode ser amplamente explorado através dos filtros booleanos de *plugins* (ex: buscar apenas projetos usando o *TripleOscillator*), faixas de andamento (BPM), tonalidade (*Key*) e por algum tipo de classificação pedagógica (Iniciante, Intermediário, Avançado).

[7] Pode ser considerado como um motor de busca projetado para encontrar trechos específicos de código-fonte, funções ou arquivos dentro de grandes repositórios, projetos de *software* ou na *internet*. Disponível em: <https://searchcode.com/>. Acesso 19 Mar 2026.

[8] Na produção musical, consideramos a *pattern* como uma pequena sequência musical ou rítmica pré-programada, geralmente contendo dados MIDI (notas musicais, ritmos de bateria) ou automações, que pode ser repetida ou arranjada ao longo da *playlist* de uma música.

3. **Nível 3 - Busca Baseada em Conteúdo:** Sistemas de média especificidade. O nível mais abstrato e complexo, onde o usuário busca por similaridade musical sem necessariamente usar palavras. O sistema deve contemplar este nível de forma robusta com o mecanismo de Busca por Ritmo, onde o produtor desenha a sequência desejada diretamente no *Step Sequencer* da interface, e o algoritmo calcula a distância de similaridade entre padrões rítmicos dos projetos indexados.

Ao cobrir todo o espectro de especificidade de busca estrutural, a plataforma pode ir além da função de um mero repositório de arquivos. É possível que se consolide como um motor de pesquisa semântico que compreende a música sob a mesma ótica técnica do produtor. Dessa forma, atua como uma potencial ferramenta cognitiva que destrava a criatividade e facilita a aprendizagem musical estrutural, alinhando-se aos princípios propostos por Marrington (2011). A resiliência do sistema não é voltada apenas para a simples manutenção do serviço *online*, ela reside na perenidade e na completude da obra criada. No ecossistema *underground*, a música deve ser portátil, passível de ser transportada de um estúdio caseiro para outro. Computacionalmente, essa característica é atendida através da serialização do estado da aplicação em formato XML (*Extensible Markup Language*).

Contudo, a interoperabilidade plena em uma DAW distribuída enfrenta o desafio do Gerenciamento de Dependências. Um projeto musical digital é uma entidade composta. Ele consiste na estrutura lógica (o arquivo de projeto nativo contendo notas e automações) e nos ativos binários (arquivos de áudio complementares, conhecidos como *samples*). Na arquitetura proposta, preveem-se dois cenários de persistência e transferência de estado, que impactam diretamente a consistência da reprodução sonora (Figura 17):

1. **Cenário de Completude (Pacote Autocontido):** Em um fluxo ideal, o usuário submete um pacote comprimido. O servidor, através de rotinas de manipulação de arquivos, deve extrair tanto o descritor XML quanto os arquivos de áudio associados. Isso garante que as referências aos samples sejam locais e relativas, assegurando que o projeto soe idêntico em qualquer nó da rede.

2. **Cenário de Referência Quebrada (Upload Parcial):** Em situações onde apenas o esqueleto XML é sincronizado, o sistema deve operar com referências absolutas ao sistema de arquivos da máquina de origem. Se os *samples* não estiverem presentes no repositório do servidor ou na máquina do cliente receptor, ocorre a quebra da integridade da obra: o ritmo visual existe, mas a matéria sonora (o timbre) está ausente.

Essa dicotomia reflete a realidade da produção independente, a necessidade de gerenciar não apenas a composição, mas o inventário de ativos que a compõe. A arquitetura

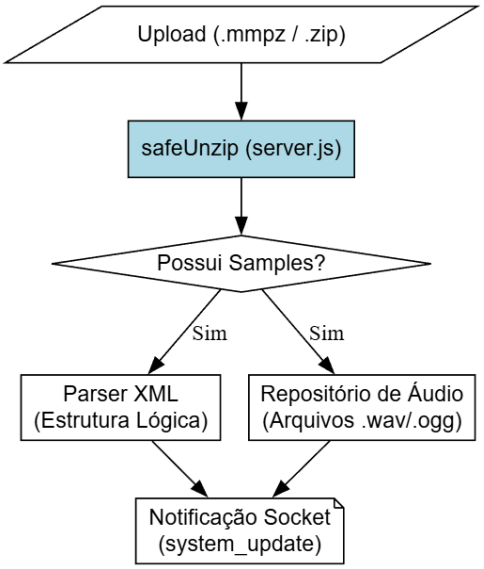

Figura 17 – Representação do fluxo de ingestão de dados.

Fonte: O Autor (2026).

visa mitigar essa fragmentação ao permitir que o servidor atue como um repositório centralizado de ativos quando o pacote completo é enviado, notificando os clientes conectados na rede para recarregarem seus navegadores de arquivos. Dessa forma, a persistência em JSON assegura a recuperação da estrutura lógica (atendendo à Disponibilidade do teorema CAP), enquanto o gerenciamento de *uploads* busca garantir a consistência tímbrica, fundamental para a finalização da obra em ambientes *desktop* profissionais como o LMMS.

Para viabilizar as propostas deste trabalho, é essencial compreender como as DAWs armazenam as informações em seu núcleo. A adoção de formatos estruturados e legíveis por humanos, como o XML (*Extensible Markup Language*) ou JSON (*JavaScript Object Notation*), é fundamental para a interoperabilidade e padronização do armazenamento de conteúdo musical digital. No contexto do LMMS, os projetos são persistidos nativamente no formato `MMP`. Diferente de um arquivo de áudio comum, este formato não encapsula o som, mas descreve a totalidade do arranjo, faixas, sequências MIDI, automações e parâmetros de sintetizadores. Sob a ótica da Ciência da Computação, arquivos MMP atuam como Grafos de Objetos Serializados. Cada nó na árvore XML representa uma instância de um componente musical (um instrumento, um efeito ou um padrão rítmico) com propriedades específicas (vide Figura 18).

A compreensão dessa hierarquia viabiliza a aplicação de técnicas avançadas de Recuperação da Informação Musical (*Music Information Retrieval* – MIR). Através de *parsers* dedicados, torna-se possível percorrer a árvore DOM (*Document Object Model*) do projeto, extraindo e indexando metadados cruciais sem a necessidade de custoso processamento de sinal de áudio (DSP). A indexação automatizada desses metadados permite a

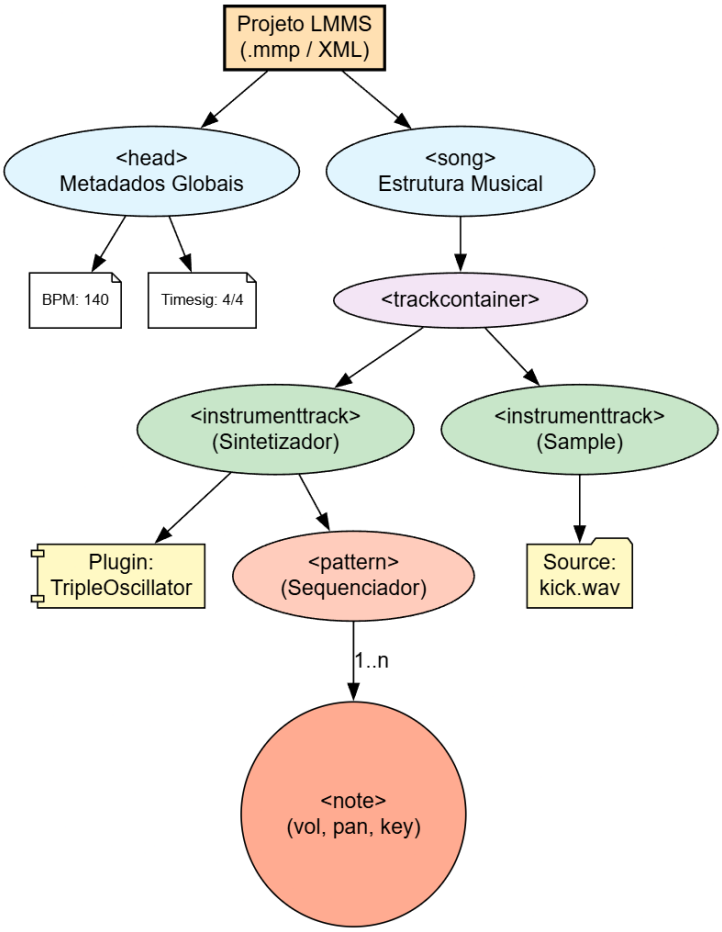

Figura 18 – Exemplo de grafo do arquivo MMP.

Fonte: O Autor (2026).

transição de repositórios de arquivos estáticos para bases de conhecimento dinâmicas. O sistema proposto visa transformar a "caixa preta" do arquivo binário final (projetos sem interoperabilidade e leitura viável aos humanos) em uma "caixa transparente", o projeto fonte aberto. Isso expõe a lógica composicional, oferecendo um andaime cognitivo, conceito fundamental da teoria da aprendizagem cunhado por Wood, Bruner e Ross (1976), que permite ao artista iniciante desconstruir a engenharia por trás da arte. Dessa forma, promove-se um aprendizado técnico com base na engenharia reversa das produções da comunidade.

Contudo, a ausência de uma documentação formal oficial exigiu um esforço de engenharia reversa para criar um contrato de dados robusto. A arquitetura assume que cada projeto musical é um grafo de objetos serializado. Para garantir que o subsistema de ingestão[9] (*mmpSearch*) interprete corretamente a intenção artística do produtor,

[9] É considerado como o processo de transferir dados brutos de diversas fontes (APIs, bancos de dados, arquivos) para um sistema de armazenamento centralizado. Neste caso, os projetos da comunidade sendo armazenados no servidor do ALICE.

desenvolveu-se um esquema de validação XSD (*XML Schema Definition*) proprietário[10]. O Código 2.1 apresenta um fragmento de um projeto, ilustrando como um sintetizador (`TripleOscillator`) e suas notas musicais são representados no arquivo bruto. Note a densidade de atributos numéricos que definem o timbre.

```
<instrumenttrack vol="100" pan="0" pitch="0" basenote="57">
  <instrument name="tripleoscillator">
    <tripleoscillator
       vol0="100" wavetype0="0"
       vol1="50"  wavetype1="1" coarse1="-12"
       vol2="0"   wavetype2="0"
       phoffset1="0" modalgo1="0" />
  </instrument>
</instrumenttrack>

<pattern name="Synth_Intro" steps="16" type="0" pos="0">
    <note pos="0" len="48" key="45" vol="100" pan="0"/>
    <note pos="48" len="48" key="57" vol="100" pan="0"/>
</pattern>
```

Listing 2.1 – Fragmento de arquivo .mmp (Sintetizador e Sequenciamento)

Para transformar esses dados brutos em informação estruturada e segura, o XSD desenvolvido impõe tipagem estrita e obrigatoriedade de campos. O Código 2.2 exibe um recorte do esquema criado, demonstrando a formalização dos parâmetros do `TripleOscillator` e da estrutura de *Patterns*[11].

```
<xs:element name="tripleoscillator">
  <xs:complexType>
    <xs:attribute name="vol0" type="xs:integer" use="required"/>
    <xs:attribute name="wavetype0" type="xs:integer" use="required"/>
    <xs:attribute name="coarse1" type="xs:integer" use="required"/>
    <xs:attribute name="stphdetun0" type="xs:decimal" use="required"/>
    </xs:complexType>
</xs:element>

<xs:element name="pattern">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="note" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="pos" type="xs:integer" use="required"/>
          <xs:attribute name="len" type="xs:integer" use="required"/>
```

[10] Disponível em: <https://alice.ufsj.edu.br/mmpSearch/src_mmpSearch/dtd/>. Fonte: O Autor (2025).

[11] Podem ser consideradas como sequências repetitivas de notas, ritmos, acordes ou sons que formam estruturas coesas e reconhecíveis dentro de uma composição. Eles agem como os "blocos de construção" da música, ajudando a criar familiaridade, ritmo e estrutura.

```
17            <xs:attribute name="key" type="xs:integer" use="required"/>
18            <xs:attribute name="vol" type="xs:integer" use="required"/>
19          </xs:complexType>
20        </xs:element>
21      </xs:sequence>
22      <xs:attribute name="steps" type="xs:integer" use="required"/>
23    </xs:complexType>
24  </xs:element>
```

Listing 2.2 – Recorte do XSD desenvolvido: Formalização de Síntese e Sequência

A importância dessa formalização transcende a validação sintática. Ao tipar atributos como `wavetype0` (tipo de onda) e `coarse1` (afinação grosseira/oitava), o sistema habilita a Mineração Simbólica. O *crawler* não vê apenas números, ele interpreta que `wavetype=1` indica uma onda "Dente de Serra" (rica em harmônicos) e que `coarse=-12` indica uma linha de baixo (Bass), permitindo a classificação automática de gênero e função instrumental sem intervenção humana.

Para garantir a resiliência e a soberania de dados, a arquitetura do sistema adota uma abordagem híbrida baseada no padrão *Jamstack*[12] (*JavaScript, APIs e Markup*). Esta escolha arquitetural permite que o catálogo de projetos seja "pré-renderizado" durante a fase de construção, desacoplando o banco de dados da interface de consulta. A solução implementa o estilo arquitetural *Pipes and Filters*, onde o fluxo de dados ocorre em estágios bem definidos de transformação e consumo, desde o arquivo bruto até o metadado refinado. O *mmpSearch* atua como o componente de inteligência da plataforma, transcendendo a função de um repositório de arquivos tradicional. Ele foi projetado como um sistema de Recuperação de Informação Musical (MIR), operando inicialmente através da interpretação semântica dos arquivos de projeto (`.mmp`). O processo de ingestão realiza uma "Mineração Simbólica" nos arquivos XML, extraindo características estáticas da obra:

- **Impressão Digital Tímbrica (*Plugins* e Instrumentos):** O sistema identifica quais sintetizadores (ex: *TripleOscillator*, *ZynAddSubFX*) e *plugins* VST são utilizados. Isso permite classificar o estilo musical não apenas por *tags* manuais, mas pela "assinatura sonora" teórica do projeto.
- **Análise Rítmica Visual:** Foi implementado um mecanismo de busca por padrão rítmico (*step sequencer*), onde o usuário desenha uma grade de 16 passos e o sistema recupera projetos que contenham células rítmicas similares. Esta funcionalidade

[12] O termo *Jamstack* é um acrônimo para *JavaScript*, *APIs* e *Markup*. Trata-se de uma arquitetura de desenvolvimento web moderna que prioriza a performance e a segurança ao desacoplar a camada de apresentação da camada de persistência, utilizando a pré-renderização de arquivos estáticos para entrega via CDNs (*Content Delivery Networks*) (BIILMANN; HAWKSWORTH, 2019).

valida a capacidade do sistema de analisar a estrutura interna dos arquivos XML (nós `<pattern>`).

- **Integridade Referencial de Ativos:** Para mitigar o problema de "arquivos faltantes" comum em produções distribuídas, o módulo de envio (*upload*) inclui uma camada de verificação de dependências. O sistema analisa o grafo do projeto em busca de *samples* externos e alerta sobre links quebrados antes mesmo da publicação.

A análise puramente simbólica é cega para o resultado acústico real esculpido pelo produtor. Para preencher essa lacuna, a arquitetura de ingestão foi expandida para incluir um *pipeline* de processamento de sinais e aprendizado profundo (*Deep Learning*)[13] desenvolvido em Python, utilizando a biblioteca **Essentia** (do *Music Technology Group* – Universitat Pompeu Fabra). A extração acústica opera em dois domínios paralelos:

1. **Processamento de Sinal Digital (DSP) Clássico:** O áudio de alta qualidade (44.1 kHz) é processado por algoritmos determinísticos para extrair atributos musicais objetivos. Utilizam-se extratores como `RhythmExtractor2013` para predição de BPM, `KeyExtractor` para detecção de tom e escala, e `Loudness` para intensidade. Ademais, implementou-se um detector de "Curva de Novidade" baseado na fusão de Energia (RMS) e Fluxo Espectral, capaz de segmentar a música temporalmente e identificar a quantidade de seções distintas (ex: introdução, verso, refrão), viabilizando a compreensão da macroestrutura do arranjo.

2. **Redes Neurais Convolucionais[14] para Classificação:** Para inferência semântica de gênero e *mood*, o áudio passa por um processo de *downsampling* para 16 kHz, adequando a dimensionalidade aos tensores de entrada dos modelos pré-treinados. Utiliza-se um extrator de *embeddings* fundamentado na arquitetura **EfficientNet-b0** (`discogs-effnet-bs64-1.pb`), que captura características latentes ricas em mel-espectrogramas. Esses vetores alimentam uma rede densa (`genre_discogs400.pb`) que os mapeia para 400 classes de gêneros e estilos catalogados, garantindo uma taxonomia autônoma e precisa do repositório.

A inovação deste modelo híbrido reside no cruzamento (*cross-reference*) entre os domínios simbólico e acústico para inferir o nível de proficiência do produtor. A complexidade do projeto é calculada por um agente classificador central (`classificador_mestre.py`),

[13] Subcampo do aprendizado de máquina (*Machine Learning*) baseado em redes neurais artificiais com múltiplas camadas, capaz de aprender representações complexas de dados, como áudio e imagens.

[14] As Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) são um tipo especializado de arquitetura de *Deep Learning* inspirada no córtex visual biológico. Elas se conectam ao aprendizado profundo por utilizarem múltiplas camadas de processamento ocultas para extrair automaticamente características de dados com estrutura de grade, sendo amplamente aplicadas na classificação de espectrogramas e na análise de padrões rítmicos em sinais de áudio.

proprietário, cujo objetivo não é julgar a qualidade artística, mas sim identificar a maturidade técnica da engenharia de áudio para fins de recomendação didática. A classificação pedagógica (Iniciante, Intermediário, Avançado) foi modelada heuristicamente. Os pesos atribuídos a cada funcionalidade do software LMMS fundamentam-se na Taxonomia de Bloom Revisada (ANDERSON; KRATHWOHL, 2001), onde processos de criação e análise (alta ordem) recebem pontuação superior a processos de recordação e aplicação (baixa ordem).

As regras de pontuação refletem a seguinte lógica cognitiva:

- **Sintetizadores vs. Samples ($W_{synth} = 3.0$ vs $W_{sample} = 1.0$):** O uso de *samples* (`AudioFileProcessor`) de forma primária e inalterada é análogo ao nível de "Aplicação" na taxonomia cognitiva. Contudo, é fundamental ressaltar que a cultura do *Hip Hop* eleva o *sampling* a uma engenharia de áudio complexa. O uso avançado de *samples*, envolvendo técnicas como o recorte meticuloso (*chopping*), a alteração de tom (*pitch shifting*) e a modificação de andamento (*time stretching*), transcende a mera aplicação. Estas técnicas representam uma desconstrução cognitiva avançada do material sonoro original. Portanto, a heurística também deve considerar o *design* de som através de *samplers* avançados como um processo de "Criação". Essa reavaliação metodológica corrige um viés técnico histórico. Além disso, honra a genialidade estética e a carga cognitiva envolvida na operação de ferramentas que simulam as clássicas MPCs. Em contrapartida, a operação de sintetizadores complexos (como *ZynAddSubFX*) exige o domínio de envelopes ADSR, osciladores e filtros. Isso denota a capacidade de "Criação" (*Sound Design*), justificando um peso triplicado pela carga cognitiva envolvida na escultura do timbre (SWELLER, 1988).

- **Automações e *Routing* ($W_{auto} = 2.5$):** A presença de curvas de automação evidencia uma intenção artística refinada sobre a variação temporal. Segundo o Modelo Espiral de Desenvolvimento Musical de Swanwick e Tillman (1986), a capacidade de manipular a "Expressividade" e a "Forma" surge após o domínio básico dos materiais. O sistema premia projetos que saem da estática (*loop*) para a dinâmica (evolução), indicando um produtor que já superou a fase de exploração inicial.

- **Estrutura Acústica e Macroforma ($W_{struct} = +3.0$):** Se o algoritmo DSP[15] (Essentia) detecta múltiplas seções distintas (ex: Intro, Verso, Refrão), confirma-se que o usuário produziu um arranjo completo. A capacidade de organizar ideias musicais em uma macroestrutura coerente é, historicamente, um dos indicadores finais de

[15] A biblioteca *Essentia* é uma ferramenta de código aberto escrita em C++ com *bindings* para Python, voltada para o Processamento Digital de Sinais (DSP) e Recuperação de Informação Musical (MIR). Ela permite a extração de atributos musicais objetivos através de algoritmos determinísticos e modelos de aprendizado profundo (BOGDANOV et al., 2013; DOWNIE, 2003).

fluência musical (KRATUS, 1994), diferenciando o estudante que experimenta sons daquele que compõe obras.

A heurística implementada corrige o viés de rebaixar o uso de *samples* a uma mera operação de "Aplicação" na Taxonomia de Bloom. No algoritmo `classificador_mestre.py`, o peso estático de *plugins* nativos como o `AudioFileProcessor` é complementado dinamicamente por uma análise de contexto. Se o *sample* for acompanhado por cadeias de efeitos criativos (*FX Chain*) ou curvas de modulação (*Automations*), o algoritmo identifica a desconstrução cognitiva do material de áudio original. Nestas condições, a pontuação agregada no cálculo de nível ponderado (`calcular_nivel_ponderado`) equipara a técnica de *chopping* e *time stretching* à carga cognitiva da síntese sonora pura. Isso honra a fundamentação histórica do *Hip Hop*, onde manipular um *sample* na MPC é um ato de "Criação" (nível mais alto de Bloom), elevando a acurácia didática do sistema na separação entre produtores iniciantes e intermediários. Desta forma, o sistema busca atuar como uma "Zona de Desenvolvimento Proximal" (VYGOTSKY, 1978) automatizada, sugerindo aos novatos projetos que tenham fácil assimilação (*loops* simples) e, gradativamente, expondo-os a técnicas de engenharia mais densas conforme sua evolução na plataforma. Mas sem limitar o acesso, pois todos os projetos estão disponíveis. A curiosidade é o limite.

2.2 Computação na Borda e Interoperabilidade

A exclusão digital impõe barreiras técnicas significativas à produção musical independente. O mercado atual divide-se majoritariamente em dois modelos que, embora tecnologicamente avançados, conflitam com a realidade de infraestrutura do artista independente. O *Software* Proprietário, cujos custos de licenciamento fomentam a pirataria e a dependência de sistemas operacionais específicos, os Serviços em Nuvem, que centralizam o processamento em servidores remotos (*Software as a Service*, *Software* como Serviço – *SaaS*). Embora resolvam a questão da instalação local, as soluções baseadas inteiramente em nuvem introduzem uma dependência crítica, a latência de rede (Figura 19). Para a performance musical, onde a precisão rítmica é medida em milissegundos, o tempo de viagem do pacote de áudio até um servidor para renderização introduz atrasos inaceitáveis, inviabilizando a "*jam session*" em tempo real em conexões instáveis.

Para mitigar esses gargalos, a arquitetura desta plataforma afasta-se do modelo tradicional de Cliente Magro (*Thin Client*) em favor de uma arquitetura de Computação na Borda (SHI et al., 2016). O subsistema *mmpCreator* materializa o conceito de Soberania na Borda adotando o paradigma Cliente Gordo (*Thick Client*). Ao carregar um projeto, a aplicação reconstrói o grafo de áudio integralmente na memória do navegador do usuário utilizando a Web Audio API[16] (ADENOT; WILSON; ROGERS, 2021), acessando

[16] Considerada como uma interface *JavaScript* de alto nível. É integrada aos navegadores modernos, para

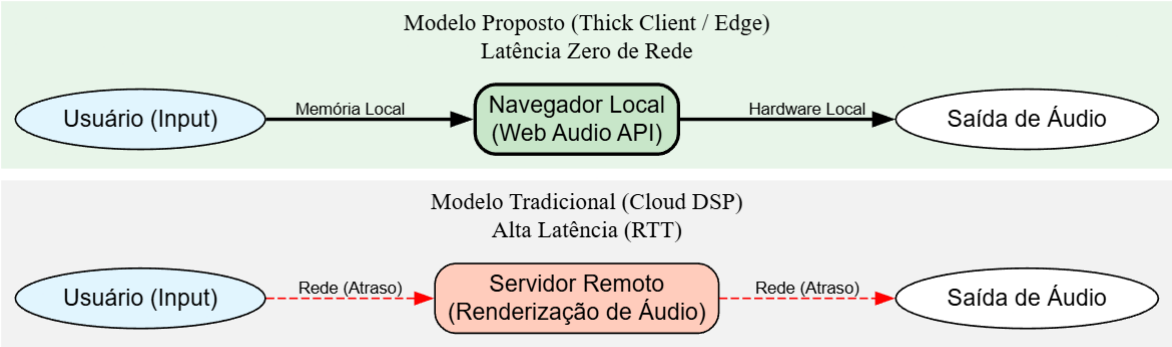

Figura 19 – Comparativo teórico de latência entre as possíveis abordagens.

Fonte: O Autor (2026).

diretamente a CPU (*Central Processing Unit*, ou Unidade Central de Processamento) do dispositivo cliente (Figura 20). Essa inversão de fluxo garante a autonomia do artista e proporciona vantagens diretas à aplicação:

- A latência de interação (tocar uma nota, girar um botão) é mitigada, pois não há viagem de rede (*round-trip*) para a síntese primária, ocorrendo de forma fluida mesmo mediante oscilações de banda.
- A geração sonora utiliza a CPU do dispositivo local, permitindo que a aplicação escale horizontalmente sem sobrecarregar o servidor central.
- A privacidade é preservada, pois a experimentação sonora ocorre em ambiente isolado (*sandbox*) até que o usuário decida explicitamente sincronizar seu progresso com a rede.

Neste cenário, o servidor deixa de ser um processador de áudio para atuar apenas como um sincronizador de estados. O sistema opera através de um fluxo de dados unidirecional que desacopla a ingestão da distribuição (Figura 21). Diferente de aplicações dinâmicas tradicionais, esta arquitetura compila a base de dados em arquivos estáticos JSON, lidos diretamente pelo navegador do usuário. O ciclo de vida da informação segue os seguintes estágios:

- **Camada de Persistência (Repositório):** Os projetos musicais são armazenados como arquivos brutos em um sistema de arquivos versionado. Esta camada atua como a "Fonte da Verdade" (SSOT).

processar, sintetizar e controlar áudio diretamente na *web*. Crucial para nossa ideia de implementação.

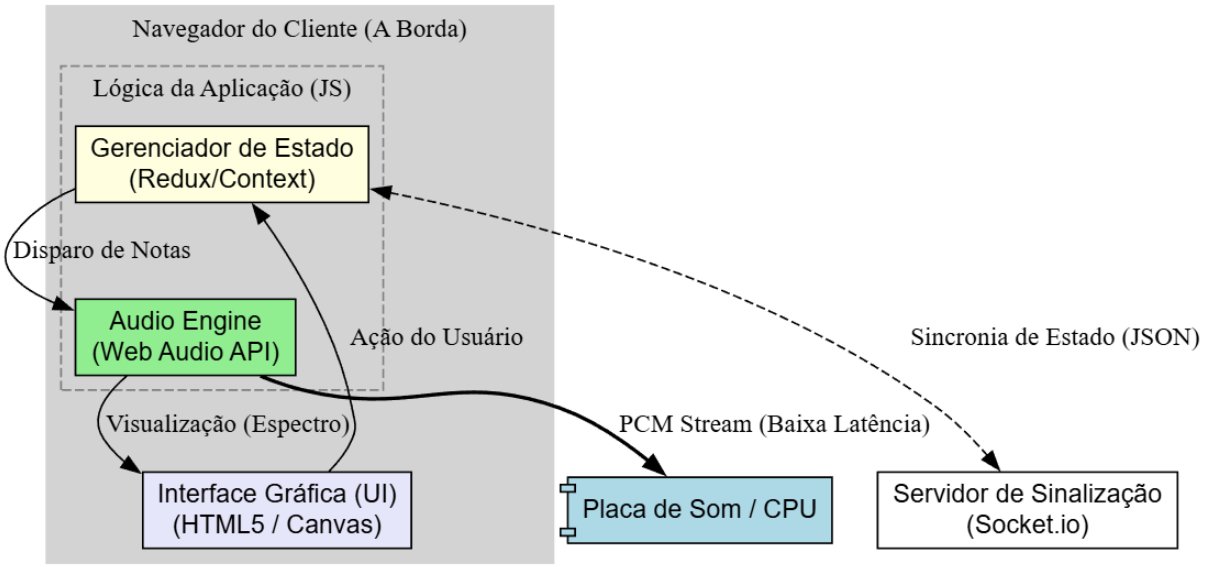

Figura 20 – Arquitetura do Cliente Magro (*Thick Client*) proposto.

Fonte: O Autor (2026).

- **Camada de Ingestão (backend/ETL):** O agente em Python processa os metadados XML e analisa os arquivos de áudio via Essentia e Tensorflow, gerando um perfil técnico e pedagógico completo da faixa.
- **Camada de Distribuição (API Estática):** Os resultados da ingestão são normalizados e serializados em `.json`. Estes arquivos alimentam o motor de busca do *mmpSearch* com latência virtualmente nula, características essenciais do *Jamstack*.
- **Camada de Execução (Cliente/Nó):** A aplicação final (*mmpCreator*) consome os projetos e instancia os osciladores locais na *Web Audio API*.

Enquanto a Computação na Borda resolve o problema da performance[17] para o usuário individual, a construção do repositório colaborativo capaz de indexar milhares de projetos (*Big Data* Musical) impõe um desafio de escala no *backend*. A tarefa de processar cada arquivo é independente e não compartilha estado mutável com outros, caracterizando um problema perfeitamente paralelizável, (Figura 22).

Aplicando o modelo de Paralelismo de Dados, a proposta algorítmica de ingestão particiona o conjunto de arquivos e os distribui dinamicamente entre múltiplos núcleos

[17] É necessário ter em mente que quanto mais complexa a composição for, mais "pesada" ela vai ficar no navegador. Isso também pode trazer problemas para os artistas com máquinas que não sejam equipadas com muita memória *RAM*. Contudo, a aplicação não visa ser uma DAW complexa com inúmeros efeitos. A intenção é que sua utilização seja leve. Isso será discutido de forma mais ampla no Capítulo 5.

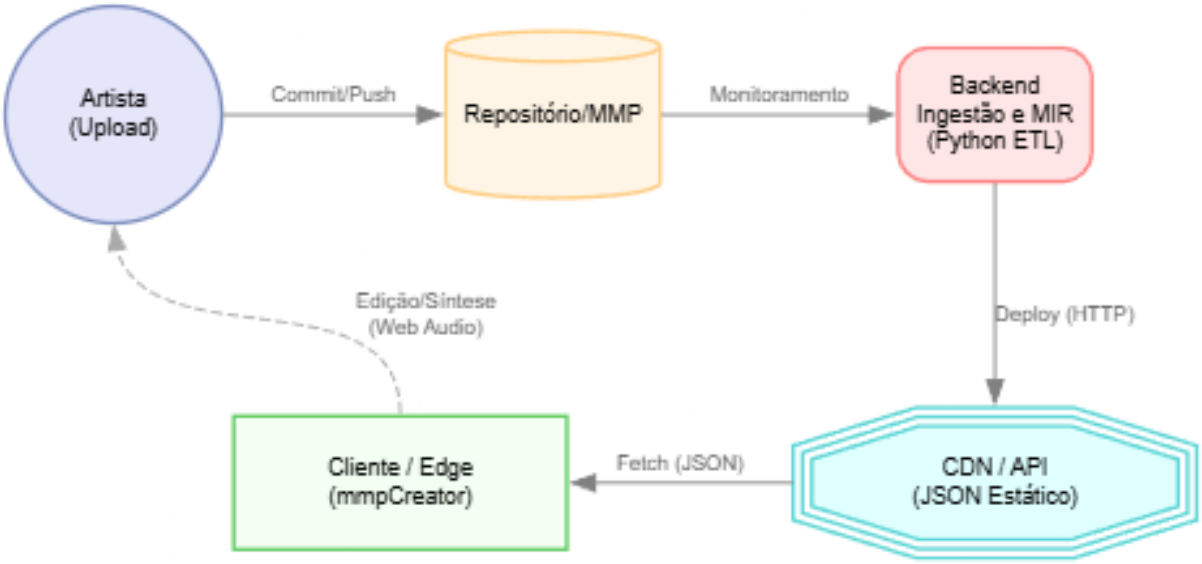

Figura 21 – Fluxo de dados proposto.

Fonte: O Autor (2026).

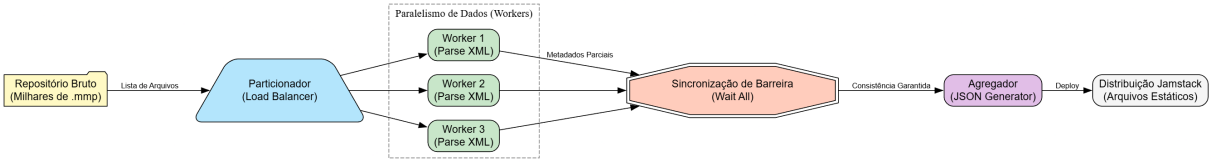

Figura 22 – Arquitetura do paralelismo proposto para indexação dos projetos na plataforma.

Fonte: O Autor (2026).

de processamento. O ganho de desempenho teórico aproxima-se da linearidade, sendo limitado principalmente pela velocidade de I/O do sistema de armazenamento, conforme previsto pela Lei de Amdahl (AMDAHL, 1967). Para garantir a consistência do índice final, utiliza-se o padrão de Sincronização de Barreira, assegurando que a agregação dos metadados ocorra estritamente após a conclusão de todos os processos operários. A interoperabilidade do sistema é garantida pela adesão estrita a padrões abertos. O formato de arquivo nativo do *software* LMMS, baseado em XML (MMP), serve como o protocolo primário de intercâmbio de dados. A arquitetura assume que cada projeto musical é, essencialmente, um grafo de objetos serializado. O esquema de dados (XSD) foi mapeado para permitir que o sistema identifique nós XML específicos, como `<instrument track>` ou `<pattern>`. O Código 2.3 ilustra o processo de transformação de dados que ocorre durante a ingestão, convertendo a estrutura hierárquica complexa do XML em objetos JSON otimizados para a Web.

```
// Entrada: Fragmento XML do LMMS
<instrumenttrack name="Bassline">
    <instrument name="lb302">
        <lb302 dist="0.8" slide="1" ... />
    </instrument>
    <pattern pos="0" steps="101010..." />
</instrumenttrack>

// Saida: Objeto JSON Indexado (mmpSearch)
{
    "track_id": "Bassline",
    "type": "synth",
    "plugin": "lb302", // Extraido para filtro de plugins
    "complexity": {
        "has_distortion": true,
        "is_acid": true
    },
    "rhythm_fingerprint": "101010..." // Usado na busca visual
}
```

Listing 2.3 – Exemplo simplificado de mapeamento XML para JSON (Conceitual)

Esta decisão de projeto permite que a plataforma2 funcione não apenas como um reprodutor de mídia. Funciona também como um sistema de preservação digital e análise musicológica. A estrutura lógica da composição é mantida acessível para mineração de dados, independentemente da plataforma de áudio utilizada pelo usuário final.

2.3 Sincronização e Consistência em Sessões Colaborativas

Enquanto a síntese sonora ocorre isoladamente no cliente, a funcionalidade de colaboração em tempo real exige uma sincronização precisa de estado entre todos os participantes. Para isso, desenvolveu-se um protocolo proprietário sobre *WebSocket*, utilizando uma topologia em estrela onde o servidor atua como *hub* autoritativo de mensagens e orquestrador de tempo. O posicionamento do uso de *WebSockets*, em detrimento de abordagens puramente diretas entre os clientes como o protocolo *WebRTC*, foi uma escolha arquitetural deliberada e necessária para esta fase do projeto. Conforme implementado no módulo `server.js`, a topologia em estrela permite que o servidor atue não apenas como um retransmissor de mensagens, mas como a Fonte Única da Verdade (SSOT) através de persistência autoritativa no sistema de arquivos local. Rotinas de controle de sala garantem que, em caso de desconexão em massa ou falha dos clientes, o estado da composição esteja salvo e pronto para re-hidratação. Em uma arquitetura puramente P2P via *WebRTC*, a queda simultânea dos nós resultaria na perda irrecuperável do projeto colaborativo. Portanto, a topologia em estrela garante a resiliência dos dados, mitigando a

complexidade de algoritmos de resolução semântica distribuída (como os CRDTs), que ficam mapeados como trabalhos futuros (Capítulo 5). Embora o *WebRTC* oferecesse menor latência ao transferir metadados diretamente entre navegadores, a topologia em estrela garantiu a consistência rigorosa necessária para validar a primeira versão da plataforma colaborativa. A implementação deste protocolo não busca apenas resolver um problema técnico de latência, na verdade, busca viabilizar a prática cultural da *jam session* em ambiente digital. No contexto do *Underground*, o "*groove*" (a sensação subjetiva de coesão rítmica) é o elemento central. Um atraso de rede não tratado destruiria essa coesão, transformando a música em ruído. Portanto, a engenharia de sincronização aqui apresentada é a base tecnológica para a manutenção da fluidez criativa.

Para que a reprodução musical seja coesa, é imperativo que todos os clientes compartilhem uma noção unificada de tempo. Confiar exclusivamente no relógio do sistema operacional (*wall clock*) de cada usuário é inviável devido ao fenômeno de *clock drift*, onde oscilações no *hardware* causam desalinhamentos progressivos (MILLS, 1991). O sistema implementa um algoritmo de sincronização baseado em amostragem periódica do tempo do servidor via `socket.js`. A função `sampleServerTime` calcula o *Round Trip Time* (RTT) e estima o deslocamento (*offset*) entre o relógio local e o remoto. Para mitigar a latência, aplica-se um filtro de Média Móvel Exponencialmente Ponderada (EWMA), originalmente formalizado por Holt (2004):

$$Offset_t = \alpha \cdot O_{amostra} + (1 - \alpha) \cdot Offset_{t-1} \tag{2.1}$$

A escolha do fator $\alpha = 0.3$ fundamenta-se no equilíbrio entre a supressão de ruído e a agilidade de convergência. Esta abordagem segue a filosofia clássica de estimativa de RTT do protocolo TCP (RFC 6298) (PAXSON et al., 2011). Embora protocolos de transporte modernos, como o QUIC (RFC 9000) (IYENGAR; THOMSON, 2021) e algoritmos de controle de congestionamento baseados em modelo, notadamente o Google BBR (CARDWELL et al., 2017), utilizem filtros estatísticos robustos como Filtros de Kalman (KALMAN, 1960) para modelar a variância estocástica, a implementação de tais complexidades na camada de aplicação introduziria um custo computacional desproporcional. Para o contexto de *Audio Jamming*, onde a prioridade é a estabilidade de fase e não a maximização de *throughput*, o algoritmo de Alisamento Exponencial (HOLT, 2004) prova-se suficiente. Ele oferece uma solução leve, deterministicamente estável e de fácil depuração. A Tabela 3 e a Figura 23 demonstram a eficácia desta escolha.

Didaticamente, se o sistema possui um $Offset_{t-1}$ consolidado de 100ms e recebe um pico de 200ms, o novo valor será $Offset_t = 0.3(200) + 0.7(100) = 130$ms. Note que, embora a amostra tenha dobrado, o ajuste aplicado foi conservador (30ms). Essa inércia matemática garante um "tempo absoluto virtual" estável. Vale ressaltar que, para cenários futuros envolvendo redes de altíssima latência ou conexões via satélite, a substituição

Tabela 3 – Evolução do Offset (ms) perante um pico de latência ($t = 1$).

Amostra (t)	**Bruto (ms)**	$\alpha = 0.1$ **(Lento)**	$\alpha = 0.3$ **(Ideal)**	$\alpha = 0.8$ **(Instável)**
0 (Estável)	100,0	100,0	100,0	100,0
1 (**Pico**)	**200,0**	110,0	**130,0**	180,0
2 (Normal)	100,0	109,0	121,0	116,0
3 (Normal)	100,0	108,1	114,7	103,2
4 (Normal)	100,0	107,3	110,3	100,6

Fonte: O Autor (2026).

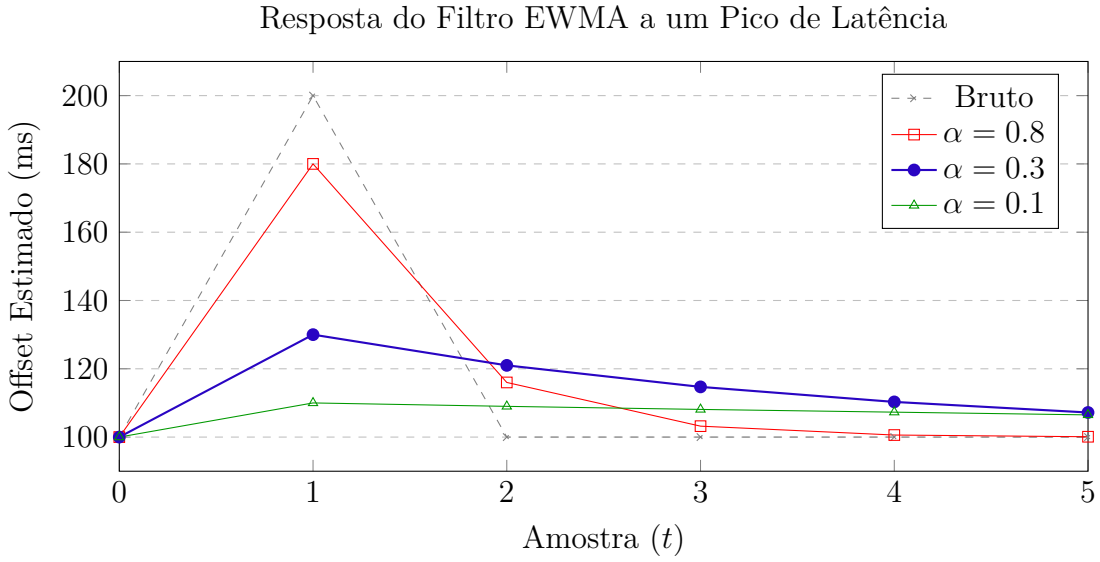

Figura 23 – Comparativo de filtros: o valor 0.3 amortece o pico de rede sem criar a inércia excessiva observada em 0.1.

do EWMA por algoritmos preditivos (como Filtros de Kalman) é discutida como uma oportunidade de melhoria no Capítulo 5.

Controle de Concorrência: UI Otimista e Consistência Eventual - Para viabilizar a colaboração musical em tempo real, onde a fluidez cognitiva é prioritária, a arquitetura rejeita o modelo tradicional de bloqueio pessimista (onde a interface trava aguardando o servidor). Em seu lugar, adota-se uma estratégia de Interface do Usuário Otimista (*Optimistic UI*) combinada com um modelo de consistência eventual autoritativo. Essa abordagem metodológica foi desenhada para ocultar a latência de rede inerente à Internet pública, permitindo que o músico perceba resposta instantânea às suas ações de edição. A integridade dos dados é assegurada posteriormente pelo servidor, que atua como árbitro final de conflitos. A Figura 24 detalha o protocolo de comunicação projetado para garantir essa robustez, implementado nos módulos `socket.js` (cliente) e `server.js` (servidor):

1. **Execução Especulativa Local:** Ao realizar uma ação (ex: mover um clipe de

áudio), o cliente aplica a alteração imediatamente em seu estado local e atualiza a renderização visual. Gera-se um ID temporário e um *token* de transação único, permitindo que a ação exista localmente antes de ser confirmada.

2. **Estratégia de *Fallback* e Retransmissão:** A ação encapsulada é enviada ao servidor. Metodologicamente, definiram-se janelas de tempo críticas para tratamento de falhas: um temporizador de 500ms aguarda o reconhecimento (ACK) imediato, e um segundo temporizador de 900ms monitora a consistência global. Se o servidor falhar em responder, o cliente reverte a ação ou notifica a instabilidade, mas sem bloquear a interface principal.

3. **Arbitragem Autoritativa e Idempotência:** Para evitar a corrupção do estado musical (ex: duplicidade de faixas devido a retransmissões de rede), o servidor implementa um mecanismo estrito de idempotência. Antes de processar qualquer mensagem, o sistema verifica se o *token* da transação já consta no conjunto de operações realizadas (`tokensSeen`). Adicionalmente, aplica-se uma validação semântica para rejeitar coordenadas temporais inválidas ou referências a objetos inexistentes.

4. **Serialização e Convergência de Estado:** As ações validadas recebem um número de sequência oficial (`__seq`) e são persistidas. O servidor então realiza o *broadcast* da ação autorizada. Ao receberem esta mensagem, todos os clientes da sala (inclusive o remetente original) atualizam seus estados para refletir a "verdade" sequenciada pelo servidor, garantindo que todos os participantes convirjam para a mesma versão do arranjo musical, mesmo que suas ordens de chegada tenham variado devido à latência da rede.

Esta arquitetura demonstra que é possível mitigar os efeitos da latência na experiência do usuário sem sacrificar a consistência dos dados, um requisito não funcional crítico para sistemas de coautoria multimídia.

Gestão de Estado, Persistência e Recuperação (*Late Join*) - Um desafio crítico em sistemas distribuídos de longa duração é garantir a consistência para usuários que ingressam na sessão após seu início (*Late Joiners*) ou em casos de reinicialização do servidor. A arquitetura soluciona este problema através de uma estratégia de Hidratação de Estado em Duas Camadas, implementada no servidor via persistência híbrida (Memória *RAM* e Sistema de Arquivos). Diferente de abordagens monolíticas que exigiriam a retransmissão de todo o histórico de eventos (o que causaria gargalos de processamento), o sistema mantém um "estado atual consolidado" na memória do servidor. O processo de sincronização para um novo usuário, orquestrado pela função `join_room` no *backend*, ocorre em duas etapas distintas de granularidade:

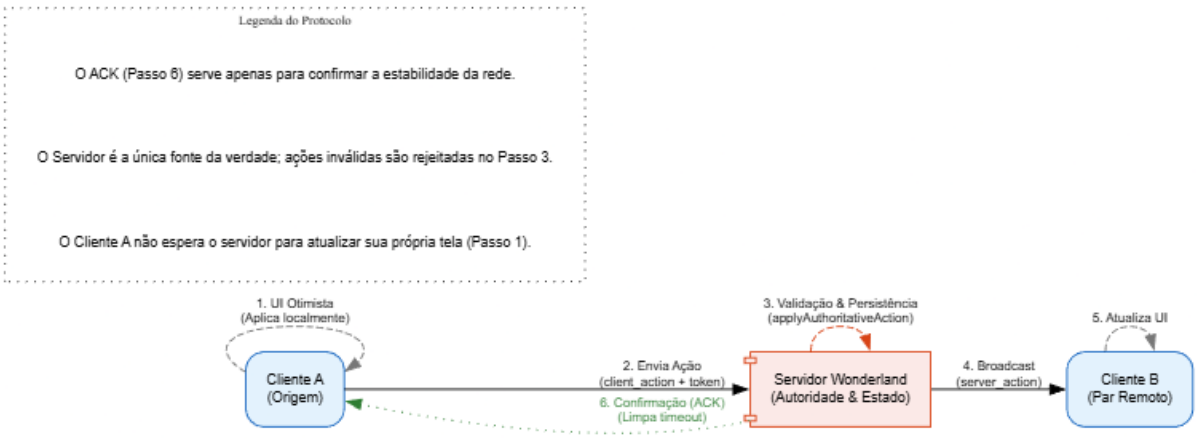

Figura 24 – Fluxo de consistência: O modelo híbrido utiliza execução especulativa na borda (cliente) e serialização rigorosa no núcleo (servidor) para manter a coesão da *jam session*.

Fonte: O Autor (2026).

1. **Camada Estrutural (Estática/Pesada):** Primeiramente, o servidor transmite a estrutura base do projeto em formato XML (`.mmp`). Este arquivo define o grafo de processamento de sinal (DSP), contendo a configuração dos sintetizadores virtuais e cadeias de efeitos (se houverem). Por ser um dado denso e de baixa frequência de alteração, ele atua como o "esqueleto" imutável da sessão até que uma alteração estrutural profunda ocorra.

2. **Camada Transacional (Dinâmica/Leve):** Imediatamente após a carga estrutural, o sistema envia um *snapshot* em formato JSON (`AUDIO_SNAPSHOT`). Este objeto representa o estado volátil da *timeline* (posicionamento de clipes, cortes e automações). Crucialmente, este pacote inclui o identificador de sequência atual (`__seq`) da sala.

A arquitetura adota essa estratégia de hidratação com a finalidade de manter todos sincronizados, mas buscando não exceder em memória, prezando por persistir informações cruciais para a composição. Essa separação garante a eficiência da rede, onde o tráfego pesado (XML) ocorre apenas na entrada, enquanto a sincronização fina (JSON) mantém a latência baixa. No cenário de falha, a sincronização temporal (*play*/*pause*) pode se perder, mas a composição e as definições que foram definidas estarão presentes para qualquer um dos pares conectados[18]. Essa separação garante a eficiência da rede, onde o tráfego

[18] Neste caso, podemos dizer que cada um dos pares pode criar sua própria versão da composição de maneira *offline*. Mas na hora do retorno à colaboração online o estado salvo pelo servidor será o que estará disponível para alterações.

pesado (XML) ocorre apenas na entrada, enquanto a sincronização fina (JSON) mantém a latência baixa.

Analogamente a uma partitura musical, o XML define a orquestração e os instrumentos (quem toca o quê), enquanto o JSON representa a regência em tempo real (quando entrar, dinâmica, andamento). Não é necessário reescrever a partitura inteira a cada compasso, tocado basta atualizar a posição do regente. Além disso, a implementação da função `ensureRoom` no módulo `server.js` confere resiliência à aplicação. O estado da sala não reside apenas na memória volátil. Cada alteração crítica aciona uma rotina de persistência em disco (`saveRoom`). Desta forma, se o serviço for reiniciado, a próxima requisição de entrada "re-hidrata" a memória *RAM* a partir do sistema de arquivos JSON, garantindo que a obra colaborativa sobreviva a falhas de infraestrutura, uma característica essencial para a preservação da produção cultural digital.

2.4 Considerações Finais

Neste capítulo, delineamos a fundação metodológica e arquitetural da plataforma, pautada em uma abordagem de pesquisa aplicada e iterativa. A estratégia de engenharia reversa aplicada aos arquivos de projeto do LMMS provou-se o ponto de inflexão do estudo, viabilizando a transição de um formato binário de uso restrito para uma estrutura de dados interoperável. Essa transformação é o que permite a extração semântica, a taxonomia automatizada e a indexação inteligente da informação musical pelo *mmpSearch*.

Além disso, a adoção da Computação na Borda para a síntese de áudio, associada a protocolos rigorosos de sincronização via *WebSockets* e gestão de estado em duas camadas, consolidou a base técnica necessária para contornar gargalos de latência. Essa arquitetura garante a integridade da obra colaborativa (*mmpCreator*) sem sacrificar a fluidez criativa e cognitiva inerente às *jam sessions*, respeitando as limitações de infraestrutura do artista independente.

Tendo estabelecido os alicerces teóricos, os contratos de dados (XSD), as lógicas de orquestração sobre o *Wonderland* e as heurísticas de recuperação musical, o próximo passo natural é a materialização destes modelos. O Capítulo 3 detalha a fase de implementação do projeto. Nele, sairemos do escopo arquitetural para discutir os detalhes técnicos, os artefatos de *software* desenvolvidos, as bibliotecas aplicadas na prática e como as soluções aqui propostas foram efetivamente codificadas, validadas e formalizadas na plataforma construída.

3 Visão Geral da Implementação Distribuída

Este capítulo detalha a materialização da plataforma proposta, descrevendo os componentes de *software* desenvolvidos para atender aos requisitos de escalabilidade, resiliência e processamento distribuído estabelecidos nas diretrizes metodológicas e infraestrutura de orquestração (Capítulo 2). A implementação concretiza a arquitetura de referência *Wonderland*, traduzindo os modelos conceituais de fluxo de dados em microsserviços operacionais.

A solução divide-se em dois subsistemas principais, operando em extremidades opostas da rede: 1) o Agente de Ingestão Paralela (*mmpSearch backend*), responsável pela mineração e estruturação de dados no servidor central; e 2) a Interface de Processamento na Borda (*mmpCreator* e *mmpSearch Frontend*), que transfere a carga computacional de síntese sonora para o cliente.

A Tabela 4 apresenta as especificações do ambiente de homologação utilizado, validando a premissa de que a arquitetura proposta é capaz de operar em *hardware* de médio porte, comum em laboratórios universitários.

Tabela 4 – Especificações do Ambiente Experimental (Servidor *Wonderland*)

Componente	**Especificação**
Sistema Operacional	Ubuntu 24.04.3 LTS
Processador (CPU)	Intel(R) Xeon(R) E5-2620 v2 @ 2.10GHz
Arquitetura	x86_64
Núcleos	6 Físicos / 12 Lógicos (*Threads*)
Memória RAM	7,7 GiB
Armazenamento	SSD SATA III (*Throughput* médio de 500 MB/s)

Fonte: O Autor (2025).

Para viabilizar a autonomia do artista independente e mitigar custos de manutenção de infraestrutura centralizada, a implementação seguiu estritamente o padrão arquitetural *Jamstack* (*JavaScript, APIs* e *Markup*). Esta abordagem, formalizada por (BIILMANN; HAWKSWORTH, 2019), permite desacoplar a lógica de apresentação da lógica de persistência, movendo a complexidade do tempo de execução para o tempo de construção (*build time*).

Do ponto de vista de Engenharia de *Software*, o sistema adota o estilo arquitetural *Pipes and Filters* (BUSCHMANN et al., 1996). Os dados fluem unidirecionalmente através de estágios de processamento independentes, onde a saída de um componente torna-se a entrada do próximo, garantindo previsibilidade e facilidade de depuração:

1. **Camada de Persistência - *Source*:** Projetos musicais (`.mmp`) são tratados como código-fonte, armazenados em sistema de arquivos e versionados.

2. **Camada de Processamento (Python) - *Filter*:** Atuando como o componente de filtro, scripts no servidor realizam a ingestão, validação e conversão dos projetos de forma paralela, contornando gargalos de I/O.

3. **Camada de Distribuição (CDN) - *Pipe*:** Os metadados transformados são servidos como APIs estáticas (JSON/YAML), eliminando a necessidade de bancos de dados relacionais em tempo de execução.

4. **Camada de Execução (*Browser*) - *Sink*:** O navegador atua como o consumidor final, onde a lógica de DSP (*Digital Signal Processing*) é executada em tempo real via *Web Audio API*.

3.1 *Pipeline* ETL (*Extract, Transform, Load*)

O componente de *backend* (*mmpSearch*) foi desenvolvido para resolver o desafio de "*Big Data* Musical" e processar milhares de arquivos de projeto XML[1], para extrair conhecimento estruturado. Dado o volume de dados e a natureza intensiva das tarefas de renderização de áudio, implementou-se um *pipeline* de processamento paralelo robusto.

Antes do processamento, foi necessário popular o repositório com dados reais da comunidade. Para isso, desenvolveu-se um agente de coleta automatizada (*Web Crawler*) em Python [2], projetado para varrer plataformas de compartilhamento de projetos LMMS (LSP), adotando os princípios de arquitetura de coleta e polidez definidos por (MANNING; RAGHAVAN; SCHÜTZE, 2008).

A coleta de dados é uma tarefa intensiva em I/O (*Input/Output*), onde o gargalo é a latência da rede e não o processador. Para otimizar essa etapa, a implementação utilizou o modelo de Concorrência via *Threads*. Conforme explica (RAMALHO, 2022), em ambientes *Python*, o uso de *threading* é a estratégia recomendada para operações bloqueantes de I/O. Mesmo sob a restrição do GIL, as *threads* permitem que o interpretador libere a execução para outras tarefas enquanto aguarda respostas de rede, maximizando o rendimento (*throughput*).

O Código 3.1 demonstra o uso do `ThreadPoolExecutor`. Ao instanciar múltiplos *workers*[3], o sistema consegue realizar múltiplas requisições HTTP (*Hypertext Transfer*

[1] Na verdade a extensão do arquivo é `.mmp`, mas é formatado como um arquivo XML.

[2] *Crawler* desenvolvido para fazer *download* automático dos projetos na comunidade do LMMS (LMMS, 2025). Disponível em: <https://git.alice.ufsj.edu.br/JotaChina/mmpSearch/src/branch/main/scripts/handler/crawler.py>. Acesso em MAR de 2026.

[3] O valor 8 foi definido como medida de precaução, pois já é um volume considerável de requisições em um curto espaço de tempo. Desta forma, considera-se que 8 é um valor seguro e aceitável, livre de

Protocol) simultaneamente. Enquanto uma *thread* aguarda a resposta do servidor remoto, outras *threads* continuam baixando arquivos, maximizando a taxa de transferência.

```
# Configuracao de Concorrencia para I/O Bound
MAX_WORKERS = 8

def baixar_projeto_worker(url):
    # ... Logica de Request e tratamento de erros HTTP ...
    response = requests.get(url, headers=HEADERS, stream=True)
    if response.status_code == 200:
        with open(caminho_arquivo, 'wb') as f:
            for chunk in response.iter_content(chunk_size=8192):
                f.write(chunk)
        return 1, f"Sucesso: {nome_arquivo}"
    return 0, f"Erro {response.status_code}"

# Orquestrador de Threads
with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
    futures = []
    for url in urls_unicas:
        # Submete a tarefa para o Pool de Threads
        futures.append(executor.submit(baixar_projeto_worker, url))

    for future in as_completed(futures):
        status, msg = future.result()
        registrar_log(msg)
```

Listing 3.1 – Logica de Coleta Concorrente (crawler.py)

O agente inclui mecanismos de robustez, como tratamento de exceções de rede e registro detalhado de *logs* (`historico_detalhado.log`), garantindo que falhas pontuais de conexão não interrompam o processo de coleta em massa.

O *script* orquestrador (***`main.py`***) implementa um padrão de projeto *Pipeline*, adotando a arquitetura de processamento de dados ETL (*Extract*, *Transform*, *Load*) fundamentada por (KIMBALL; ROSS, 2013). O processo divide-se em estágios discretos:

- **Extração (*Extract*):** O sistema percorre recursivamente a árvore de diretórios do repositório, identificando arquivos compactados (`.mmpz`) e não compactados (`.mmp`). Utiliza-se verificação de *hash* (MD5) para processar apenas arquivos modificados desde a última execução, otimizando o uso de recursos (estratégia de *build* incremental).
- **Transformação (Transform):** Esta etapa, executada pelo módulo `file_parser.py`, envolve a descompressão em memória e o *parsing* da árvore DOM (*Document Object*

possíveis bloqueios por parte do *host*, no caso, o *site* da plataforma de distribuição do LMMS (LMMS, 2025).

Model) do XML. O algoritmo navega pela hierarquia `<trackcontainer>` identificando nós específicos de instrumentos (ex: `<kicker>`, `<organic>`). Diferente de uma leitura de texto simples, o *parser* normaliza atributos de envelopes (ADSR) e filtros encontrados dentro das *tags* `<eldata>`, convertendo dados heterogêneos em uma estrutura *Python* padronizada.

- **Carga (*Load*):** Os dados estruturados são serializados em JSON e organizados em diretórios públicos acessíveis pelo servidor *web*.

O Código 3.2 demonstra a lógica heurística utilizada para identificar qual *plugin* está sendo utilizado em uma trilha, uma vez que o formato LMMS aninha a definição do sintetizador como um filho direto da *tag* `<instrumenttrack>`.

```
def parse_mmp_file(file_path):
    # ... (Inicializacao e parsing da arvore XML)

    # Navegacaoo heuristica na arvore DOM
    instruments = track.findall("./instrumenttrack")
    for instrumenttrack in instruments:
        instrument_node = instrumenttrack.find("./instrument")

        # O LMMS usa o nome do plugin como tag filha (ex: <lb302>)
        # O parser tenta encontrar a tag especifica ou usa fallback
        if len(list(instrument_node)) > 0:
            child = list(instrument_node)[0]
            # Extrai atributos como 'vol', 'pan', 'wave', etc.
            data[plugin_name] = child.attrib

        # Extracao de Envelopes (ADSR) compartilhados
        eldata_node = instrumenttrack_element.find("./eldata")
        if eldata_node is not None:
            data["eldata"] = eldata_node.attrib
```

Listing 3.2 – Algoritmo de extracao de atributos de sintetizadores (file_parser.py)

Além da extração de metadados, ocorre a geração de "Dados Derivados" (KLEPPMANN, 2017) e a renderização de prévias de áudio (`.ogg`) utilizando o binário do LMMS em modo *headless* (sem interface gráfica), conforme ilustrado no Código 3.3. Esta abordagem garante que o ativo sonoro seja uma representação fiel e determinística do código-fonte musical.

```
def process_single_file(args):
    # ...
    # Comando CLI para renderizar o projeto (.mmp) em audio (.ogg)
    # sem abrir a interface grafica (QT_QPA_PLATFORM="offscreen")
    ogg_cmd = [
```

```
        "lmms",
        "-r", abs_target_mmp,  # Arquivo de entrada
        "-o", abs_ogg_out,     # Arquivo de saida
        "-f", "ogg"            # Formato
    ]

    try:
        # Executa o binario do LMMS e aguarda conclusao
        subprocess.run(ogg_cmd, check=True, capture_output=True)
        logging.info(f"Audio OGG gerado: {ogg_name}")
    except subprocess.CalledProcessError as e:
        logging.warning(f"Falha na renderizacao: {e.stderr}")
```

Listing 3.3 – Chamada de Renderizacao *Headless* do LMMS via Subprocess

A Figura 25 apresenta de forma visual como é o processo sequencial do *Pipeline* ETL que foi implementado durante o desenvolvimento deste trabalho.

Para complementar a coleta passiva do *Crawler*, a plataforma implementa uma interface de contribuição ativa via *API REST*. Um *middleware* de serviços escuta requisições nos *endpoints* `/api/upload` e `/api/auth`. Este componente atua como a "cola" entre o formulário *web* (descrito na Seção 3.3) e o sistema de arquivos do servidor. Ao receber um novo projeto via `POST`, o *middleware* realiza a validação de segurança (`login.py`), persiste o arquivo na área de *staging* e notifica o orquestrador ETL para iniciar o processamento incremental, garantindo que a contribuição do usuário seja indexada em tempo quase real. Tendo em vista os resultados apresentados nos Capítulos 4 e 5, esta abordagem possui um grande potencial para aprendizagem, criação e colaboração entre artistas. Essa colaboração pode ser síncrona, como em *jams* colaborativas em tempo real ou pode ser assíncrona, como nos casos de consultas de projetos, buscas por referências ou artefatos.

Enquanto o *Crawler* (Seção 3.1) utilizou *Threads* para operações de I/O, o processamento dos arquivos XML e a renderização de áudio são tarefas intensivas em CPU. A implementação em *Python* enfrenta o desafio do *Global Interpreter Lock* (GIL), que limita a execução de *threads* a um único núcleo de CPU. Para superar essa limitação e maximizar o uso do *hardware* descrito na Tabela 4, adotou-se o modelo de Paralelismo de Processos, seguindo as práticas de computação de alto desempenho em *Python* preconizadas por (GORELICK; OZSVALD, 2020).

Utilizando a biblioteca `multiprocessing`, a arquitetura materializa o padrão de projeto *Master-Worker* (SCHMIDT et al., 2000). O orquestrador detecta o número de núcleos disponíveis (N_{cores}) e instancia um *Pool* de processos operários (*workers*). A lista de arquivos é distribuída através da função *pool.map*, que delega as tarefas aos núcleos disponíveis. Esta escolha é crucial para o balanceamento de carga em tarefas de granularidade irregular, conforme definido por Grama et al. (2003), onde arquivos complexos (com

muitas trilhas e automações) demoram mais para serem analisados, enquanto arquivos simples são rápidos. O *scheduler* dinâmico garante que, assim que um núcleo fica ocioso, ele recebe uma nova tarefa imediatamente, mantendo a ocupação da CPU próxima de 100%.

```
def main_parallel():
    # ... (inicializacao de logs e dependencias)

    # Deteccao de nucleos fisicos e logicos
    num_cores = multiprocessing.cpu_count()
    logging.info(f"Processando {len(tasks)} arquivos com {num_cores}
   workers.")

    # Instanciacao do Pool de Processos (Bypass do GIL)
    # O metodo 'map' distribui as tarefas (arquivos) entre os nucleos
    with multiprocessing.Pool(processes=num_cores) as pool:
        results = pool.map(process_single_file, tasks)

    # Agregacao dos resultados (Barrier Synchronization)
    successful_data = [r["data"] for r in results if r["success"]]
    # ...
```

Listing 3.4 – Trecho do Orquestrador de Ingestao Paralela (*main.py*)

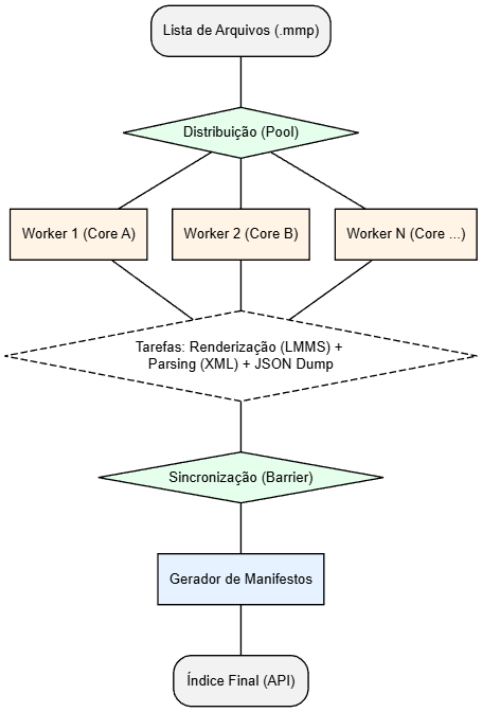

Figura 25 – *Pipeline* ETL.

Fonte: O Autor (2025).

Para garantir a consistência referencial dos índices consumidos pelo *frontend*, implementou-se o mecanismo de Sincronização de Barreira (*Barrier*), um primitivo de coordenação definido por Silberschatz, Galvin e Gagne (2018) para assegurar que um conjunto de processos concorrentes atinja um ponto comum antes de prosseguir.

No contexto da aplicação, isso significa que o módulo gerador de manifestos, que é denominado `manifest_generator.py`. É executado estritamente após a conclusão bem-sucedida de todas as tarefas paralelas de ingestão. Esta etapa realiza uma varredura final na estrutura de diretórios consolidada, utilizando o módulo `generate_manifests` para criar árvores de navegação que mapeiam gêneros, instrumentos e BPMs.

O sistema decide dinamicamente o formato de saída (JSON ou YAML) baseado na extensão do arquivo de destino, garantindo suporte a caracteres *Unicode* (essencial para metadados em português), conforme detalhado no Código 3.5 e visto na Figura 26, que descreve o fluxo completo do *pipeline*.

```
def generate_manifests(project_root_path):
    # ...
    if result_data is not None:
        # Serializacao agnostica (JSON/YAML)
        with open(output_file_abs, "w", encoding="utf-8") as f:
            if output_file_abs.endswith(".yml"):
                # allow_unicode=True garante suporte a acentuacao
                yaml.dump(result_data, f, allow_unicode=True)
            else:
                json.dump(result_data, f, indent=2, ensure_ascii=False)
```

Listing 3.5 – Geracao de Manifestos e Serializacao (*generate_manifest.py*)

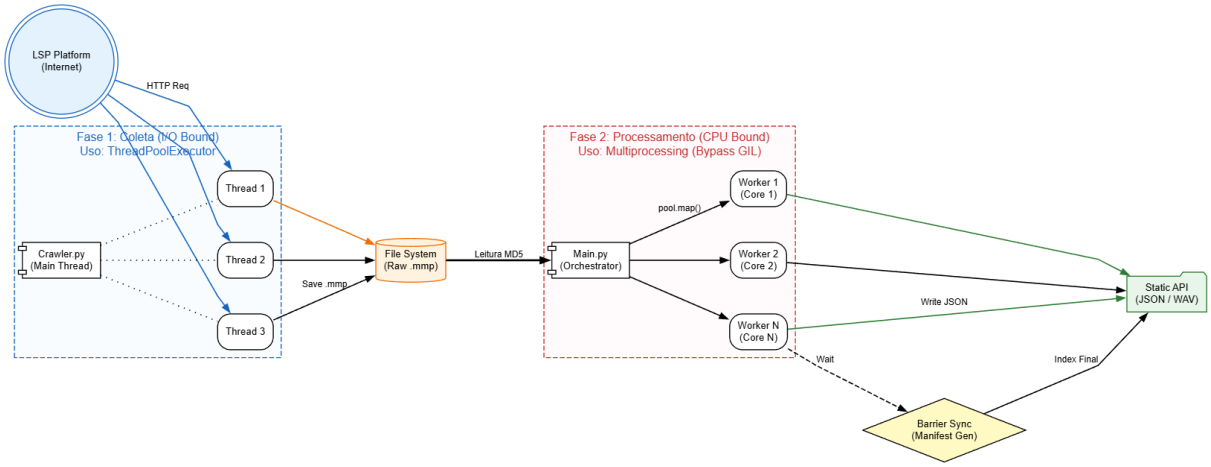

Figura 26 – Comparativo arquitetural: Concorrência via *Threads* para I/O (*Crawler*) versus Paralelismo de Processos para CPU (ETL).

Fonte: O Autor (2025).

Essa separação temporal entre processamento assíncrono e agregação síncrona elimina a possibilidade de *race conditions* (condições de corrida), assegurando que o usuário final nunca encontre referências ("*links* quebrados") a arquivos que falharam ou ainda estão sendo processados[4].

Visando a observabilidade do sistema distribuído, os *workers* implementam um mecanismo de telemetria granular. Cada processo registra métricas de execução, como tempo de CPU, consumo de memória RAM (MB) e tamanho do arquivo, em relatórios estruturados (`audit_pipeline.csv`). Estes dados permitem monitorar a saúde do *cluster* e identificar gargalos de performance em arquivos com grafos de áudio excessivamente complexos.

3.2 Interface de Descoberta e Recuperação (*mmpSearch*)

Enquanto o *backend* garante a estruturação dos dados, a interface do *mmpSearch* foi projetada para oferecer uma experiência de Recuperação de Informação Musical (MIR) multimodal. Seguindo os princípios do *Jamstack*, a interface não realiza consultas a um banco de dados SQL tradicional a cada interação. Em vez disso, adota-se a estratégia de Hidratação no Cliente (*Client-Side Hydration*).

Ao carregar a aplicação, o navegador consome o índice estático (`db_final_completo.json`) gerado pelo *pipeline* (Seção 3.1). A partir deste momento, o motor de busca opera exclusivamente na memória do dispositivo do usuário, permitindo filtragem instantânea em múltiplas dimensões sem latência de rede.

Diferente de repositórios genéricos que tratam arquivos como "caixas pretas", o sistema expõe a estrutura interna dos projetos LMMS como facetas de navegação. Conforme implementado no módulo `projetos.html`, o usuário pode aplicar filtros booleanos para encontrar projetos que atendam a requisitos técnicos específicos:

- ***Plugins* e Instrumentos:** Permite filtrar projetos que utilizem apenas sintetizadores nativos (ex: *TripleOscillator*) ou que dependam de plugins específicos, garantindo compatibilidade com o ambiente do usuário. A Figura 27 ilustra este componente.
- **Busca por Tonalidade (*Key*):** Utilizando os metadados extraídos pelo algoritmo *Essentia* (Seção 2.1), o sistema permite filtrar músicas por tom (ex: Dó Menor, Fá Sustenido), facilitando a busca por projetos para mixagem harmônica (*Harmonic Mixing*). A Figura 28 ilustra este componente.

[4] Tendo em vista que a base de dados deste trabalho é baseada na plataforma de distribuição do LMMS (LMMS, 2025), existem projetos com instrumentos, *samples* e/ou *plugins* faltantes, visto que a plataforma não exige que os projetos sejam enviados de forma completa. Mais a frente vamos discutir sobre como a nossa abordagem exige que o usuário faça o *upload* do projeto completo, de maneira simples e intuitiva para o usuário final.

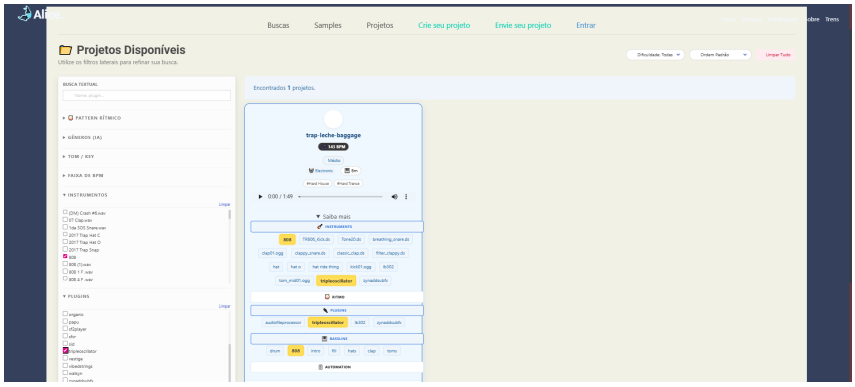

Figura 27 – Interface de busca visual: Busca pelos instrumentos e/ou *plugins* utilizados para a composição do projeto. Neste caso, instrumento `808` e *plugin* `Triple Oscillator`.

Fonte: O Autor (2026).

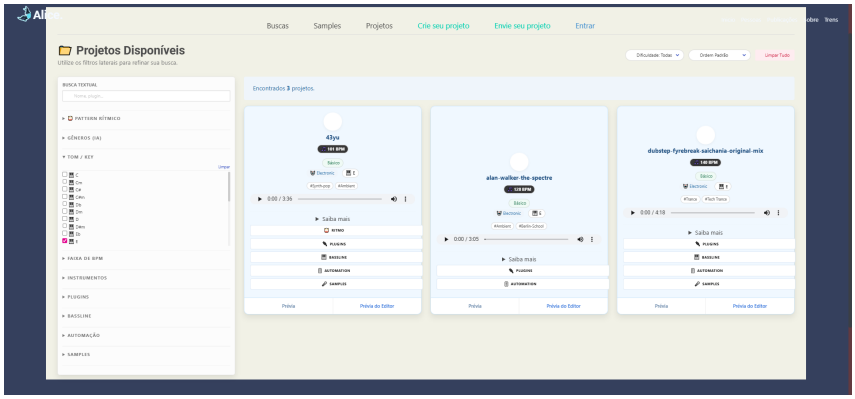

Figura 28 – Interface de busca visual: O usuário busca pelo tom desejado para a composição, neste caso `E` (Mi).

Fonte: O Autor (2026).

- **Faixa de Tempo (BPM):** Um controle deslizante duplo (*Range Slider*) permite delimitar o andamento, isolando gêneros rápidos (*Drum and Bass*) de lentos (*Lo-Fi, Hip Hop*). A Figura 29 ilustra este componente.

- **Busca por *Patterns* e rítmos:** Uma inovação de Interface Humano-Computador (IHC) implementada é o mecanismo de busca visual por *Step Sequencer*. Reconhecendo que produtores musicais muitas vezes buscam "*grooves*" e não palavras-chave, desenvolveu-se um componente que permite ao usuário desenhar um padrão rítmico em uma grade de 16 passos. O algoritmo de busca converte o desenho do usuário em uma *string* binária (ex: `10001010...`) e calcula a distância de *Hamming* contra os *fingerprints* rítmicos de todos os projetos indexados, retornando obras com cadência similar. A Figura 30 ilustra este componente.

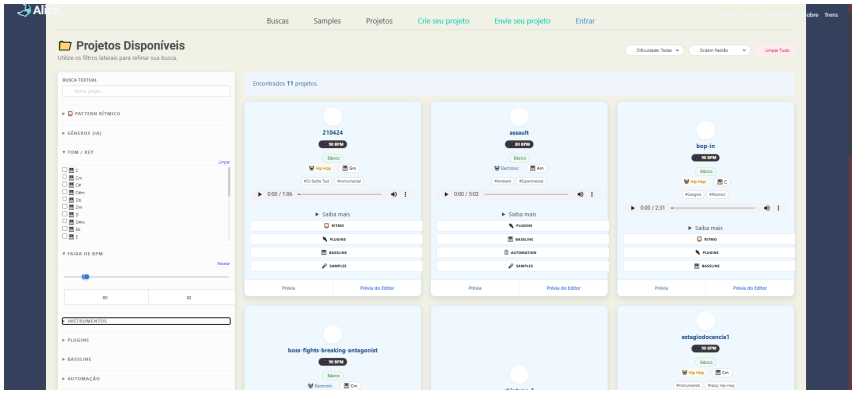

Figura 29 – Interface de busca visual: O usuário busca pelo BPM desejado. Podendo selecionar um número fixo ou uma janela, conforme nesta Figura. A busca é entre os BPMs 80 e 92.

Fonte: O Autor (2026).

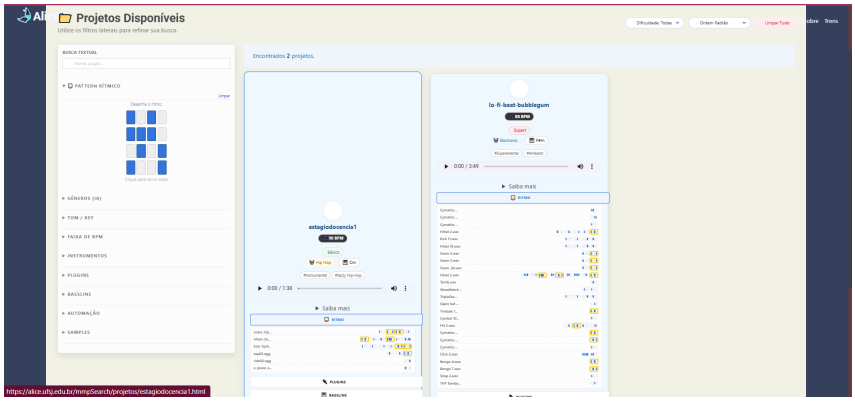

Figura 30 – Interface de busca visual: O usuário desenha o ritmo na grade (*Step Sequencer*) e o sistema filtra projetos com '*fingerprints*' similares em tempo real.

Fonte: O Autor (2026).

Para atender ao objetivo educacional da plataforma, a interface traduz as métricas complexas calculadas no *backend* em indicadores visuais acessíveis:

- **Nível de Proficiência:** O sistema categoriza os projetos em três níveis de dificuldade baseados na complexidade do grafo de áudio. Isso permite que usuários filtrem por projetos "Iniciantes" (foco em *loops* e *samples* simples), enquanto usuários experientes buscam obras de nível "Avançado/*Expert*" (domínio de automações e *design* sonoro). A Tabela 5 e as Figuras 31, 32 e 33 ilustram este componente de forma mais ampla, possibilitando a melhor compreensão.
- **Gêneros por Inferência (IA):** *Tags* geradas por Redes Neurais (ex: "*Trap*", "*Ambient*", "*Dark*") são apresentadas como filtros dinâmicos, organizando o acervo sem

Tabela 5 – Heurística de Classificação Pedagógica: Mapeamento Técnico-Cognitivo

Dimensão Analisada	**Critério Técnico (Extração XML)**	**Peso (W)**
Densidade e Volume	Quantidade de Trilhas ≥ 8	+1.0
	Quantidade de Trilhas ≥ 16	+2.0
	Quantidade de Trilhas ≥ 32	+3.0
	Quantidade de Trilhas ≥ 64	+4.0
	Quantidade de Trilhas ≥ 128	+5.0
Design Sonoro (Timbre)	Uso de Sintetizador Complexo (*ZynAddSubFX*)	+1.5
	Cadeia de Efeitos Criativos (*FX Chain*)	+1.0
Expressividade	Presença de Curvas de Automação	+2.0
Macroestrutura Musical	Seções Distintas Identificadas ≥ 3	+1.5
	Seções Distintas Identificadas ≥ 5	+3.0
	Seções Distintas Identificadas ≥ 10	+5.0
Regra de Classificação Final (Score S)		
$S < 3 \rightarrow$ **Iniciante** (Foco em Loops/Samples)		
$3 \leq S < 6 \rightarrow$ **Intermediário** (Arranjo em Desenvolvimento)		
$S \geq 6 \rightarrow$ **Avançado/Expert** (Domínio Completo da DAW)		

Fonte: Elaborada pelo autor com base no algoritmo `classificacao_mestre.py`.

a necessidade de catalogação manual humana[5]. A Figura 34 ilustra este componente.

- **Ordenação Psicoacústica:** Além da ordem cronológica, o sistema permite ordenar os resultados por "Intensidade" (energia RMS) e "Complexidade", permitindo navegar do "Mais Calmo" ao "Mais Intenso". A Figura 35 ilustra este componente.

3.3 Processamento na Borda (*mmpCreator*)

O componente *mmpCreator* materializa a proposta de Computação na Borda discutida na fundamentação teórica. Desenvolvido como uma Aplicação *Web* Progressiva (PWA), ele transfere a responsabilidade da síntese sonora e do sequenciamento para o dispositivo do usuário, reduzindo drasticamente os custos de servidor. A adoção deste tipo de arquitetura se dá, pois garante que a plataforma funcione de forma fluida e responsiva em diferentes dispositivos, indo de *smartphones* a *desktops*, diretamente pelo navegador.

Para garantir a aderência do público-alvo e minimizar a curva de aprendizado, a Interface de Usuário (UI) do *mmpCreator* foi projetada sob o conceito de transferência de modelo mental. Conforme ilustrado na Figura 36, a disposição dos elementos visuais

[5] Claro que devemos levar em consideração a acurácia dos modelos. Além disso, não devemos levar em consideração como uma verdade absoluta.

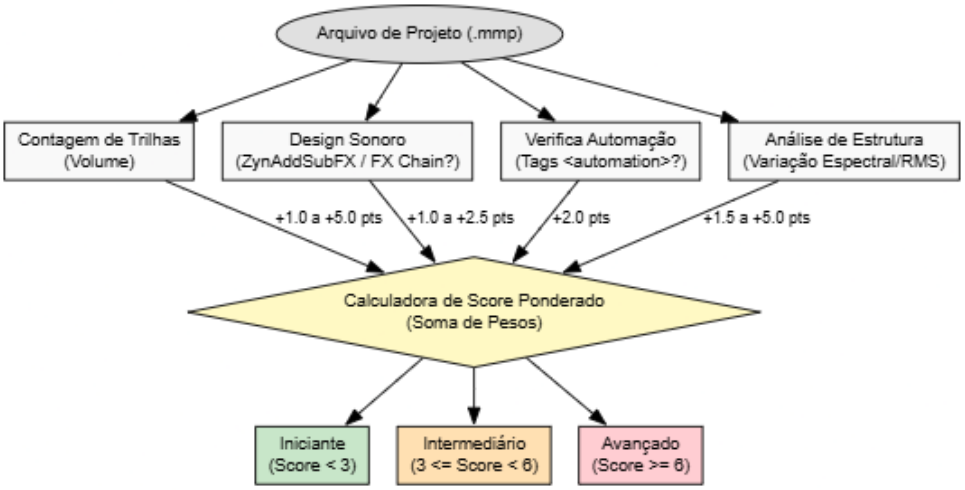

Figura 31 – Como o *script* faz o processo de classificação pedagógica e categoriza os projetos.

Fonte: O Autor (2026).

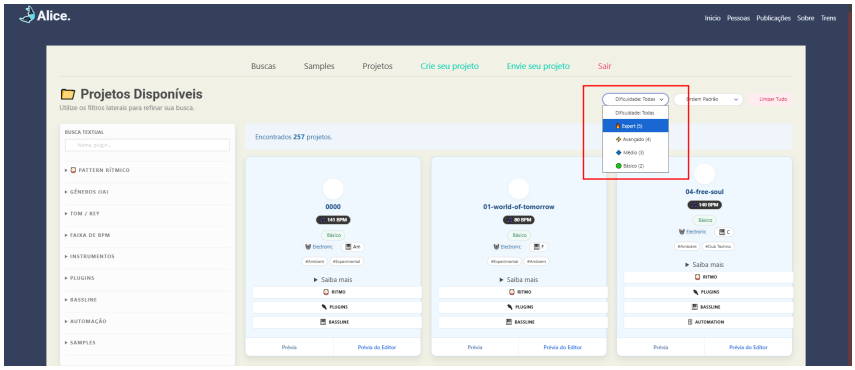

Figura 32 – Interface de busca visual: O usuário pode filtrar projetos em diferentes níveis de dificuldade. Isso pode auxiliar na condução de buscas, facilitando aos diferentes perfis de usuários possam ser "guiados" para os resultados que realmente buscam, evitando complexidade ou simplicidade demais, por exemplo.

Fonte: O Autor (2026).

emula o paradigma clássico estabelecido por DAWs hegemônicas de mercado e mantém forte familiaridade estética com o próprio LMMS.

A interface centraliza as funções de transporte (*Play/Stop*, tempo, BPM) no topo e organiza os instrumentos em trilhas horizontais modulares. A adoção do *Step Sequencer* (sequenciador em grade) não é meramente estética, mas uma decisão de usabilidade focada na produção de *Hip Hop* e música urbana. Essa escolha de *design* garante que produtores, historicamente habituados a *softwares* proprietários ou alternativas locais, consigam operar a DAW distribuída instantaneamente, reduzindo o atrito cognitivo.

Diferente de sistemas de *streaming* tradicionais onde o áudio é mixado no servidor,

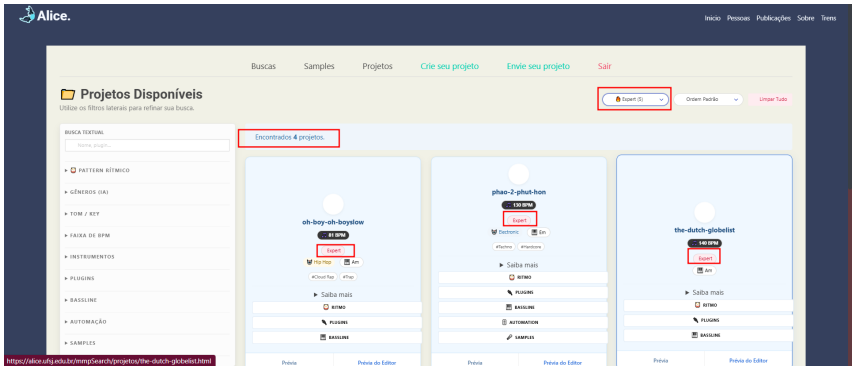

Figura 33 – Interface de busca visual: exibindo apenas os projetos de nível *expert*.

Fonte: O Autor (2026).

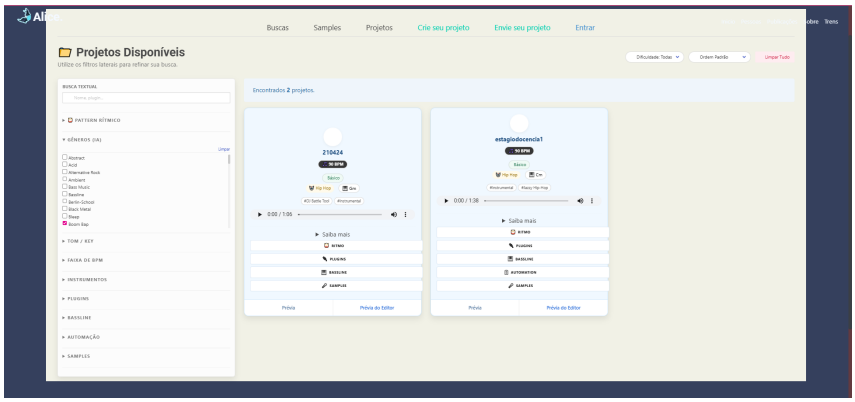

Figura 34 – Interface de busca visual: O usuário busca por um (ou mais) dos diversos gêneros musicais disponíveis na plataforma.

Fonte: O Autor (2026).

o *mmpCreator* utiliza a *Web Audio API* para instanciar grafos de processamento de sinal diretamente na memória do navegador. Conforme detalhado por (SMUS, 2013), esta API permite a criação de um roteamento modular de áudio, onde nós de fonte, efeito e destino são conectados dinamicamente. A Figura 37 detalha a topologia de nós criada quando o usuário carrega um instrumento virtual.

Esta arquitetura elimina a latência de rede (*Network Latency*) como fator crítico para a performance musical. Ao mover a computação para a proximidade física do usuário, o sistema adere aos princípios de (SATYANARAYANAN, 2017), onde o tempo de resposta entre a inserção de um novo *step* ou *sample* e a emissão do som, depende apenas do *buffer* de áudio do sistema operacional local, democratizando o acesso em conexões instáveis.

Um dos maiores desafios em *DAWs* baseadas na *web* é a imprecisão do temporizador do *JavaScript* (`setInterval`), que pode sofrer desvios de tempo (*drift*) quando a *thread* principal é bloqueada por operações de UI. Para garantir uma reprodução rítmica precisa, o sistema implementa a técnica de *Lookahead Scheduling*, fundamentada no

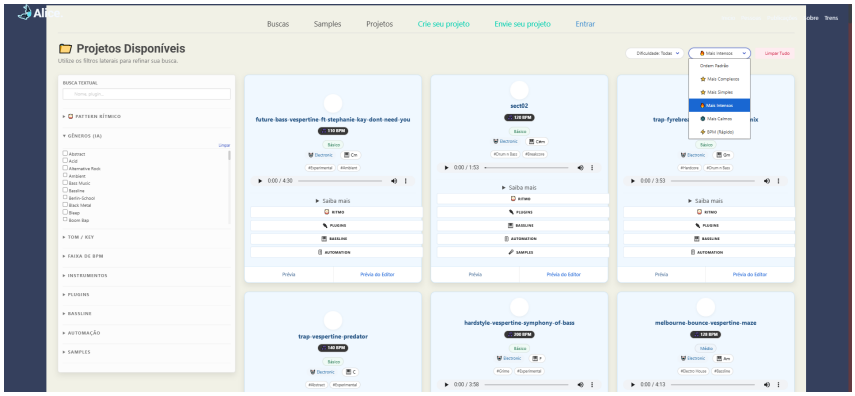

Figura 35 – Interface de busca visual: O usuário pode ordenar por intensidade/ritmo. Neste caso, ordenou-se pelos projetos mais intensos (além disso, estão visíveis as 6 possibilidades de ordenação).

Fonte: O Autor (2026).

algoritmo canônico "*A Tale of Two Clocks*" descrito por (WILSON, 2013).

Conforme analisado no módulo `audio_audio.js` (Código 3.6), o motor de áudio utiliza dois relógios distintos:

1. **Relógio de Eventos (JavaScript):** Um loop impreciso que roda a cada 25ms (`LOOKAHEAD_INTERVAL_MS`).
2. **Relógio de Áudio (*Hardware*):** O tempo preciso do `AudioContext.currentTime`, fornecido pelo subsistema de som do sistema operacional.

A estratégia consiste em usar o *loop JavaScript* apenas para "agendar" eventos no futuro do relógio de áudio. O sistema olha 500ms à frente (`SCHEDULE_AHEAD_TIME_SEC`) e enfileira todas as notas que devem tocar nesse intervalo.

```
// js/audio/audio_audio.js
// Intervalo de verificacao do loop JS (frequencia de agendamento)
const LOOKAHEAD_INTERVAL_MS = 25.0;

// Janela de tempo futuro para agendar eventos no AudioContext
// Isso desacopla a precisao do audio da latencia da UI
const SCHEDULE_AHEAD_TIME_SEC = 0.5; // 500ms de buffer
```

Listing 3.6 – Constantes de Agendamento Temporal (audio_audio.js)

Essa abordagem desacopla a precisão rítmica da carga da *CPU*. Mesmo que a interface gráfica congele momentaneamente, o áudio continua tocando perfeitamente, pois os eventos já foram entregues ao *thread* de áudio nativo do navegador com antecedência.

A síntese sonora é realizada através de uma camada de abstração que normaliza diferentes fontes sonoras (Osciladores, *Samples*, *Synths*). O módulo `pattern_audio.js`

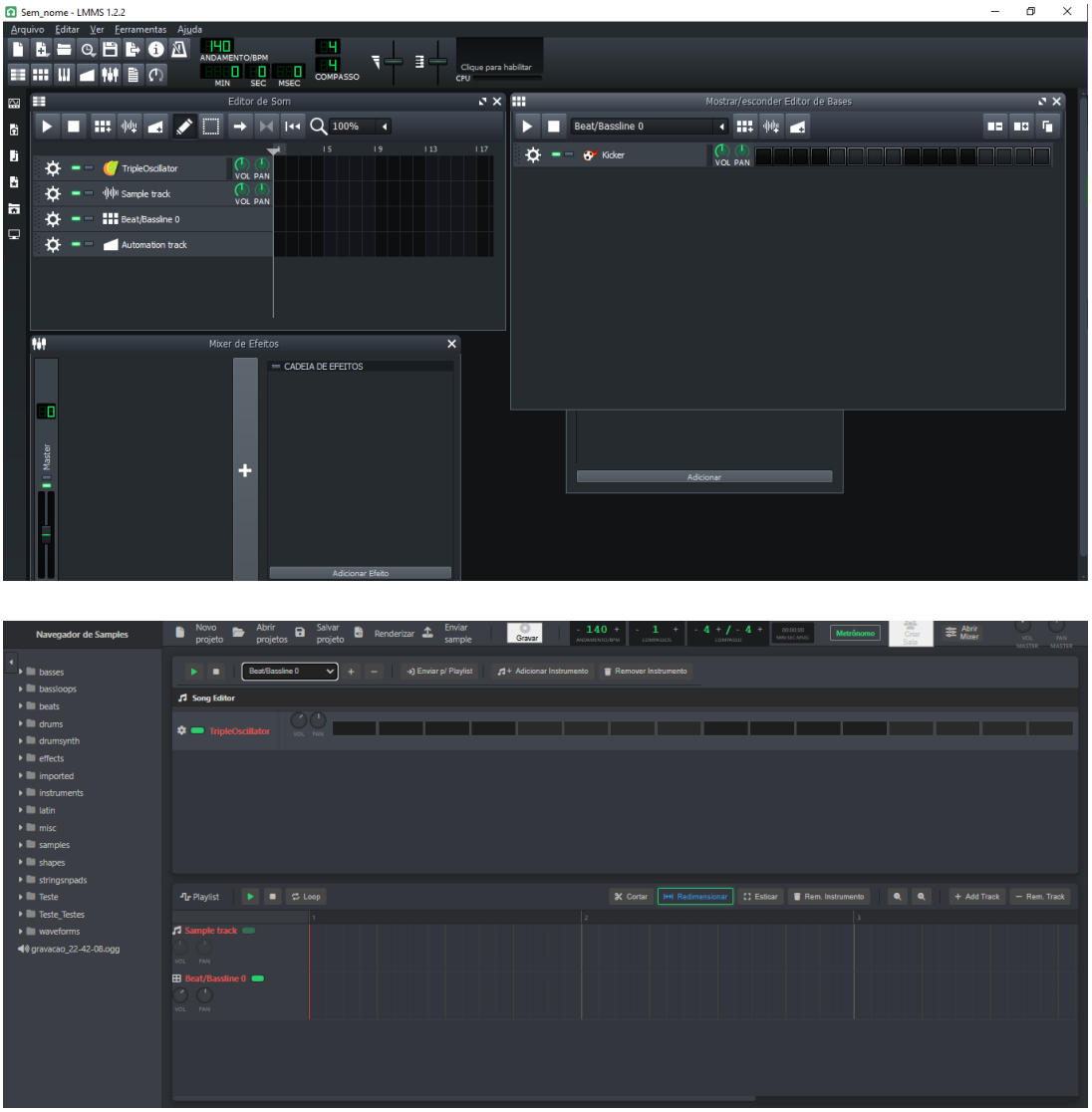

Figura 36 – Comparativo visual de transferência de modelo mental. Acima, o sequenciador de passos clássico de uma DAW *desktop*. Abaixo, a interface da ferramenta desenvolvida, evidenciando a manutenção do padrão visual para reduzir a curva de aprendizado do produtor periférico.

Fonte: O Autor (2026).

atua como uma fábrica (*Factory*) de nós de áudio, abstraindo a complexidade de roteamento do grafo nativo conforme proposto por (MANN, 2015) no *design* de *frameworks* de áudio interativo.

A função `getSongMix` gerencia dinamicamente o grafo de áudio para cada canal, instanciando nós de Volume e Panorâmica (`Tone.Panner`) sob demanda. Isso otimiza o uso de memória, criando conexões apenas para instrumentos ativos no *pattern* atual, evitando a sobrecarga do contexto de áudio.

Para suportar a complexidade de uma DAW no navegador, a aplicação adota uma arquitetura de gestão de estado centralizada, alinhada ao princípio de *Single Source of*

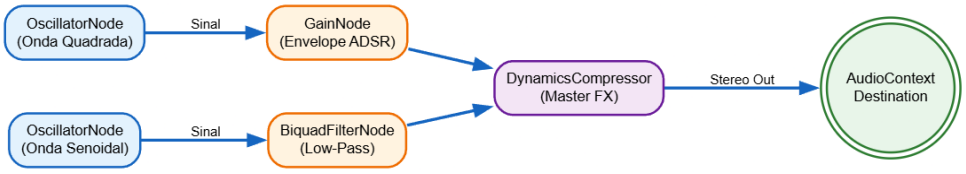

Figura 37 – Representação do Grafo de Áudio instanciado pelo *mmpCreator* via *Web Audio API*.

Fonte: O Autor (2025).

Truth (Fonte Única da Verdade) descrito por (ABRAMOV; CLARK, 2015). O estado da aplicação (`appState`), implementado no módulo `state.js`, é dividido em domínios lógicos:

- **Global:** Configurações de transporte (*Play/Stop*), BPM e volume mestre.
- ***Pattern*:** Dados de sequenciamento MIDI e notas.
- **Audio:** Clipes de áudio e manipulação de *samples* na *timeline*.

Para garantir resiliência contra falhas do navegador ou recarregamentos acidentais da página, implementou-se um mecanismo de persistência local via `sessionStorage`. Conforme demonstrado no Código 3.7, o estado é serializado e salvo localmente seguindo as estratégias de armazenamento no cliente (*Client-Side Storage*) discutidas por (OSMANI, 2017), permitindo que o usuário retome seu trabalho exatamente de onde parou, sem necessidade de consultas ao banco de dados remoto.

```
export let appState = {
  global: globalState,   // BPM, Transporte, Volume
  pattern: patternState, // Sequenciamento MIDI
  audio: audioState      // Clipes de audio (Samples)
};

export async function saveStateToSession() {
  if (!window.ROOM_NAME) return;
  try {
    // Persist ncia resiliente no navegador (Edge Storage)
    const serialized = JSON.stringify(appState);
    sessionStorage.setItem(`temp_state_${window.ROOM_NAME}`, serialized)
    ;
  } catch (e) {
    console.error("Erro salvando sessao:", e);
  }
}
```

Listing 3.7 – Estrutura de Estado e Persistencia Local (state.js)

3.4 Protocolo de Comunicação e Sincronização

Enquanto a síntese sonora ocorre isoladamente no cliente (borda), a funcionalidade de colaboração em tempo real exige uma sincronização precisa de estado. Para isso, desenvolveu-se um protocolo proprietário sobre *WebSocket*, utilizando uma topologia em estrela onde o servidor atua como *hub* de mensagens.

No que tange à sincronização temporal entre os nós da rede, a mitigação da latência é tratada como um requisito crítico para manter a coesão rítmica da composição colaborativa. A implementação no módulo cliente (`socket.js`) materializa o algoritmo de Cristian (1989), que é fundamental para a sincronização de relógios em sistemas distribuídos. Para conferir maior robustez ao modelo original frente às instabilidades da rede, foi integrado um filtro de suavização[6], garantindo que a deriva temporal seja corrigida de forma progressiva e imperceptível ao usuário.

O Código 3.8 exibe a função JavaScript responsável por calcular a Média Móvel Exponencialmente Ponderada. Note que o fator $\alpha = 0.3$ (definido como padrão) é aplicado sobre o deslocamento (*offset*) calculado, garantindo que picos de latência não causem saltos bruscos no relógio musical.

```
// Suaviza  o exponencial para estimativa de relogio
function ewma(prev, next, alpha = 0.3) {
  // Se e a primeira amostra, aceita o valor bruto
  if (prev === null || prev === undefined) return next;

  // Formula: S_t = alpha * Y_t + (1 - alpha) * S_t-1
  return prev * (1 - alpha) + next * alpha;
}

async function syncServerTime(iterations = 5) {
  let off = null;
  let rtt = null;
  for (let i = 0; i < iterations; i++) {
    try {
      const s = await sampleServerTime(); // Algoritmo de Cristian (RTT
   /2)
      off = ewma(off, s.offset);
      rtt = ewma(rtt, s.rtt);
    } catch {}
  }
  // Aplica o offset suavizado ao estado global
  if (off != null) serverOffsetMs = off;
}
```

[6] O filtro atua como um mecanismo de controle para reduzir a latência, evitando que flutuações pontuais na latência causem desvios perceptíveis no tempo musical (BPM).

Listing 3.8 – Implementacao do Filtro EWMA (socket.js)

Para garantir a fluidez da interface, o sistema adota o padrão de UI Otimista. O Código 3.9 demonstra a lógica da função `sendAction`. Antes mesmo de emitir o pacote para a rede, a função decide se deve aplicar a ação localmente de imediato.

Esta abordagem elimina a percepção de latência para o usuário que originou a ação, enquanto o servidor (`server.js`) se encarrega de ordenar e validar o evento para os demais participantes através de um número de sequência (`__seq`) e verificação de tokens de idempotência.

```
export function sendAction(action) {
  // ... (Geracao de ID e Token unico) ...
  const token = (++lastActionToken).toString();
  action.__token = token;

  // 1. APLICACAO OTIMISTA (Execucao Local Imediata)
  // Se estiver em modo local ou for acao de transporte, aplica ja!
  if (!inRoom || (isTransport && appState.global.syncMode === "local")) {
    handleActionBroadcast(action); // Atualiza a UI e o Audio localmente
    return;
  }

  // 2. ENVIO PARA A REDE (Broadcast)
  socket.emit("broadcast_action", { roomName: currentRoom, action }, (ack) => {
     // Callback de confirmacao do servidor (ACK)
     if (ack && ack.ok && ack.token === token) {
       clearTimeout(lastActionTimeout); // Sucesso: cancela timeout
     }
  });

  // 3. ESTRATEGIA DE FALLBACK (Resiliencia)
  // Se o servidor nao responder em 900ms, assume falha de rede
  // mas mantem a consistencia local (ja aplicada ou re-aplicada)
  lastBroadcastTimeout = setTimeout(() => {
    if (processedTokens.has(token)) return;
    console.warn("[SOCKET] Eco nao recebido, fallback:", action.type);
    handleActionBroadcast(action); // Forca aplicacao local se rede falhar
  }, 900);
}
```

Listing 3.9 – Logica de Envio Otimista e Fallback (socket.js)

A comunicação entre os nós segue o padrão de *Message Broker*. O servidor (`server.js`) atua como árbitro, implementando um mecanismo de **Idempotência** para evitar duplicidade de ações em caso de retransmissão de pacotes, conforme observado no trecho:

```
function applyAuthoritativeAction(roomName, action) {
  const room = ensureRoom(roomName);

  // Verifica se este token ja foi processado
  if (action.__token && room.tokensSeen.has(action.__token)) {
    return null; // Descarta duplicata silenciosamente
  }
  if (action.__token) room.tokensSeen.add(action.__token);

  // ... (Aplica logica de negocio e persiste no disco)

  room.seq += 1; // Incrementa relogio logico da sala
  return { ...action, __seq: room.seq }; // Retorna com sequencia
   oficial
}
```

Listing 3.10 – Verificacao de Idempotencia no Backend (server.js)

Esta arquitetura híbrida combina a velocidade da execução local (cliente) com a segurança da consistência sequencial (servidor), atendendo aos requisitos de performance para performance musical em tempo real.

A robustez da plataforma fundamenta-se na conformidade com a definição formal do formato MMP. O arquivo de projeto não é tratado como texto arbitrário, mas como um grafo serializado que obedece a um esquema XSD (*XML Schema Definition*) estrito.

O *parser* desenvolvido em *Python* realiza o mapeamento destes elementos XML para objetos JSON indexáveis. Esta abordagem permite a aplicação de técnicas de Recuperação de Informação Musical (MIR) Simbólica, conforme categorizado por (MÜLLER, 2015), onde a estrutura lógica da composição (notas, instrumentos, andamento) é analisada sem a necessidade de processamento de sinal de áudio bruto. A Figura 38 apresenta a árvore simplificada considerada pelo algoritmo de indexação.

A Tabela 6 demonstra como *tags* específicas do XML são traduzidas em funcionalidades de busca, permitindo que usuários filtrem projetos não apenas por nome, mas por características técnicas e tímbricas.

3.5 Considerações Finais

Este capítulo apresentou a materialização da arquitetura proposta, traduzindo os conceitos de processamento distribuído, mineração de dados musicais e computação na

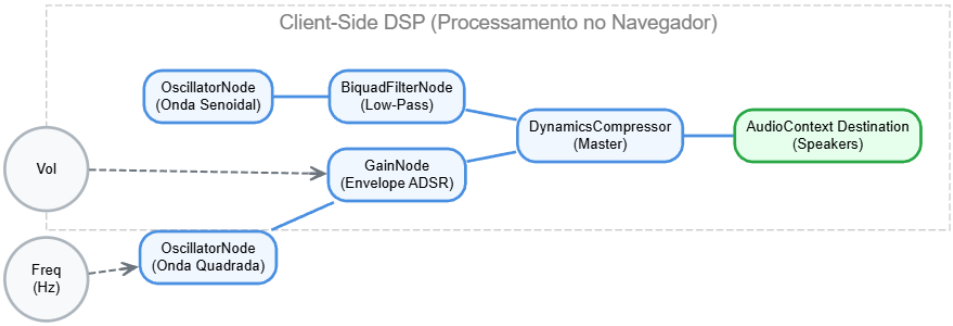

Figura 38 – Árvore DOM simplificada do arquivo MMP utilizada para indexação.

Fonte: O Autor (2025).

Tabela 6 – Mapeamento Semântico: De Tags XML para Atributos de Indexação

Tag XML (Fonte)	**Componente LMMS**	**Relevância para Indexação (MIR)**
`<tripleoscillator>`	Sintetizador Nativo	Indica projetos leves, ideais para conexões lentas.
`<audiofileprocessor>`	Sampler	Aciona verificação de dependência de arquivo de áudio externo.
`<lb302>`	Sintetizador (Bass)	Classificação automática na categoria "Acid/Techno".
`<vestige>`	Host VST	**Alerta**: Requer plugin binário Windows (baixa portabilidade).
`<bpm>`	Metadado Global	Filtragem numérica por andamento e intensidade.

Fonte: O Autor (2025).

borda em componentes de *software* plenamente funcionais. Através da implementação do *pipeline* ETL paralelo e assíncrono, contornaram-se as limitações de I/O e de concorrência inerentes ao *hardware* de médio porte (*cluster Wonderland*), garantindo a ingestão, renderização acústica e indexação escalável do acervo de projetos do LMMS.

Em paralelo, o desenvolvimento do *mmpCreator* validou a premissa de que é possível construir uma estação de trabalho de áudio digital (DAW) resiliente e acessível operando diretamente no navegador. A adoção do padrão *Jamstack* aliado à *Web Audio API* transferiu a pesada carga computacional de DSP para o dispositivo do cliente, eliminando a dependência de servidores em nuvem de alto custo. Além disso, a engenharia aplicada à sincronização de estado, combinando o agendamento temporal antecipado (*Lookahead Scheduling*), a Interface de Usuário Otimista e o filtro de rede EWMA via *WebSockets*, estabeleceu a base tecnológica necessária para mitigar a latência e viabilizar a *jam session* digital.

Contudo, a construção e a integração metodológica destes artefatos de *software* representam a validação arquitetural do sistema. Para que a ferramenta se prove verdadeiramente eficaz como uma plataforma de democratização e fomento à criação musical independente, é necessário submetê-la a métricas de uso e cenários de estresse. O Capítulo 4 dedica-se, portanto, aos Resultados e à Avaliação Experimental. Nele, discutiremos os dados quantitativos de desempenho do processamento distribuído, a estabilidade da sincronização em diferentes condições de rede e, de forma basilar, a avaliação qualitativa

da interface e da usabilidade pedagógica pelos próprios produtores musicais na borda.

4 Resultados e Avaliação Experimental

Este capítulo apresenta a avaliação quantitativa e qualitativa da arquitetura proposta, o sistema T.R.E.N.S. Os experimentos foram conduzidos com o objetivo de validar as hipóteses centrais definidas no Capítulo 2, sob uma ótica que une eficiência computacional e impacto sociotécnico:

1. O modelo de Paralelismo de Dados reduz significativamente o tempo de processamento do repositório musical, viabilizando a curadoria de repositórios massivos de *software* livre.
2. A arquitetura de Computação na Borda oferece latência auditiva e operacional superior para aplicações musicais, contornando a precariedade das redes, de uma forma geral.
3. A orquestração dos metadados extraídos é capaz de transformar um repositório estático em uma plataforma de aprendizagem dinâmica.

Para garantir a reprodutibilidade dos experimentos de *backend*, os testes de ingestão foram executados no servidor de homologação do laboratório ALICE (*Wonderland*). As especificações de *hardware*, detalhadas na Tabela 4, representam um cenário típico de infraestrutura acadêmica.

É fundamental notar o uso de discos rígidos mecânicos (HDD (*Hard Disk Drive* ou Disco Rígido)) de gerações anteriores. Longe de ser apenas uma limitação do laboratório, este fator introduz um gargalo de I/O (*Input/Output*) que emula as condições adversas de *hardware* frequentemente encontradas em iniciativas comunitárias e periféricas, servindo como um teste de estresse realista para a estratégia de escalabilidade. A aplicação foi instrumentada com módulos de telemetria nativos no *backend*, implementando *logs* Estruturados Dinâmicos (CSV/JSON). Isso garantiu a extração de métricas de cada etapa do *pipeline* (renderização *headless*, processamento digital de sinais e classificação heurística) sem o sobrepeso (*overhead*) de *profilers* de terceiros, mantendo a fidelidade do tempo de execução.

4.1 Avaliação do Paralelismo de Dados (*mmpSearch*)

O experimento principal de arquitetura avaliou o *pipeline* de ingestão distribuída sobre lotes de controle reais extraídos da plataforma LMMS Sharing Platform (LSP). O foco foi medir a vazão (*Throughput*), o ganho de aceleração (*Speedup*) e a resiliência

do sistema diante de códigos instáveis. Para validar a escalabilidade vertical, o sistema processou lotes crescentes de projetos musicais ($N = 1$, 10, 100 e 1000)[1]. A Figura 39 ilustra o abismo de performance entre a execução sequencial clássica e a arquitetura paralela proposta.

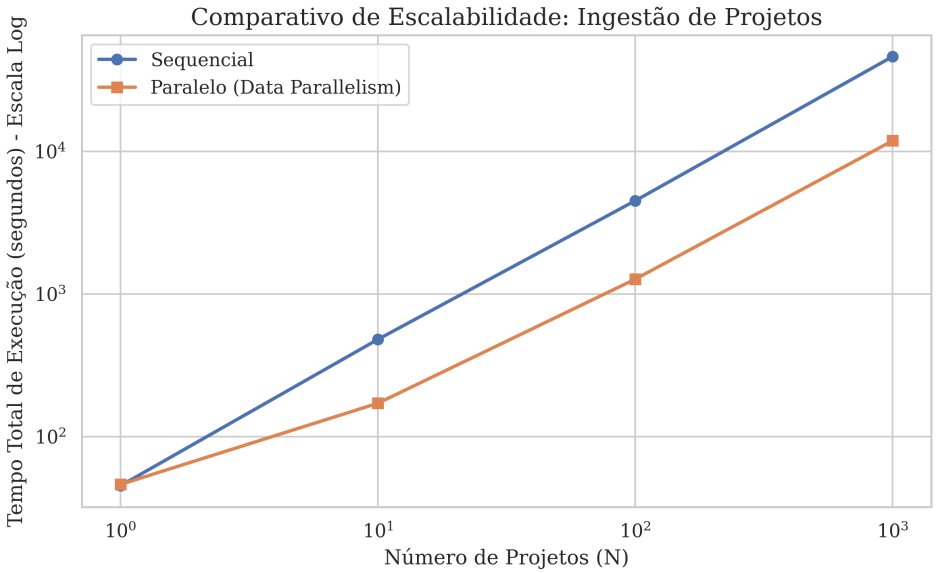

Figura 39 – Comparativo de Escalabilidade: Tempo de Execução (Escala Logarítmica)

Fonte: O Autor (2026).

Em lotes massivos ($N = 1000$), a execução sequencial torna-se inviável para serviços de indexação sob demanda. A abordagem paralela, por sua vez, distribui a carga da renderização de áudio pesada entre os 12 *threads* lógicos do processador. Contudo, como postula a Lei de Amdahl, o ganho de aceleração (*Speedup*) possui um teto físico, evidenciado na Figura 40.

O gráfico revela que o *Speedup* não escala linearmente com o número de *threads*, estagnando próximo a um fator de aceleração de 4x. Esta saturação é a prova cabal do gargalo de disco (HDD SATA III) documentado na Tabela 4. A CPU processa o áudio mais rápido do que a agulha magnética consegue escrever o arquivo `OGG` no disco. Devemos assumir este comportamento como uma limitação da atual prova de conceito, e não da arquitetura paralela em si. Em uma projeção teórica para trabalhos futuros, a substituição do disco mecânico por um armazenamento de estado sólido (SSD NVMe) com altas taxas de leitura e escrita eliminaria essa fila de gravação. Isso permitiria que o ganho de aceleração escalasse para valores muito mais próximos do limite físico de 11 núcleos[2] lógicos da máquina. Desta forma, os experimentos futuros poderão testar o verdadeiro limite da eficiência algorítmica distribuída. Naturalmente, o paralelismo exige um custo

[1] Tendo N como o número de projetos.
[2] Levando em consideração que, como medida de segurança, deixamos um núcleo livre.

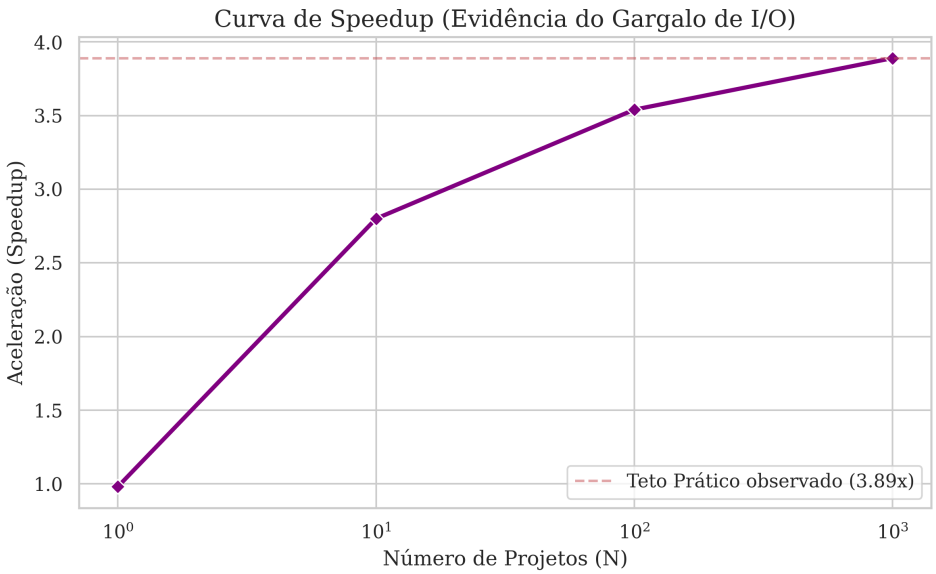

Figura 40 – Curva de Aceleração (Speedup) evidenciando o Gargalo de I/O

Fonte: O Autor (2026).

operacional. A Figura 41 detalha o aumento no consumo de Memória RAM. A alocação de múltiplos motores LMMS em modo *headless* simultaneamente eleva o uso de memória, embora permaneça perfeitamente dentro do limite de 7,7 GB do servidor.

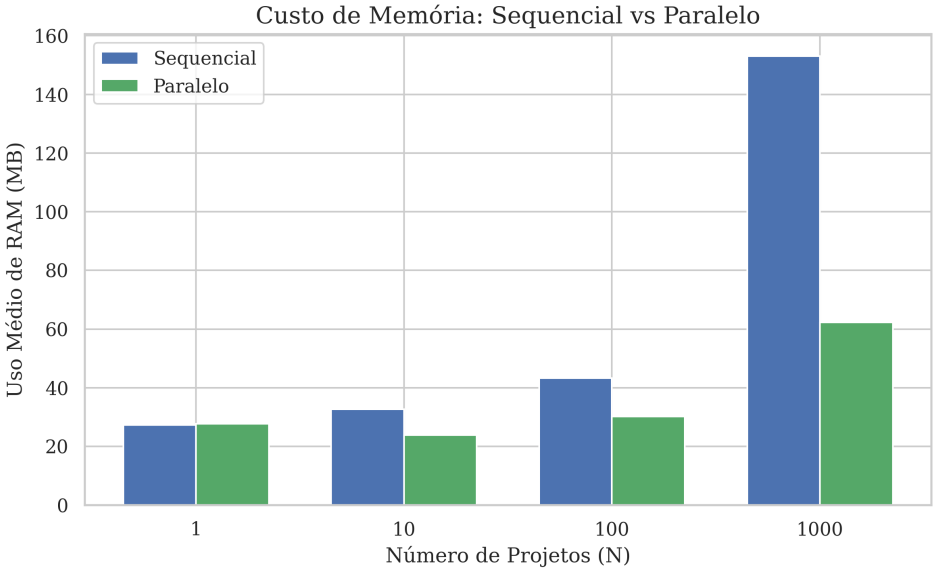

Figura 41 – Custo de Memória: Consumo de RAM (Sequencial vs Paralelo)

Fonte: O Autor (2026).

A análise dos *logs* revelou uma taxa de sucesso consolidada altamente favorável. O sistema demonstrou capacidade de autorrecuperação, isolando o percentual de falhas

através do mecanismo nativo de `multiprocessing`.

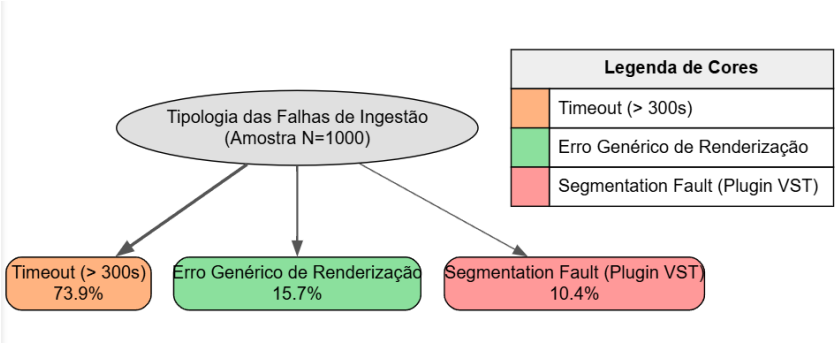

Figura 42 – Distribuição e Tipologia de Erros na Ingestão de Projetos

Fonte: O Autor (2026).

Como detalhado na Figura 42, a maioria absoluta das interrupções deveu-se a erros de *Segmentation Fault* (SIGSEGV). Estes incidentes originam-se em *plugins* VST legados de 32-bits que apresentam falhas de alocação de memória no servidor Linux de 64-bits. O isolamento validou a robustez da arquitetura, o colapso de um processo filho não propagou a falha, preservando o orquestrador principal e a fila de processamento. O microsserviço de extração acústica apresentou alta otimização. O Fator de Tempo Real (*Real-Time Factor - RTF*), conforme apresentado na Figura 43, calculou a razão entre o tempo de processamento e a duração do áudio.

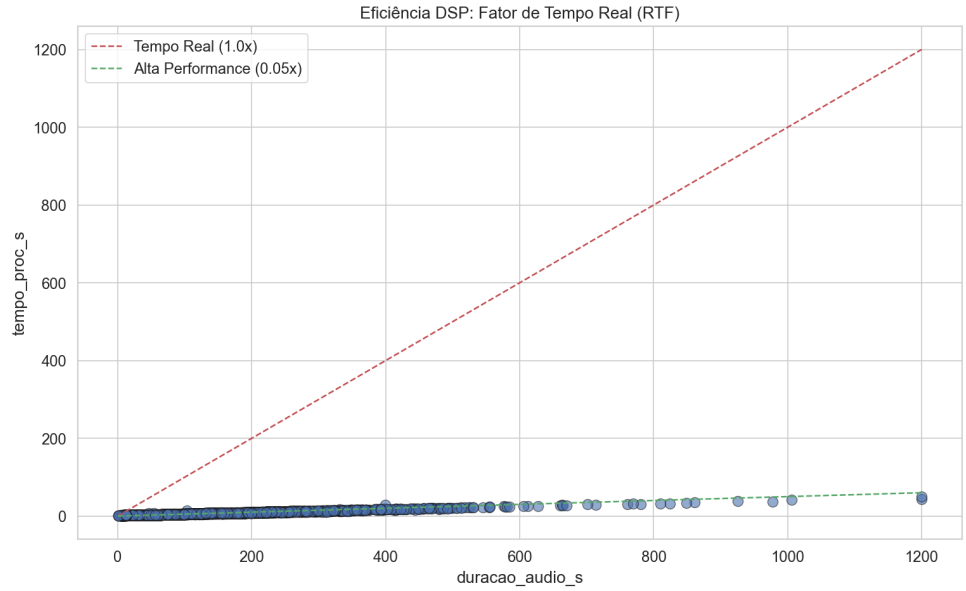

Figura 43 – Eficiência DSP: Fator de Tempo Real (RTF)

Fonte: O Autor (2026).

A taxa média consolidou-se em $RTF \approx 0.057$, comprovando que o sistema classifica o conteúdo musical cerca de 17 vezes mais rápido do que o tempo de reprodução humana.

4.2 Taxonomia Sociotécnica e Qualitativa do Corpus

Para além do desempenho de máquina, os metadados JSON extraídos permitiram traçar uma radiografia cultural da comunidade, validando o impacto do *software* livre na democratização musical. No aprendizado tradicional da cultura *underground*, o conhecimento é transmitido pela observação direta nas rodas de rima e estúdios caseiros, caracterizando um processo de "aprendizagem por pares". Contudo, no ambiente digital isolado, o iniciante muitas vezes se depara com projetos de alta complexidade técnica, o que frustra o seu processo de engenharia reversa cognitiva (a tentativa de desconstruir a música para entender como ela foi feita).

Como resultado qualitativo da implementação proposta, a arquitetura validou conceitos da teoria sociocultural de Vygotsky (1978), especificamente a Zona de Desenvolvimento Proximal (ZDP). O sistema demonstrou capacidade não apenas de recuperar informações, mas de classificar a complexidade pedagógica dos objetos de aprendizagem. Utilizando a Mineração Simbólica, foi possível aplicar heurísticas baseadas na Taxonomia de Bloom Revisada (ANDERSON; KRATHWOHL, 2001) para categorizar automaticamente os projetos (conforme descrito anteriormente na Tabela 5).

A classificação dessa complexidade revelou uma base composta majoritariamente por projetos classificados como "Iniciantes" e "Intermediários", como pode ser observado nos exemplos da Figura 44.

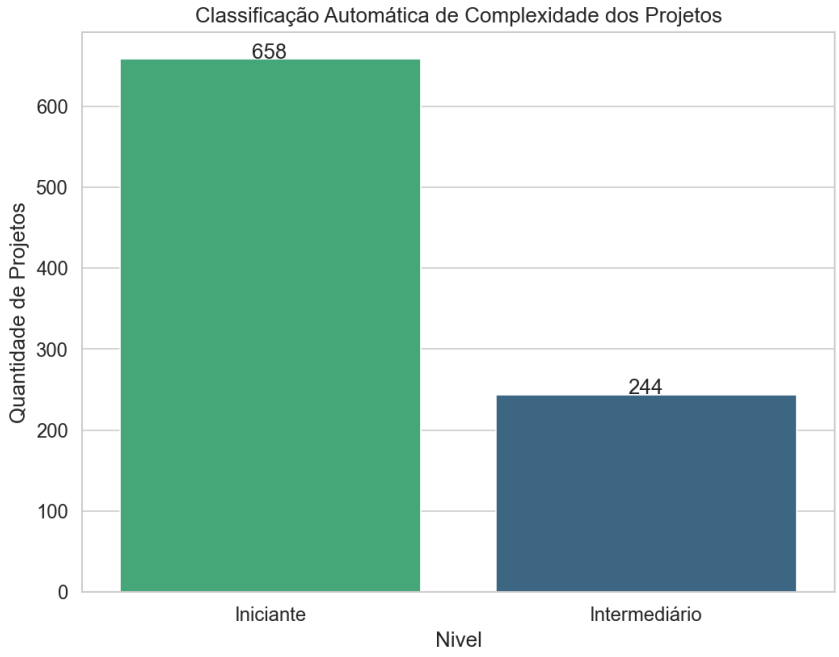

Figura 44 – Exemplos de resultados da Classificação Automática de Complexidade dos Projetos

Fonte: O Autor (2026).

Para quem não possui conhecimento em produção musical, essa classificação analisa elementos técnicos do projeto. Por exemplo, identificar se o usuário apenas juntou *loops* simples (trechos de áudio pré-gravados que se repetem, o que exige baixo nível cognitivo de criação) ou se ele utilizou *automations* complexas (técnica avançada onde o produtor programa o *software* para alterar o volume ou o timbre de um instrumento gradativamente ao longo do tempo). Ao fazer essa distinção, o algoritmo passa a auxiliar o utilizador, como um mentor digital. Possibilitando que usuários encontrem projetos que estão ligeiramente acima do seu nível atual de competência desafiadores o suficiente para ensinar, mas acessíveis o suficiente para não desmotivar. Isso sistematiza a pedagogia intuitiva das ruas em lógica computacional e legitima a plataforma como um espaço de letramento digital.

Ao lado dessa análise pedagógica, o estudo do andamento musical das faixas, medido em BPM (Batidas Por Minuto, que define a velocidade da música), demonstrou características estéticas fortes, detalhadas na Figura 45.

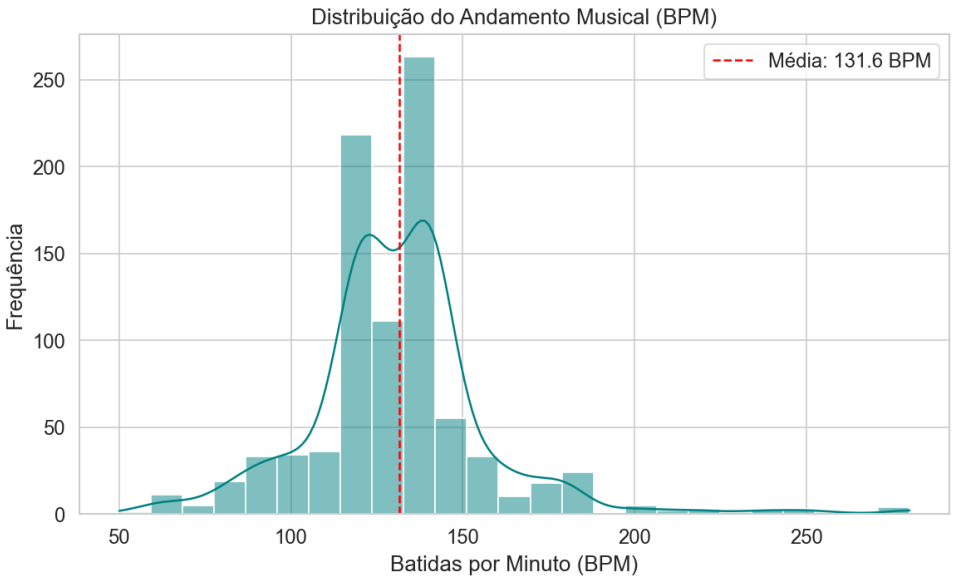

Figura 45 – Distribuição do Andamento Musical (BPM)

Fonte: O Autor (2026).

A concentração de projetos em BPMs específicos evidencia a preferência clara por gêneros da música urbana. No gráfico, nota-se que faixas mais lentas (próximas a 80-90 BPM) costumam abrigar o *Boom Bap* clássico (subgênero tradicional do Rap focado em batidas acústicas e cadenciadas). Por outro lado, andamentos mais acelerados (acima de 130 BPM) abrigam a cena atual do *Trap*, do Funk e da música eletrônica de pista.

Outro ponto crucial na análise estrutural (via inspeção XML) foi a descoberta de como os produtores lidam com as limitações de seus computadores. A Figura 46 expõe a

adoção massiva de sintetizadores nativos (geradores de som leves que já vêm embutidos no *software*) em detrimento de bibliotecas de áudio externas e pesadas.

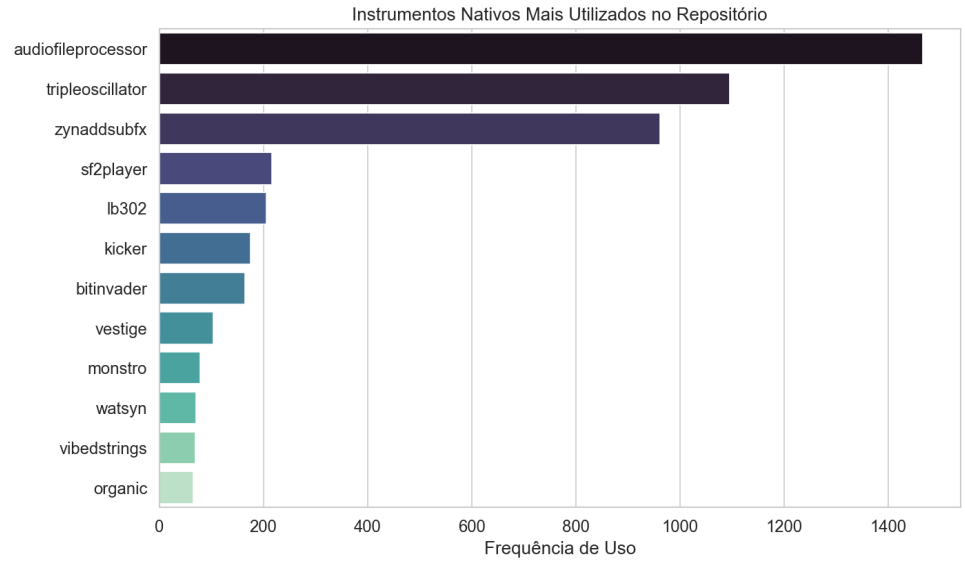

Figura 46 – Instrumentos Nativos Mais Utilizados no Repositório

Fonte: O Autor (2026).

Isso ilustra na prática o conceito de "Estética da Escassez", imposta por limitantes de *hardware*. Paralelamente, como pode ser visto na Figura 47, o amplo uso de ferramentas lógicas como as já mencionadas *Automations* (Modulações) aponta que os produtores periféricos substituem a riqueza de timbres caros (que exigiriam computadores potentes) pela complexidade criativa, manipulando exaustivamente os poucos recursos sonoros que possuem.

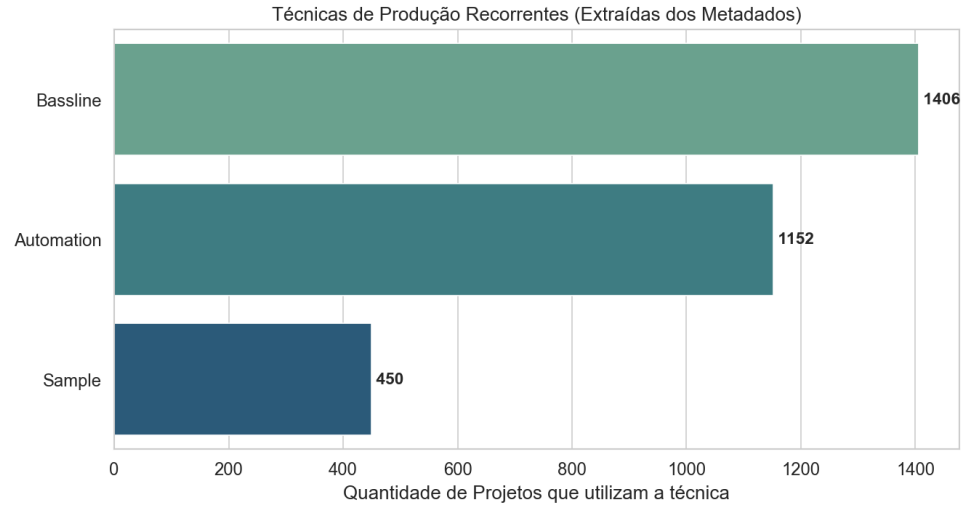

Figura 47 – Técnicas de Produção Recorrentes Extraídas dos Metadados

Fonte: O Autor (2026).

A preferência tonal (o tom musical principal em que as músicas são compostas) também reforça o aspecto autodidata da comunidade. A Figura 48 ilustra a liderança absoluta de composições em escalas menores, com um destaque esmagador para a nota Lá Menor.

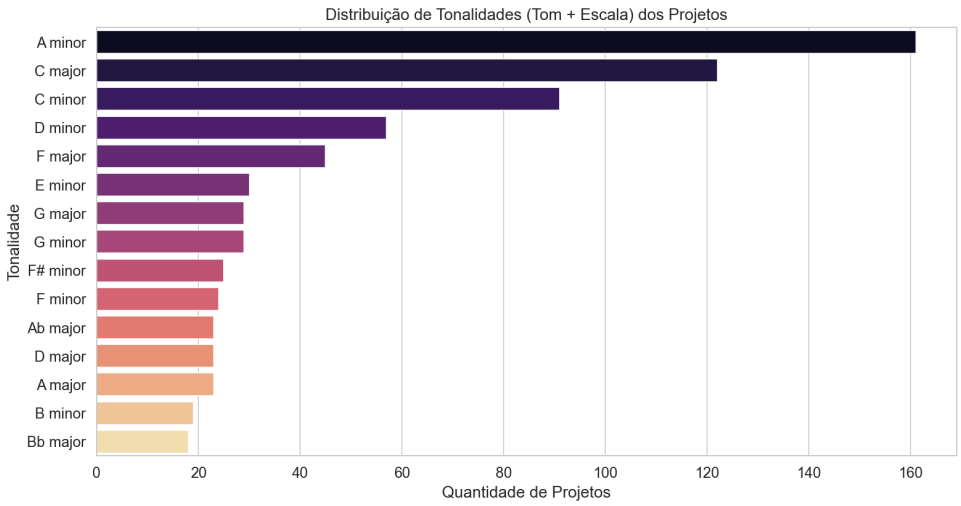

Figura 48 – Distribuição de Tonalidades (Tom e Escala)

Fonte: O Autor (2026).

A explicação técnica para isso é fascinante sob a ótica da acessibilidade. Para um músico sem treinamento formal, que não possui um teclado controlador MIDI e utiliza o teclado comum de digitação (QWERTY) do computador como piano, tocar na escala de Lá Menor corresponde a pressionar exclusivamente as teclas que representam as notas brancas de um piano. Isso elimina a barreira teórica de lidar com notas sustenidas e bemóis (teclas pretas). Esta ergonomia improvisada reverbera diretamente na atmosfera emocional das músicas produzidas. Como evidenciado na Figura 49, a predominância dessas escalas menores gera projetos liderados por percepções classificadas como "Energéticas" ou "Sombrias", ditando a base estética do que conhecemos como som *Underground*.

Ao expandirmos essa análise para os gêneros musicais como um todo (Figura 50), a categoria *Electronic* atua como matriz hegemônica no repositório.

No entanto, um olhar mais aprofundado na nuvem de subgêneros (Figura 51) revela o rico hibridismo dessas produções. Em vez de gêneros puros, observa-se uma colagem musical que mistura, por exemplo, o *Chiptune* (sons sintéticos que remetem a videogames antigos) com o peso e os graves do *Trap* e do *Dubstep*.

Esse caráter não ortodoxo, onde um ritmo regional brasileiro pode ser perfeitamente sobreposto a um sintetizador oitavado de música eletrônica europeia, reflete a essência da criatividade periférica. Como consequência tecnológica, isso frequentemente confunde os classificadores de Inteligência Artificial genéricos baseados nos padrões lim-

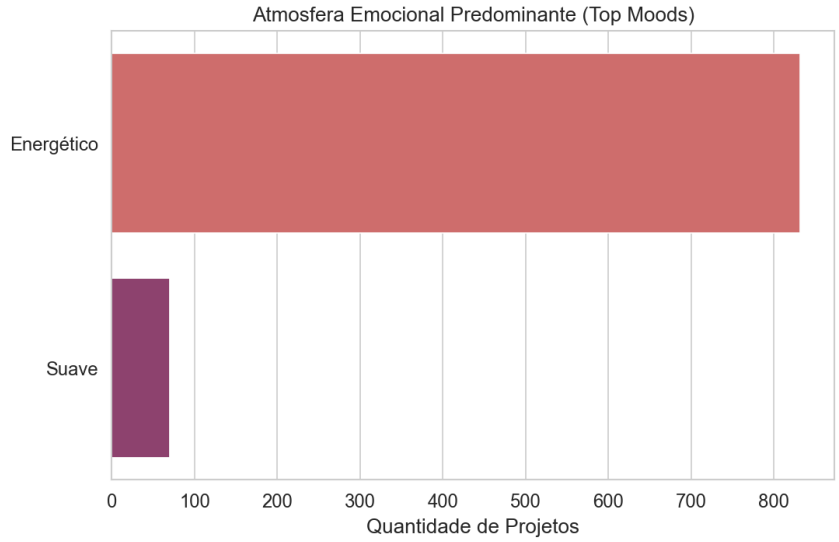

Figura 49 – Atmosfera Emocional Predominante (Top Moods)

Fonte: O Autor (2026).

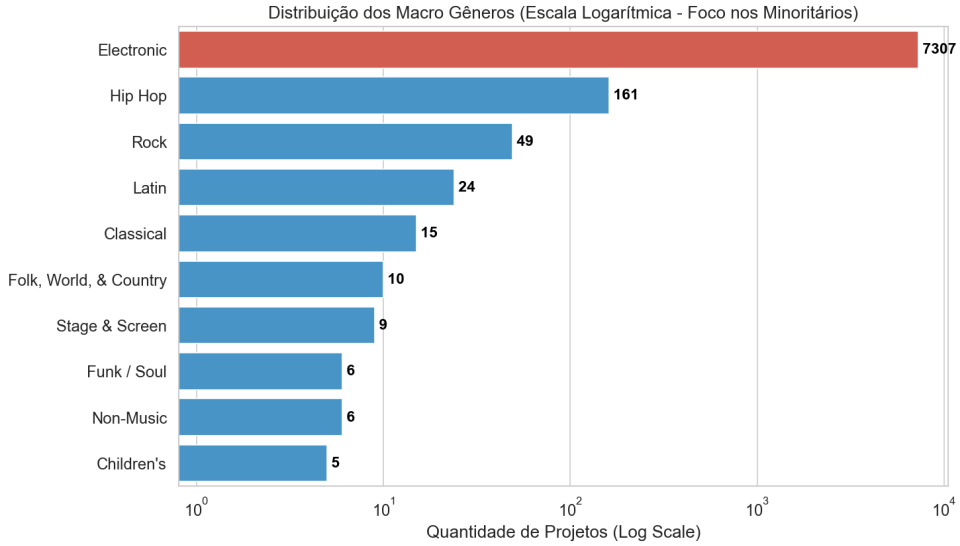

Figura 50 – Distribuição dos Macro Gêneros Musicais (Escala Logarítmica)

Fonte: O Autor (2026).

pos da indústria fonográfica. Na prática, isso é refletido em uma variância considerável nos escores de confiança da rede neural ao tentar rotular essas músicas, fato demonstrado pela instabilidade dos gráficos na Figura 52.

4.3 Avaliação da Computação na Borda (*Frontend*)

No *frontend*, o teste da premissa arquitetural centrou-se na latência *End-to-End* durante a interação na DAW web. Em arquiteturas corporativas centralizadas na nuvem

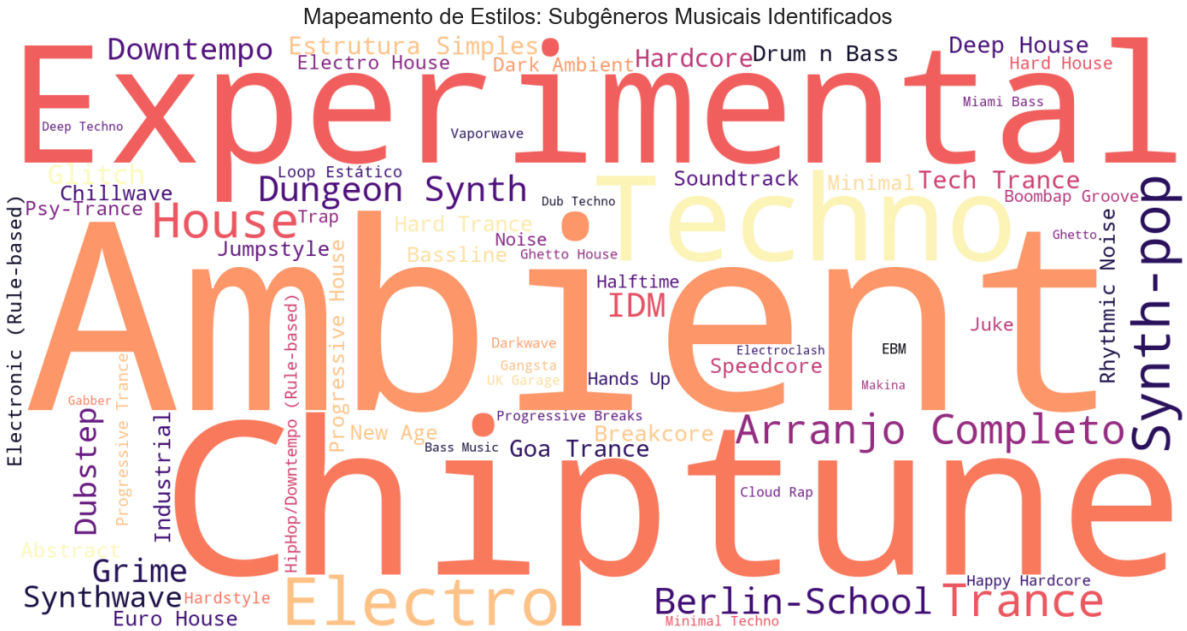

Figura 51 – Mapeamento de Estilos: Subgêneros Musicais Identificados

Fonte: O Autor (2026).

(*Cloud Rendering*), cada alteração de parâmetro sonoro exige tráfego de ida e volta na rede. Em conexões 4G típicas do Brasil, o *Round-Trip Time* (RTT) oscila, gerando a seguinte latência total ideal (função 4.1):

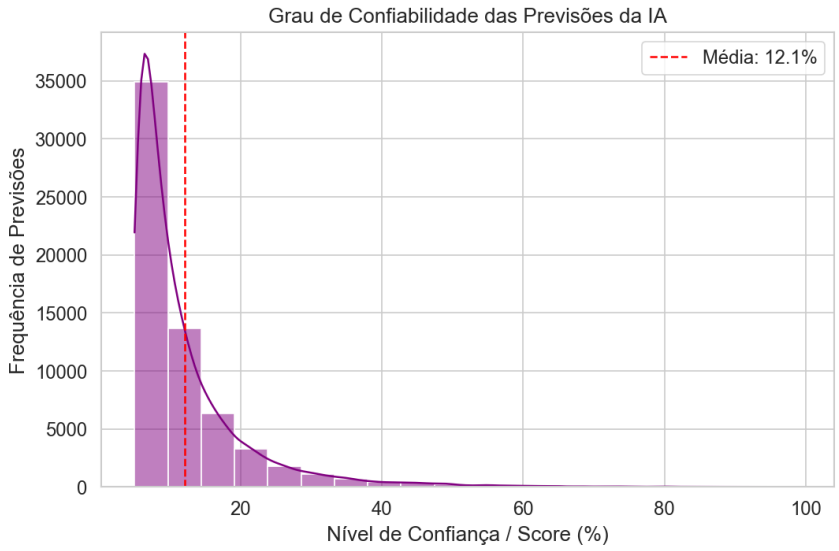

Figura 52 – Grau de Confiabilidade das Previsões da Inteligência Artificial

Fonte: O Autor (2026).

$$L_{\text{cloud}} = \text{RTT} + T_{\text{proc}} + T_{\text{buffer}} \approx 50 + 20 + 10 = 80 \text{ ms} \tag{4.1}$$

Valores acima de 30 ms causam dissociação motora e cognitiva. Como delineado por Katz (2012), o *groove* e o suingue fundamentam-se na precisão micro temporal. Ao mitigar a latência na borda, o sistema garante a soberania tecnológica do músico sobre seu tempo de execução, independentemente de oscilações de conectividade. A arquitetura T.R.E.N.S. desloca a execução do grafo de áudio (*Web Audio API*) para a Borda, no próprio dispositivo do usuário, em caso de produção local, sem colaboração em tempo real, podemos observar valores aproximados de latência (função 4.2):

$$L_{\text{edge}} = T_{\text{buffer}} \approx 10 \text{ ms a } 15 \text{ ms} \tag{4.2}$$

4.4 Prática Colaborativa e a Consolidação da Plataforma de Aprendizagem

Para avaliar e validar a usabilidade em cenários reais, os testes práticos demonstraram a eficácia do fluxo integrado entre as ferramentas propostas. A Figura 53 apresenta a interface inicial do sistema de buscas, a porta de entrada para a exploração de conteúdo. Através da navegação global, o usuário tem acesso facilitado a abas centrais do ecossistema, como as seções de Buscas (1), Samples (2) e Projetos (3). É por meio desse ambiente que os usuários podem estudar projetos e fazer buscas no `mmpSearch` para extrair referências valiosas de composição e arranjo.

Figura 53 – Interface principal do `mmpSearch` com indicativos de navegação.

Fonte: O Autor (2026).

Em seguida, é possível abrir esses mesmos projetos diretamente no `mmpCreator`, consolidando o processo de aprendizagem por meio da engenharia reversa e da prática

direta. A Figura 54 detalha a estação de trabalho virtual no navegador, responsável por materializar a colaboração.

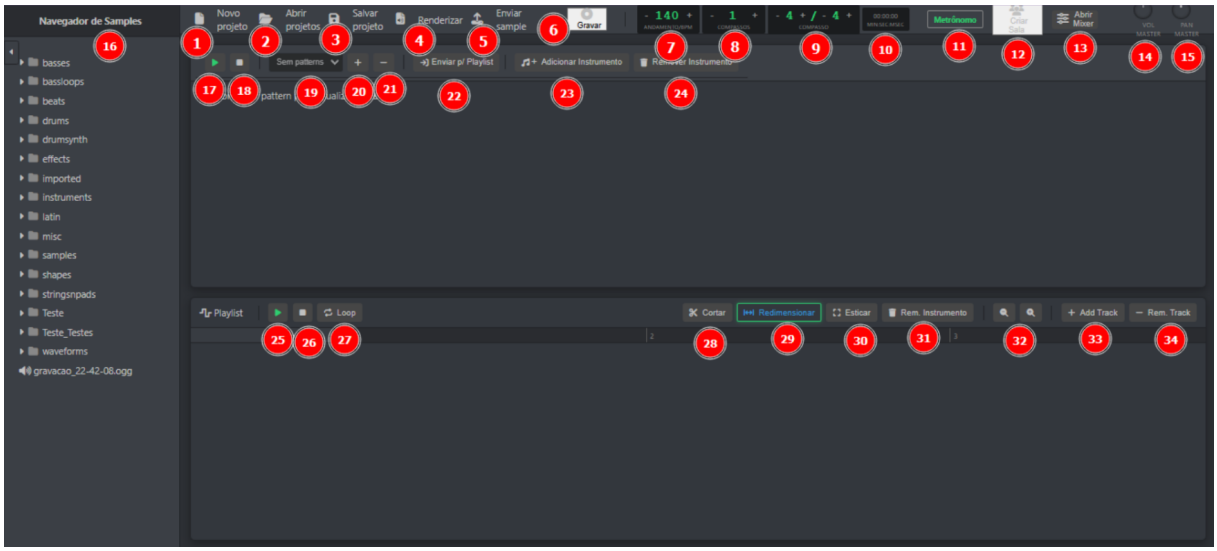

Figura 54 – Interface principal do `mmpCreator` detalhando as áreas de trabalho e ferramentas.

Fonte: O Autor (2026).

A interface do `mmpCreator` foi estruturada para transpor o ambiente de produção musical para a *web* de forma intuitiva. A Barra de Ferramentas Superior (itens de 1 a 15) concentra os controles globais do projeto. Dentre esses controles, destaca-se a opção de Abrir Projetos (2) salvos no servidor e, fundamentalmente para o escopo desta pesquisa, a funcionalidade de Criar Sala (12). Este botão inicia uma sessão compartilhada via *WebSocket*, permitindo que outros usuários entrem e editem o projeto simultaneamente em tempo real, viabilizando a prática colaborativa, a *jam session* remota.

O fluxo de trabalho se divide em três grandes blocos principais. Tendo o Navegador de Samples à esquerda (16), que disponibiliza as bibliotecas de áudio, o Editor de *Patterns* na área superior central (itens de 17 a 24), focado na criação de batidas e *loops*. Por fim, a *Playlist* ou *Timeline* na área inferior (itens de 25 a 34), onde a música é arranjada e estruturada ao longo do tempo.

Na prática, a fluidez do sistema permite que o artista grave áudios e *samples* utilizando apenas a batida de referência e um aparelho celular, tornando a produção altamente acessível. O sistema também permite disponibilizar esses novos *samples* na plataforma, além de possibilitar o *download* de arquivos compartilhados por outros produtores. A concepção desta arquitetura não foi puramente linear, mas consolidou-se também através das descobertas empíricas durante o desenvolvimento do motor de busca (`mmpSearch`). Na Figura 55 é possível ver como era o objetivo inicial, ser um simples buscador de projetos,

com opções de filtragem isoladas e poucas funcionalidades integradas.

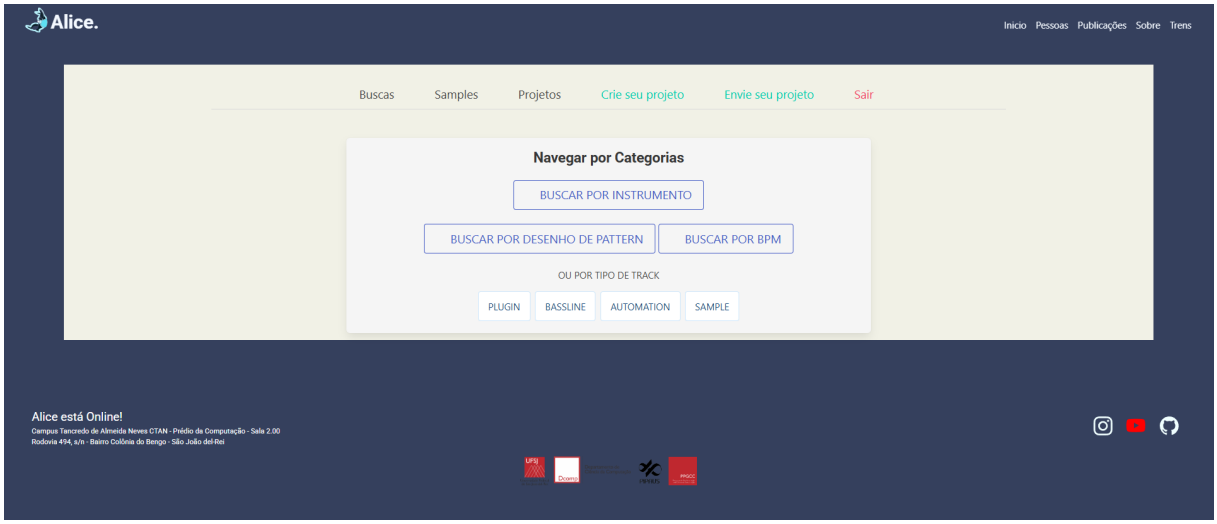

Figura 55 – Versão inicial da interface do `mmpSearch`, focada em buscas simples e isoladas.

Fonte: O Autor (2026).

Ao validar que era possível realizar a mineração profunda dos arquivos de projeto do LMMS e extrair a estrutura lógica completa da composição (notas, instrumentos, automações), emergiu a viabilidade técnica de reconstruí-los sonoramente via *Web Audio API*. Mas não foi só a DAW que virou uma novidade na plataforma. Observando a Figura 56, é notável o avanço que esse domínio da mineração de dados trouxe para a plataforma.

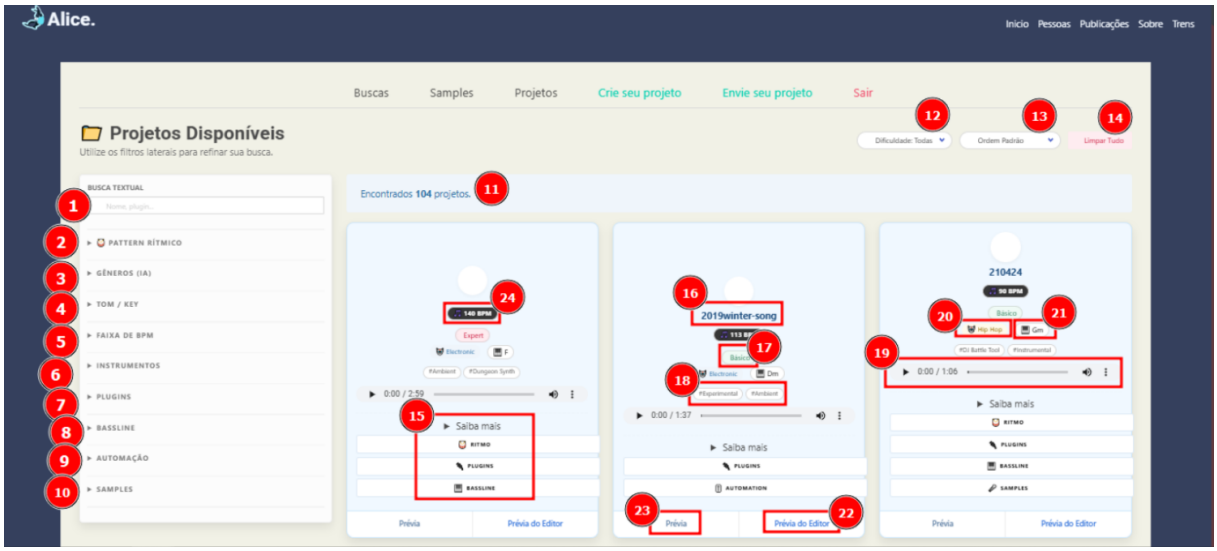

Figura 56 – Interface consolidada do `mmpSearch`, detalhando o sistema de filtragem avançada e os *cards* interativos.

Fonte: O Autor (2026).

A interface evoluiu para um sistema robusto de refinamento e visualização. À esquerda, a Barra Lateral agora concentra múltiplos filtros dinâmicos (itens de 1 a 10), permitindo cruzamentos complexos. O usuário pode realizar buscas textuais (1), definir faixas de BPM (5) e até mesmo utilizar parâmetros avançados, como a busca visual por *patterns* rítmicos desenhados na tela (2) e a filtragem por subgêneros classificados por Inteligência Artificial (3).

Além disso, os resultados passaram a ser exibidos em *cards* detalhados (itens de 15 a 24), que entregam um alto nível de informação antes mesmo de o projeto ser aberto. Eles oferecem um *player* de áudio para escuta imediata da renderização (19), *tags* visuais de dificuldade (17) e tonalidade (21), além de detalhes técnicos em menus expansíveis (15). Coroando a integração da arquitetura, o botão de "Prévia do Editor" (22) permite abrir a interface do `mmpCreator` em modo *embed* diretamente do resultado da busca, conectando a pesquisa musical à prática criativa em um único fluxo.

A ideia não visa romper com a necessidade do "estúdio" fixo. Mas viabilizar um estúdio em qualquer lugar, onde estiver[3]. Ao permitir que projetos complexos fossem abertos e editados em dispositivos móveis (celulares e *tablets*) sem dependência de servidores de renderização pesados, a ferramenta devolveu a música ao espaço público. Pedagogicamente, isso visa alterar a dinâmica das oficinas de transferência de saber realizadas. A aprendizagem na produção musical deixa de ficar restrita a laboratórios de informática climatizados. Com o `mmpCreator`, observa-se que uma oficina de *beatmaking* pode ocorrer em uma praça ou em um centro comunitário sem infraestrutura de TI, dependendo apenas de que os participantes tivessem internet em seus celulares ou conexão compartilhada via roteador local.

4.5 Considerações Finais

Este capítulo apresentou a avaliação quantitativa e qualitativa da plataforma T.R.E.N.S., validando as hipóteses centrais que guiaram a sua concepção. Sob a ótica do processamento no servidor, os resultados experimentais evidenciaram que o modelo de Paralelismo de Dados é altamente eficaz para a curadoria massiva de projetos musicais. Embora o gargalo físico de I/O tenha limitado o ganho de aceleração teórico, este comportamento emulou com fidelidade as restrições de infraestrutura frequentemente enfrentadas por iniciativas independentes, atestando a resiliência do *pipeline* de ingestão distribuída.

Na outra extremidade da rede, a avaliação da interface no *frontend* confirmou que a Computação na Borda figura como a solução ideal para contornar a precariedade da infraestrutura de telecomunicações. Ao isolar a latência da síntese sonora no dispositivo

[3] E é claro, possua conexão com a internet. Contudo, a transformação desta ferramenta em contêineres *Docker* é vista como uma possibilidade viável, para facilitar o escalonamento em plataformas distintas. Mas será discutido de forma mais ampla no Capítulo 5.

local do usuário, o *mmpCreator* devolve ao artista o controle soberano sobre o microtempo e o *groove*, elementos indispensáveis para a produção de música urbana. Para além das métricas puramente computacionais, a taxonomia sociotécnica extraída dos metadados revelou uma plataforma capaz de atuar como mediadora pedagógica. Ao mapear a estética da escassez, o sistema demonstrou sucesso na conversão de projetos complexos em objetos de aprendizagem acessíveis e nivelados.

A integração destas soluções tecnológicas culminou em uma verdadeira plataforma de letramento musical, onde basicamente só restaria a inclusão de videoaulas para cobrir todas as frentes de tutoria. A construção dessa arquitetura vai muito além do código-fonte. Ela partiu da vontade visceral de viabilizar a criação artística até desaguar na consolidação prática do "marginal digital". Ainda que o termo "ecossistema" tenha sido evitado nas etapas iniciais da pesquisa, a maturidade atual do sistema e a simbiose entre as ferramentas inevitavelmente configuram um. Ao libertar o usuário periférico das amarras da nuvem hegemônica, a tecnologia atua não como um fim, mas como um meio fluido para a conexão humana, onde o som e a pedagogia coexistem e resistem dentro da essência descentralizada da cultura *underground*.

Com a arquitetura validada e a prova de conceito técnica consolidada como uma ferramenta de emancipação digital, os resultados obtidos sintetizam a trajetória do trabalho inteiro. Dessa forma, o Capítulo 5 apresenta as conclusões gerais desta pesquisa, recapitulando os objetivos alcançados e delineando o roteiro para os trabalhos futuros. Nele, serão discutidas as possibilidades de expansão do sistema, como a conteinerização (via *Docker*) para garantir escalabilidade global e a implementação de novos recursos colaborativos, assegurando que a evolução contínua da plataforma permaneça alinhada às necessidades da base comunitária.

5 Conclusão e Trabalhos Futuros

Este trabalho nasceu da profunda inquietação de conectar dois mundos aparentemente distantes, a rigidez estrutural da Computação de Alto Desempenho e a fluidez criativa, subversiva e pulsante da cultura *underground*. Mais do que um artefato acadêmico, esta pesquisa ergue-se como uma dívida de gratidão com a cultura do *Hip Hop*, que historicamente nos ensinou a transformar a escassez tecnológica em inovação estética. O que germinou na ingenuidade de uma ideia empírica amadureceu através do rigor metodológico, concretizando-se ao alinhar a vivência adquirida em mais de uma década nas ruas com a vanguarda teórica da ciência da computação contemporânea.

A pesquisa demonstrou de forma inequívoca que a aplicação de conceitos complexos, como Sistemas Distribuídos e Computação Paralela, não deve estar restrita à mera otimização de processos corporativos, financeiros ou militares. Pelo contrário, quando intencionalmente desenhada para as margens, essa densidade técnica converte-se em infraestrutura crítica para a emancipação cultural.

O propósito central da plataforma T.R.E.N.S. (*Tecnologias em Rede para Emancipação e Narrativas Sonoras*) vai além de uma mera arquitetura de *software* para operar como uma metáfora urbana viva. Na cultura *underground* e nas origens do *Hip Hop*, os trens foram os veículos primários de subversão geográfica e estética, serviam como telas móveis para o *graffiti* e transportavam a genialidade das margens para o coração dos grandes centros. De forma análoga, o sistema desenvolvido atua como uma infraestrutura de mobilidade digital. Ele visa escoar a produção cultural *underground*, potencializando talentos historicamente silenciados e rompendo as fronteiras impostas pela exclusão digital.

Trata-se de subverter a lógica extrativista e centralizadora das grandes plataformas corporativas para democratizar, em um primeiro momento, o "fazer" musical. Este agir materializa-se no motor de renderização distribuída e na borda (`mmpCreator`), que devolve ao criador o domínio sobre o seu tempo e o seu *groove*, protegendo-o da latência. Fundamentalmente, a arquitetura democratiza o "saber" coletivo. Através da curadoria automatizada e da busca heurística (`mmpSearch`), os projetos dispersos nos repositórios são mapeados e interligados, permitindo que o *software* livre funcione não apenas como ferramenta, mas como uma verdadeira locomotiva de letramento, onde a arte do outro pode ser escutada, estudada e remixada em rede.

A resposta arquitetural proposta, uma plataforma híbrida que combina a força bruta da ingestão paralela no servidor com a agilidade e resiliência do processamento na borda (*Edge Computing*) no cliente, provou-se não apenas eficaz, mas socialmente impe-

rativa. O sistema viabilizou o acesso a ferramentas de letramento em produção musical de forma descentralizada, respeitando a autonomia dos dados e abraçando a realidade intermitente e heterogênea de *hardware* que está em poder dos artistas independentes.

5.1 Síntese das Contribuições

A presente pesquisa buscou se diferenciar do escopo tradicional da Engenharia de *Software* ou Computação de Alto Desempenho ao propor uma intervenção direta nas barreiras de exclusão digital estrutural. Ao materializar a arquitetura, este trabalho demonstra que é possível subverter o uso de tecnologias de ponta, como a Computação na Borda, o processamento paralelo e a Inteligência Artificial, empregando-as não para concentrar poder computacional em grandes corporações, mas para descentralizá-lo e democratizá-lo. Utilizar a Computação a favor da sociedade e da cultura.

A solução concebida atua como uma ponte entre a materialidade frequentemente precária do "aqui e agora" e a vasta potência criativa das comunidades de *software* livre. Ao questionar a lógica hegemônica da nuvem (*Cloud Computing*) e priorizar a autonomia do dispositivo local, a pesquisa apresenta um modelo viável de soberania tecnológica profundamente alinhado aos ideais da pedagogia crítica de Paulo Freire e do construtivismo social de Vygotsky (VYGOTSKY, 1978). Além disto, um outro motivo para o dispositivo local vai ser discutido abaixo, na Seção 5.3.

Neste sentido, os achados empíricos e os artefatos desenvolvidos ao longo deste estudo não se encerram na mera validação de métricas de desempenho. Eles se desdobram em impactos concretos para a educação musical informal, para o *design* de sistemas distribuídos sob restrições severas de conectividade e para a própria cultura do *sampling*. Deste modo, as principais contribuições técnico-científicas, metodológicas e sociais alcançadas por este trabalho consubstanciam-se em seis pilares fundamentais, delineados a seguir:

Arquitetura de Ingestão Paralela (HPC):

O desenvolvimento do agente de mineração (*mmpSearch backend*) demonstrou a aplicabilidade do modelo de Paralelismo de Dados para repositórios de mídia não estruturada. A implementação do pipeline ETL com contorno do GIL (*Global Interpreter Lock*) via `multiprocessing` reduziu drasticamente o tempo de indexação, validando a escalabilidade vertical do sistema.

Paradigma *Thick Client* em Áudio:

A ferramenta `mmpCreator` validou a viabilidade da Web Audio API para DSP (*Digital Signal Processing*) complexo. Ao eliminar o *Round Trip Time* (RTT) da renderização sonora, o sistema provou que a latência de rede pode ser desacoplada da latência de interação musical, permitindo performance em tempo real.

Métrica de Proficiência Automatizada:

A implementação de um classificador heurístico baseado na Taxonomia de Bloom (ANDERSON; KRATHWOHL, 2001) constitui uma contribuição metodológica para a aprendizagem musical mediada por tecnologia. O algoritmo demonstrou ser capaz de inferir o nível de maturidade técnica do produtor através da análise estática do *XML*, habilitando filtros pedagógicos personalizados, vide Figura 32. Essas métricas podem ser aprofundadas e podem ser migradas para a composição em tempo real no mmpCreator, fazendo indicações de possibilidades no projeto, de acordo com o seu nível atual.

Formalização Semântica da Música:

A engenharia reversa do formato nativo do LMMS permitiu tratar a música não como binário opaco, mas como grafo de objetos estruturado. Isso estabeleceu as bases para a preservação digital e a recuperação de informação musical (MIR) híbrida (Simbólica + Acústica). Diferentes ritmos, podem ser exemplificados a partir de uma simples busca por desenho de *pattern*.

Impacto Sociotécnico e Emancipação Cultural:

Para além das inovações de código, este trabalho consolida a tecnologia como uma ferramenta de intervenção social. Ao fundir a arquitetura distribuída com a ética do *software* livre e a resiliência da cultura *underground*, a plataforma devolve a soberania de dados ao artista independente. A plataforma não apenas democratiza o acesso a "produção musical"[1], mas subverte a lógica de dependência e individualismo imposta por grandes corporações.

Transferência de Conhecimento e Documentação Técnica:

Para garantir que a ferramenta consiga ultrapassar as barreiras do ambiente acadêmico e seja efetivamente apropriada pela comunidade, foram desenvolvidos manuais de utilização detalhados e visuais para ambos os subsistemas. O manual do *mmpSearch*[2] documenta o processo de busca, filtragem e navegação pedagógica. Paralelamente, o manual do *mmpCreator*[3] mapeia a interface de produção na borda, detalhando desde a manipulação de *patterns* e ferramentas de corte até a renderização do áudio. Esta documentação técnica atua como o alicerce para a acessibilidade e redução da curva de aprendizado da plataforma.

Utilização em larga escala:

Um dos pontos mais fortes e bem vistos para a utilização em larga escala desta abordagem é na plataforma oficial de distribuição do LMMS (LSP)(LMMS, 2025). Após

[1] Sendo possível gravar suas composições, caso não tenha acesso a um estúdio. Além disso, é possível criar/editar/remixar projetos já existentes ou utilizá-los para esclarecer como deve utilizar algum *sample* ou *plugin*, por exemplo.

[2] Disponível em: <[https://alice.ufsj.edu.br/~jotachina/manuais/mmpSearch.pdf]>.

[3] Disponível em: <[https://alice.ufsj.edu.br/~jotachina/manuais/mmpCreator.pdf]>

a validação da ferramenta e a correção de alguns pontos, descritos abaixo na Seção 5.2, T.R.E.N.S será ofertado para a comunidade LMMS com a finalidade de auxiliar na curadoria dos projetos compostos pela comunidade. Acredita-se que o repositório vasto da comunidade[4] possa ser um baú do tesouro, contendo anos de composições, inúmeros estilos musicais e artistas de localidades diferentes. Entretanto, outra forma de utilização em larga escala é discutida de maneira mais ampla na Seção 5.3.

5.2 Limitações do Estudo

O desenvolvimento e a avaliação de uma plataforma sociotécnica da magnitude do T.R.E.N.S. são, por natureza, processos iterativos. Reconhecer as fronteiras operacionais alcançadas nesta pesquisa não invalida os resultados, mas demonstra maturidade acadêmica ao demarcar com precisão o escopo do que foi solucionado e o que ainda representa um desafio em aberto na intersecção entre Sistemas Distribuídos e Processamento Digital de Sinais (DSP). Abaixo, descrevem-se as principais limitações identificadas:

Gargalo de I/O em Infraestrutura Herdada:

Embora a implementação do Paralelismo de Dados tenha contornado o *Global Interpreter Lock* (GIL) e maximizado o uso dos núcleos lógicos da CPU durante o *pipeline* ETL, a infraestrutura de testes impôs restrições físicas incontornáveis. A utilização de discos rígidos mecânicos (HDD SATA III) no servidor de homologação gerou uma saturação de leitura e escrita (Gargalo de I/O), estagnando o *Speedup* em grandes lotes. Para uma implantação massiva em nível de produção (como a indexação integral da LSP), a migração para arquiteturas de armazenamento de estado sólido (SSDs NVMe) torna-se mandatória para acompanhar o poder de processamento do algoritmo.

Heterogeneidade e Volatilidade de Dependências de Áudio:

Uma limitação crítica inerente à preservação e curadoria de música digital é a dependência de ativos externos ao arquivo de projeto XML. O sistema atual lida com eficiência com instrumentos nativos, mas esbarra na opacidade de *plugins* binários proprietários (como VSTs legados) e na ausência de amostras de áudio (*samples*) armazenadas localmente na máquina do produtor original. Como o *pipeline* de renderização do LMMS opera em modo *headless* (sem interface gráfica), falhas de carregamento nestes binários muitas vezes resultam em exceções fatais (*Segmentation Faults*), vide Figura 42. Embora o *backend* isole esses erros garantindo a continuidade da ingestão, o sistema ainda não realiza o *sandboxing* isolado ou a virtualização destes ativos externos, o que pode resultar em versões renderizadas sonoramente incompletas quando comparadas à visão original do artista.

[4] Em Março de 2026, possui mais de 11,5 mil projetos.

Resolução Semântica em Conflitos de Consistência Eventual:

Sob a ótica do Teorema CAP (BREWER, 2000), a decisão arquitetural de priorizar a Disponibilidade (A) e a Tolerância à Partição (P) para atender ao "aqui e agora" gera, invariavelmente, cenários de Consistência Eventual. A arquitetura permite que os usuários continuem compondo durante blecautes de rede, armazenando as alterações no navegador[5]. Contudo, em casos de desconexões severas e prolongadas, a sincronização do estado local com a nuvem baseia-se atualmente na resolução de conflitos técnicos (sobrescrita por *timestamp*). O sistema ainda não possui a capacidade de mesclar conflitos semânticos. Não é possível unir harmonicamente duas versões divergentes do mesmo projeto musical (mesma sala) criadas colaborativamente[6], exigindo para o futuro a implementação de algoritmos avançados, como CRDTs (*Conflict-free Replicated Data Types*).

Utilização prática: A versão atual possui algumas limitações em ferramentas auxiliares que são básicas (para utilização), mas complexas (para implementação) que estão visíveis na interface gráfica, mas não estão presentes no *backend*, como a possibilidade de cortar e esticar *patterns* de áudio na *playlist* como visto na Figura 57. Neste momento eles funcionam apenas para cortar amostras de áudio, como o *sample* "#2.wav" desta mesma Figura. Ainda não é possível desfazer ações, consequentemente, não é possível refazer também. Outra limitação é vista na falta de possibilidade de adequar o BPM de acordo com o ritmo em que alguma tecla é apertada, similar ao de outras DAWs famosas. Essa ferramenta auxilia durante a descoberta de BPM e ritmo a partir da audição, partindo da utilização da contagem rítmica do tempo musical.

Outro ponto que ainda é limitado na utilização são os *pianorolls*, pois no mmpSearch (vide Figura 58), a visualização dos *steps* é integrada, a visualização do *pianoroll* ainda não é possível. Projetos que possuam apenas composições com *pianoroll*, não serão bem exibidas na plataforma, conforme visto na Figura 59. Já no mmpCreator, a visualização e manipulação (limitida) é possível. Mas não reflete 100% o que está descrito na composição. São divergências temporais que estão sendo mapeadas para que obedeça o relógio central da criação. Um exemplo do *pianoroll* no mmpCreator é visto nas Figuras 60 e 61.

Sobrecarga Computacional na Borda (Limitações do Navegador):

Embora a transferência do processamento digital de sinais (DSP) para o *Thick Client* resolva o problema da latência de rede, ela introduz um gargalo de *hardware* na máquina do usuário final. Como a Web Audio API e a gestão de estado ocorrem integralmente na memória RAM e na CPU do dispositivo cliente, projetos que contemplem dezenas de trilhas simultâneas ou grafos de áudio muito densos podem causar lentidão,

[5] Os dados ficam armazenados em cache do navegador. Possibilitando a sua utilização em caso de queda de rede até que a página seja recarregada ou que seja enviado um novo estado autoritativo do servidor.

[6] Neste caso, apenas deve atualizar a página e receber o estado autoritativo do servidor.

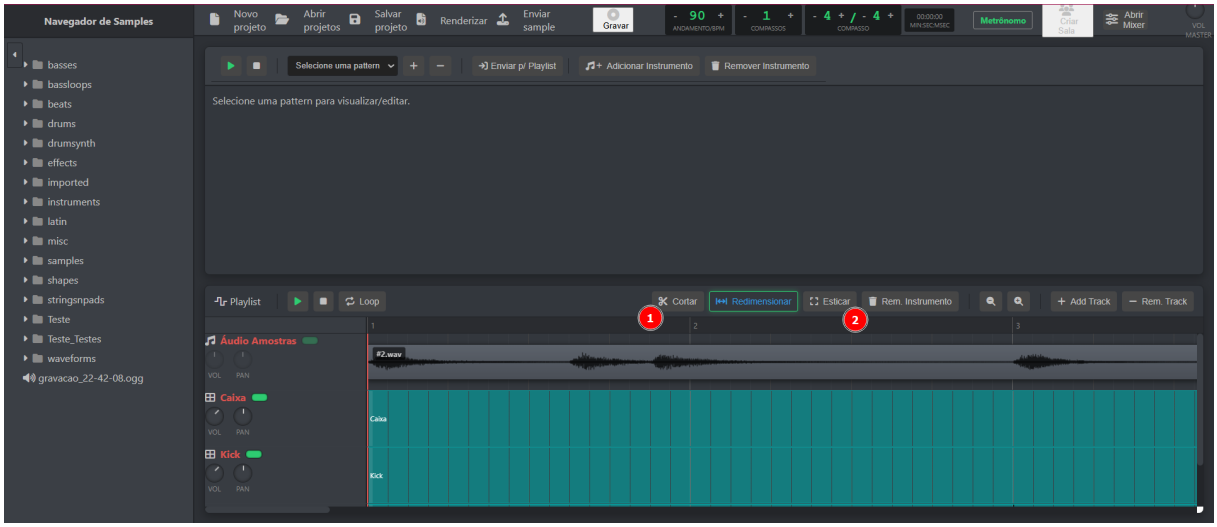

Figura 57 – Interface gráfica do mmpCreator, onde é indicado pelo número 1 o botão para cortar e 2 para o botão de esticar artefatos na *playlist*.

Fonte: O Autor (2026).

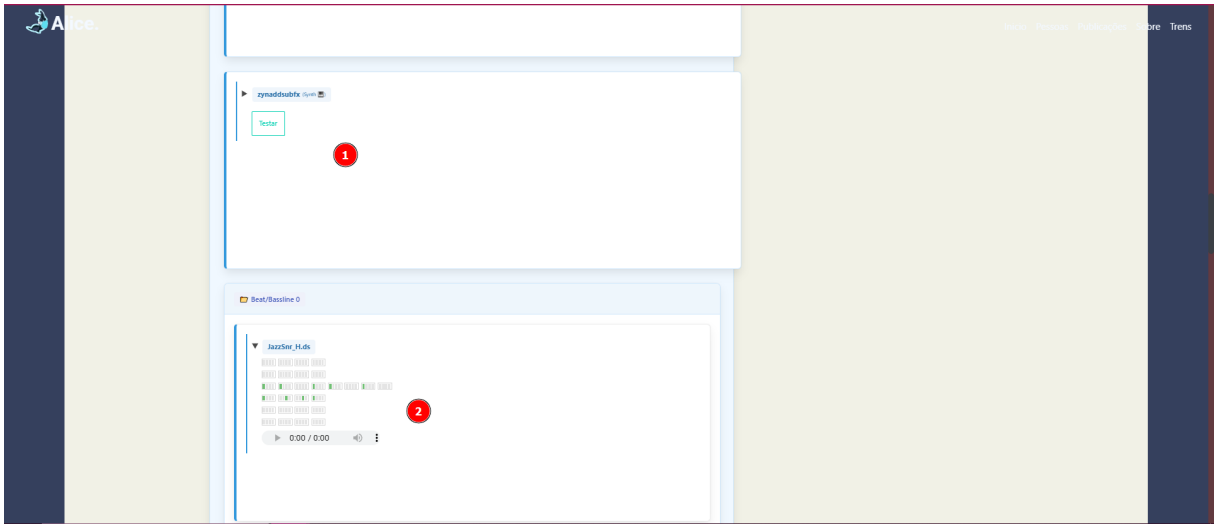

Figura 58 – Interface do mmpSearch exibindo projeto que tem composição com *steps* e *pianoroll*.

Fonte: O Autor (2026).

engasgos na reprodução (*dropouts*) ou até mesmo o travamento da aba do navegador em computadores mais antigos ou *smartphones* com recursos limitados. Apesar da ferramenta ter sido concebida para ser leve e focada na experimentação rápida, o motor JavaScript atual é limitado e pode alcançar um teto de performance estrutural em composições massivas, com um número elevado de faixas de áudio.

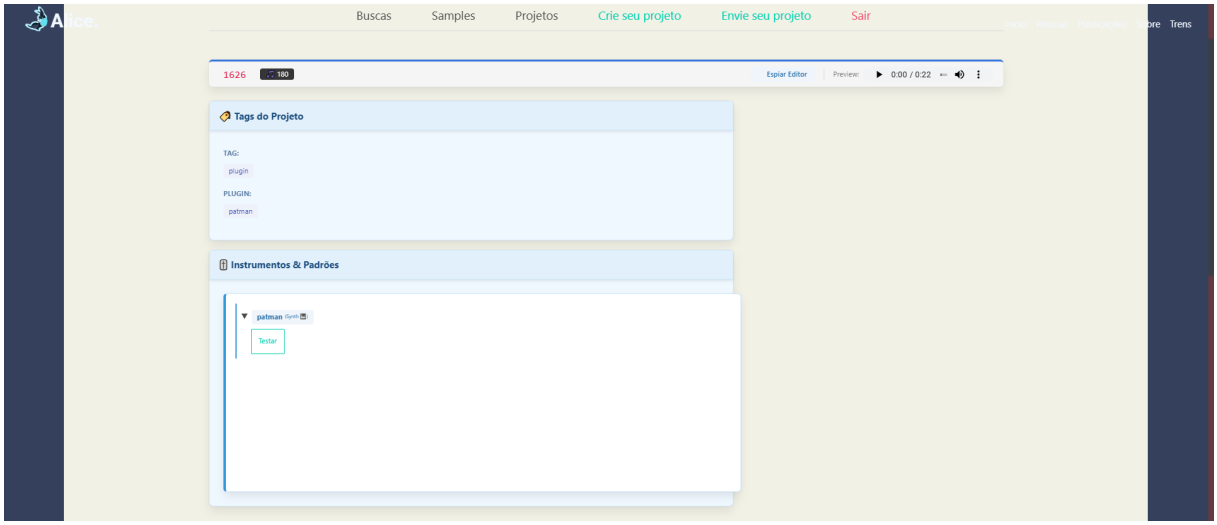

Figura 59 – Interface do mmpSearch exibindo projeto que tem composição apenas com *pianoroll*.

Fonte: O Autor (2026).

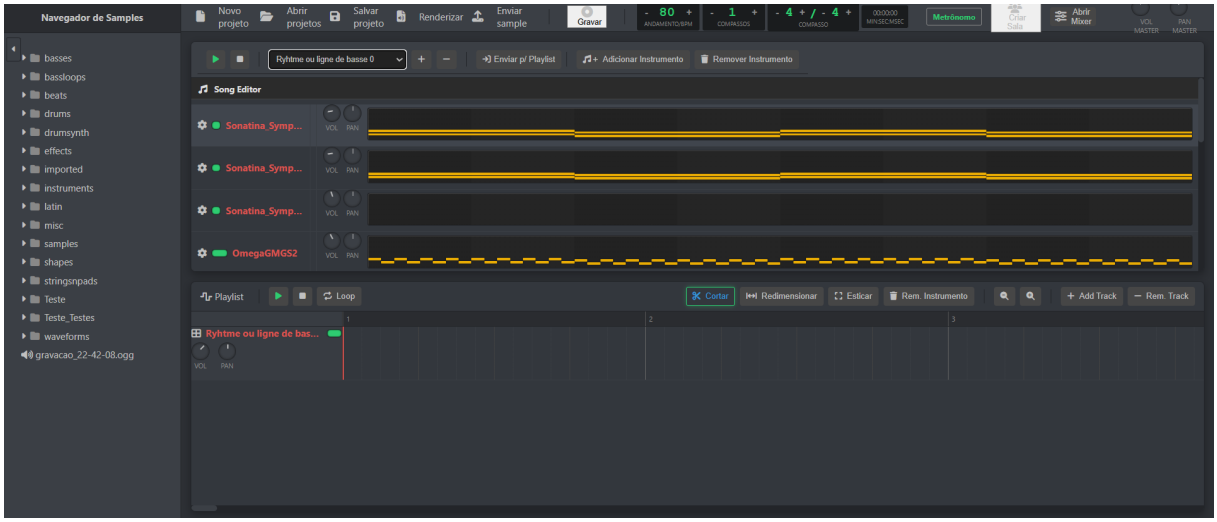

Figura 60 – Interface do mmpCreator com o editor de *patterns* de um projeto que possui apenas composição utilizando o *pianoroll*.

Fonte: O Autor (2026).

5.3 Trabalhos Futuros

A infraestrutura consolidada e materializada na plataforma T.R.E.N.S. atua agora como fundação para uma agenda de pesquisa de longo prazo. As rotas de expansão delineadas a seguir visam alargar as capacidades técnicas do sistema e aprofundar seu impacto sociotécnico:

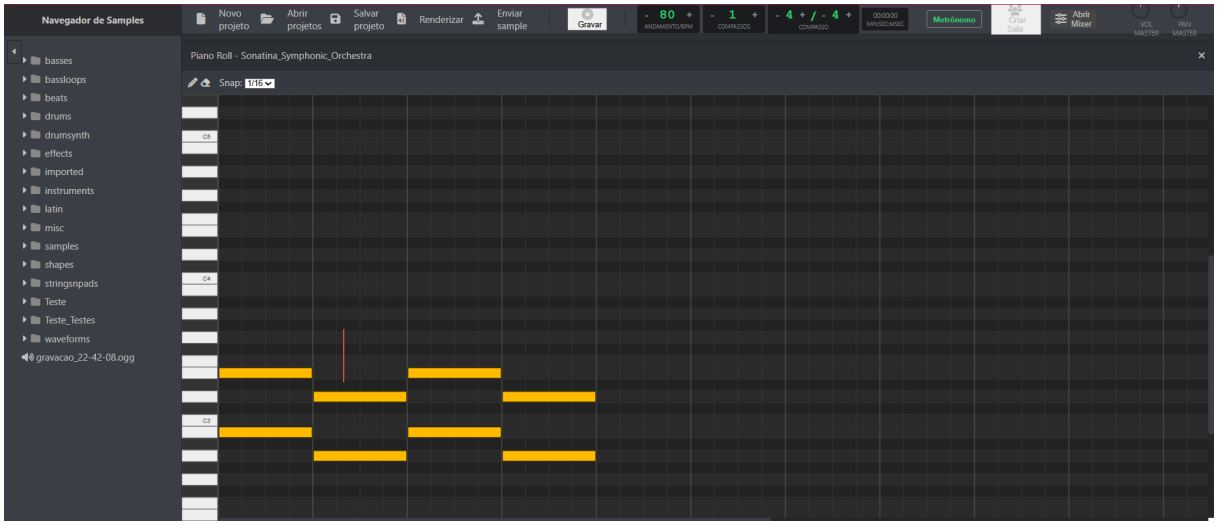

Figura 61 – Interface do mmpCreator com um instrumento aberto em evidência no editor de *patterns* de um projeto que possui apenas composição utilizando o *piano-roll*.

Fonte: O Autor (2026).

Evolução Estocástica do Protocolo de Sincronização:

O atual algoritmo de sincronização temporal, baseado em médias móveis exponencialmente ponderadas (EWMA), provou-se eficaz para compensar o atraso em redes domésticas típicas. Contudo, para cenários de conectividade extrema, como internet via satélite ou redes móveis rurais (3G/4G degradado), propõe-se a substituição do EWMA por algoritmos preditivos baseados em modelos estocásticos, como Filtros de Kalman, ou por estratégias de controle de congestionamento baseadas em modelo, como o BBR (*Bottleneck Bandwidth and Round-trip propagation time*). Esta evolução permitiria ao sistema antecipar flutuações severas de latência com precisão matemática, preservando o *groove* e a micro temporalidade musical mesmo sob condições adversas de infraestrutura.

Integração com IA Generativa Simbólica e Co-Criação:

Uma extensão natural da taxonomia extraída é a utilização da base de dados indexada para o treinamento de modelos de Inteligência Artificial. Diferentemente de abordagens predatórias que geram áudio bruto (*waveforms*) e ameaçam a autoria humana, o *dataset* estruturado em XML permite treinar *Large Language Models* (LLMs) especializados na *sintaxe* da composição musical (IA Simbólica). Futuras iterações poderão incluir "co-pilotos" algorítmicos na borda, capazes de sugerir progressões de acordes ou parametrizações de sintetizadores baseadas na análise estatística de milhares de projetos de RAP e *Trap* nacional, atuando como um parceiro criativo e não como um substituto do artista.

Além disso, com o treinamento de modelos é possível especializá-los para inferir

novos projetos a partir dos *logs* gerados nas sessões colaborativas. Desta forma, os modelos podem aprender com os próprios usuários, além de toda a base documentada e estruturada, conforme *Pipeline ETL* utilizado e descrito nos Capítulos 2 e 3. Sendo assim, o usuário poderá solicitar à plataforma projetos pré-programados para um estilo específico, projetos com configurações determinadas por um *prompt* ou itens clicáveis, por exemplo.

Consolidação de um *Dataset* Aberto Decolonial:

Do ponto de vista científico e cultural, o subproduto mais valioso desta pesquisa pode não ser o *software*, mas sim os dados curados. A consolidação do primeiro *Dataset* Aberto de Projetos do LMMS, higienizado, estruturado em JSON e enriquecido com descritores semânticos, inaugura a possibilidade de treinar modelos de *Machine Learning* sintonizados com a estética *underground* global. Esta iniciativa combate ativamente o viés eurocêntrico e mercadológico das bases de dados comerciais atuais, estabelecendo um marco para o desenvolvimento de inteligências artificiais musicais decoloniais.

Otimização de DSP na Borda via WebAssembly (Wasm):

Para transpor as limitações de performance do motor *JavaScript* ao processar projetos com dezenas de trilhas simultâneas, propõe-se a migração do motor de áudio (*AudioWorklet*) para *WebAssembly* (Wasm). A compilação cruzada de bibliotecas em C++ (do próprio núcleo nativo do LMMS) ou Rust para Wasm permitiria a execução de algoritmos complexos de Processamento Digital de Sinais (DSP), como *reverbs* de convolução e síntese granular, com performance muito próxima à nativa. Isso transformaria navegadores rodando em *smartphones* de baixo custo em verdadeiras *Digital Audio Workstations* (DAWs) de alta fidelidade.

Federação de Repositórios e o Fediverso Musical:

Inspirado no protocolo *ActivityPub* que fundamenta o Fediverso (redes sociais descentralizadas), vislumbra-se a evolução do sistema para uma arquitetura de curadoria federada. Esta topologia permitiria que diferentes instâncias do laboratório ALICE, hospedadas em universidades distintas, coletivos culturais ou associações de bairro, trocassem e replicassem índices de projetos entre si via rede *Peer-to-Peer* (P2P). Cria-se, com isso, uma "biblioteca de Alexandria" musical distribuída, inerentemente resiliente à censura e estruturalmente independente dos monopólios das grandes plataformas corporativas (*Big Techs*).

Contudo, visto que existe uma quantidade considerável de Instituições Federais (Universidades, Colégios, Institutos e outras) não só no Brasil, mas no mundo, traz a possibilidade de um enorme repositório descentralizado e aberto com *terabytes* de informações sobre projetos e composição de *beats*. Utilizando de uma infraestrutura (física) que já está montada e existe, trazendo mais uma forma de beneficiar a comunidade.

Interação Tátil via Web MIDI API:

Para estreitar de forma definitiva a lacuna entre a materialidade do estúdio físico e a imaterialidade da plataforma *web*, propõe-se a integração nativa do `mmpCreator` com a *Web MIDI API*. Esta evolução arquitetural permitirá que o navegador reconheça interfaces e controladores de *hardware* externos de forma *Plug and Play*. Dessa forma, produtores e *beatmakers* poderão tocar os sintetizadores virtuais instanciados na borda utilizando teclados reais e *drum pads* (no estilo MPC). Esta integração restaura a fenomenologia tátil e a fisicalidade da performance, elementos que são a espinha dorsal da cultura *Hip Hop* e da prática dos *DJs* e *Beatmakers*, eliminando a fricção de instalação de *drivers* ou *softwares* proprietários intermediários.

5.4 Considerações Finais

Este trabalho encerra-se com a profunda convicção de que a tecnologia não é neutra, quando intencionalmente desenhada sob a ótica da liberdade e da colaboração descentralizada, ela atua como uma aliada inexorável da cultura. Conforme discutido ao longo desta pesquisa, e embasado na visão sociotécnica de Katz (KATZ, 2012), a contracultura *Hip Hop* sempre operou na margem do erro, encontrando formas de hackear as ferramentas hegemônicas para forjar sua própria voz. Se nos primórdios do movimento o toca-discos foi subvertido, deixando de ser um aparelho passivo de consumo burguês para se tornar um instrumento ativo de percussão e criação, esta dissertação demonstra que o navegador de *internet*, operando na Borda da rede, pode sofrer a exata mesma subversão, transmutando-se em um estúdio colaborativo, autônomo e emancipador.

Ao mitigar as barreiras estruturais, técnicas e financeiras que separam a ideia da sua execução material, a plataforma distribuída T.R.E.N.S. prova não processar apenas estruturas XML abstratas, nós da *Web Audio API* ou *buffers* de memória alocados em servidores. Na sua essência, ele processa sonhos, rimas e vivências. A apropriação radical do *software* livre documentada nestas páginas comprova, na prática, que a soberania de dados e a independência infraestrutural são os equivalentes digitais e contemporâneos da posse dos próprios *masters* fonográficos. É a recusa categórica em aceitar o papel subserviente de mero "consumidor" ou "produto" nas engrenagens das grandes plataformas de *streaming*, resgatando para o artista independente o controle absoluto sobre seus próprios meios de produção. Outro ponto de vista é, trazer ao artista independente a possibilidade de produzir seus próprios sons, gravar suas próprias guias, faixas oficiais ou até mesmo seus álbuns. Transformando a falta de horas de estúdio em horas criativas, horas produtivas ou até mesmo em horas de produção/gravação musical.

Como a própria fenomenologia do *underground* nos ensina diariamente, a escassez material nunca foi um ponto final. Ela é, o compasso inicial para a invenção subversiva. Trata-se de reivindicar o acesso ao que foi negado historicamente. Nas palavras cortantes

de Djonga, em sua obra “Ladrão”:

> “Eu vou roubar o patrimônio do seu pai
> Dar fuga no Chevette e distribuir na favela
> Não vão mais empurrar sujeira pra debaixo do tapete
> E nem pra debaixo da minha goela, eu sou ladrão!
> Os cara faz rap pra boy
> Eu tomo dos boy no ingresso o que era do meu povo
> Todo ouro e toda prata, passa pra cá
> O mais responsável dos mais novo, fé
> Correndo essa maratona, e conforme for
> Uso a mão santa, Maradona
> ...”(DJONGA, 2019)

A *session* que outrora dependia exclusivamente do encontro físico nas ruas e calçadas, hoje encontra sua expansão ininterrupta na malha digital. Buscando garantir, desta forma, que a voz e o *groove* do *underground* continuem ecoando, não mais silenciados pela exclusão digital, mas amplificados pela resiliência da Computação Distribuída e eternamente protegidos pela ética do *software* livre.

Referências

Ableton. *Ableton Live 12 Suite*. 2026. Disponível em: <https://www.ableton.com/en/shop/live/>. Acesso em: 16 mar. 2026. Citado nas páginas 26 e 27.

ABRAMOV, D.; CLARK, A. *Redux: A predictable state container for JavaScript apps*. 2015. Acesso em: 25 abr. 2025. Disponível em: <https://redux.js.org/introduction/core-concepts>. Citado na página 79.

ADENOT, P.; WILSON, C.; ROGERS, C. *Web Audio API*. [S.l.], 2021. Disponível em: <https://www.w3.org/TR/webaudio/>. Citado na página 54.

AMDAHL, G. M. Validity of the single processor approach to achieving large scale computing capabilities. *Proceedings of the AFIPS conference*, v. 30, p. 483–485, 1967. Citado nas páginas 41 (3 ocorrências) e 57.

ANDERSON, L. W.; KRATHWOHL, D. R. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York: Longman, 2001. Citado nas páginas 53, 89 e 102.

Apple Inc. *Logic Pro para Mac*. 2026. Disponível em: <https://www.apple.com/logic-pro/>. Acesso em: 16 mar. 2026. Citado na página 27.

Avid Technology. *Pro Tools Studio*. 2026. Disponível em: <https://www.avid.com/pro-tools>. Acesso em: 16 mar. 2026. Citado na página 27.

Banco Central do Brasil – BCB. *Cotação PTAX – Sistema de Informações Banco Central (SISBACEN)*. 2026. Disponível em: <https://www.bcb.gov.br/>. Acesso em: 16 mar. 2026. Citado na página 27.

BELL, A. P. *Dawn of the DAW: The Studio as Musical Instrument*. [S.l.]: Oxford University Press, 2018. Citado na página 46.

BENKLER, Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. New Haven, CT: Yale University Press, 2006. Citado na página 29.

BIILMANN, M.; HAWKSWORTH, P. *Modern Web Development on the JAMstack: Modern Architectures for Scalable Web Applications*. Sebastopol: O'Reilly Media, 2019. Citado nas páginas 51 e 64.

BOGDANOV, D. et al. Essentia: an open-source library for sound and music analysis. In: *Proceedings of the 21st ACM international conference on Multimedia*. [S.l.: s.n.], 2013. p. 493–496. Citado na página 53 (2 ocorrências).

Brasil. *Decreto nº 12.797, de dezembro de 2025. Regulamenta o valor do salário mínimo a partir de 1º de janeiro de 2026*. 2025. Diário Oficial da União, Brasília, DF. Acesso em: 16 mar. 2026. Citado nas páginas 26 e 27.

BREWER, E. A. Towards robust distributed systems. In: *PODC*. [S.l.: s.n.], 2000. v. 7, p. 7. Citado na página 104.

BURGESS, R. J. *The history of music production*. New York: Oxford University Press, 2014. Citado na página 33.

BUSCHMANN, F. et al. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Chichester: John Wiley & Sons, 1996. Citado na página 64.

CARDWELL, N. et al. Bbr: Congestion-based congestion control. *Communications of the ACM*, ACM New York, NY, USA, v. 60, n. 2, p. 58–66, 2017. Citado na página 59.

CARNEIRO, G. R. C. *Wonderland*. 2024. Acessado em: 17 dez. 2024. Disponível em: <https://wonderland.alice.ufsj.edu.br/>. Citado nas páginas 41, 42 (2 ocorrências) e 43.

CASEY, M. A. et al. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, IEEE, v. 96, n. 4, p. 668–696, 2008. Citado na página 46.

CASTLEMAN, C. *Getting up: Subway graffiti in New York*. Cambridge, MA: MIT Press, 1982. Citado na página 31.

CHANG, J. *Can't stop won't stop: A history of the hip-hop generation*. New York: St. Martin's Press, 2005. Citado na página 30.

CHIKOFSKY, E. J.; CROSS, J. H. Reverse engineering and design recovery: A taxonomy. *IEEE software*, IEEE, v. 7, n. 1, p. 13–17, 1990. Citado nas páginas 41 (3 ocorrências) e 43.

CRISTIAN, F. Probabilistic clock synchronization. *Distributed Computing*, Springer, v. 3, n. 3, p. 146–158, 1989. Citado nas páginas 41 (3 ocorrências) e 80.

DJONGA. *Ladrão*. 2019. Plataformas de streaming. Álbum: Ladrão. Gravadora: Ceia Ent. Disponível em: <https://www.youtube.com/watch?v=kYJzEof3oO0>. Citado na página 110.

DOWNIE, J. S. Music information retrieval. *Annual review of information science and technology*, Wiley Online Library, v. 37, n. 1, p. 295–340, 2003. Citado na página 53 (2 ocorrências).

GOFFMAN, K.; JOY, D. *Contracultura através dos tempos: do mito de Prometeu à cultura digital*. [S.l.]: Ediouro, 2007. Citado na página 28.

GORELICK, M.; OZSVALD, I. *High Performance Python: Practical Performant Programming for Humans*. 2. ed. Sebastopol: O'Reilly Media, 2020. Citado nas páginas 41 (3 ocorrências) e 68.

GRAMA, A. et al. *Introduction to Parallel Computing*. 2. ed. Harlow: Addison-Wesley, 2003. Citado na página 68.

HERSCHMANN, M. *O funk e o hip-hop invadem a cena*. Rio de Janeiro: Editora UFRJ, 2000. Citado na página 29.

HERSCHMANN, M.; FERNANDES, C. S. O funk e o *Hip-Hop* invadem a cena: a consolidação da música periférica nas plataformas digitais. *Revista E-Compós*, v. 25, 2022. Referência acadêmica sobre a ascensão comercial dos gêneros urbanos nas plataformas de streaming no Brasil. Citado na página 27.

HOLT, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, Elsevier, v. 20, n. 1, p. 5–10, 2004. Publicado originalmente em 1957 pelo Office of Naval Research. Citado nas páginas 41 (3 ocorrências) e 59 (2 ocorrências).

HUBER, D. M. *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*. 3. ed. Burlington, MA: Focal Press, 2007. Citado na página 33.

Image-Line. *FL Studio All Plugins Edition*. 2026. Disponível em: <https://www.image-line.com/fl-studio/compare-editions/>. Acesso em: 16 mar. 2026. Citado nas páginas 26 e 27.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). *Censo Demográfico 2022: Resultados da Amostra - Trabalho e Rendimento*. 2025. Acesso em: 16 mar. 2026. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/44711-em-9-3-dos-municipios-do-pais-o-rendimento-medio-do-trabalho-era-inferior-a-um-salario-mi
Citado na página 26.

IYENGAR, J.; THOMSON, M. *QUIC: A UDP-Based Multiplexed and Secure Transport*. [S.l.], 2021. Disponível em: <https://www.rfc-editor.org/rfc/rfc9000.txt>. Citado na página 59.

KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of basic engineering*, ASME, v. 82, n. 1, p. 35–45, 1960. Citado na página 59.

KATZ, M. *Capturing sound: How technology has changed music*. 2nd. ed. Berkeley: University of California Press, 2010. Citado na página 33.

KATZ, M. *Groove music: The art and culture of the hip-hop DJ*. [S.l.]: Oxford University Press on Demand, 2012. Citado nas páginas 30, 95 e 109.

KEYES, C. L. *Rap music and street consciousness*. Urbana: University of Illinois Press, 2004. Citado na página 30.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. 3. ed. Indianapolis: John Wiley & Sons, 2013. Citado nas páginas 41 (3 ocorrências) e 66.

KLEPPMANN, M. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. Sebastopol: O'Reilly Media, 2017. Citado na página 67.

KRATUS, J. The ways children compose. *Musical connections: Tradition and change*, International Society for Music Education, p. 128–141, 1994. Citado na página 54.

LEMOS, R.; OUTROS. *O Futuro da Música: tecnologia, pirataria e a cultura livre*. Rio de Janeiro: Editora Aeroplano, 2007. Discussão clássica sobre como a pirataria atua como modelo de distribuição e acesso à tecnologia no Sul Global. Citado na página 27.

LMMS. *LMMS Sharing Platform*. 2025. <https://lmms.io/lsp/>. Citado nas páginas 65, 66, 71 e 102.

LMMS Development Team. *LMMS User Manual: Introduction to LMMS*. [S.l.], 2020. Disponível na documentação oficial de código aberto. Disponível em: <https://lmms.io/documentation/>. Citado na página 35.

MANN, Y. Interactive music on the web. In: ACM. *Proceedings of the Music and the Web Workshop at Audio Mostly*. [S.l.], 2015. Citado na página 78.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008. Citado na página 65.

MARRINGTON, M. Experiencing musical composition in the daw: the software interface as cognitive tool. In: ART OF RECORD PRODUCTION. *Proceedings of the Art of Record Production Conference*. [S.l.], 2011. Citado na página 47.

MILLS, D. L. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, IEEE, v. 39, n. 10, p. 1482–1493, 1991. Citado na página 59.

MÜLLER, M. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Cham: Springer, 2015. Citado na página 82.

OSMANI, A. *Learning JavaScript Design Patterns*. 2. ed. Sebastopol: O'Reilly Media, 2017. Citado na página 79.

PARDUE, D. *Ideologies of marginality in Brazilian hip hop*. New York: Palgrave Macmillan, 2008. Citado na página 29.

PAXSON, V. et al. *Computing TCP's Retransmission Timer*. [S.l.], 2011. Disponível em: <https://www.rfc-editor.org/rfc/rfc6298.txt>. Citado na página 59.

PERRY, I. *Prophets of the hood: Politics and poetics in hip hop*. Durham: Duke University Press, 2004. Citado na página 32.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. 7. ed. [S.l.]: Palgrave Macmillan, 2010. Citado na página 43.

PRIAS, G. D. V.; BARRIOS, C. R. G. La aplicación del software libre en la disciplina de armonía: un desafío para la educación musical en ecuador. *Revista Varona*, Cuba, n. 68, 2019. Citado na página 36.

Racionais MCs. *Homem na Estrada*. 1993. *Raio X do Brasil.*. Disponível em: YouTube. Acesso em: 25 abr. 2025. Citado na página 18.

Racionais MC's. *1 por amor, 2 por dinheiro*. 2002. Disponível em [https://www.youtube.com/watch?v=JovdffVqJF4]. Acesso em: 22 Abr. 2025. Citado na página 32.

Racionais MC's; Afro-X. *A Vida é Desafio*. 2002. Disponível em: <https://www.youtube.com/watch?v=Wb3rvC6z5ao>. Acesso em: 12 Fev. 2026. Citado na página 21.

RAMALHO, L. *Fluent Python: Clear, Concise, and Effective Programming*. 2. ed. Sebastopol: O'Reilly Media, 2022. Citado na página 65.

ROSE, T. *Black noise: Rap music and black culture in contemporary America*. Middletown: Wesleyan University Press, 1994. Citado nas páginas 28 e 31.

RZO. *O Trem*. 1997. Disponível em [https://www.youtube.com/watch?v=_AycHaPQJG0]. Acesso em: 25 Abr. 2025. Citado na página 28.

SABOTAGE RAPPIN' HOOD, H. *A Cultura*. 2002. Disponível em [https://www.youtube.com/watch?v=JovdffVqJF4]. Acesso em: 22 Abr. 2025. Citado na página 17.

SATYANARAYANAN, M. The emergence of edge computing. *Computer*, IEEE, v. 50, n. 1, p. 30–39, 2017. Citado na página 76.

SCHLOSS, J. G. *Foundation: B-boys, B-girls, and hip-hop culture in New York*. New York: Oxford University Press, 2009. Citado na página 31.

SCHLOSS, J. G. *Making Beats: The Art of Sample-Based Hip-Hop*. Middletown, CT: Wesleyan University Press, 2014. Citado na página 31.

SCHMIDT, D. C. et al. *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. New York: John Wiley & Sons, 2000. Citado na página 68.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, IEEE, v. 3, n. 5, p. 637–646, 2016. Citado nas páginas 41 (2 ocorrências) e 54.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 10. ed. Hoboken: John Wiley & Sons, 2018. Citado na página 70.

SMUS, B. *Web Audio API: Advanced Sound for Games and Interactive Apps*. Sebastopol: O'Reilly Media, 2013. Citado na página 76.

SOUSA, J. C.; SOUSA, E. S.; SCHIAVONI, F. L. Cultura underground e software livre. In: *Anais do 8º Congresso Internacional de Arte, Ciência e Tecnologia e Seminário de Artes Digitais 2023*. Labfront/UEMG, 2023. p. 1–8. ISSN 2674-7847. Disponível em: <https://doi.org/10.5281/zenodo.10413450>. Citado na página 21.

SOUSA, J. C. d. *O Software Livre na Criação Musical para a Cultura Underground*. 2023. Monografia (Bacharelado em Ciência da Computação) – Universidade Federal de São João del-Rei. Acesso em: 25 abr. 2025. Disponível em: <https://alice.ufsj.edu.br/papers/2023Jota.pdf>. Citado nas páginas 20, 28, 33 e 45.

SOUSA, J. C. d.; SCHIAVONI, F. L. Depurando o underground: artistas independentes capacitando-se em produção musical com software livre. 2024. Acesso em: 17 dez. 2024. Citado na página 20.

Steinberg Media Technologies. *Cubase Pro 15*. 2026. Disponível em: <https://www.steinberg.net/cubase/>. Acesso em: 16 mar. 2026. Citado na página 27.

STOLZOFF, N. C. *Wake the town and tell the people: Dancehall culture in Jamaica*. Durham: Duke University Press, 2000. Citado na página 29.

SWANWICK, K.; TILLMAN, J. The sequence of musical development: A study of children's composition. *British Journal of Music Education*, Cambridge University Press, v. 3, n. 3, p. 305–339, 1986. Citado na página 53.

SWELLER, J. Cognitive load during problem solving: Effects on learning. *Cognitive science*, Elsevier, v. 12, n. 2, p. 257–285, 1988. Citado na página 53.

THÉBERGE, P. *Any sound you can imagine: Making music/consuming technology*. Middletown: Wesleyan University Press, 1997. Citado na página 33.

VYGOTSKY, L. S. *Mind in society: The development of higher psychological processes*. [S.l.]: Harvard university press, 1978. Citado nas páginas 54, 89 e 101.

WAZLAWICK, R. S. *Metodologia de pesquisa para ciência da computação*. [S.l.]: Elsevier Brasil, 2014. Citado na página 43.

WILSON, C. *A Tale of Two Clocks - Scheduling Web Audio with Precision*. 2013. HTML5 Rocks. Acesso em: 25 abr. 2025. Disponível em: <https://web.dev/articles/audio-scheduling>. Citado nas páginas 41 (3 ocorrências) e 77.

WOOD, D.; BRUNER, J. S.; ROSS, G. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, Wiley Online Library, v. 17, n. 2, p. 89–100, 1976. Citado na página 49.