USO DE TECNOLOGIAS DART E FLUTTER PARA O ENSINO DO INSTRUMENTO MUSICAL TROMBONE DE VARA

Misael Mendes de Souza[1], Diorgines Mattos Machado[2]

Resumo: O presente trabalho apresenta o desenvolvimento de um aplicativo móvel voltado ao ensino do trombone de vara, integrando elementos de gamificação, análise sonora em tempo real e comunicação por meio da *Musical Instrument Digital Interface*. O objetivo da pesquisa é criar uma ferramenta interativa que utilize *Dart* e *Flutter* para auxiliar trombonistas na prática musical, contemplando funcionalidades como metrônomo, afinador, reprodução de áudio e mecanismos básicos de gamificação voltados ao engajamento do usuário. A metodologia adotada envolveu etapas de pesquisa, desenvolvimento e testes práticos para verificar o funcionamento das principais funcionalidades. Os resultados obtidos indicam que o aplicativo apresenta funcionamento consistente, precisão na detecção de notas e uma interface com usabilidade adequada ao propósito do aplicativo. Conclui-se que a proposta contribui para a democratização do ensino do trombone de vara e favorece experiências de aprendizagem interativas no contexto da educação musical.

Palavras-chave: gamificação; Flutter; Dart; trombone de vara; ensino musical

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), misaelmendesdesouza@gmail.com

[2]Orientador. Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), diorginesmattos@unesc.net

ABSTRACT: This work presents the development of a mobile application for trombone education, integrating gamification elements, real-time audio analysis, and Musical Instrument Digital Interface communication. The research objective is to create an interactive tool using Dart and Flutter to assist trombonists in musical practice, including features such as metronome, tuner, audio playback, and basic gamification mechanisms aimed at user engagement. The adopted methodology involved research, development, and practical testing phases to verify the operation of the main functionalities. The obtained results indicate that the application presents consistent operation, accuracy in note detection, and an interface with adequate usability for the application's purpose. It is concluded that the proposal contributes to the democratization of trombone education and promotes interactive learning experiences in the context of music education.

Keywords: gamification; Flutter; Dart; slide trombone; music education;

1 INTRODUÇÃO

Atualmente existem diversos aplicativos que auxiliam no aprendizado de instrumentos, especialmente voltados para cordas, teclados, percussão e sopro. Contudo, a maioria desses aplicativos exige uma licença paga ou possui uma versão gratuita limitada (Alves et al., 2021). Esses aplicativos costumam incluir recursos como verificação de afinação, metrônomo para controle de ritmo e tempo, além de suporte ao aprendizado de teoria musical e composição. Alguns também aplicam técnicas de gamificação para tornar o ensino mais interativo (Manosso, 2023). Um exemplo é o *Tonestro - Learning Wind Instruments*, que permite ao usuário tocar o instrumento escolhido enquanto o aplicativo verifica a afinação e precisão do som emitido (Hernández, 2020), apesar de ser uma ferramenta eficaz, ela oferece uma versão gratuita restrita e funcionalidades completas apenas na versão paga (Alves et al., 2021).

Os instrumentos de sopro exigem precisão no controle de embocadura, respiração e afinação (Junior et al., 2023), o que torna essencial que as plataformas de aprendizado proporcionem uma experiência adequada por meio de interfaces gráficas interativas e responsivas, que integram a experiência do usuário com o som produzido (Acquilino; Scavone, 2022). A aplicação de elementos de gamificação contribui para manter o músico motivado, promovendo desafios progressivos, conteúdos variados e feedbacks em tempo real que aprimoram a experiência de aprendizado e reduzem o tédio e a ansiedade (Studart, 2021).

A escolha de plataformas móveis para esse tipo de aplicação é vantajosa devido à portabilidade e à ampla acessibilidade dos dispositivos móveis. Com o crescimento contínuo desse mercado e a evolução do desenvolvimento mobile, diversas tecnologias e ferramentas têm surgido, como Swift, Java, Kotlin, Flutter e React Native (Fava, 2023).

O Flutter é uma ferramenta que utiliza a linguagem Dart, é fácil de utilizar, multiplataforma, e possui uma ótima performance e rapidez. Ele possui uma biblioteca de gerenciamento de dados NoSQL chamada Hive (Fontinelli et al., 2023), que é um banco de dados muito eficiente que se baseia em chave-valor, onde os dados e informações são acessadas por meio de uma chave única (Junior, 2023). Diversas bibliotecas desenvolvidas para Flutter, como a biblioteca *flutter_sound*, *flutter_midi_command* e *flutter_metronome*, oferecem recursos de reconhecimento de áudio, afinação, interação com a *Musical Instrument Digital Interface* (MIDI) e metrônomo (PUB.DEV, 2024). O protocolo MIDI é essencial na comunicação entre instrumentos e softwares musicais, possibilitando identificar notas, afinação e tempo de execução (Valente, 2022).

Uma plataforma desenvolvida para a área musical deve compreender suas ferramentas, como o metrônomo, que é responsável por marcar o tempo e indicar o ritmo declarado pelo compositor (Acquilino; Scavone, 2022). O metrônomo teve seu início com Galileu Galilei no século XVI, por meio do descobrimento do pêndulo. O primeiro protótipo do metrônomo foi criado em 1696 por Étienne Loulié, mas não havia alteração do tempo da batida nem som, que surgiram somente em 1816 por Dietrik Nikolaus Winkel. Este, no entanto, não conseguiu patentear a invenção, perdendo a patente para Johan Nepomuk Maelzel com o Metrônomo Maelzel. A ferramenta foi evoluindo e adquiriu extrema importância nas performances musicais e, por meio dos avanços tecnológicos, muitos dispositivos móveis podem instalar um software de metrônomo (Alves et al., 2021).

Os artigos que exploram o uso de tecnologias no ensino de instrumentos musicais foram pesquisados, servindo como base de comparação e inspiração para o desenvolvimento deste projeto, no qual, destacam-se o Patch BoneAux, que propõe uma metodologia gratuita para o aprendizado do trombone de vara, o Metrônomo para surdos, voltado à inclusão de pessoas com deficiência auditiva por meio de estímulos visuais e táteis, e sistemas de conversão de áudio em dados digitais baseados no protocolo MIDI, como os apresentados por Valente (2022). Já nos estudos de Hernández (2020) observa-se o uso da gamificação para tornar a prática

instrumental mais interativa, enquanto Moschen (2025) propõe uma plataforma web que conecta alunos e professores em um ambiente virtual adaptável. Estudos que demonstram a relevância da integração entre música e tecnologia, reforçando a importância de soluções acessíveis e interativas para a aprendizagem instrumental.

Apesar dos avanços observados, existe escassez de soluções educacionais móveis gratuitas para o ensino do trombone de vara que integrem análise sonora em tempo real, comunicação MIDI e gamificação. A maioria das ferramentas disponíveis apresenta limitações de acessibilidade financeira ou não contempla especificidades do instrumento, como controle de embocadura, precisão de afinação e feedback imediato. Há também carência de estudos sobre desenvolvimento de aplicativos educacionais para instrumentos de sopro de metal utilizando tecnologias como Flutter e Dart.

O objetivo geral deste trabalho é desenvolver um aplicativo utilizando Dart e Flutter que incorpore gamificação e integração com o protocolo MIDI, oferecendo análise sonora em tempo real. Para alcançar esse propósito, o estudo contempla o aprofundamento na linguagem Dart e no framework Flutter e suas bibliotecas, e integração com o protocolo MIDI. As etapas de desenvolvimento compreenderam desde a pesquisa sobre técnicas de ensino do instrumento até a implementação de funcionalidades que envolvem afinação, prática musical, reprodução de áudio e comunicação em tempo real entre o som produzido e o sistema.

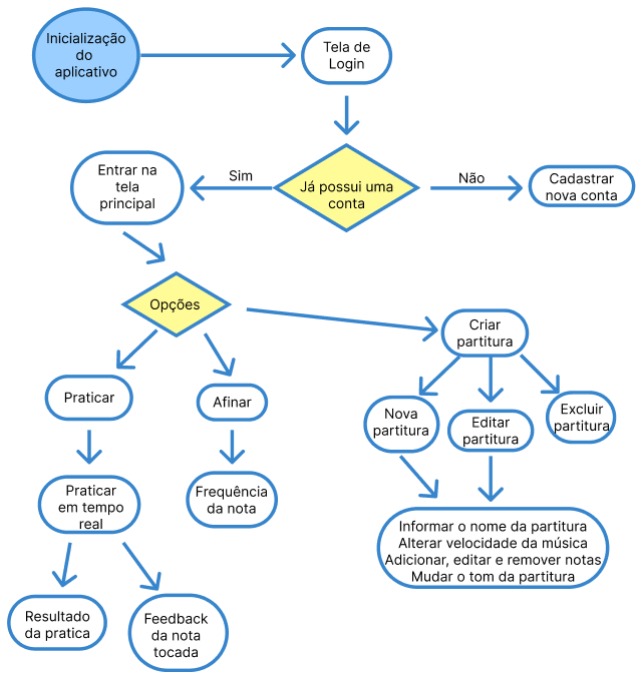
2 MATERIAIS E MÉTODOS

Esta pesquisa caracteriza-se como aplicada, de natureza experimental, com abordagem quali-quantitativa. O objetivo foi desenvolver e validar um aplicativo móvel educacional completo para o ensino do trombone de vara. A metodologia envolveu quatro etapas principais, revisão bibliográfica sobre ensino musical e tecnologias aplicadas, definição de requisitos e modelagem do aplicativo, desenvolvimento da aplicação utilizando Flutter e Dart, e validação por meio de testes práticos comparativos com instrumentos de referência.

Esta pesquisa busca gerar conhecimentos voltados à aplicação prática no ensino do trombone de vara. Investigando soluções tecnológicas para a educação musical, documentando o desenvolvimento e funcionamento do aplicativo e realizando testes comparativos para validação. Combinando análise de funcionalidades e usabilidade com medições de precisão, frequências e desempenho.

A pesquisa resultou no desenvolvimento de um aplicativo móvel independente de conexão com a internet, capaz de interagir com o usuário ajudando o mesmo a evoluir como músico trombonista. Elaborou-se um diagrama de uso para descrever o funcionamento da aplicação, desde o login até a execução de uma prática musical, o diagrama na Figura 1 representa os passos que o usuário poderá fazer em meio a execução do aplicativo.

Figura 1 - Diagrama de Uso



Fonte: Do autor.

No primeiro contato do usuário com o aplicativo aparece a tela de *login*, por meio dela é possível criar uma nova conta ou entrar com uma conta existente. Após cadastrar e entrar na mesma, o usuário será encaminhado para a tela principal do aplicativo, onde poderá praticar exercícios pré-carregados ou afinar seu instrumento. Em praticar partitura, carrega uma partitura padrão, que é a escala de Dó Maior. Na partitura está a armadura de clave, as notas, qual a posição da nota no trombone, o que foi detectado quando o usuário tocou e a precisão da nota. Além disso, aparece o símbolo de um microfone para o início da prática. Nessa mesma tela é possível iniciar, pausar e reiniciar a prática. Também é possível trocar a partitura, sendo uma padrão do aplicativo ou criada pelo usuário. Nas partituras é possível ver sua melhor pontuação ao praticá-la, sendo de 0 a 100. Também é possível iniciar o metrônomo, para ter uma referência de velocidade. No menu também é possível entrar na tela de afinação, onde

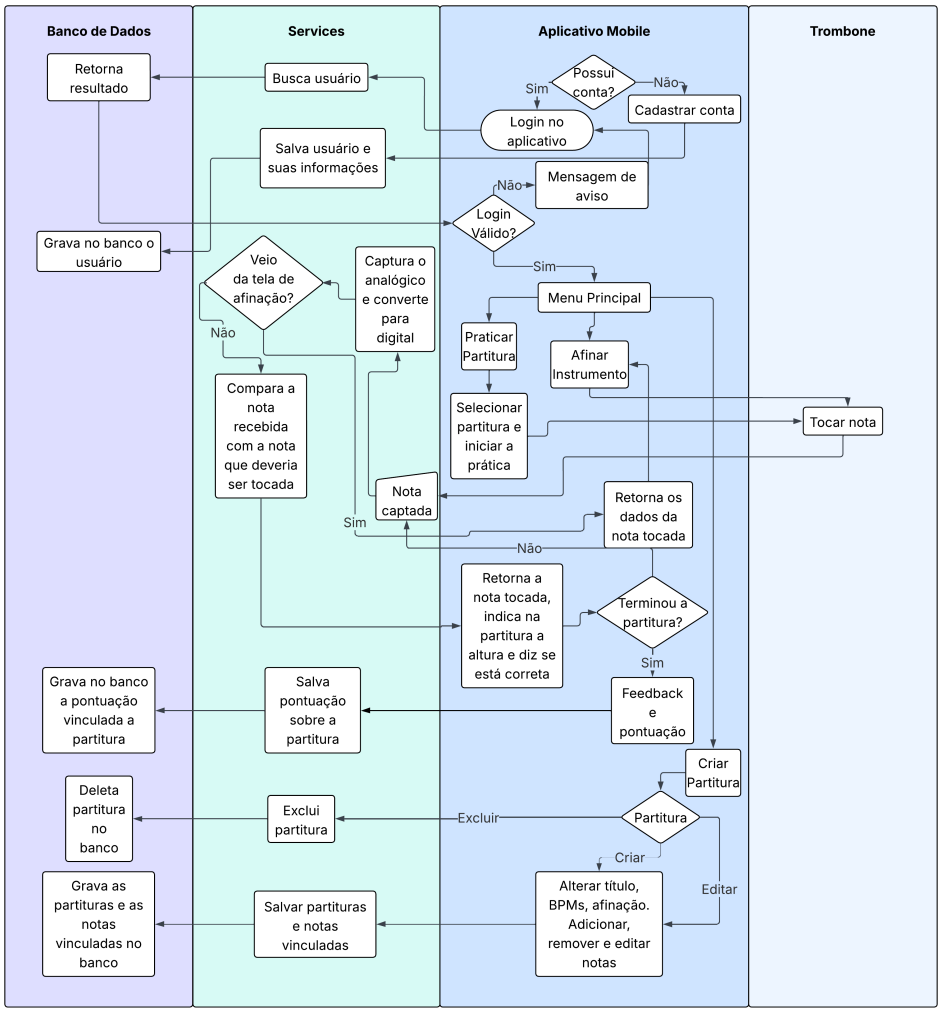
é possível afinar o trombone, ao tocar uma nota irá dizer se ela está acima ou abaixo da frequência padrão. A tela de criação de partitura, também encontrado no menu, permite criar, editar e excluir partituras. Ao criar e editar partituras, é possível adicionar, editar e excluir notas, alterar os BPMs (batidas por minuto), o compasso, o tom e o título da partitura.

2.1 DESENVOLVIMENTO DO APLICATIVO

O aplicativo foi desenvolvido garantindo que cada componente do aplicativo tivesse uma função bem definida. A Figura 2 apresenta o fluxo geral de funcionamento, desde o momento em que o usuário interage com o aplicativo até o processamento das informações e o retorno dos resultados.

O diagrama está dividido em quatro partes: **Banco de Dados**, **Serviços (Services)**, **Aplicativo Mobile** e **Trombone**. Cada uma dessas partes tem um papel específico dentro do funcionamento do aplicativo.

Figura 2 - Fluxograma



Fonte: Do autor.

O banco de dados armazena as informações do usuário, as partituras criadas e editadas, e os resultados das práticas, os serviços conectam o aplicativo ao banco de dados, realiza operações de salvamento, busca e análise de dados, e processa em tempo real o som captado pelo microfone do dispositivo. O aplicativo representa a camada de interface visível, onde o usuário loga, cria e gerencia partituras, pratica exercícios com *feedback* imediato e afina o instrumento, o trombone representa o instrumento físico, seu som é captado pelo microfone e processado pelo aplicativo para análise de frequência e afinação.

O funcionamento do aplicativo inicia no login do usuário e segue até o momento da prática musical. Durante o estudo, o som produzido pelo trombone é captado continuamente e analisado em tempo real, sendo comparado à nota exibida na partitura. Quando o usuário toca a nota correta e afinada e finaliza a partitura, o aplicativo registra a pontuação e salva o progresso no banco de dados. Além disso, o aplicativo permite criar, editar e excluir partituras personalizadas.

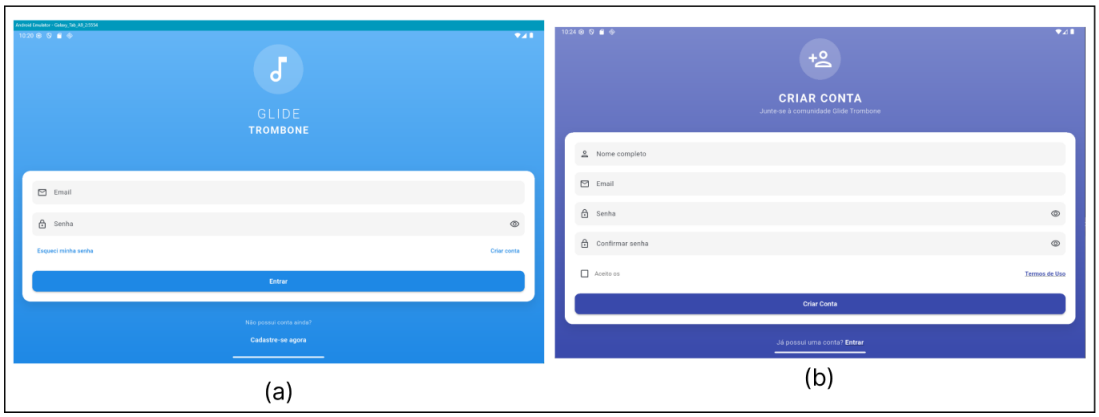
2.1.1 Ambiente do Aplicativo

O aplicativo é composto por diversas telas, entre elas, a tela inicial, a de prática, afinador e de criação de partitura. Algumas delas possuem modais subtelas, abaixo elas estão sendo explicadas juntamente com suas figuras.

a) **Tela de login**

Nesta tela, o usuário pode realizar o *login* e cadastrar uma nova conta, conforme a Figura 3, onde (a) é a tela de login, e (b) a tela de cadastro de usuário.

Figura 3 - Tela de login e cadastro



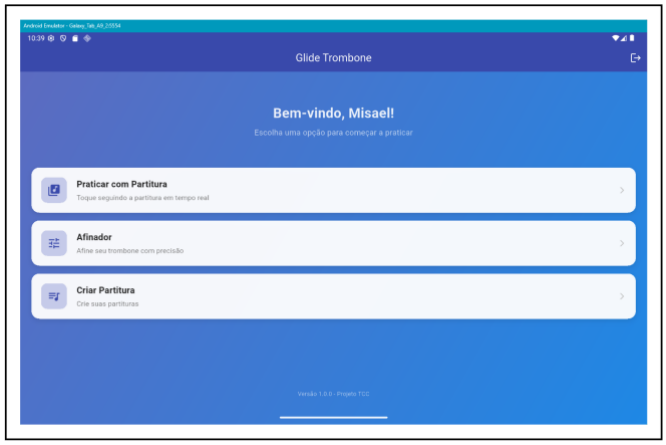
(a) (b)

Fonte: Do autor.

b) **Menu principal**

Após o login, o usuário é direcionado ao menu principal, onde pode escolher entre as opções: *Praticar*, *Afinador* e *Partituras*, conforme a Figura 4.

Figura 4 - Menu principal

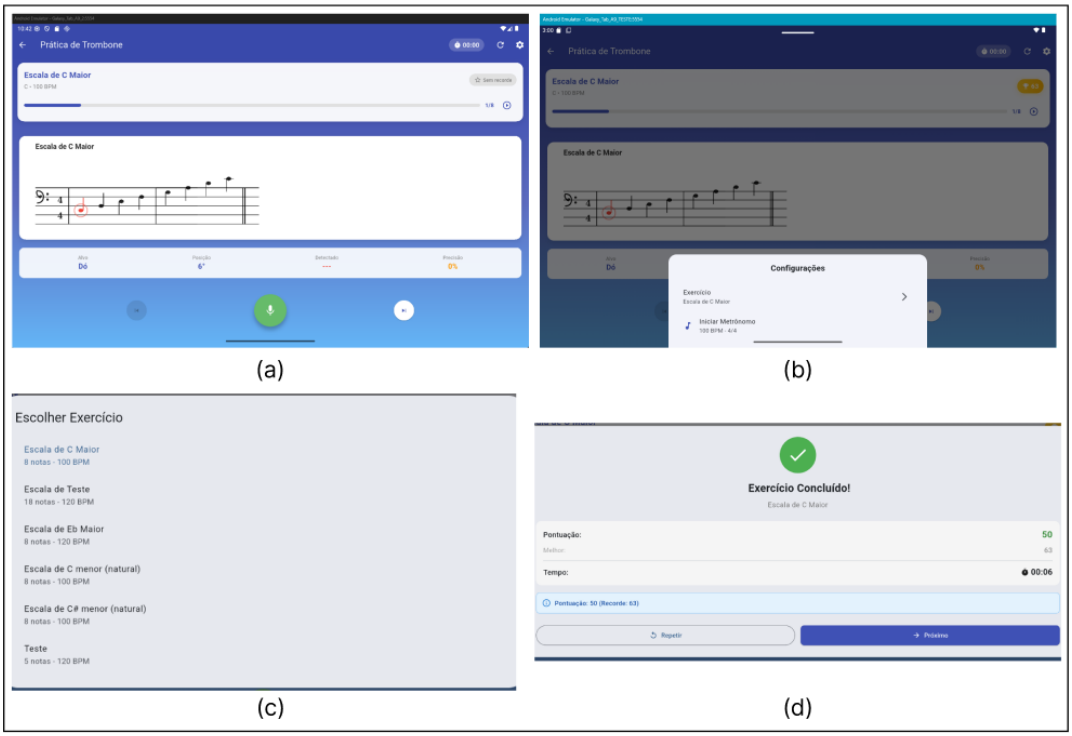


Fonte: Do autor.

c) **Tela de praticar**

A prática está visualizada na Figura 5, por ela é possível selecionar uma partitura e pratica-la.

Figura 5 - Prática de Trombone



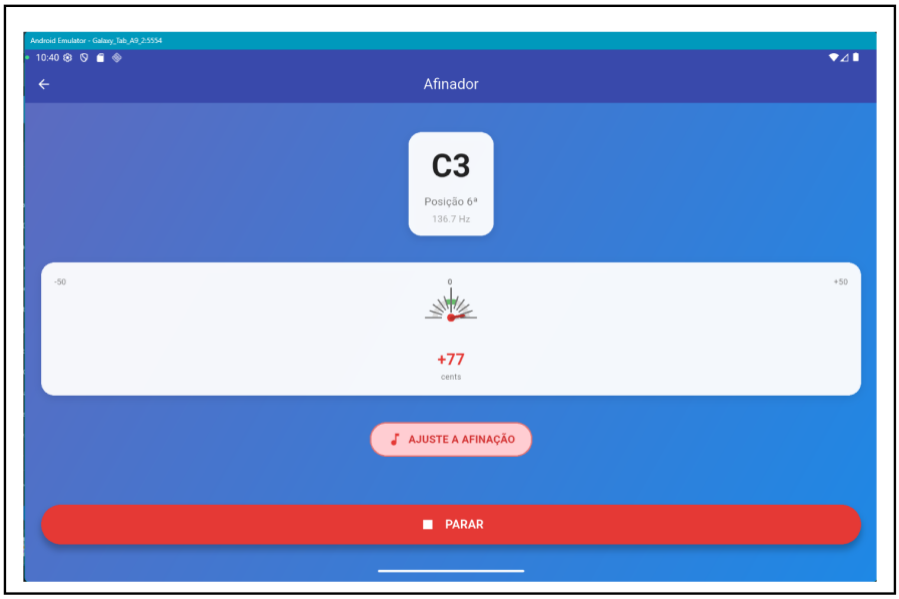
Fonte: Do autor.

Na imagem (a) está a tela de prática, onde se inicia clicando no ícone do microfone. Em (b) estão as configurações, onde é possível iniciar o metrônomo ou ir para a (c) onde ocorre a troca de partitura. Ao finalizar a prática da partitura selecionada, aparece a tela de feedback (d), indicando a pontuação obtida com base na precisão e afinação das notas executadas, bem como o tempo que levou.

d) **Afinador**

A Figura 6 mostra a tela de afinação, que permite ao usuário verificar se o instrumento está ajustado. O aplicativo capta o som do trombone, analisa a frequência e indica visualmente se a nota está abaixo, acima ou afinada.

Figura 6 - Afinador

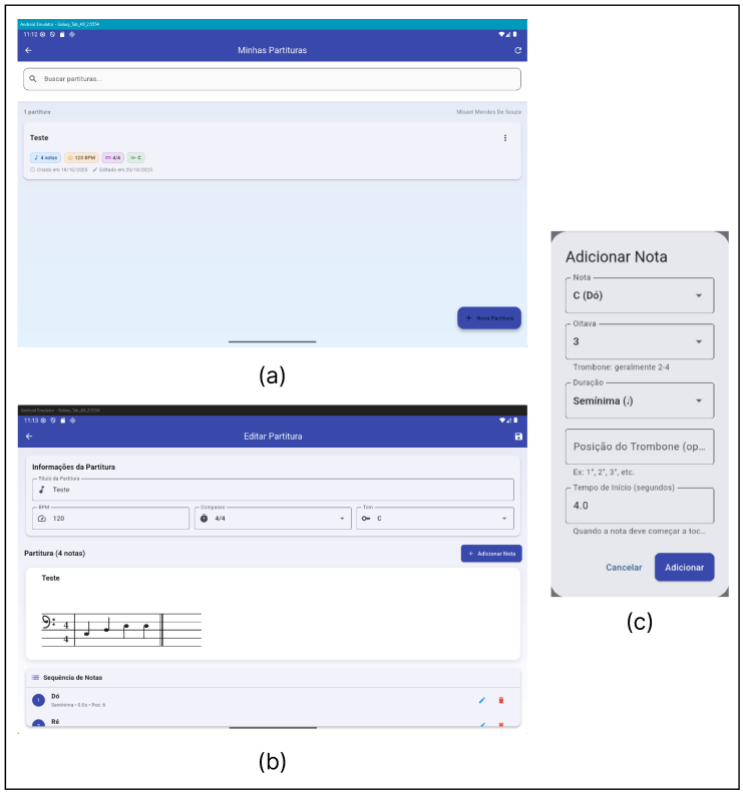


Fonte: Do autor.

e) **Tela de partituras**

O usuário tem acesso às partituras previamente cadastradas no aplicativo, podendo visualizar, criar, editar e excluir partituras. Na Figura 7 (a) é possível visualizar e filtrar as partituras criadas pelo usuário, clicando em uma partitura específica ou em "Nova Partitura"o usuário entra na (b), onde pode editar as informações gerais da partitura, incluindo título, andamento em BPM, compasso e tonalidade, e adicionar e editar notas como na imagem (c), ou excluir notas.

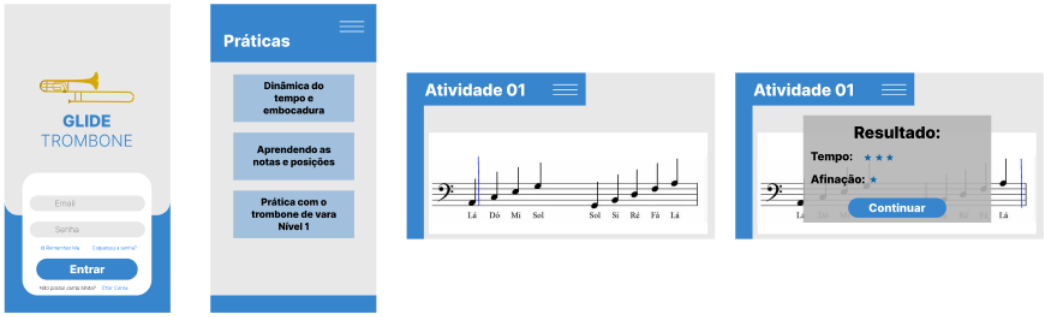
Figura 7 - Partituras



Fonte: Do autor.

O layout apresentado na Figura 8 representa uma visão inicial do funcionamento do aplicativo, funcionando como uma simulação do fluxo de navegação e das principais funcionalidades.

Figura 8 - Layout Figma



Fonte: Do autor.

2.1.2 Ambiente de Desenvolvimento

A aplicação foi desenvolvida com o framework Flutter (versão 3.27.1) e linguagem Dart (versão 3.6.0), utilizando a IDE IntelliJ IDEA. Foram integradas bibliotecas específicas para manipulação de áudio e proto-

colo MIDI, possibilitando a captação sonora, identificação das notas musicais e comparação em tempo real com a partitura exibida no aplicativo.

As principais dependências utilizadas no projeto incluem *flutter_sound* (versão 9.2.13), responsável pela captura e processamento de áudio em tempo real, *audioplayers* (versão 5.0.0), utilizada para reprodução de arquivos de áudio, *hive* (versão 2.2.3) e *hive_flutter* (versão 1.1.0), para o banco de dados NoSQL e sua integração com Flutter, *permission_handler* (versão 11.0.0), para gerenciamento de permissões do sistema.

O gerenciamento de dados foi realizado com o banco NoSQL Hive, que garantiu a execução do aplicativo em modo offline. O aplicativo foi configurado para Android com SDK mínimo 21 (Android 5.0 Lollipop). Os testes foram conduzidos no simulador do Android Studio e em um dispositivo Samsung Galaxy Tab A9, utilizando um trombone Yamaha YSL-354S para validação das funcionalidades. O ambiente de desenvolvimento contou com um computador com Windows 11, 32 GB de RAM, processador Intel Core i5-10400F e placa de vídeo NVIDIA GeForce RTX 3060.

2.1.3 Banco de Dados

O armazenamento de dados foi implementado com o banco Hive, uma solução NoSQL para Flutter, que oferece alto desempenho, baixo consumo de recursos e dispensa um servidor externo, sendo ideal para sistemas móveis que funcionam de forma offline, como neste projeto.

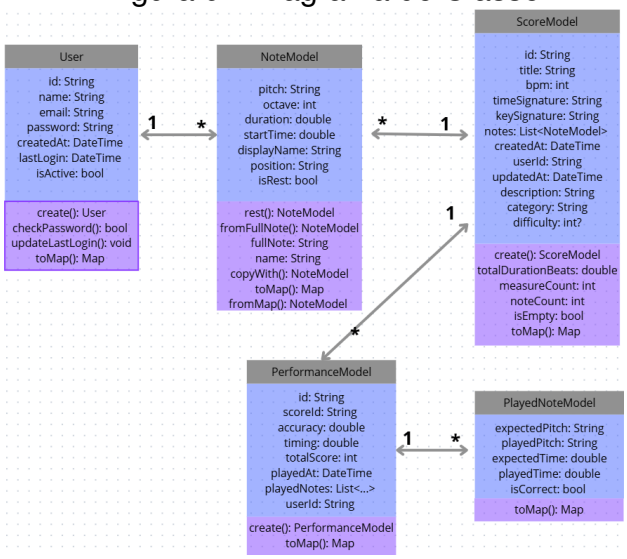
Enquanto soluções como Firebase Firestore e MongoDB Realm requerem conexão com servidores externos e sincronização em nuvem, o Hive opera completamente offline, eliminando dependências de rede e garantindo funcionamento integral sem internet. Em relação ao SQLite, embora também seja uma solução local, o Hive apresenta desempenho superior para operações de leitura e escrita devido ao seu armazenamento em formato binário otimizado, além de dispensar a estruturação rígida de esquemas SQL. Ele permite armazenar objetos complexos serializados, com simplicidade, eficiência e de forma offline (Santos, 2019).

São armazenados dados essenciais, as informações do usuário, notas musicais representadas em objetos *NoteModel*, e partituras e desempenhos nos modelos *ScoreModel* e *PerformanceModel*. Cada modelo implementa um adaptador de tipo (*TypeAdapter*) com identificador único (*typeId*), que permite ao Hive serializar e desserializar objetos em formato binário, garantindo operações rápidas de leitura e escrita.

A estrutura de dados implementada é ilustrada no diagrama de

classes da Figura 9, que apresenta os relacionamentos entre os modelos de dados persistidos no Hive.

Figura 9 - Diagrama de Classe



Fonte: Do autor.

A inicialização do Hive ocorre no serviço *DatabaseService*, responsável por registrar os adaptadores e abrir as *Boxes* necessárias. A manipulação de partituras e notas é realizada pelo *ScoreDatabaseService*.

2.2 PRINCIPAIS SERVIÇOS

O *AudioService* é responsável pela captação, análise e reprodução sonora. Ele calcula as frequências captadas pelo microfone, identifica em tempo real a nota tocada e integra bibliotecas como *flutter_sound* e *audioplayers*, além de gerenciar as permissões e o fluxo de áudio. Na Figura 18, encontrada no Apêndice A, estão as principais funções que compõem o serviço de áudio, essenciais para o processamento e análise dos sons.

A função *processAudioChunk*, Apêndice A, Figura 18 (a), prepara o áudio capturado, convertendo os dados em amostras numéricas para análise. A *_playMetronomeAsync*, Apêndice A, Figura 18 (b), gera o som do metrônomo digital, com batidas acentuadas que simulam o modelo tradicional. A *detectPitch*, Apêndice A, Figura 18 (c), identifica a nota tocada pelo usuário, analisando a frequência captada e comparando-a com a nota exibida na partitura. A função *_playSynthesizedNote*, Apêndice A, Figura 18 (d), realiza a síntese sonora diretamente no código. Ela calcula a frequência da nota, gera uma onda senoidal com envelope dinâmico e reproduz o som pela biblioteca *flutter_sound*, garantindo baixa latência na execução.

Outro componente essencial é o *DatabaseService*, responsável por gerenciar a comunicação entre os modelos de dados e o restante do aplicativo, ele executa operações como cadastro, autenticação e gerenciamento das informações dos usuários. O *ScoreDatabaseService* é responsável pelo controle das partituras e desempenhos musicais.

A interação entre esses serviços possibilita o funcionamento offline do aplicativo e a integração contínua entre os módulos de áudio, partitura e desempenho do usuário.

2.2.1 Biblioteca de Sons

O projeto faz uso das bibliotecas *flutter_sound* e *audioplayers* para funcionalidades complementares, como gravação, reprodução de áudio, e comparação do som produzido pelo usuário com o som de referência.

2.2.2 Captura de Áudio e Conversão

A captura de áudio é realizada pela biblioteca *flutter_sound*, que acessa o microfone e converte o sinal analógico em dados digitais, com taxa de amostragem de 22.050 Hz e codificação PCM de 16 bits em formato mono. O sinal é recebido como um fluxo contínuo de bytes, no qual cada amostra sonora é representada por dois bytes (Mu et al., 2021).

Os dados são armazenados em um buffer de 1024 amostras, tamanho utilizado em análises de frequência com transformada rápida de Fourier (FFT), por equilibrar resolução espectral e desempenho computacional (Mu et al., 2021; Audio, 2025). A FFT é um algoritmo otimizado para calcular a Transformada Discreta de Fourier (DFT), que decompõe um sinal temporal em suas componentes de frequência (Audio, 2025). No contexto musical, a FFT permite identificar quais frequências estão presentes no sinal de áudio capturado, transformando a representação do domínio do tempo para o domínio da frequência. Essa transformação é essencial para a detecção de notas musicais, pois cada nota corresponde a uma frequência fundamental específica (Goto, 2009; Pontes; Madruga, 2019).

O buffer de 1024 amostras determina a resolução em frequência da análise, com taxa de amostragem de 22.050 Hz, obtém-se resolução de aproximadamente 21,5 Hz por bin (22050/1024), suficiente para distinguir notas musicais adjacentes (Audio, 2025). Em seguida, cada par de bytes é convertido em um valor inteiro de 16 bits e normalizado pela divisão por 32768, ajustando as amplitudes para o intervalo [-1, 1] antes da aplicação dos algoritmos de detecção e identificação das notas (Akiyama et al., 2019).

2.2.3 Reprodução de Notas

Para a reprodução de notas é implementado um sintetizador baseado em um oscilador senoidal (Pontes; Madruga, 2019), seguindo o modelo matemático $y(t) = A \cdot \sin(2\pi f \mathrm{t}) \cdot E(\mathrm{t})$

Onde $y(t)$ representa a amplitude da onda no tempo t, A é a amplitude máxima controlada pelo envelope, f é a frequência da nota em Hertz, e $E(t)$ corresponde à função envelope ADSR. A função seno permite gerar um tom puro que serve como referência para comparação com o som produzido pelo instrumentista (Goto, 2009).

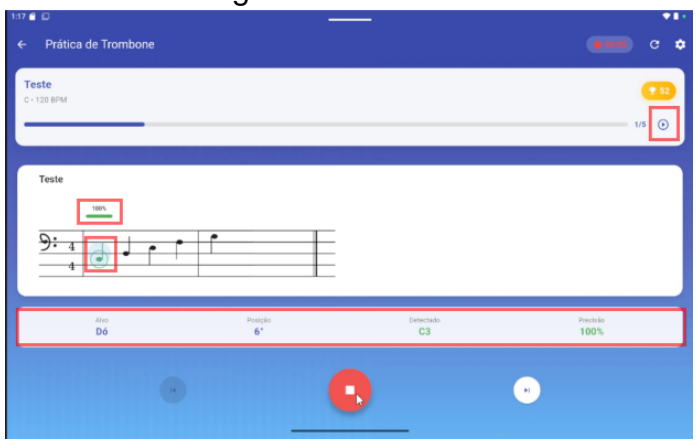
A frequência de cada nota é calculada usando a fórmula de temperamento (Ferreira et al., 2021), adotado como padrão na música ocidental e no protocolo MIDI (Harasim; al., 2022): $f = 440 \cdot 2^{\frac{(n-69)}{12}}$

Onde n representa o número MIDI da nota, sendo 69 correspondente à nota Lá4 (A4 = 440 Hz), utilizada como referência internacional de afinação. Este sistema permite a conversão precisa entre representação simbólica (nota musical) e representação física (frequência em Hertz) (Valente, 2022). Por exemplo, para a nota Dó3 (C3), que possui número MIDI 48: $f_{C3} = 440 \cdot 2^{\frac{(48-69)}{12}} = 440 \cdot 2^{-1.75} \approx 130{,}81 Hz$.

2.2.4 *Feedback*

O aplicativo fornece *feedback* visual por meio de indicadores gráficos que mostram correspondência e afinação das notas tocadas. O *feedback* sonoro permite reproduzir notas de referência e utilizar o metrônomo para compreender os BPMs. Esses estímulos visuais e auditivos tornam a aprendizagem mais imersiva, reforçando acertos e erros. A Figura 10 apresenta os indicadores visuais e sonoros.

Figura 10 - Feedback



Fonte: Do autor.

2.3 DISPOSITIVOS COMPATÍVEIS

O aplicativo foi desenvolvido especificamente para dispositivos com o sistema operacional Android, entre a versão 7.0 e 15.0. Pode ser utilizado por smartphones e tablets porém possui maior foco em tablets, por conta do tamanho e conforto na hora de ler e tocar uma partitura. O aplicativo requer a permissão *RECORD_AUDIO* que é necessária para capturar o som do instrumento.

2.4 MÉTODO DE AVALIAÇÃO DO SISTEMA

O processo de avaliação do aplicativo foi realizado por meio de testes práticos conduzidos pelo autor, com o objetivo de verificar o funcionamento do afinador, a detecção de notas durante a prática musical e a usabilidade geral da aplicação. A validação baseou-se em uma abordagem comparativa, utilizando um trombone Yamaha YSL-354S, e comparando os resultados com dois afinadores, o afinador físico Tonante AF10 e o aplicativo *Soundcorset Tuner* (Soundcorset, 2025), permitindo observar a consistência entre as leituras do aplicativo e os dispositivos já consolidados no uso musical. Foram analisadas a detecção correta de nota e o desvio de frequência de cents, sendo valores entre -10 e +10 cents considerados aceitáveis (Alemi; Lehmann; Deroche, 2020).

O procedimento de teste para afinação consistiu em executar a nota fundamental do trombone, $Si\flat$ ($B\flat2$), na primeira posição, registrando a frequência detectada por cada dispositivo e o desvio em cents em relação à frequência padrão. Para o teste de prática musical, foi criada uma partitura no próprio aplicativo com título Teste, andamento 120 BPM, compasso 4/4 e tonalidade Dó Maior, contendo cinco notas, Dó (C3), Ré (D3), Mi (E3), Fá (F3) e Sol (G3). Cada nota foi executada no trombone e analisada pelos três dispositivos, registrando a nota detectada e o desvio de afinação. Os critérios de aceitação estabelecidos foram desvio máximo de ±10 cents em relação aos afinadores de referência.

3 RESULTADOS E DISCUSSÃO

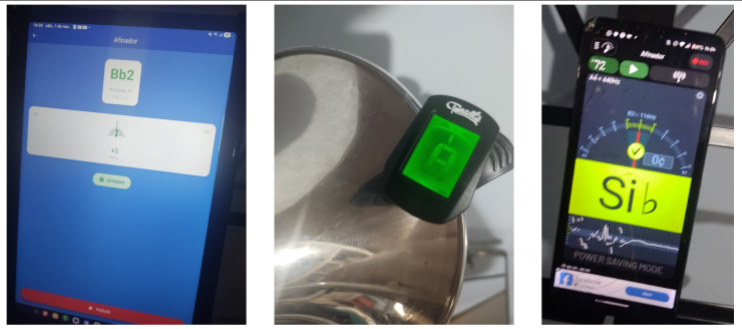
A avaliação do aplicativo foi realizada por meio de uma simulação prática conduzida pelo autor, atuando como usuário e trombonista. O objetivo foi analisar a precisão do afinador e a usabilidade do aplicativo, seguindo o fluxo definido no diagrama de casos de uso, desde o cadastro e login até as etapas de afinação, criação de partitura e prática do trombone.

A simulação teve início com a criação do cadastro no aplicativo,

etapa que possibilita o registro de usuários e o acompanhamento individual do progresso. Em seguida, o aplicativo direcionou para a tela inicial, onde estavam disponíveis as funcionalidades principais: prática, afinação e criação de partituras.

Na tela de afinação, foi realizado o teste de detecção de frequência utilizando um trombone de vara Yamaha YSL-354S. Ao executar a nota fundamental do instrumento, $Si\flat$ ($B\flat 2$) na primeira posição, comparou-se a frequência detectada pelo afinador do aplicativo com as leituras dos afinadores de referência: o afinador físico Tonante AF10 e o aplicativo *Soundcorset Tuner*. A Figura 11 mostra os resultados dos testes de afinação

Figura 11 - Afinadores



(a) (b) (c)

Fonte: Do autor.

O aplicativo desenvolvido indicou $Si\flat 2$ com variação de +5 cents (unidade que mede pequenas diferenças de afinação), conforme mostrado na Figura 11 (a), enquanto o Tonante AF10 indicou afinação centralizada, sem variação perceptível, conforme Figura 11 (b), e o Soundcorset Tuner apresentou a leitura da nota $Si\flat 2$ centralizada, visualizada na Figura 11 (c).

Durante a execução, observou-se que as frequências tendem a oscilar levemente devido às variações naturais de embocadura e pressão de ar características do trombone de vara. No entanto, o aplicativo manteve as medições dentro de uma margem estável, demonstrando comportamento compatível com os afinadores de referência. Os resultados indicaram que o afinador apresentou medições consistentes e precisas, com baixa variação em relação a outros dispositivos, validando o bom funcionamento do módulo de afinação.

A função de criação de partitura também foi testada, sendo criada uma partitura intitulada *Teste*, configurada com andamento de 120 BPM, compasso 4/4, afinação em Dó (C) e contendo as notas Dó (C3),

Ré (D3), Mi (E3), Fá (F3) e Sol (G3). A qual não ocorreu erros durante sua criação. A etapa seguinte consistiu em praticar as notas criadas na partitura, utilizando o modo de prática do aplicativo. O aplicativo exibiu as notas na tela e analisou, em tempo real, as frequências captadas pelo microfone durante a execução no trombone.

A tabela 1 mostra a comparação entre as notas tocadas no trombone, ao comparar o afinador físico Tonante AF10 e o aplicativo *Soundcorset Tuner* com a tela de prática do aplicativo desenvolvido:

Tabela 1 - Comparação das notas tocadas

Nota	Tonante AF10	Soundcorset
C3	Central	Central
D3	Central	-1 cent
E3	Central	Central
F3	Central	+2 cents
G3	Central	+8 cents

Fonte: Do autor.

A Figura 12 apresenta a comparação da nota Dó detectada pelo aplicativo desenvolvido (a), o afinador físico Tonante AF10 (b) e o aplicativo *Soundcorset Tuner* (c), indicando o som centralizado em todos.

Figura 12 - Nota Dó

Fonte: Do autor.

Na Figura 13 está a análise da nota Ré, todos afinadores identificaram corretamente a nota Ré, mantendo a afinação dentro do padrão esperado, tendo apenas uma variação no *Soundcorset Tuner* (c) onde está 1 cent à esquerda.

Figura 13 - Nota Ré

Fonte: Do autor.

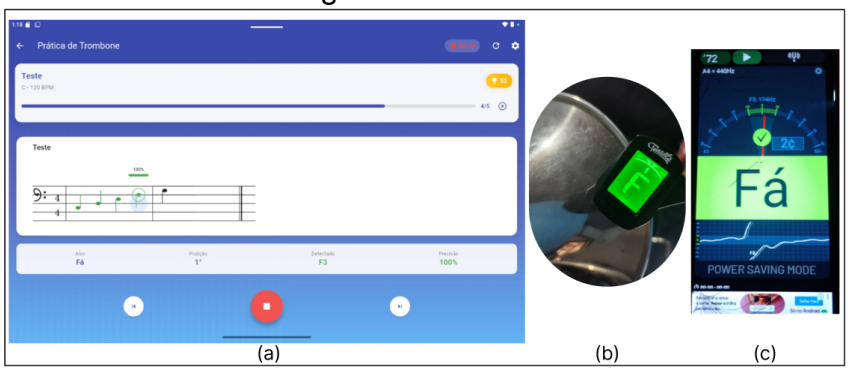
Os afinadores e o aplicativo identificaram corretamente a nota Mi, mantendo ela centralizada, conforme Figura 14.

Figura 14 - Nota Mi

Fonte: Do autor.

A Figura 15 apresenta a análise da nota Fá, observa-se que todos identificaram corretamente a nota, com o afinador (c) apresentando leve variação de 2 cents à direita. Essa diferença é considerada normal e dentro do limite aceitável de precisão.

Figura 15 - Nota Fá



Fonte: Do autor.

O resultado da nota Sol é exibido na Figura 16, onde as notas foram identificadas corretamente, porém o afinador (c) indicou uma leve variação de 8 cents à direita, uma diferença considerada pequena e aceitável dentro do padrão de precisão dos afinadores.

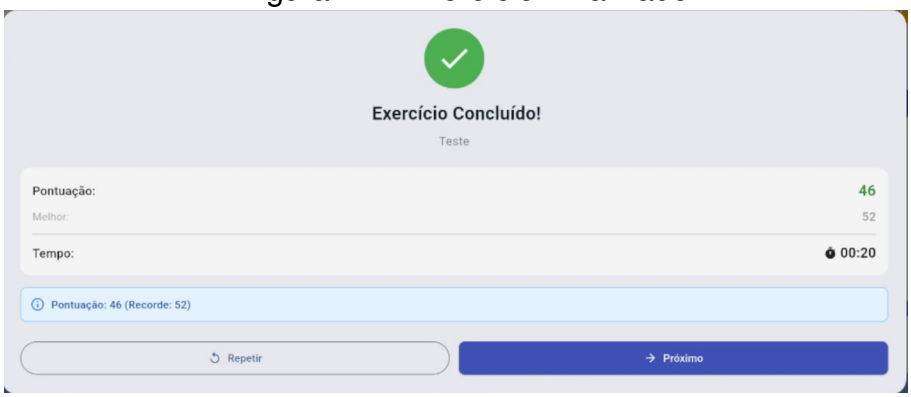
Figura 16 - Nota Sol

(a) (b) (c)

Fonte: Do autor.

Ao finalizar a partitura aparece a Figura 17, indicando que o exercício foi finalizado.

Figura 17 - Exercício Finalizado



Fonte: Do autor.

A simulação demonstrou que o aplicativo identifica as notas com precisão, contudo observou-se uma latência no processo de análise das frequências captadas. Embora a detecção em tempo real e o *feedback* visual ocorram de forma funcional, a resposta do aplicativo apresenta um pequeno atraso. Essa diferença impacta a sincronização com o andamento (BPM), evidenciando a necessidade de otimizar do processamento de áudio para reduzir o tempo de resposta e aprimorar a fluidez da prática musical.

4 CONCLUSÃO

O desenvolvimento do aplicativo proposto resultou em uma aplicação móvel funcional para o ensino do trombone de vara, integrando tecnologia móvel, análise sonora em tempo real e protocolo MIDI. A combinação de Dart e Flutter, aliada a bibliotecas especializadas de áudio e ao banco NoSQL Hive, permitiu a implementação de funcionalidades essenciais ao aprendizado musical com operação offline, criação e edição de partituras personalizadas, modo de prática com detecção em tempo real, afinador, metrônomo digital e feedback visual e sonoro.

Os testes de validação técnica demo;nstraram precisão na detecção das notas musicais, com resultados próximos aos afinadores Tonante AF10 e Soundcorset Tuner. O teste de afinação da nota fundamental ($Sib2$) apresentou desvio de 5 cents. O teste de prática com cinco notas (C3, D3, E3, F3, G3) resultou em identificação correta de todas as notas, com variações até 8 cents, dentro do limite aceitável de 10 cents.

Apesar dos bons resultados, foi identificada latência no processamento do áudio, indicando necessidade de otimização dos algoritmos para garantir sincronização precisa. A validação contemplou apenas testes técnico-funcionais conduzidos pelo autor. Trabalhos futuros devem incluir estudos com estudantes de trombone, professores e múltiplos usuários, avaliando aspectos pedagógicos por meio de métricas estatísticas e metodologias quali-quantitativas, além de implementar melhorias como filtros de suavização, análise de intensidade sonora, suporte a pausas e dinâmiFcas, e síntese MIDI com maior fidelidade.

Conclui-se que o aplicativo alcançou seus objetivos técnicos, confirmando a viabilidade de aplicações móveis offline para suporte ao ensino de instrumentos musicais, com potencial de expansão para outros instrumentos de sopro. O projeto contribui para o avanço das tecnologias educacionais musicais e reafirma o papel das plataformas digitais como instrumentos de inovação no ensino prático de música.

REFERÊNCIAS

ACQUILINO, A.; SCAVONE, G. **Current state and future directions of technologies for music instrument pedagogy**. [S.l.]: Frontiers Media SA, 2022. 835609 p. Acesso em: 14 out. 2025. Disponível em: <https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2022.835609/full>.

AKIYAMA, O. et al. Dcase2019 task: preprocessing details. In: **DCASE**

Workshop Proceedings. [S.l.: s.n.], 2019. Exemplo prático: divisão por 32768 para normalizar amostras 16-bit. Acesso em: 05 nov. 2025. Disponível em: <https://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Akiyama_63.pdf>.

ALEMI, R.; LEHMANN, A.; DEROCHE, M. L. D. Adaptation to pitch-altered feedback is independent of one's own voice pitch sensitivity. **Scientific Reports**, Nature Publishing Group, v. 10, p. 16860. Acesso em: 02 dez. 2025. Disponível em: <https://www.nature.com/articles/s41598-020-73932-1>.

ALVES, B. N. et al. **Patch BoneAux: uma ferramenta gratuita desenvolvida para auxiliar o aprendizado e aprimoramento do trombone de vara**. [S.l.]: Universidade Federal da Paraíba, 2021. Acesso em: 20 out. 2025. Disponível em: <https://repositorio.ufpb.br/jspui/handle/123456789/25694>.

AUDIO, N. **Transformação rápida de Fourier FFT – Noções básicas**. 2025. Artigo online. Acesso em: 05 nov. 2025. Disponível em: <https://www.nti-audio.com/pt/suporte/saber-como/transformacao-rapida-de-fourier-fft>.

FAVA, F. B. **Ferramenta para visualização do consumo de recursos de hardware para aplicações flutter**. [S.l.]: Universidade Federal do Pampa, 2023. Acesso em: 20 out. 2025. Disponível em: <https://repositorio.unipampa.edu.br/server/api/core/bitstreams/da587ca2-dffd-45c7-9f38-189bb3fe5f7d/content>.

FERREIRA, R. R. R. d. S. et al. **Som em blocos: método para construção de conhecimento musical a partir de programação em MIDI**. [S.l.]: Universidade Federal do Pará, 2021. Acesso em: 15 out. 2025. Disponível em: <https://repositorio.ufpa.br/items/f786ed87-1d75-40d8-957d-13adb73495d3>.

FONTINELLI, F. N. d. S. et al. **DOEBRASIL. ORG: Plataforma para intermediação de doações**. [S.l.]: Instituto Federal Goiano, 2023. Acesso em: 04 nov. 2025. Disponível em: <https://repositorio.ifgoiano.edu.br/handle/prefix/4148>.

GOTO, M. **Física e música em consonância**. [S.l.]: Revista Brasileira de Ensino de Física, 2009. 2303 p. Acesso em: 04 nov. 2025. Disponível em: <https://www.scielo.br/j/rbef/a/s9TwczdtxDR8QPJCndqJDmb/>.

HARASIM, D.; AL. et. **midiVERTO: A Web Application to Visualize Tonality in Real Time**. [S.l.]: Cornell University, 2022. Acesso em: 05 nov. 2025. Disponível em: <https://arxiv.org/abs/2203.13158>.

HERNÁNDEZ, A. P. **Gamification-based Tool to Practice Basic Exercises with Wind Instruments**. [S.l.]: Universidad de Castilla-La Mancha, 2020. Acesso em: 05 nov. 2025. Disponível em: <https://www.esi.uclm.es/www/David.Vallejo/TFE/TFG_Antonio_Pulido.pdf>.

JUNIOR, F. J. C. **Proposta inicial de um aplicativo móvel para recomendação de leitoras RFID**. [S.l.]: Universidade Federal do Ceará, 2023. Acesso em: 05 nov. 2025. Disponível em: <https://repositorio.ufc.br/handle/riufc/76269>.

JUNIOR, L. O. R. P. et al. **O multi-instrumentista de metais: aspectos técnicos e similaridades na prática de diferentes instrumentos**. [S.l.]: Universidade Federal de Uberlândia, 2023. Acesso em: 05 nov. 2025. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/38491/1/MultiInstrumentistaMetais.pdf>.

MANOSSO, M. S. **Educação Musical: Discussão acerca da Elitização da Música e Proposta de Aplicativo Mobile para a Educação Musical**. [S.l.]: Instituto Federal do Paraná, 2023. Acesso em: 05 nov. 2025. Disponível em: <https://ifpr.edu.br/londrina/wp-content/uploads/sites/18/2023/05/MARIANNA-SOARES-MANOSSO-Proposta-de-Aplicativo-Mobile-para-Educacao-Mu pdf>.

MOSCHEN, A. B.S. thesis, **Sistema web para o aprendizado de instrumentos musicais**. 2025. Acesso em: 07 nov. 2025. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/36567/1/sistemaaprendizadoinstrumentosmusicais.pdf>.

MU, W. et al. Environmental sound classification using temporal-frequency attention based convolutional neural network. **Scientific Reports**. Resampled to 22050 Hz; window size 1024, hop 512. Acesso em: 05 nov. 2025. Disponível em: <https://www.nature.com/articles/s41598-021-01045-4.>

PONTES, R.; MADRUGA, J. Música e modelagem matemática: representações de notas musicais por meio da função seno. **Revista Tangram**. Acesso em: 05 nov. 2025. Disponível em: <https://ojs.ufgd.edu.br/index.php/tangram/article/view/10215>.

PUB.DEV. **Pub.dev: the package repository for Dart.** 2024. Acesso em: 14 out. 2024. Disponível em: <https://pub.dev/>.

SANTOS, V. **Flutter — O que é e como utilizar o banco de dados Hive**. 2019. Acesso em: 07 maio 2025. Disponível em: <https://medium.com/flutter-comunidade-br/flutter-o-que-%C3%A9-e-como-utilizar-o-banco-de-dados-hive-2839f9ada04f>.

SOUNDCORSET. **Soundcorset Tuner & Metronome**. 2025. Acesso em: 1 nov. 2025. Aplicativo disponível no Google Play, <https://play.google.com/store/apps/details?id=com.soundcorset.client.android>.

STUDART, N. **A gamificação como design instrucional**. [S.l.]: SciELO Brasil, 2021. e20210362 p. Acesso em: 05 nov. 2025. Disponível em: <https://www.scielo.br/j/rbef/a/TFcKMNMYWRRhBGNxNmHRn3v/?format=html&lang=pt>.

VALENTE, L. P. **Sistema de análise de características de sinais de áudio para conversão em protocolo MIDI**. [S.l.]: Universidade Federal do Ceará, 2022. Acesso em: 05 nov. 2025. Disponível em: <https://repositorio.ufc.br/handle/riufc/66633>.

APÊNDICE A – CÓDIGO-FONTE DAS PRINCIPAIS FUNÇÕES

```
List<double>? processAudioChunk(Uint8List audioData) {
  _audioBuffer.addAll(audioData);

  int requiredBytes = REQUIRED_SAMPLES * 2;
  if (_audioBuffer.length < requiredBytes) {
    return null;
  }

  List<double> samples = [];
  int bytesToProcess = math.min(_audioBuffer.length,
requiredBytes);

  for (int i = 0; i < bytesToProcess - 1; i += 2) {
    int sample16bit = (_audioBuffer[i + 1] << 8) |
_audioBuffer[i];

    if (sample16bit > 32767) {
      sample16bit -= 65536;
    }

    double normalizedSample = sample16bit / 32768.0;
    samples.add(normalizedSample);

    if (samples.length >= REQUIRED_SAMPLES) {
      break;
    }
  }

  int samplesToRemove = math.min(_audioBuffer.length,
REQUIRED_SAMPLES);
  _audioBuffer.removeRange(0, samplesToRemove);

  return samples.length >= REQUIRED_SAMPLES ? samples :
null;
 }
```

(a)

```
Future<void> _playMetronomeAsync(bool isAccent) async {
  try {
    const int sampleRate = 44100;
    double duration = isAccent ? 0.08 : 0.05;
    int numSamples = (sampleRate * duration).toInt();

    List<int> samples = [];
    final random = math.Random();

    for (int i = 0; i < numSamples; i++) {
      double time = i / sampleRate;

      double noise = (random.nextDouble() * 2 - 1) * 0.4;
      double tone = math.sin(2 * math.pi * (isAccent ? 1200
: 1000) * time) * 0.6;

      double value = noise + tone;
      double envelope = math.exp(-20 * time / duration);
      double volume = isAccent ? 0.8 : 0.6;

      int sample = (value * envelope * 32767 *
volume).toInt();

      samples.add(sample & 0xFF);
      samples.add((sample >> 8) & 0xFF);
    }

    Uint8List audioData = Uint8List.fromList(samples);

    await _player!.startPlayer(
      fromDataBuffer: audioData,
      codec: Codec.pcm16,
      numChannels: 1,
      sampleRate: sampleRate,
    );

  } catch (e) {
    print('Erro ao reproduzir metrônomo: $e');
  }
 }
```

(b)

```
String? detectPitch(List<double> audioData, {double
sampleRate = 22050}) {
    if (audioData.length < 512) {
      return null;
    }

    double energy = _calculateEnergy(audioData);
    if (energy < 0.001) {
      return null;
    }

    List<double> filteredData =
_highPassFilter(audioData);
    double? frequency =
_detectFrequencyYIN(filteredData, sampleRate);

    if (frequency == null || frequency < 80 ||
frequency > 2000) {
      return null;
    }

    String? pitch = _frequencyToPitch(frequency);
    return _smoothDetection(pitch);
  }
```

(c)

```
Future<void> _playSynthesizedNote(String pitch, int octave,
double duration) async {
    print('Iniciando síntese: $pitch$octave');

    double frequency = _getNoteFrequency(pitch, octave);
    print('Frequência: $frequency Hz');

    const int sampleRate = 44100;
    int numSamples = (sampleRate * duration).toInt();

    List<int> samples = [];
    for (int i = 0; i < numSamples; i++) {
      double time = i / sampleRate;
      double value = math.sin(2 * math.pi * frequency * time);
      double envelope = _calculateEnvelope(i, numSamples);
      int sample = (value * envelope * 32767 * 0.9).toInt();
      samples.add(sample & 0xFF);
      samples.add((sample >> 8) & 0xFF);
    }

    Uint8List audioData = Uint8List.fromList(samples);
    print('Dados gerados: ${audioData.length} bytes');

    try {
      await _player!.startPlayer(
        fromDataBuffer: audioData,
        codec: Codec.pcm16,
        numChannels: 1,
        sampleRate: sampleRate,
      );
      print('Player iniciado');

      await Future.delayed(Duration(milliseconds: (duration *
1000).toInt()));

      await _player!.stopPlayer();
      print('Player parado');
    } catch (e) {
      print('Erro no player: $e');
      rethrow;
    }
  }
```

(d)

Fonte: Do autor.