Ambiente Interativo para Aprendizagem de Violão Assistida por Inteligência Artificial

Nome do Estudante: Alexandre Salgado
Curso: Engenharia de Software
Data: 7 de junho de 2025

Resumo

Este projeto tem como objetivo principal desenvolver um software interativo para auxílio no aprendizado e na prática de violão, focado no reconhecimento de acordes em tempo real. A aplicação será desenvolvida em Flutter, permitindo a criação de versões para desktop (Windows) e mobile (Android/iOS) a partir de uma única base de código. A plataforma integrará funcionalidades didáticas, como uma partitura interativa inspirada no Songsterr, um metrônomo visual e sonoro, e um braço de violão virtual que exibe as digitações corretas. O aplicativo oferecerá um sistema de feedback instantâneo ao usuário sobre a precisão da execução dos acordes. A inteligência artificial para o reconhecimento de acordes será processada em um backend em Python, utilizando bibliotecas como TensorFlow/Keras e Librosa, com o aplicativo Flutter comunicando-se com este backend. O software busca não apenas auxiliar no aprendizado e prática, mas também engajar o usuário através de elementos de gamificação que tornam o processo mais motivador e divertido.

Introdução

- **Contexto:** O aprendizado de violão, especialmente a correta execução dos acordes e a leitura de partituras, é um desafio comum para iniciantes e mesmo para músicos intermediários. A ausência de feedback preciso e interativo durante a prática solo é uma barreira significativa para o aprimoramento técnico e musical. Ferramentas digitais existentes muitas vezes carecem de uma integração fluida entre reconhecimento em tempo real, visualização interativa da música e recursos didáticos completos.

- **Justificativa:** O desenvolvimento deste software se justifica pela necessidade de uma ferramenta centralizada e multiplataforma que ofereça feedback preciso e interativo para praticantes de violão. Ao combinar o reconhecimento de acordes por Inteligência Artificial com elementos didáticos visuais e auditivos (partitura interativa, metrônomo e braço do violão), o projeto visa preencher uma lacuna no mercado de ferramentas de aprendizado musical, oferecendo uma experiência imersiva e eficaz. Essa eficácia será potencializada pela inclusão de elementos de gamificação, que visam aumentar o engajamento do usuário e a motivação a longo prazo, transformando a prática em uma jornada mais divertida e recompensadora. A escolha do Flutter se justifica pela sua capacidade de criar aplicativos de alta performance para desktop e mobile a partir de uma única base de código, otimizando o desenvolvimento, a manutenção e a acessibilidade da solução. No campo da engenharia de software, o projeto se destaca pelo uso de tecnologias modernas de IA e desenvolvimento multiplataforma, oferecendo uma solução eficiente, escalável e com ampla compatibilidade para um problema de aprendizado relevante.

- **Objetivos:** O objetivo principal do projeto é criar um software intuitivo e funcional, compatível com sistemas desktop (Windows) e dispositivos móveis (Android/iOS), que auxilie praticantes de violão a dominar a execução de acordes com precisão e confiança.

  Como objetivos secundários, destacam-se:

  - Implementar um sistema de reconhecimento de acordes baseado em IA que opere em tempo real, fornecendo feedback instantâneo.

- Implementar elementos de gamificação, como sistema de níveis, desafios diários e conquistas, para aumentar o engajamento e a motivação do usuário.
- Desenvolver uma interface de usuário rica e performática em Flutter, com foco na interatividade da partitura.
- Integrar um metrônomo configurável com feedback visual e sonoro, sincronizado com o reconhecimento de acordes e a partitura.
- Criar uma visualização interativa do braço do violão que exiba as digitações corretas dos acordes da partitura.
- Prover a opção de escolha entre microfone embutido e interfaces de áudio externas como fonte de entrada de áudio.
- Gerenciar a comunicação eficiente e de baixa latência entre o aplicativo Flutter (frontend) e o backend Python (IA).

Descrição do Projeto

Tema do Projeto: O projeto consiste no desenvolvimento de um software que atua como um tutor virtual de violão, utilizando Inteligência Artificial para reconhecimento de acordes em tempo real, complementado por um sistema de gamificação robusto. O aplicativo principal será construído em Flutter, permitindo sua execução em ambientes desktop (Windows) e mobile (Android/iOS). A lógica de reconhecimento de acordes baseada em IA será hospedada em um backend em Python, com o aplicativo Flutter comunicando-se com ele para enviar o áudio capturado e receber o feedback do acorde. A plataforma interativa oferecerá um ambiente de prática guiada com feedback instantâneo sobre a precisão dos acordes tocados, e uma experiência de leitura de partitura dinâmica, similar ao Songsterr.

Problemas a Resolver:

- Dificuldade de iniciantes em violão em identificar se estão executando os acordes corretamente sem auxílio externo.
- Desmotivação no aprendizado devido à falta de clareza sobre o progresso e erros , e a dificuldade de manter-se no tempo, que será mitigada por um sistema de recompensas e desafios.
- Ausência de feedback imediato e preciso durante a prática solo de violão, especificamente no contexto de uma partitura.
- Falta de ferramentas centralizadas que combinem partitura interativa, metrônomo e visualização de acordes com reconhecimento em tempo real em uma plataforma multiplataforma.
- Desmotivação no aprendizado devido à falta de clareza sobre o progresso e erros, e a dificuldade de manter-se no tempo.

Limitações:

- O reconhecimento de acordes será focado em acordes comuns (maiores, menores, sétimas), podendo ter limitações com acordes muito complexos, inversões específicas ou técnicas avançadas de toque (ex: harmônicos, bending, arpejos).
- A precisão do reconhecimento pode ser afetada por ruídos ambientais significativos ou pela qualidade do dispositivo de entrada de áudio (embora a opção de interface de áudio minimize isso).

- A partitura inicial será carregada em um formato simplificado (e.g., JSON para acordes e tempos), sem suporte para notação musical complexa (duração de notas individuais, pausas, dinâmica, etc.) ou importação de arquivos MusicXML/MIDI (inicialmente).

Para minimizar riscos, o aplicativo contará com:

- Feedback visual claro e detalhado no braço do violão e na partitura para guiar o usuário em caso de erro.
- Termos de uso claros sobre as capacidades e limitações do software.

Elementos de Gamificação

Elementos de Gamificação Propostos:

Sistema de Progressão (Níveis e XP):

- Pontos de Experiência (XP): O usuário ganhará XP por cada acorde tocado corretamente, por completar sequências ou músicas, e por atingir metas de precisão.
- Níveis: A cada certo número de XP, o usuário avançará de nível, desbloqueando novos exercícios, músicas ou personalizações visuais para o aplicativo.

Desafios Diários e Missões:

- Serão propostos exercícios curtos e focados diariamente, como "Domine o Acorde Sol em 30 segundos" (tocar o acorde 'Sol' com alta precisão um determinado número de vezes dentro do tempo limite) ou "Toque 5 acordes em sequência perfeitamente" (executar uma progressão de acordes pré-definida sem erros).
- A conclusão dos desafios renderá XP e moedas virtuais.

Conquistas e Emblemas (Achievements):

- O aplicativo concederá emblemas virtuais por marcos específicos, como "Primeiro Acorde Perfeito", "Músico Persistente" (prática por X dias consecutivos), "Mestre do Ritmo" (manter o metrônomo por X minutos).

Especificação Técnica

O software adotará uma arquitetura cliente-servidor híbrida, onde o aplicativo Flutter atuará como cliente e um servidor Python como backend para a IA.

- **Frontend (Aplicativo Flutter - Desktop/Mobile):**
  - **Linguagem:** Dart.
  - **Framework:** Flutter.
  - **Captura de Áudio:** Utilizará plugins Flutter específicos para acesso ao microfone do dispositivo (ex: record, audio_stream), capturando o áudio e enviando-o para o backend.
  - **Interface do Usuário:** Construída com os widgets de alta performance do Flutter, garantindo uma UI rica, fluida e consistente em todas as plataformas (Windows, Android, iOS).

- **Partitura Interativa (similar ao Songsterr):** Renderizará dinamicamente a sequência de acordes da partitura. À medida que o metrônomo avança, a partitura rolará e destacará o acorde atual, fornecendo feedback visual de acerto/erro diretamente sobre o acorde.
- **Visualizações:** Desenho personalizado e animado do braço do violão e do metrônomo visual através dos recursos gráficos do Flutter.
- **Comunicação:** Realizará chamadas HTTP/HTTPS ou WebSockets para o backend Python para enviar *chunks* de áudio em tempo real e receber as previsões de acordes.

- **Backend (Python - Servidor):**
  - **Linguagem:** Python.
  - **Framework Web:** Flask (para expor a API de reconhecimento de forma leve e eficiente).
  - **Processamento de Áudio:** Librosa para janelamento, pré-processamento (normalização, remoção de ruído) e extração de características (principalmente chromagramas e possivelmente espectrogramas/Mel espectrogramas) dos *chunks* de áudio recebidos do Flutter.
  - **Modelo de Reconhecimento de Acordes:** Carregará um modelo de Deep Learning (CNN ou CNN-LSTM) treinado com TensorFlow e Keras, realizando a inferência sobre as características extraídas do áudio.
  - **Lógica de Negócio:** Gerenciará a partitura carregada, a sincronização com o metrônomo, a comparação entre o acorde tocado e o esperado, e o envio do feedback detalhado (acorde reconhecido, status de acerto/erro) de volta para o aplicativo Flutter.

Frameworks e Bibliotecas:

- **Frontend (Flutter - Dart):**
  - Flutter: Framework principal para desenvolvimento multiplataforma.
  - [Plugin de Captura de Áudio como 'record' ou 'audio_stream']: Para acessar o microfone do dispositivo e gerenciar o fluxo de áudio.
  - [Plugin de comunicação HTTP/WebSockets como 'http' ou 'web_socket_channel']: Para interagir com o backend Python.
  - [Flutter_bloc ou Provider]: Para gerenciamento de estado (opcional, mas altamente recomendado para complexidade da UI).
- **Backend (Python):**
  - Flask: Framework web para o backend.
  - TensorFlow / Keras: Para desenvolvimento e execução do modelo de IA.
  - Librosa: Para processamento de áudio e extração de características musicais.
  - NumPy: Para manipulação eficiente de dados numéricos.

Protocolos e Algoritmos:

- **Comunicação Cliente-Servidor:** HTTP/HTTPS para requisições assíncronas e WebSockets para comunicação de áudio e feedback em tempo real de baixa latência entre o aplicativo Flutter e o backend Python.
- **Algoritmo de Reconhecimento:** Modelo de Deep Learning CNN-LSTM.
- **Algoritmo de Sincronização:** Lógica baseada no tempo do metrônomo (BPM) e no janelamento preciso do áudio, permitindo a comparação do acorde tocado com o esperado na partitura em um determinado momento.
- **Pós-processamento de Reconhecimento:** Algoritmos de suavização (e.g., média móvel ou filtros de votação) para estabilizar as previsões ruidosas do modelo de IA e fornecer um feedback mais consistente.

Procedimentos:

1. **Fase de Pesquisa e Prototipagem de IA (Offline):**
    - Exploração e, se necessário, aumento do dataset "Guitar Chords v2" do Kaggle para cobrir a variedade de acordes desejada.
    - Desenvolvimento, treinamento e otimização do modelo CNN-LSTM para reconhecimento offline utilizando Python, TensorFlow/Keras e Librosa.
    - Avaliação detalhada das métricas de desempenho do modelo de IA (acurácia, precisão, recall, F1-score, matriz de confusão) para diferentes condições.
2. **Fase de Desenvolvimento do Backend (Python):**
    - Criação da API Flask que expõe o modelo de IA e as funções de processamento de áudio.
    - Implementação da lógica musical para carregar e gerenciar a partitura, a sincronização com o metrônomo e a comparação de acordes.
3. **Fase de Desenvolvimento do Frontend (Flutter):**
    - Construção da interface de usuário em Flutter, incluindo o design do braço do violão interativo, metrônomo visual e sonoro, e a exibição da partitura com rolagem e destaque.
    - Implementação da captura de áudio via plugins Flutter e a funcionalidade de envio dos *chunks* para o backend.
    - Criação da lógica para receber e exibir o feedback detalhado do backend (acorde reconhecido, status de acerto/erro) e sua integração com a partitura e o braço do violão.
4. **Fase de Integração e Testes:**
    - Conexão e comunicação robusta entre o aplicativo Flutter e o backend Python, otimizando a latência.
    - Implementação e refinamento da lógica de sincronização detalhada e do sistema de feedback completo.
    - Testes rigorosos de usabilidade e desempenho do sistema completo, incluindo latência do reconhecimento e precisão do feedback em diferentes cenários de

prática.

- Testes com diferentes violões, técnicas de toque e condições de áudio (microfone do dispositivo, interfaces de áudio).

5. **Fase de Otimização e Refinamento:**
    - Otimização de performance do modelo de IA e do fluxo de comunicação para garantir fluidez.
    - Refinamento da interface do usuário e da experiência do usuário (UX) com base em testes e feedback.

Requisitos de Software:

Lista de Requisitos Funcionais (RF)

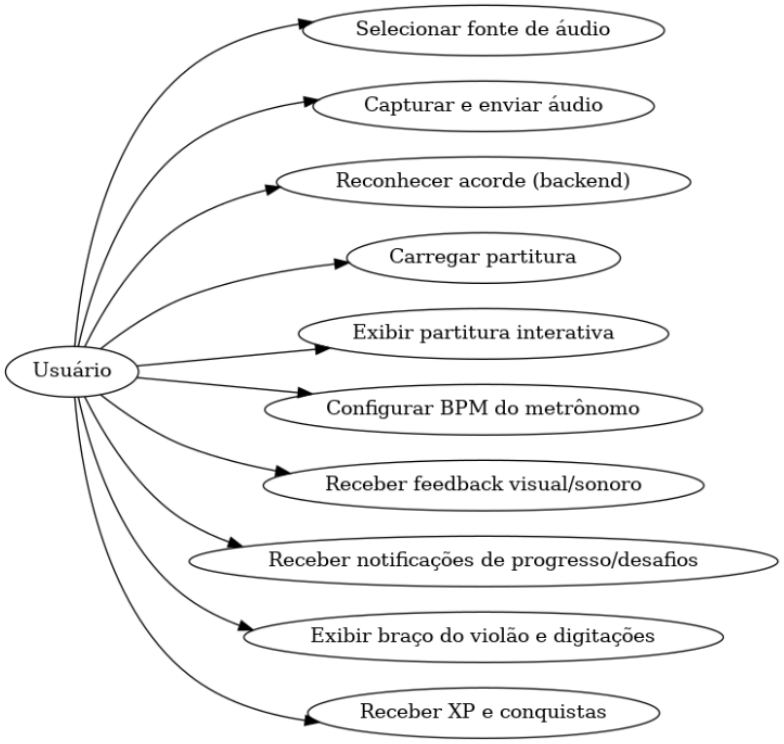
- RF01 - O usuário deve poder selecionar a fonte de entrada de áudio (microfone do dispositivo ou interface de áudio externa).
- RF02 - O aplicativo deve capturar e enviar *chunks* de áudio do violão para o backend em tempo real.
- RF03 - O backend deve processar o áudio e reconhecer o acorde tocado pelo usuário.
- RF04 - O usuário deve poder carregar uma partitura contendo uma sequência de acordes no aplicativo.
- RF05 - O aplicativo deve exibir a partitura de forma interativa, rolando e destacando o acorde atual.
- RF06 - O usuário deve poder configurar o BPM (Batidas Por Minuto) do metrônomo.
- RF07 - O metrônomo deve fornecer feedback sonoro e visual da batida.
- RF08 - O sistema deve sincronizar o reconhecimento do acorde com o metrônomo e a partitura.
- RF09 - O aplicativo deve exibir no braço do violão a digitação correta do acorde esperado.
- RF10 - O aplicativo deve fornecer feedback visual (ex: cores verde/vermelha) no braço do violão e na partitura indicando se o acorde tocado está correto ou incorreto.
- RF11 - O sistema deve permitir iniciar, pausar e parar a sessão de prática.
- RF12 - O sistema deve conceder XP ao usuário com base na precisão da execução e no cumprimento de metas.
- RF13 - O aplicativo deve exibir o nível de progresso do usuário e a quantidade de XP acumulada.
- RF14 - O sistema deve apresentar desafios diários com objetivos claros e recompensas.
- RF15 - O aplicativo deve notificar o usuário sobre novas conquistas e permitir a visualização de emblemas.
- RF16 - O usuário deve poder usar uma moeda virtual para desbloquear itens ou personalizações.

Lista de Requisitos Não Funcionais (RNF):

- RNF01 - O aplicativo deve ser compatível com sistemas operacionais Windows, Android e iOS.
- RNF02 - O tempo de resposta total do reconhecimento de acordes (latência entre tocar e feedback) deve ser minimizado para uma experiência fluida (e.g., inferior a [definir um valor, ex: 300ms]).
- RNF03 - A precisão do reconhecimento de acordes deve ser superior a [Definir um percentual alvo, ex: 85%] para os acordes de treinamento.
- RNF04 - A interface do usuário deve ser intuitiva, responsiva e performática.
- RNF05 - O sistema deve lidar com variações de timbre de diferentes violões e técnicas de toque para um reconhecimento robusto.
- RNF06 - A gamificação deve manter a fluidez e performance do aplicativo, sem comprometer a latência do reconhecimento de acordes.

Representação dos Requisitos

Diagrama de Casos de Uso (UML)



Considerações de Design

- **Interface:** Design limpo, moderno e intuitivo, priorizando a clareza das informações cruciais (partitura, metrônomo, braço do violão). Utilizar as capacidades de animação e personalização de UI do Flutter para uma experiência visualmente rica, incorporando elementos visuais da gamificação de forma harmoniosa..
- **Feedback:** O feedback visual e auditivo claro será integrado aos elementos de gamificação, mostrando o progresso em tempo real e as recompensas.
- **Responsividade:** Design adaptável para diferentes tamanhos de tela e orientações (desktop, tablet, smartphone).
- **Acessibilidade:** Cores e contrastes adequados, e potencial para futuras opções de acessibilidade (ex: ajuste de fonte, customização de tema).

Stack Tecnológica

- **Linguagens de Programação:** Dart (Frontend), Python (Backend).
- **Frameworks e Bibliotecas:**
    - **Flutter (Frontend):**
        - Flutter: Framework principal para o desenvolvimento da aplicação multiplataforma.
        - [Plugin de Captura de Áudio como 'record' ou 'audio_stream']: Para gerenciar a entrada de áudio.
        - [Plugin de comunicação HTTP/WebSockets como 'http' ou 'web_socket_channel']: Para a interação cliente-servidor.
        - [Flutter_bloc ou Provider]: Para gerenciamento de estado (altamente recomendado).
    - **Python (Backend):**
        - Flask: Framework web leve para a API de reconhecimento.
        - TensorFlow / Keras: Para desenvolvimento e execução do modelo de IA.
        - Librosa: Para processamento de áudio e extração de características musicais.
        - NumPy: Para manipulação eficiente de dados numéricos.
- **Ferramentas de Desenvolvimento e Gestão de Projeto:**
    - Visual Studio Code (VSCode) como ambiente de desenvolvimento (com extensões para Dart/Flutter e Python).
    - GitHub para versionamento de código e colaboração.
    - Trello (ou similar como Jira, ClickUp) para gestão do projeto e acompanhamento de tarefas.
    - Figma: Para design e prototipagem da interface do usuário.

Considerações de Segurança

- **Comunicação:** Utilização de HTTPS para a comunicação entre o aplicativo Flutter e o backend Python para garantir a segurança dos dados em trânsito.
- **Segurança da API:** Implementação de medidas de segurança na API do Flask para prevenir acessos não autorizados ou uso indevido.
- **Modelo de IA:** Avaliação da robustez do modelo contra "adversarial attacks" (se relevante e dentro do escopo do TCC).

Referências

- **Flutter:** https://flutter.dev/
- **Dart:** https://dart.dev/
- **Flask:** https://flask.palletsprojects.com/
- **TensorFlow:** https://www.tensorflow.org/
- **Keras:** https://keras.io/
- **Librosa:** https://librosa.org/
- **Kaggle(Dataset "Guitar Chords v2"):** https://www.kaggle.com/datasets/fabianavinci/guitar-chords-v2
- **Visual Studio Code (VSCode):** https://code.visualstudio.com
- **GitHub:** https://github.com
- **Trello:** https://trello.com
- **Songsterr:** https://www.songsterr.com/

Avaliações de Professores

Considerações Professor/a:

Considerações Professor/a:

Considerações Professor/a: