Documento de Especificação do Projeto

AudioCleaner

Aplicação para tratamento de áudio

Feito por: Leonardo Oliva Neves
Curso: Engenharia de Software 7º Fase

Data: 09/04/2024

Documento de Especificação do Projeto

AudioCleaner

Sumário

Sumário

1 Introdução

1.1 | Contexto

O sistema é uma aplicação web para remoção de ruído e aplicação de efeitos criativos em áudio chamado "AudioCleaner". O objetivo principal do "AudioCleaner" é fornecer aos usuários uma ferramenta acessível e fácil de usar. A plataforma visa atender a uma ampla gama de usuários, desde músicos e produtores de áudio até podcasters e entusiastas da gravação caseira. O serviço deve ser rápido e eficaz.

1.2 | Justificativa

A aplicação "AudioCleaner" surge como uma resposta direta à crescente demanda por uma solução acessível e eficaz para tratar os audios da maneira desejada. Com a proliferação do conteúdo de áudio em diversas plataformas, desde músicas e podcasts até vídeos online, a qualidade do áudio tornou-se um aspecto crucial para garantir a satisfação dos usuários.

1.3 Objetivos

O objetivo primordial da aplicação "AudioCleaner" é fornecer aos usuários uma ferramenta acessível e de fácil utilização para remover ruídos indesejados de arquivos de áudio e aplicar efeitos criativos . Ao atender a essa necessidade, a aplicação visa melhorar significativamente a qualidade do áudio em uma variedade de contextos, desde produções musicais profissionais até gravações caseiras de áudio e vídeo. Além disso, a aplicação se esforça para oferecer uma experiência intuitiva e agradável, garantindo que usuários de

todos os níveis de habilidade possam utilizar a ferramenta sem complicações.

2 Descrição do Projeto

2.1 Tema do Projeto

A proposta deste projeto é desenvolver uma aplicação web denominada "AudioCleaner", dedicada à aplicação de efeitos em audios. A "AudioCleaner" será uma ferramenta acessível e eficaz para usuários que buscam melhorar a qualidade do áudio.

2.2 Problemas a resolver

Baixa Qualidade de Áudio: Muitas gravações de áudio podem conter ruídos indesejados, como estática, zumbidos e ruídos de fundo, que afetam negativamente a qualidade do áudio final. A "AudioCleaner" visa resolver esse problema oferecendo uma maneira eficaz de remover esses ruídos, melhorando assim a qualidade geral do áudio.
Dificuldade de Uso de Ferramentas Complexas: As ferramentas de remoção de ruído existentes no mercado podem ser complexas e exigir conhecimentos técnicos avançados para serem utilizadas corretamente.

A "AudioCleaner" pretende resolver esse problema oferecendo uma interface simples e intuitiva, acessível a usuários de todos os níveis de habilidade.
Necessidade de Resultados Rápidos e Eficientes: Em muitos casos, os usuários precisam de resultados rápidos e eficientes ao limpar seus arquivos de áudio. A "AudioCleaner" visa resolver esse problema oferecendo algoritmos de remoção de ruído eficazes e uma plataforma otimizada para processamento rápido de áudio.
Aplicação de efeitos: Em muitos casos, os usuários precisam alterar os audios utilizados em podscasts, filmes e musicas de maneiras criativas e a aplicação visa resolver essa questão.

2.3 Limitações

Eficiência de Remoção de Ruído: Embora a aplicação se esforce para fornecer uma remoção eficaz de ruído, pode haver situações em que certos tipos de ruído sejam mais difíceis de remover completamente.

Ruídos complexos ou de alta intensidade podem apresentar desafios adicionais para os algoritmos de processamento de áudio.

Compatibilidade de Formatos de Áudio: A "AudioCleaner" pode ter limitações em relação aos formatos de áudio suportados para upload e processamento. Nem todos os formatos de áudio podem ser facilmente processados pelos algoritmos de remoção de ruído ou podem exigir conversão para um formato compatível antes do processamento.

Dependência da Qualidade Original do Áudio: A eficácia da remoção de ruído pode ser influenciada pela qualidade original do áudio. Em alguns casos, arquivos de áudio com baixa qualidade ou ruídos excessivos podem apresentar resultados subótimos mesmo após o processamento.

3 Especificação Técnica

3.1 Requisitos de Software

Requisitos Funcionais (RF):

RF1: Os usuários devem ser capazes de fazer upload de arquivos de áudio no formato suportado (WAV).

RF2: O sistema deve aplicar algoritmos de processamento de áudio para aplicação dos efeitos desjados dos arquivos carregados.

RF3: Os usuários devem ter a opção de baixar o áudio processado após a aplicação do efeito. O sistema deve fornecer o áudio alterado no mesmo formato do arquivo original.

RF4: Os usuários devem poder ouvir uma prévia do áudio processado antes de salvar as alterações.

RF5: A aplicação deve conter um sistema de login e autenticação do usuário para usar a mesma.

Requisitos Não Funcionais (RF):

RNF1: O tempo de processamento do áudio deve ser razoável (5 segundos para cada 1 mega de tamanho do arquivo), garantindo uma experiência responsiva para os usuários. A latência do sistema deve ser minimizada para garantir tempos de resposta rápidos durante a interação do usuário.

RNF2: Os dados dos usuários, incluindo os arquivos de áudio enviados, devem ser tratados com confidencialidade e protegidos contra acesso não autorizado.

RNF3: A interface do usuário deve ser intuitiva e fácil de usar, mesmo para usuários iniciantes.

O feedback do sistema, como mensagens de erro e notificações de progresso, deve ser claro e compreensível.
RFN4: O sistema deve ser projetado para lidar com um aumento no número de usuários e na carga de trabalho sem degradação significativa do desempenho.

Diagrama de Casos de Uso

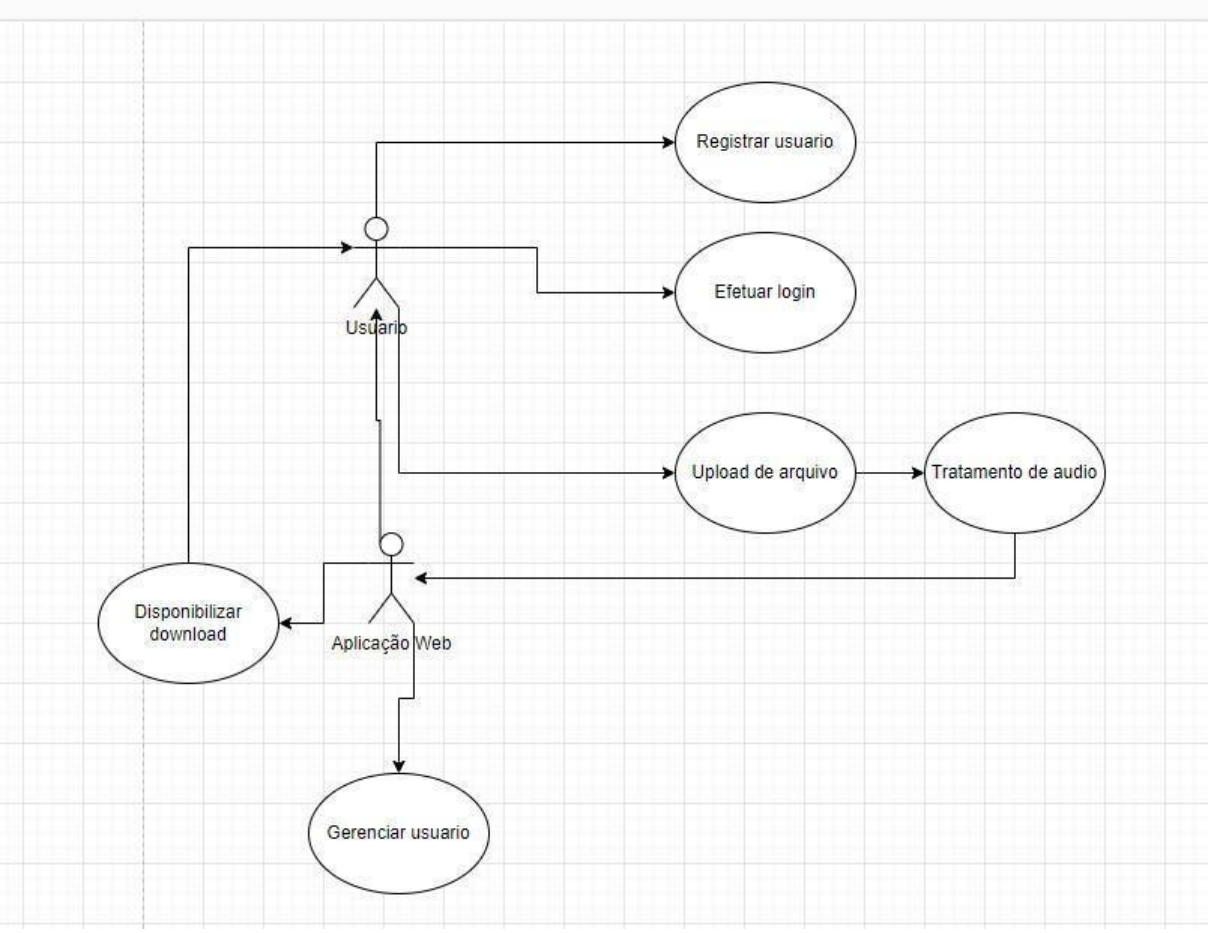
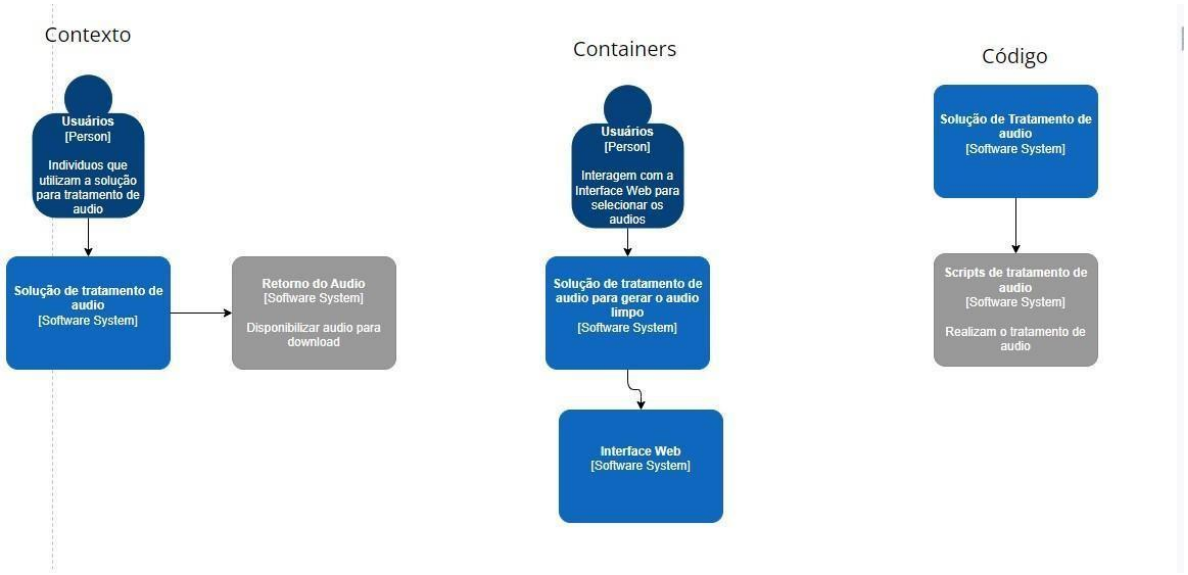


Diagrama C4



3.2 Considerações de design

Visão Inicial da Arquitetura

1. Integração via API
    a. Escolha: Será usado uma API em python para conectar o front-end ao back-end para execução do script de tratamento de audio.

    b. Justificativa: Deixa mais fácil a conexão entre usuário e interface.

    c. Alternativa considerada: Conexão direta entre interface e executável do script.

    d. Razão para rejeição: Menor flexibilidade e maior complexidade de gestão

Padrões de Arquitetura

1. Microserviços:
    - A arquitetura é dividida em serviços independentes, como o Serviço de Processamento de Áudio, API Gateway e Autenticação. Isso permite escalabilidade, manutenção e desenvolvimento independentes.
2. Serverless:
    - Utilização de API com Heroku para processamento de áudio, eliminando a necessidade de gerenciar servidores e facilitando a escalabilidade automática.

3.3 Stack Tecnológica

Linguagens de Programação

1. JavaScript
    a. Justificativa: JavaScript é amplamente utilizado para desenvolvimento frontend devido à sua capacidade de criar interfaces de usuário interativas e dinâmicas. Frameworks como React.js e Vue.js oferecem estruturas robustas para construir interfaces modulares e reutilizáveis, facilitando a manutenção e a escalabilidade do código.

2. Python
    a. Justificativa: Python é uma linguagem poderosa e versátil, especialmente adequada para desenvolvimento backend. Oferece excelente performance, segurança e uma vasta biblioteca de classes que facilita o desenvolvimento de aplicações robustas e escaláveis.

Frameworks e Bibliotecas

1. Front-end
    a. React.js ou Vue.js - Justificativa: Ambos os frameworks oferecem componentes reutilizáveis, integração fácil com outras bibliotecas e ferramentas, e uma grande comunidade de suporte. A escolha entre React.js e Vue.js pode depender da preferência da equipe e dos requisitos específicos do projeto.

2. Back-end
    a. Python - Justificativa: Python é uma linguagem poderosa e versátil, especialmente adequada para desenvolvimento backend.
3. Banco de Dados
    a. Firestore Database – Justificativa: Firestore Database é uma maneira facil de integrar o banco de dados com o sistema de armazenamento de arquivo.

4. Processamento de audio
    a. Noisereduce - Justificativa: A biblioteca utiliza algoritmos avançados baseados no método de redução de ruído de Spectral Gating, uma abordagem científica que analisa a frequência do áudio para remover ruídos de fundo.

    b. Perdalboard - Justificativa: Criada pelo Spotify, a biblioteca é projetada para processamento de áudio de alta qualidade, incluindo efeitos como equalização, compressão, reverb, delay e muito mais.

Ferramentas de Desenvolvimento e Gestão de Projeto

1. IDE e Editores de Codigo
    a. Pycharm - Justificativa: IDE robusta e poderosa, especialmente otimizada para desenvolvimento em Python, oferecendo recursos avançados de depuração e integração com serviços de nuvem.

b. Visual Studio Code - Justificativa: Popular editor de código com suporte para várias extensões, facilitando o desenvolvimento em JavaScript e C#

2. Controle de Versão
    a. Git - Justificativa: Sistema de controle de versão distribuído, essencial para colaboração e controle de versões do código.

    b. GitHub ou GitLab - Justificativa: Plataformas de hospedagem de repositórios Git que oferecem funcionalidades de colaboração, CI/CD (integração contínua/entrega contínua) e gestão de projetos.

3. Gestão de Projeto
    a. Jira - Justificativa: Ferramenta de gestão de projetos ágil, oferecendo recursos para planejamento, rastreamento e gerenciamento de tarefas e sprints.

3.4 Considerações de Segurança

1. Autenticação e Autorização
    a. Questão: Garantir que apenas usuários autorizados possam acessar a aplicação e seus recursos.

    b. Mitigação: Utilizar o serviço AWS Cognito para gerenciar a autenticação e autorização dos usuários. Implementar autenticação de dois fatores (2FA) para uma camada adicional de segurança.

2. Proteção de Dados Sensíveis
    a. Questão: Proteger os dados sensíveis dos usuários, como informações de login, arquivos de áudio e dados pessoais.

    b. Mitigação: Utilizar práticas de criptografia para proteger dados em repouso e em trânsito. Utilizar o AWS KMS (Key Management Service) para gerenciar chaves de criptografia.

3. Testes de Segurança
    a. Questão: Identificar e corrigir vulnerabilidades de segurança durante o desenvolvimento e implantação da aplicação.

    b. Mitigação: Realizar testes de segurança regulares, incluindo testes de penetração, varreduras de vulnerabilidades e revisões de código. Utilizar ferramentas de análise estática de código (SAST) e análise dinâmica de segurança (DAST) para identificar possíveis vulnerabilidades.

4 Deploy

4.1 Diagrama de Deploy

a. **Usuário final (Navegador):**
- Os usuarios acessam o site hospedado no Vercel.
- O site, por sua vez, se comunica com a API hospedada no Heroku para obter ou enviar dados.

b. **Site na Vercel**:
- Desenvolvido com Next.js.
- Deploy continuo integrado com o repositório(Github).
- Responsável pelo front-end e renderização das páginas.

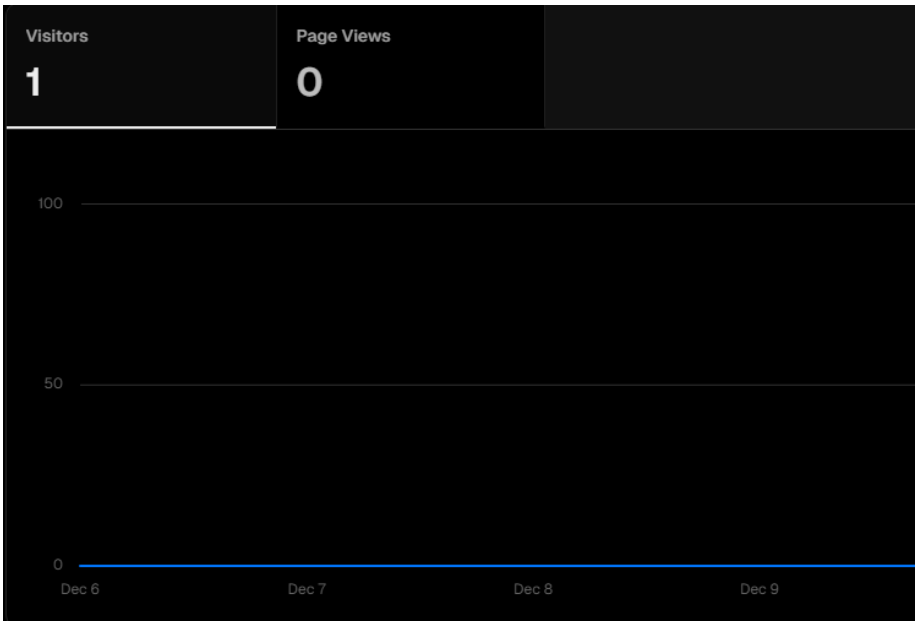
c. **API no Heroku**:

- Desenvolvida em Python(FastAPI).
- Hospeda a lógica de negócios.
- Exposto com um serviço RESTful.
- Deploy continuo integrado com o repositorio(Github)

5 Monitoramento

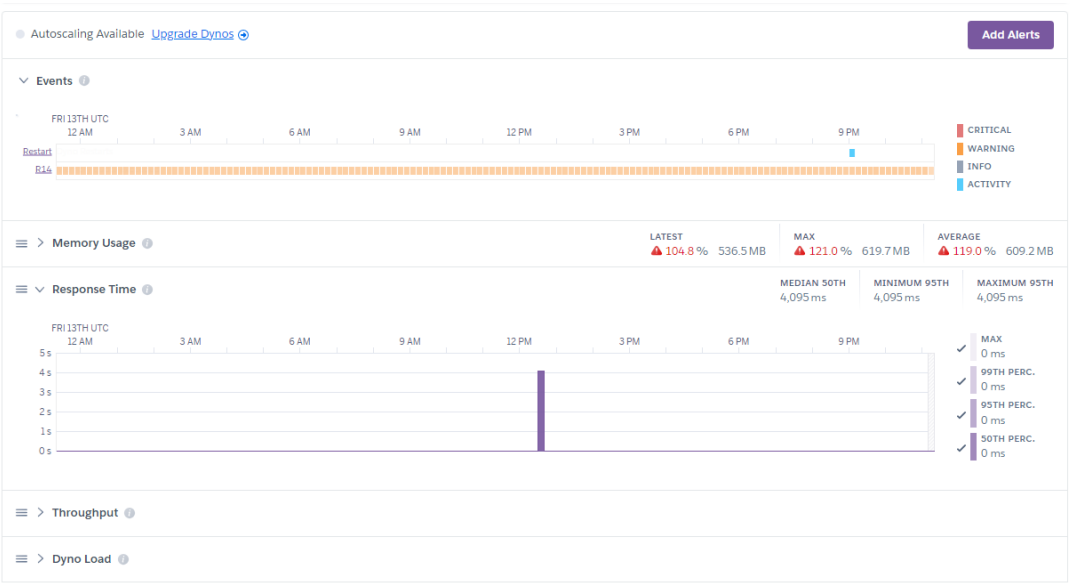
5.1 **Front-end**:

Vercel Web Analytics: Vercel oferece um sistema de monitoramento onde inclui visitas ao site e eventos customizados como quantas vezes foi logado no site ou quantos acessos foram feitos por bots.



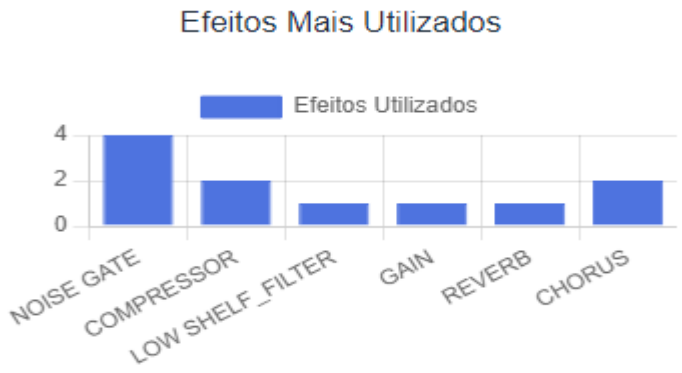
5.2 **Back-end**:

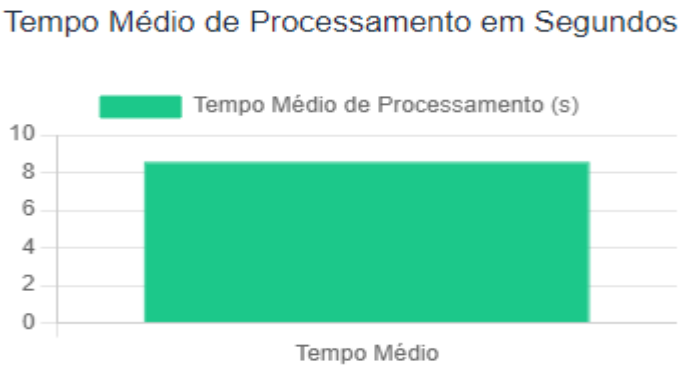
Heroku Metrics: O Heroku oferece uma ferramente de Metrics onde são informados informações como tempo de resposta da API, uso de memoria e eventos como deploys e restarts.



5.3 **Banco de Dados:**

Utilizado para salvar o tempo das execuções e os efeitos mais utilizados para mostrar em forma de grafico ao usuario na página de Dashboard.





6 Próximos Passos

6.1 **Alterar aplicação para funcionar com Streaming**:

A biblioteca utilizada atualmente(pedalboard) não aceita a aplicação de efeitos em streaming, procurar alternativas da biblioteca para poder fazer a alteração dos audios em streaming.

6.2 **Aumentar a quantidade de extensões de audio suportadas**:

A aplicação atualmente só funciona com .wav, assim não sendo tão abragente. Procurar aumentar a disponibilidade de extensões.

7 Referências

7.1 Fontes de Pesquisa

- Documentação oficial da AWS (https://docs.aws.amazon.com/)
- Documentação oficial do ASP.NET Core (https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0)
- Documentação oficial do React.js (https://reactjs.org/docs/getting-started.html)
- Documentação oficial do Vue.js (https://vuejs.org/v2/guide/)
- Livros e artigos acadêmicos sobre processamento de áudio e remoção de ruído.

7.2 Frameworks e bibliotecas

- React.js (https://reactjs.org/)
- Vercel (https://vercel.com/docs)
- Heroki (https://devcenter.heroku.com/categories/reference)
- Python (https://www.python.org/doc/)

- Pedalboard (https://spotify.github.io/pedalboard/)