Discente: Pedro Henrique Pires

Disciplina: Projetos de Arquitetura de Software

Professor: Jacson Rodrigues Barbosa

Curso: Engenharia de Software

Documento de Arquitetura de Software

Introdução

Finalidade

Este documento possui como objetivo definir os aspectos da Arquitetura do software objeto de avaliação da matéria de Padrões de Arquitetura de Software para análise de emoções em mídias e é direcionado aos stakeholders do software a ser desenvolvido, representados pelo professor Jacson Rodrigues, porém, possuindo grande foco para os Desenvolvedores e a Equipe de implantação.

Escopo

Este documento se baseia no documento de requisitos do projeto para definir os atributos de qualidade a serem priorizados, bem como, os estilos arquiteturais que favorecem tais atributos e as representações das visões arquiteturais e seus sub-produtos.

Visão Geral

Os próximos tópicos descrevem requisitos funcionais e não funcionais da aplicação, assim como atributos de qualidade priorizados. Também serão definidos os padrões arquiteturais e suas motivações.

Contexto da Arquitetura

Requisitos Funcionais (RF)

Os requisitos funcionais definem **o que a aplicação deve fazer**.

1. Upload de Arquivos

RF-01 – O usuário deve poder fazer upload de arquivos de áudio e vídeo pela interface web.
RF-02 – O sistema deve permitir uploads de arquivos nos formatos **MP3, MP4, WAV, AVI, MKV, MOV**.
RF-03 – O tamanho máximo do arquivo permitido deve ser de **50 MB** (configurável no backend).
RF-04 – O sistema deve exibir uma **barra de progresso** indicando o status do upload.

2. Processamento e Análise de Emoções

RF-05 – O backend deve processar o áudio do arquivo enviado.
RF-06 – O sistema deve extrair e interpretar **as emoções presentes no áudio** utilizando um

modelo de **IA**.
RF-07 – As emoções identificadas devem incluir pelo menos:

- **Felicidade**
- **Tristeza**
- **Raiva**
- **Surpresa**
- **Neutro**
  RF-08 – O backend deve enviar ao frontend o resultado da análise, informando:
- As emoções predominantes.
- Um percentual de confiança para cada emoção detectada.

3. Retorno ao Usuário

RF-09 – O frontend deve exibir o resultado da análise emocional após o processamento do arquivo.
RF-10 – O usuário deve ser notificado caso o arquivo não contenha áudio válido ou se houver erro na análise.

4. Segurança

RF-11 – O backend deve validar os arquivos enviados para garantir que sejam formatos suportados.
RF-12 – O sistema deve bloquear uploads de arquivos suspeitos ou potencialmente maliciosos.

Requisitos Não Funcionais (RNF)

Os requisitos não funcionais definem **como a aplicação deve se comportar**.

1. Performance

RNF-01 – O tempo de processamento da análise de emoções deve ser inferior a **10 segundos** para arquivos de até **20 MB**.
RNF-02 – O sistema deve permitir que múltiplos usuários façam uploads simultaneamente sem comprometer o desempenho.

2. Inteligência Artificial

RNF-03 – O modelo de IA utilizado deve ser capaz de identificar emoções com uma **precisão mínima de 80%**.
RNF-04 – A aplicação deve utilizar bibliotecas de processamento de áudio, como **librosa, pydub, ou SpeechRecognition**.

RNF-05 – O backend pode usar um modelo pré-treinado, como **Wav2Vec, OpenSmile, DeepSpeech ou outros**.

3. Usabilidade

RNF-06 – A interface deve ser responsiva, permitindo uso em **desktops, tablets e smartphones**.
RNF-07 – O sistema deve apresentar a análise de forma intuitiva, usando gráficos ou tabelas.

4. Segurança

RNF-08 – O backend deve impedir injeções de código e ataques de upload malicioso.
RNF-09 – As requisições entre frontend e backend devem ser protegidas com **CORS e autenticação JWT (se necessário)**.

5. Manutenção e Escalabilidade

RNF-10 – O código deve seguir boas práticas para facilitar futuras melhorias.
RNF-11 – A aplicação deve permitir integração futura com serviços externos de IA, como **Google Speech-to-Text, AWS Transcribe ou IBM Watson**.

Estilo arquitetural

1. Estilo Arquitetural: Cliente-Servidor + Camada de IA

A aplicação mantém uma **arquitetura Cliente-Servidor**, mas agora o backend executa uma análise **baseada em Inteligência Artificial (IA)** para processar emoções a partir do áudio enviado.

Os principais componentes agora incluem:

- **Frontend (Vue.js)** → Interface para upload e exibição dos resultados.
- **Backend (FastAPI + IA)** → Processamento de áudio e análise de emoções.
- **Módulo de IA** → Utiliza um modelo treinado para identificar emoções a partir do áudio.
- **Armazenamento (Sistema de Arquivos ou Cloud)** → Guarda os arquivos enviados e pode ser expandido para serviços em nuvem.

2. Arquitetura em Camadas

Agora, a arquitetura é composta por quatro camadas principais:

Camada de Apresentação (Frontend - Next.js)

- Fornece a interface gráfica para o usuário.
- Envia arquivos para o backend via API REST.
- Exibe o resultado da análise de emoções.

Camada de Aplicação (Backend - FastAPI)

- Gerencia requisições HTTP do frontend.
- Valida e armazena os arquivos recebidos.
- Aciona o Módulo de IA para análise do áudio.

Camada de Processamento de IA (Machine Learning)

- Converte áudio em espectrograma ou outro formato para análise.
- Executa modelos de Deep Learning para detectar emoções.
- Retorna uma lista de emoções identificadas com seus níveis de confiança.

Camada de Persistência (Sistema de Arquivos / Cloud Storage)

- Armazena os arquivos enviados na pasta `uploads/` (ou em um serviço na nuvem).
- Possível integração futura com AWS S3, Google Cloud Storage ou Banco de Dados.

3. Padrões Arquiteturais Utilizados

Arquitetura RESTful

- O backend expõe endpoints REST para comunicação com o frontend.
- Permite escalabilidade e integração futura com outros serviços.

Arquitetura Baseada em Serviços (Service-Oriented Architecture - SOA)

- A análise de emoções é tratada como um serviço independente dentro do backend.
- Possível evolução para um microserviço de IA no futuro.

Pipeline de Processamento de Áudio

- O áudio enviado passa por um fluxo de transformação e análise.
- Pode incluir pré-processamento (remoção de ruído, normalização).

4. Possíveis Evoluções

- Microservices: Separar o serviço de IA como um microserviço independente.
- Integração com Cloud AI: Utilizar Google Speech-to-Text, AWS Transcribe ou IBM Watson.
- Armazenamento em Banco de Dados: Para manter históricos de análises

Atributos de qualidade

1. Desempenho e Escalabilidade

Prioridade: Alta

A aplicação precisa processar arquivos de áudio/vídeo e realizar análise de emoções sem grandes atrasos. Para isso:

- **Tempo de processamento da IA ≤ 10 segundos** para arquivos de até **20 MB**.
- **Suporte a múltiplos uploads simultâneos** sem comprometer o desempenho.
- **Possibilidade de escalabilidade** com processamento distribuído ou serviços em nuvem.

2. Precisão e Confiabilidade da IA

Prioridade: Alta

A análise de emoções deve fornecer **resultados confiáveis** para o usuário. Portanto:

- O modelo de IA deve ter uma **precisão mínima de 80%** na detecção de emoções.
- O sistema deve informar um **percentual de confiança** para cada emoção detectada.
- Implementação de **validação e normalização de áudio** para melhorar a precisão.

3. Segurança

Prioridade: Alta

Como a aplicação recebe arquivos do usuário, medidas de segurança são essenciais:

- **Validação de arquivos** (formato e tamanho) antes do processamento.
- **Proteção contra uploads maliciosos**, impedindo execução de código ou ataques de injeção.
- **Restrições de CORS** para evitar requisições indevidas ao backend.
- **Autenticação JWT** (caso seja necessário controle de acesso no futuro).

4. Usabilidade e Experiência do Usuário

Prioridade: Média-Alta

A interface deve ser clara, intuitiva e fornecer feedback adequado:

- **Responsividade**, permitindo uso em dispositivos móveis e desktop.
- **Barra de progresso** indicando o status do upload.
- **Exibição visual do resultado da análise**, usando gráficos ou tabelas.
- **Mensagens claras de erro/sucesso** durante o upload e análise.

5. Manutenibilidade e Extensibilidade

Prioridade: **Média**

A aplicação deve ser fácil de atualizar e expandir:

- Código organizado seguindo **boas práticas de Clean Code e SOLID**.
- Estrutura modular que permita **adicionar novos modelos de IA** no futuro.
- Logs detalhados para **debugging e auditoria de uploads**.

Principais casos de uso e arquitetura relacionada

A análise da arquitetura com base em **casos de uso** nos ajuda a visualizar como os diferentes componentes do sistema interagem para atender aos objetivos dos usuários.

Caso de Uso 1: Upload de Arquivo (Áudio/Vídeo)

Descrição: O usuário seleciona e faz o upload de um arquivo de áudio ou vídeo para análise.
Fluxo Arquitetural:

1. O usuário acessa a interface web (Next.js).
2. Seleciona um arquivo compatível (MP3, MP4, WAV, etc.).
3. O frontend envia o arquivo para o backend via **API REST** (`POST /upload/`).
4. O backend (FastAPI) valida o arquivo:
    - Verifica **tamanho e formato**.
    - Salva o arquivo na pasta `uploads/`.
5. Retorna uma **resposta de sucesso** ao frontend

Componentes Envolvidos:

- **Frontend (Next.js):** Formulário de upload.
- **Backend (FastAPI):** Validação e armazenamento do arquivo.
- **Sistema de Arquivos:** Armazena o arquivo temporariamente.

Caso de Uso 2: Processamento do Áudio e Análise de Emoções

Descrição: O backend processa o arquivo de áudio para detectar emoções.
Fluxo Arquitetural:

1. O backend **extrai o áudio do arquivo**, caso seja um vídeo.

2. O áudio é convertido para um formato analisável (**WAV, espectrograma, etc.**).
3. O backend chama um **módulo de IA** para análise de emoções.
4. O modelo de IA **identifica emoções** e retorna resultados com níveis de confiança.
5. O backend armazena os resultados temporariamente e os retorna ao frontend.

Componentes Envolvidos:

- **Backend (FastAPI):** Controla o fluxo de análise.
- **Módulo de IA:** Modelo pré-treinado (exemplo: Wav2Vec, OpenSmile).
- **Bibliotecas de Processamento de Áudio:** `librosa`, `pydub`, `SpeechRecognition`.
- **Armazenamento Temporário:** Arquivo convertido e espectrograma.

Caso de Uso 3: Exibição dos Resultados da Análise

Descrição: O usuário visualiza a análise de emoções do áudio enviado.
Fluxo Arquitetural:

1. O frontend recebe os resultados do backend.
2. Exibe as emoções identificadas com **percentuais de confiança**.
3. Pode exibir um **gráfico de emoções ao longo do tempo**.

Componentes Envolvidos:

- **Frontend (Next.js):** Interface gráfica para exibição dos resultados.
- **Backend (FastAPI):** API que fornece os resultados.
- **Bibliotecas de Visualização:** (exemplo: Chart.js, Recharts)

Caso de Uso 4: Tratamento de Erros e Feedback ao Usuário

Descrição: Se houver erro durante o upload ou análise, o usuário recebe um aviso.
Fluxo Arquitetural:

1. O backend detecta erros, como **formato inválido, arquivo corrompido ou falha na IA**.
2. Retorna uma **mensagem de erro padronizada** ao frontend.
3. O frontend exibe um aviso claro ao usuário.

Componentes Envolvidos:

- **Frontend (Next.js):** Exibição de mensagens de erro.
- **Backend (FastAPI):** Detecção e tratamento de erros.

Caso de Uso 5: Exclusão de Arquivos Processados *(Opcional, Futuro)*

finalizar…

Visão de Projetista

O projetista da aplicação precisa considerar diversos fatores, incluindo **modularidade, escalabilidade, manutenibilidade e desempenho**. A seguir, detalho a visão do projetista sobre a arquitetura, suas decisões e desafios técnicos.

1. Decisões Arquiteturais Fundamentais

Arquitetura Cliente-Servidor com IA Acoplada

O sistema segue um **modelo Cliente-Servidor**, onde:

- **Frontend (Next.js)** → Responsável pela interface e envio de arquivos.
- **Backend (FastAPI + IA)** → Processa arquivos, executa a análise de emoções e retorna os resultados.
- **Módulo de IA** → Realiza a interpretação emocional do áudio.

Esse modelo foi escolhido porque:

- Separa responsabilidades, facilitando a **manutenção e escalabilidade**.
- Permite que o módulo de IA seja substituído por uma API externa no futuro.
- Adota **REST APIs**, tornando a integração com outros serviços mais simples.

2. Estrutura Modular do Projeto

A organização do código foi planejada para facilitar a manutenção e futuras expansões.

Frontend (Next.js)

```
upload-app/
├── components/
│   ├── FileUpload.js  # Componente de upload
│   ├── ResultDisplay.js  # Exibe os resultados da análise
├── pages/
│   ├── index.js  # Tela principal
├── services/
│   ├── api.js  # Comunicação com o backend
├── styles/
├── public/  # Ícones, favicon
├── next.config.js
├── package.json
```

Backend (FastAPI)

```
backend/
├── models/
│   ├── emotion_model.py  # Carregamento e execução do modelo de IA
├── services/
│   ├── file_service.py  # Manipulação dos arquivos
│   ├── audio_processing.py  # Conversão de áudio
├── routes/
│   ├── upload.py  # Endpoint de upload
│   ├── analyze.py  # Endpoint de análise de emoções
├── main.py  # Inicialização da API
├── requirements.txt
```

Decisões:
Separação de responsabilidades evita código monolítico.
Facilidade para testar cada módulo individualmente.
Extensibilidade → Caso seja necessário trocar a IA ou adicionar suporte a novos formatos de arquivo.

3. Principais Desafios Técnicos e Soluções

Desafio 1: Tempo de Processamento da IA

O processamento de áudio pode ser demorado, dependendo do modelo utilizado.

Solução:

- **Pré-processamento do áudio** (remover ruídos e converter para espectrograma).
- **Uso de modelos otimizados** para inferência rápida (ex: Wav2Vec, OpenSmile).
- **Execução assíncrona** no backend para evitar bloqueios.

Desafio 2: Upload de Arquivos Grandes

O usuário pode enviar arquivos grandes, impactando o tempo de upload e armazenamento.

Solução:

- **Limitação de tamanho** no frontend e backend (ex: **máx. 50MB**).
- **Streaming de upload** para processar arquivos enquanto chegam.
- **Armazenamento temporário** e limpeza automática após a análise.

Desafio 3: Precisão da Análise de Emoções

A identificação de emoções pode ter variações dependendo da qualidade do áudio.

Solução:

- Aplicação de **normalização e remoção de ruído** antes da análise.
- Uso de **modelos treinados com grande diversidade de dados**.
- Adição de **percentual de confiança** nas respostas da IA.

Desafio 4: Escalabilidade e Manutenção do Backend

Se a aplicação crescer, pode haver sobrecarga no processamento de IA.

Solução:

- **Uso de filas de processamento assíncrono** (ex: Celery, Redis).
- **Deploy escalável via Docker e Kubernetes**.
- **Possível terceirização da análise de IA** para serviços como Google Speech-to-Text.

4. Tecnologias e Justificativas

Tecnologia	Motivo da Escolha
Next.js	Framework React otimizado para desempenho e SEO.
FastAPI	Melhor desempenho que Flask, com suporte a async.
Pydub / Librosa	Processamento eficiente de áudio.
TensorFlow / PyTorch	Frameworks robustos para IA.
PostgreSQL / MongoDB *(futuro)*	Para armazenamento estruturado de análises.
Docker	Facilita deploy e escalabilidade.

Inserir Diagrama de Componentes

Possíveis melhorias

1. Melhorias Técnicas

1.1 Processamento Assíncrono para Melhor Escalabilidade

- Atualmente, a análise do áudio ocorre de forma síncrona.
- **Solução:** Implementar uma **fila de tarefas** usando **Celery + Redis** para processar os arquivos em background.
- Isso evita **tempo de espera excessivo** para o usuário.

1.2 Suporte a Múltiplos Formatos de Entrada

- Atualmente, são aceitos apenas alguns formatos de áudio e vídeo.
- **Evolução:** Expandir para aceitar **formatos adicionais** (ex: FLAC, OGG) e melhorar a **extração de áudio de vídeos**.

1.3 Melhoria na Precisão da IA

- O modelo atual pode ser otimizado para maior precisão.
- **Evolução:**
    - Usar **redes neurais mais avançadas** como **Wav2Vec 2.0** ou **DeepSpeech**.
    - **Treinar com datasets mais amplos** para cobrir diferentes **tons de voz, sotaques e emoções**.

1.4 Implementação de API GraphQL *(Opcional)*

- Atualmente, a API usa **REST**.
- **Evolução:** Adicionar **GraphQL** para maior **flexibilidade na consulta de dados**, permitindo ao frontend requisitar **somente os dados necessários**.

2. Melhorias na Experiência do Usuário (UX/UI)

🔹 2.1 Feedback Visual e Barra de Progresso

- No upload e processamento, o usuário não sabe exatamente quanto tempo falta.
- **Solução:** Adicionar uma **barra de progresso** e feedbacks dinâmicos no frontend.

2.2 Gráficos Interativos de Emoções

- A exibição atual dos resultados pode ser mais visual.

- **Evolução:** Implementar **gráficos interativos** mostrando como as emoções mudam ao longo do tempo.

2.3 Versão Mobile-Friendly

- Melhorias na responsividade para **telas menores**.
- **Evolução:** Criar um **PWA (Progressive Web App)** para melhor experiência em smartphones.

3. Expansão para Novas Funcionalidades

3.1 Geração de Relatórios e Exportação

- Possibilidade de exportar resultados para **PDF, CSV** ou compartilhar via link.

3.2 Integração com APIs Externas *(Exemplo: Google Cloud Speech-to-Text)*

- Para melhorar o reconhecimento de fala e precisão da análise.

3.3 Análise Multimodal (Áudio + Texto)

- Além da emoção no áudio, a IA pode **transcrever a fala** e analisar o significado do que foi dito.

4. Melhorias em Segurança e Infraestrutura

4.1 Autenticação de Usuários

- Atualmente, qualquer pessoa pode usar a aplicação sem controle.
- **Evolução:** Implementar **login e permissões** para uso mais seguro.

4.2 Armazenamento em Nuvem

- Hoje, os arquivos são salvos localmente.
- **Evolução:** Integrar com **Amazon S3 ou Google Cloud Storage** para melhor escalabilidade.

4.3 Logs e Monitoramento com Observabilidade

- Implementação de **logs estruturados** para melhor **debug e análise de falhas**.
- Uso de ferramentas como **Prometheus + Grafana** para monitoramento.