RELATÓRIO DE ARQUITETURA DO SISTEMA DE ANÁLISE DE ÁUDIO

Este relatório apresenta a documentação da arquitetura do *Sistema de Análise de Áudio*, o qual recebe um arquivo de áudio no formato WAV, realiza a transcrição do áudio para texto utilizando a Google Web Speech API e, em seguida, efetua a análise de sentimento do texto com `TextBlob`. A documentação segue o *C4 Model*, com diagramas de Contexto, Containers e Componentes. Além disso, apresenta-se uma proposta de solução arquitetural baseada em microserviços e processamento assíncrono, visando escalabilidade e robustez.

1 Introdução

O presente trabalho tem por objetivo documentar a arquitetura de um sistema desenvolvido em `Python` com o framework FastAPI, cuja finalidade é processar arquivos de áudio. Após receber um arquivo no formato WAV, o sistema realiza a transcrição do áudio para texto, utilizando a Google Web Speech API, e analisa o sentimento do texto transcrito com a biblioteca `TextBlob`. Para a documentação arquitetural, adota-se o *C4 Model*, que organiza a visualização do sistema em diferentes níveis de abstração.

2 Arquitetura do Sistema

A documentação segue a abordagem do C4 Model, dividida nos seguintes níveis:

- **Diagrama de Contexto:** Apresenta as interações do sistema com seus atores e serviços externos.
- **Diagrama de Containers:** Detalha os principais módulos (*containers*) do sistema e suas responsabilidades.
- **Diagrama de Componentes:** Descreve a decomposição interna do container principal (FastAPI), evidenciando os seus componentes.

3 Proposta de Solução Arquitetural

A proposta arquitetural visa garantir escalabilidade, resiliência e facilidade de manutenção por meio de uma abordagem orientada a microserviços e processamento assíncrono. A seguir, descrevem-se os principais componentes e o fluxo de dados da solução:

3.1 Componentes Principais

1. **API Gateway / Serviço de Frontend (FastAPI):** Recebe o arquivo de áudio via requisição HTTP, valida o formato (apenas WAV) e encaminha a tarefa para o processamento.

2. **Serviço de Armazenamento:** Armazena temporariamente o arquivo de áudio, permitindo que os serviços de processamento o acessem. Pode ser implementado utilizando soluções de armazenamento em nuvem (por exemplo, AWS S3).

3. **Serviço de Processamento Assíncrono:** Consome mensagens de um *broker* de mensagens (como RabbitMQ, AWS SQS ou Kafka) e realiza o processamento do áudio. Esse serviço executa as seguintes etapas:

    - **Transcrição:** Utiliza a biblioteca `SpeechRecognition` para extrair o texto do áudio, fazendo uso da Google Web Speech API.

- **Análise de Sentimento:** Processa o texto com a biblioteca `TextBlob` para determinar a polaridade (positivo, negativo ou neutro).

4. **Serviço de Persistência dos Resultados:** Armazena os resultados do processamento (texto transcrito e análise de sentimento) em um banco de dados (SQL ou NoSQL) e pode notificar o cliente ou disponibilizar uma API para consulta.

3.2 Fluxo de Dados

- O **Cliente** envia o arquivo de áudio para a **API Gateway**.
- A **API** armazena o arquivo e enfileira uma mensagem no **Broker de Mensagens**.
- O **Serviço de Processamento** consome a mensagem, recupera o arquivo do armazenamento, realiza a transcrição e a análise de sentimento.
- Os resultados são armazenados no **Banco de Dados** e disponibilizados para consulta ou notificação ao cliente.

3.3 Benefícios da Abordagem

- **Escalabilidade:** A separação entre a camada de recepção (API) e o processamento assíncrono possibilita escalar cada serviço de forma independente.
- **Resiliência:** O uso de um *broker* de mensagens desacopla as camadas, permitindo retentativas e isolamento de falhas.
- **Manutenibilidade:** A arquitetura modular permite a atualização ou substituição de componentes individuais sem impactar o sistema como um todo.
- **Integração Facilitada:** A encapsulação da chamada à Google Web Speech API no serviço de processamento possibilita a implementação de políticas de retries e fallback.

4 Diagrama de Contexto

O diagrama de contexto ilustra as interações do sistema com seus principais atores e serviços externos.

5 Diagrama de Containers

O diagrama de containers apresenta os principais módulos do sistema e suas funções.

Diagrama Gerado

6 Diagrama de Componentes

Focando no container da aplicação FastAPI, este diagrama detalha os componentes internos responsáveis pelo processamento das requisições.

Diagrama Gerado

7 Conclusão

A solução arquitetural proposta utiliza uma abordagem orientada a microserviços e processamento assíncrono, garantindo escalabilidade, resiliência e facilidade de manutenção. A separação entre a camada de recepção (API), armazenamento, processamento e persistência dos resultados permite que cada componente seja escalado e atualizado independentemente. Essa estratégia também facilita a integração com serviços externos, como a Google Web Speech API, e a implementação de mecanismos robustos de monitoramento e logging.

Por meio do *C4 Model*, foi possível documentar a arquitetura do Sistema de Análise de Áudio de forma clara e hierárquica, contribuindo para a melhor comunicação entre os membros da equipe e os stakeholders.

Referências

- C4 Model: `https://c4model.com/`
- C4-PlantUML: `https://github.com/plantuml-stdlib/C4-PlantUML`

Figura 1: Diagrama de Contexto

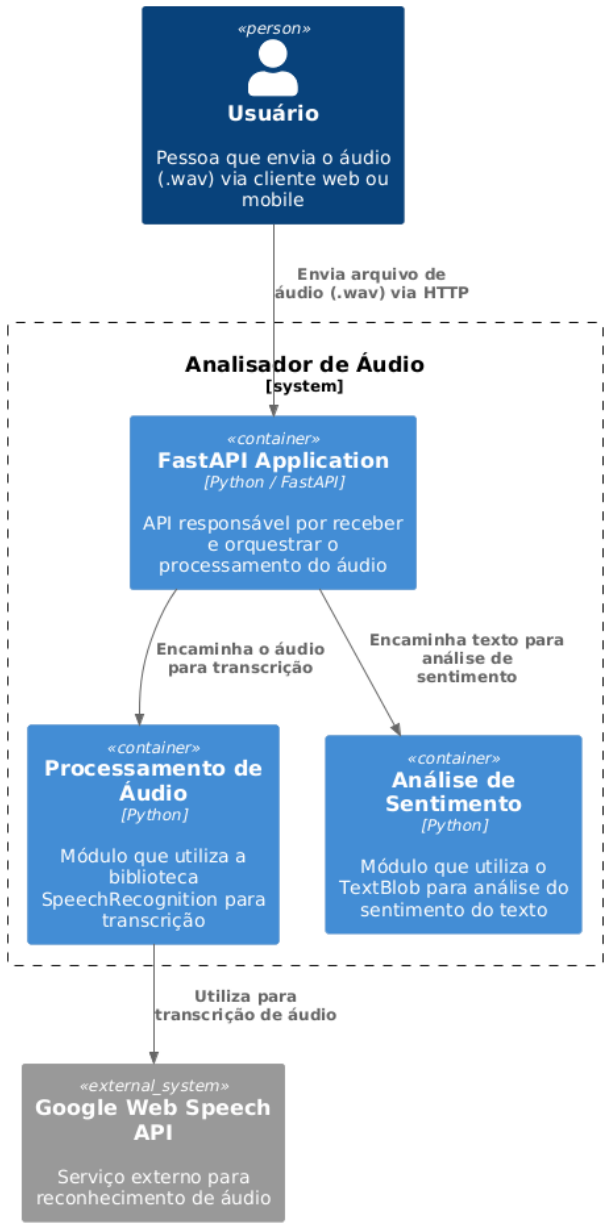

Figura 2: Diagrama de Containers do Sistema

Figura 3: Diagrama de Componentes da FastAPI Application