UNIVERSIDADE NOVE DE JULHO - UNINOVE
PROJETO DE DESENVOLVIMENTO DE APLICAÇÕES MULTIPLATAFORMA

EXCLUÍDOS OS DADOS SOBRE OS AUTORES EM ATENDIMENTO A LGPD - LEI GERAL DE PROTEÇÃO DE DADOS

Speech4text.ai
Transcrição e Geração de Texto Criativo por IA

São Paulo
2024

EXCLUÍDOS OS DADOS SOBRE OS AUTORES EM ATENDIMENTO A LGPD - LEI GERAL DE PROTEÇÃO DE DADOS

Speech4text.ai
Transcrição e Geração de Texto Criativo por IA

Projeto apresentado a Universidade Nove de Julho - UNINOVE, como parte dos requisitos obrigatórios para obtenção do título de bacharel em Ciência da computação.

Prof. Orientador: Edson Melo de Souza, Dr.

São Paulo
2024

RESUMO

Neste projeto desenvolvemos uma aplicação web composta por um frontend em React e um backend em Node.js, que utiliza Prisma para gerenciar o banco de dados e a API do OpenAI para gerar respostas automáticas baseadas em texto. O objetivo é criar uma plataforma que permite aos usuários fazer upload de vídeos e obter transcrições e sugestões automáticas de conteúdo. A conversão de fala em texto (STT) é um processo que transforma áudio falado em texto escrito. Essa tecnologia tem diversas aplicações em áreas como jornalismo, pesquisa, educação, acessibilidade e atendimento ao cliente. A demanda por soluções STT precisas e eficientes está crescendo, impulsionada pelo aumento do uso de dispositivos móveis, mídias sociais e conteúdo de áudio. speech4text.ai visa facilitar a conversão de audio em texto. Um servidor back-end funcional para conversão de fala em texto utilizando a API OpenAI e o framework Fastify. Transcrições de texto precisas e confiáveis para arquivos de áudio no formato MP3. Um banco de ados relacional que armazena prompts e transcrições de forma organizada e eficiente. Uma API RESTful que permite o acesso às funcionalidades do servidor por meio de aplicações clientes. Uma avaliação da precisão das transcrições geradas pelo servidor. Uma análise do desempenho do servidor em termos de tempo de resposta, consumo de recursos e escalabilidade.

Palavras-chave: Conversão de fala em texto (STT), Inteligência artificial, OpenAI API, Fastify, Processamento de linguagem natural, Transcrição de áudio, Banco de dados relacional, API RESTful, Precisão, Desempenho, Escalabilidade

ABSTRACT

In this project we developed a web application composed of a frontend in React and a backend in Node.js, which uses Prisma to manage the database and the OpenAI API to generate automatic text-based responses. The goal is to create a platform that allows users to upload videos and get transcripts and automatic content suggestions. Speech-to-text (STT) is a process that transforms spoken audio into written text. This technology has diverse applications in areas such as journalism, research, education, accessibility and customer service. Demand for accurate and efficient STT solutions is growing, driven by the increased use of mobile devices, social media and audio content. speech4text.ai aims to facilitate the conversion of audio to text. A functional back-end server for speech-to-text conversion using the OpenAI API and the Fastify framework. Accurate and reliable text transcriptions to audio files in MP3 format. A relational database that stores prompts and transcripts in an organized and efficient way. A RESTful API that allows access to server functionalities through client applications. An assessment of the accuracy of server-generated transcripts. An analysis of server performance in terms of response time, resource consumption, and scalability.

Keywords: Speech to Text (STT), Artificial Intelligence, OpenAI API, Fastify, Natural Language Processing, Audio Transcription, Relational Database, RESTful API, Accuracy, Performance, Scalability

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

MP3	MPEG Audio Layer III (formato de arquivo de áudio comprimido)
MP4	MPEG-4 Part 14 (formato de arquivo de vídeo comprimido)
URI	Uniform Resource Identifier (Identificador Uniforme de Recursos)
JSON	JavaScript Object Notation (Notação de Objeto JavaScript)
FFmpeg	Fast Forward MPEG (framework de software para processamento de áudio e vídeo)
API	Application Programming Interface (Interface de Programação de Aplicativos)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
CORS	Cross-Origin Resource Sharing (Compartilhamento de Recursos entre Origens Cruzadas)
DNS	Domain Name System (Sistema de Nomes de Domínio)
DB	Database (Banco de Dados)
JWT	JSON Web Token (Token Web JSON)
CLI	Command Line Interface (Interface de Linha de Comando)
RAM	Random Access Memory (Memória de Acesso Aleatório)
TCP	Transmission Control Protocol (Protocolo de Controle de Transmissão)
HTTPS	Hypertext Transfer Protocol Secure (Protocolo de Transferência de Hipertexto Seguro)
TLS	Transport Layer Security (Segurança da Camada de Transporte)
REST	Representational State Transfer (Transferência de Estado Representacional)
STT	Speech-to-Text (fala para texto, em português).

1 INTRODUÇÃO

Resumo do capítulo

Com o aumento significativo do consumo de conteúdo multimídia, a necessidade de ferramentas eficientes para gerenciar e otimizar esse conteúdo tornou-se evidente. Este projeto visa abordar essa necessidade desenvolvendo uma aplicação web integrada com inteligência artificial, capaz de automatizar a transcrição de vídeos e gerar respostas inteligentes.

1.1 Contextualização

Com o avanço da inteligência artificial, aplicações que automatizam processos de geração e análise de conteúdo tornaram-se viáveis. Este projeto visa explorar essas possibilidades. A crescente digitalização e o consumo massivo de vídeos aumentaram a demanda por ferramentas que possam transcrever e analisar conteúdo multimídia de forma eficiente. Empresas e criadores de conteúdo enfrentam desafios significativos na gestão e otimização de vídeos, o que pode resultar em perda de tempo e recursos.

1.2 Justificativa

Facilitar a criação e o gerenciamento de conteúdo multimídia com a ajuda de tecnologias modernas. A necessidade de soluções automatizadas que possam lidar com a transcrição e análise de vídeos é clara, dada a crescente importância do conteúdo de vídeo no marketing digital, na educação e em outras indústrias. Ferramentas tradicionais muitas vezes são caras e não conseguem acompanhar a demanda por rapidez e precisão, justificando a criação de uma solução inovadora e acessível.

1.3 Objetivos

1.3.1 Geral

Desenvolver uma aplicação web que facilite a transcrição automática de vídeos e a geração de respostas inteligentes utilizando inteligência artificial.

1.3.2 Específicos

Implementar um backend robusto utilizando Node.js, Fastify e Prisma para gerenciar dados e integrações. Desenvolver um frontend interativo com React para oferecer uma experiência de usuário fluida. Integrar a API do OpenAI para fornecer transcrições precisas e gerar conteúdo automatizado de alta qualidade. Validar a eficácia da solução através de testes e feedback de usuários.

2 FUNDAMENTAÇÃO TEÓRICA

Resumo do capítulo

As tecnologias fundamentais escolhidas para o desenvolvimento da aplicação frontend. São elas: React, TypeScript e FFmpeg. O React foi selecionado devido à sua eficácia na construção de interfaces de usuário dinâmicas e reativas, utilizando uma abordagem componentizada que facilita a manutenção e escalabilidade do projeto, além de ser um framework de uma linguagem de programação que o meu grupo tinha mais familiaridade. O TypeScript foi adotado para garantir a segurança e robustez do código, fornecendo detecção de erros em tempo de compilação e tipagem estática, o que melhora a experiência de desenvolvimento e aumenta a confiabilidade do código (FREEMAN, 2019). Por fim, o FFmpeg foi integrado para possibilitar operações avançadas de processamento de vídeo diretamente no navegador, oferecendo uma ampla gama de funcionalidades para edição, conversão e manipulação de mídia em tempo real. Essas tecnologias foram escolhidas com base em sua maturidade, popularidade e capacidade de atender aos requisitos do projeto, proporcionando uma base sólida para o desenvolvimento de uma aplicação frontend moderna e eficiente para processamento de vídeo.

2.1 Visão Geral

Para entendermos por que as tecnologias escolhidas foram fundamentais para o desenvolvimento do projeto, é essencial analisar cada uma delas individualmente.

2.2 React Framework

React é uma biblioteca JavaScript amplamente utilizada para a construção de interfaces de usuário. segundo (BANKS; PORCELLO, 2017), react permite aos desenvolvedores criar grandes aplicações web que podem ser atualizadas e renderizadas de forma eficiente em resposta às mudanças de dados, com um foco na simplicidade, declaratividade e componibilidade. Sua popularidade se deve à sua abordagem declarativa e componentizada, o que permite o desenvolvimento de interfaces complexas de forma mais organizada e reutilizável. Ao dividir a interface em componentes independentes, o React facilita a manutenção do código e a colaboração entre os membros da equipe de desenvolvimento. Além disso, o uso de Virtual DOM proporciona um alto desempenho, garantindo uma experiência fluida para o usuário final.

2.3 Superset TypeScript

TypeScript é um superset de JavaScript que adiciona tipagem estática ao código. Essa adição de tipos traz diversos benefícios, incluindo detecção de erros em tempo de compilação, melhorias na produtividade do desenvolvedor e aumento da robustez do código. Ao fornecer um sistema de tipos forte e ferramentas de desenvolvimento avançadas, o TypeScript ajuda a evitar bugs comuns e facilita a manutenção e escalabilidade do projeto. Além disso, sua integração perfeita com o ecossistema JavaScript existente permite uma transição suave para equipes que já estão familiarizadas com JavaScript. (HEJLSBERG, 2012)

2.4 FFmpeg framework

FFmpeg é uma biblioteca de software livre e código aberto amplamente utilizada para o processamento de áudio e vídeo. Sua versatilidade e poder tornam-no uma escolha popular para aplicações de processamento de vídeo em diversas plataformas. FFmpeg é uma ferramenta essencial para trabalhar com áudio e vídeo, com sua interface de linha de comando e vasta gama de funcionalidades, tornando-se indispensável para profissionais de vídeo e entusiastas (SHARKEY, 2018). A integração de FFmpeg com o projeto permite a execução de operações avançadas de manipulação de mídia diretamente no navegador, ampliando significativamente as possibilidades de interação e personalização para o usuário final. Com suporte para uma ampla gama de formatos e codecs, o FFmpeg oferece flexibilidade e confiabilidade no processamento de vídeo, contribuindo para a robustez e eficiência da aplicação desenvolvida.

2.4.1 Por que essas tecnologias foram escolhidas?

2.4.1.1 React

A escolha do React se deve à sua popularidade e eficácia na construção de interfaces de usuário dinâmicas e reativas. Sua arquitetura componentizada permite uma fácil manutenção e escalabilidade do projeto, enquanto sua performance otimizada garante uma experiência de usuário fluida.

2.4.1.2 TypeScript

O TypeScript foi adotado para garantir a segurança e a robustez do código, proporcionando uma melhor experiência de desenvolvimento através de ferramentas avançadas de detecção de erros e tipagem estática. Sua interoperabilidade com JavaScript facilita a

integração em projetos existentes e aumenta a confiabilidade do código.

2.4.1.3 FFmpeg

A integração de FFmpeg traz um conjunto poderoso de ferramentas para o processamento de vídeo, permitindo operações avançadas de edição, conversão e manipulação diretamente no navegador. Sua ampla compatibilidade e desempenho eficiente tornam-no uma escolha ideal para aplicações que exigem manipulação de mídia em tempo real.

Tabela 2.1 – *Comparação das Tecnologias Utilizadas no Projeto*

Critério	**React**	**TypeScript**	**FFmpeg**
Popularidade	Uma das bibliotecas mais populares para o desenvolvimento de interfaces de usuário, com uma grande comunidade de desenvolvedores e amplo suporte.	Crescimento significativo de adoção, impulsionado pela demanda por uma tipagem mais robusta e segura em projetos JavaScript.	Amplamente reconhecido como uma ferramenta essencial para processamento de áudio e vídeo, utilizado em diversas indústrias e aplicações.
Flexibilidade	Arquitetura componentizada permite a construção de interfaces complexas e reativas, com grande flexibilidade e reusabilidade de código.	Integra-se perfeitamente ao ecossistema JavaScript existente, permitindo uma transição suave para equipes já familiarizadas com JavaScript.	Oferece uma ampla variedade de funcionalidades para manipulação de mídia, incluindo conversão de formatos, edição e aplicação de efeitos.
Desempenho	Utiliza Virtual DOM para otimizar a atualização da interface, garantindo um desempenho eficiente mesmo em aplicações de grande escala.	Oferece um sistema de tipos estáticos que ajuda a identificar e corrigir erros em tempo de compilação, reduzindo a ocorrência de bugs e melhorando a performance do código.	Apresenta um desempenho robusto e eficiente, mesmo ao lidar com grandes arquivos de mídia e operações de processamento intensivas.
Compatibilidade	Compatível com uma ampla variedade de navegadores e ambientes de desenvolvimento, garantindo uma experiência consistente para os usuários.	Integra-se facilmente a frameworks e bibliotecas JavaScript populares.	Suporta uma ampla gama de formatos e codecs de áudio e vídeo, garantindo compatibilidade com diferentes tipos de mídia e sistemas operacionais.

Fonte: *(FREECODECAMP, 2012)* - Artigo sobre as vantagens do React: "Why React is So Popular for Frontend Development"

3 METODOLOGIA

Resumo do capítulo

Neste capítulo, descrevemos a abordagem metodológica adotada para o desenvolvimento do projeto, incluindo as tecnologias utilizadas, a estrutura do projeto e as ferramentas de desenvolvimento empregadas. A metodologia seguida neste projeto inclui a análise de requisitos, escolha das tecnologias, desenvolvimento incremental dos componentes, e testes contínuos para garantir a funcionalidade e performance da aplicação.

3.1 Desenvolvimento

A aplicação foi desenvolvida utilizando uma abordagem modular, onde cada componente foi implementado e testado de forma independente antes de ser integrado ao projeto principal.

3.2 Visão Geral

Fornecer uma estrutura sólida para o desenvolvimento do projeto, delineando os princípios, ferramentas e processos-chave que serão seguidos para alcançar os objetivos estabelecidos.

3.3 Testes

Os testes foram conduzidos para garantir que todas as funcionalidades da aplicação funcionassem conforme o esperado. Foram realizados testes unitários nos componentes individuais e testes de integração na aplicação como um todo.

3.4 Backend

O backend, também conhecido como servidor ou camada de servidor, é a parte de um sistema de software que lida com o processamento dos dados, a lógica de negócios e a interação com o banco de dados. Ele é responsável por receber requisições do frontend (interface do usuário) e retornar as respostas apropriadas.

Em uma arquitetura de software típica, o frontend é a parte da aplicação com a qual o usuário interage diretamente, enquanto o backend é responsável por fornecer os recursos e funcionalidades necessários para que o frontend funcione corretamente.

3.4.1 NodeJS

Escolhido devido à sua eficiência na construção de aplicativos escaláveis e de alto desempenho.
Node.js é uma plataforma construída sobre o motor JavaScript do Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Muitos de seus módulos básicos são escritos em JavaScript, e os desenvolvedores podem escrever novos módulos em JavaScript. A plataforma pretende tornar o desenvolvimento de aplicações web escaláveis e eficientes, permitindo que os desenvolvedores usem JavaScript tanto no lado do cliente quanto no lado do servidor. (DAHL, 2009)

3.4.2 Fastify

Segundo (WALT M., 2020) o Fastify é um framework web extremamente rápido e eficiente para Node.js. Ele é especialmente conhecido por sua velocidade e baixa sobrecarga, tornando-o uma escolha popular para o desenvolvimento de aplicativos web e APIs de alto desempenho. Algumas de suas principais características incluem:

1. Desempenho: O Fastify é otimizado para velocidade, oferecendo tempos de resposta rápidos mesmo sob cargas pesadas.

2. Baixa Sobrecarga: O framework é projetado para ter uma sobrecarga mínima, o que significa que consome poucos recursos do sistema.

3. Ecossistema de Plugins: O Fastify possui um ecossistema robusto de plugins que facilitam a adição de funcionalidades extras ao aplicativo.

4. Suporte a Validação de Dados: Ele possui suporte embutido para validação de dados, permitindo garantir a integridade e segurança dos dados recebidos pelo servidor.

5. Documentação Abundante: O Fastify oferece uma documentação completa e fácil de seguir, facilitando o aprendizado e a utilização do framework.

3.4.3 PrismaORM

Prisma: Framework ORM (Object-Relational Mapping) utilizado para facilitar o acesso e manipulação de dados no banco de dados.
Prisma é um ORM moderno e de código aberto para Node.js e TypeScript. Ele simplifica o acesso a bancos de dados SQL com segurança e desempenho. Com Prisma, os desenvolvedores podem modelar os dados de forma segura e acessá-los com facilidade usando uma API do tipo idioma. (DOCS, 2024)

3.4.4 OpenAI API

Integrada para fornecer recursos avançados de processamento de linguagem natural. Segundo a documentação oficial (OPENAI, 2020), OpenAI API é uma interface de programação de aplicativos que permite que os desenvolvedores usem os modelos de linguagem de ponta da OpenAI em seus próprios aplicativos. Os modelos de linguagem são treinados em uma ampla gama de textos para gerar respostas coerentes e úteis a uma variedade de solicitações de texto.

3.5 Frontend

O frontend é a parte de um sistema de software que os usuários interagem diretamente. Também é conhecido como a interface do usuário (UI). Ele consiste em todos os elementos visíveis e interativos de um aplicativo ou site, incluindo botões, menus, formulários, gráficos, entre outros.

As principais responsabilidades do frontend incluem:

1. Apresentação de Conteúdo: O frontend é responsável por exibir informações e conteúdo de maneira clara e organizada para os usuários.

2. Interatividade: Ele fornece meios para os usuários interagirem com o sistema, como clicar em botões, preencher formulários e navegar por diferentes páginas.

3. Usabilidade: O frontend é projetado para ser intuitivo e fácil de usar, proporcionando uma experiência agradável ao usuário.

4. Estilo e Design: Ele trata do layout, estilo e design visual do aplicativo ou site, garantindo uma aparência atraente e coesa.

5. Performance: O frontend também é responsável por garantir uma resposta rápida e eficiente às interações do usuário, minimizando os tempos de carregamento e proporcionando uma experiência fluida.

3.5.1 React

Framework JavaScript amplamente utilizado para criar interfaces de usuário dinâmicas e reativas. React é uma biblioteca JavaScript de código aberto para criar interfaces de usuário ou componentes de IU. É mantido pelo Facebook e uma comunidade de desenvolvedores individuais e empresas. React pode ser usado como uma base no desenvolvimento de aplicativos de página única ou móveis. (DOCS, 2019)

3.5.2 Axios

Biblioteca para fazer requisições HTTP do lado do cliente, proporcionando uma comunicação eficiente entre o frontend e o backend. Axios é uma biblioteca popular para fazer solicitações HTTP no navegador e no Node.js. Ele fornece uma interface fácil de usar para solicitações HTTP, manipulando automaticamente solicitações e respostas em formatos JSON.

3.5.3 FFmpeg

Utilizado para operações avançadas de processamento de vídeo diretamente no navegador. FFmpeg é uma biblioteca de software livre e código aberto amplamente utilizada para o processamento de áudio e vídeo. Sua versatilidade e poder tornam-no uma escolha popular para aplicações de processamento de vídeo em diversas plataformas.

Tabela 3.1 – *A tabela a seguir apresenta um comparativo entre as principais bibliotecas de processamento de vídeo disponíveis para aplicações web.*

Biblioteca	**Funcionalidades**	**Vantagens**	**Desvantagens**
FFmpeg	Conversão, edição	Ampla compatibilidade	Complexidade de uso
Video.js	Reprodução, plugins	Fácil de integrar	Menos focada em edição
HandBrake.js	Compressão, conversão	Interface amigável	Funcionalidades limitadas

Fonte: *(SMITH, 2019)* - HandBrake: Open Source Video Transcoder"

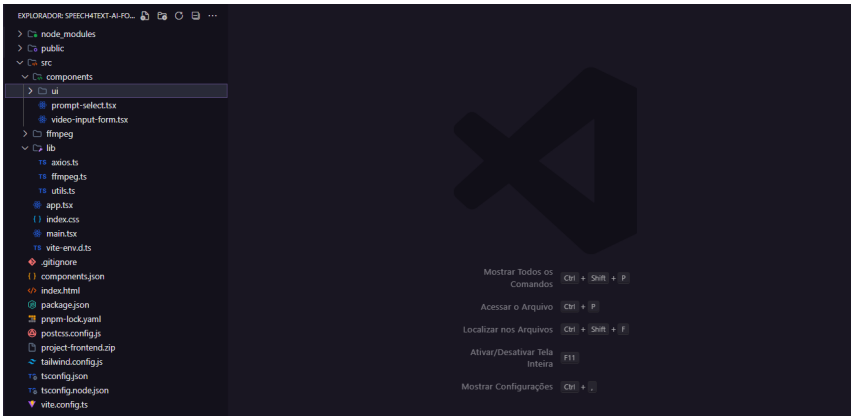
4 FRONTEND FILES

Resumo do capítulo

Neste capitulo será apresentado todos os arquivos essenciais do projeto frontend organizados e descritos. Este capítulo é crucial para a compreensão da estrutura do projeto, pois ele detalha os diferentes tipos de arquivos que compõem a interface do usuário, desde imagens e estilos até scripts e componentes.

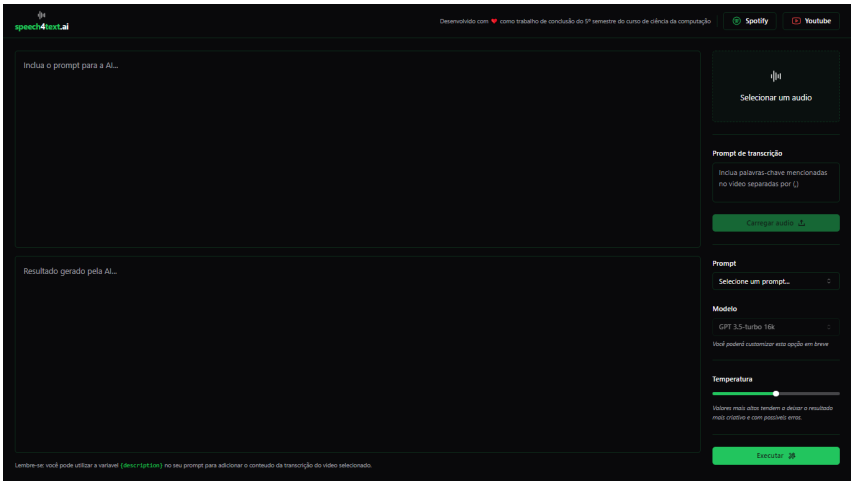
4.1 Estrutura

Figura 4.1 – *Estrutura de pastas do Frontend*



Fonte: PrintScreen: Izael Silva

Figura 4.2 – *Tela inicial do projeto na web*



Fonte: PrintScreen: Izael Silva

4.2 Código fonte dos principais arquivos

4.2.1 index.html

```
<!doctype html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale
        =1.0" />
    <title>speech4text.ai</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

4.2.2 main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import './index.css'
import { App } from './app'

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

4.2.3 app.tsx

```
import { Button } from "@/components/ui/button";
import { SpotifyLogo, YoutubeLogo, Waveform, MagicWand } from "
    @phosphor-icons/react";
import { Separator } from "./components/ui/separator";
import { Textarea } from "./components/ui/textarea";
import { Select, SelectContent, SelectItem, SelectTrigger,
    SelectValue } from "./components/ui/select";
import { Slider } from "./components/ui/slider";
import { Label } from "./components/ui/label";
import { VideoInputForm } from "./components/video-input-form";
```

```
import { PromptSelect } from "./components/prompt-select";
import { useState } from "react";
import { useCompletion } from 'ai/react';

export function App() {
  const [temperature, setTemperature] = useState(0.5);
  const [videoId, setVideoId] = useState<string | null>(null);

  const { input, setInput, handleInputChange, handleSubmit,
      completion, isLoading } = useCompletion({
    api: 'http://localhost:3333/ai/complete',
    body: {
      videoId,
      temperature
    },
    headers: {
      "Content-Type": "application/json"
    }
  });

  return (
    <div className="min-h-screen flex flex-col">
      <div className="px-6 py-3 flex items-center justify-between
          border-b border-green-900/50">
        <div className="flex flex-col items-center justify-center">
          <Waveform className="size-5" />
          <h1 className="text-base font-bold">
            <span className="text-green-500">speech</span>4<span
                className="text-green-500">text</span>.ai
          </h1>
        </div>
        <div className="flex items-center gap-3">
          <span className="text-xs text-muted-foreground">
            Desenvolvido com     como trabalho de conclus o do 5
                semestre do curso de ci ncia da computa  o
          </span>
          <Separator orientation="vertical" className="h-6" />
          <Button variant={"outline"}>
            <SpotifyLogo className="size-5 mr-2 text-green-500" />
            Spotify
          </Button>
          <Button variant={"outline"}>
            <YoutubeLogo className="size-5 mr-2 text-red-500" />
            Youtube
          </Button>
        </div>
      </div>
```

```
52
53        <main className="flex-1 p-6 flex gap-6">
54          <div className="flex flex-col flex-1 gap-4">
55            <div className="grid grid-rows-2 gap-4 flex-1">
56              <Textarea
57                placeholder="Inclua o prompt para a AI..."
58                className="resize-none p-4 leading-relaxed text-base"
59                value={input}
60                onChange={handleInputChange}
61              />
62              <Textarea
63                readOnly
64                placeholder="Resultado gerado pela AI..."
65                className="resize-none p-4 leading-relaxed text-base"
66                value={completion}
67              />
68            </div>
69            <p className="text-xs text-muted-foreground">
70              Lembre-se: voc  pode utilizar a vari vel <code
                    className="text-green-500">{'{description}'}</code>
                    no seu prompt para adicionar o conte do da
                    transcri  o do v deo selecionado.
71            </p>
72          </div>
73
74          <aside className="w-64 space-y-6">
75            <VideoInputForm onVideoUploaded={setVideoId} />
76
77            <Separator />
78
79            <div className="space-y-2">
80              <Label>Prompt</Label>
81              <PromptSelect onPromptSelected={setInput} />
82            </div>
83
84            <form onSubmit={handleSubmit} className="space-y-6">
85              <div className="space-y-2">
86                <Label>Modelo</Label>
87                <Select defaultValue="gpt-3.5" disabled>
88                  <SelectTrigger>
89                    <SelectValue />
90                  </SelectTrigger>
91                  <SelectContent>
92                    <SelectItem value="gpt-3.5">GPT 3.5-turbo 16k</
                          SelectItem>
93                  </SelectContent>
94                </Select>
```

```
95                <span className="block text-xs text-muted-foreground
                      italic">
96                  Voc  poder  customizar esta op  o em breve
97                </span>
98              </div>
99
100             <Separator />
101
102             <div className="space-y-4">
103               <Label>Temperatura</Label>
104               <Slider
105                 className="bg-zinc-900"
106                 min={0}
107                 max={1}
108                 step={0.1}
109                 value={[temperature]}
110                 onValueChange={vl => setTemperature(vl[0])}
111               />
112               <span className="block text-xs text-muted-foreground
                      italic leading-relaxed">
113                 Valores mais altos tendem a deixar o resultado mais
                        criativo e com poss veis erros.
114               </span>
115             </div>
116
117             <Separator />
118
119             <Button disabled={isLoading} type="submit" className="w-
                    full">
120               Executar
121               <MagicWand className="size-4 ml-2" weight="bold" />
122             </Button>
123           </form>
124         </aside>
125       </main>
126     </div>
127   );
128 }
```

4.2.4 axios.ts

```
1 import axios from 'axios'
2 export const api = axios.create({
3     baseURL: 'http://localhost:3333'
4 })
```

4.2.5 ffmpeg.ts

```
import { FFmpeg } from '@ffmpeg/ffmpeg'

import coreURL from '../ffmpeg/ffmpeg-core.js?url'
import wasmURL from '../ffmpeg/ffmpeg-core.wasm?url'
import workerURL from '../ffmpeg/ffmpeg-worker.js?url'

let ffmpeg: FFmpeg | null

export async function getFFmpeg() {
  if (ffmpeg) {
    return ffmpeg
  }

  ffmpeg = new FFmpeg()

  if (!ffmpeg.loaded) {
    await ffmpeg.load({
      coreURL,
      wasmURL,
      workerURL
    })
  }

  return ffmpeg
}
```

4.2.6 util.ts

```
import { type ClassValue, clsx } from "clsx"
import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}
```

4.2.7 prompt-select.tsx

```
import { useEffect, useState } from "react";
import { Select, SelectContent, SelectItem, SelectTrigger,
    SelectValue } from "./ui/select";
import { api } from "@/lib/axios";

```

```
interface Prompts {
  id: string
  title: string
  template: string
}

interface PromptSelectProps {
  onPromptSelected: (template: string) => void
}

export function PromptSelect({ onPromptSelected }: PromptSelectProps)
    {
  const [prompts, setPrompts] = useState<Prompts[] | null>(null)

  useEffect(() => {
    api.get('/prompts').then(response => {
      setPrompts(response.data)
    })
  }, [])

  function handlePromptSelected(promptId: string) {
    const selectedPrompt = prompts?.find(prompt => prompt.id ===
        promptId)

    if (!selectedPrompt) {
      return
    }

    onPromptSelected(selectedPrompt.template)
  }

  return (
    <Select onValueChange={handlePromptSelected}>
      <SelectTrigger>
        <SelectValue placeholder="Selecione um prompt..." />
      </SelectTrigger>
      <SelectContent>
        {prompts?.map(prompt => {
          return (
            <SelectItem key={prompt.id} value={prompt.id}>
              {prompt.title}
            </SelectItem>
          )
        })}
      </SelectContent>
    </Select>
  )
```

```
50 }
```

4.2.8 video-input-form.tsx

```
1  import { UploadSimple, Waveform } from "@phosphor-icons/react";
2  import { Separator } from "./ui/separator";
3  import { Label } from "./ui/label";
4  import { Textarea } from "./ui/textarea";
5  import { Button } from "./ui/button";
6  import { ChangeEvent, FormEvent, useMemo, useRef, useState } from "
       react";
7  import { getFFmpeg } from "@/lib/ffmpeg";
8  import { fetchFile } from "@ffmpeg/util";
9  import { api } from "@/lib/axios";
10
11 type Status = 'waiting' | 'converting' | 'uploading' | 'generating' |
       'success'
12
13 const statusMessages = {
14   converting: 'Convertendo...',
15   uploading: 'Fazendo Upload...',
16   generating: 'Gerando Transcri  o...',
17   success: 'Sucesso!',
18 }
19
20 interface VideoInputFormProps {
21   onVideoUploaded: (id: string) => void
22 }
23
24 export function VideoInputForm({ onVideoUploaded }:
       VideoInputFormProps) {
25   const [videoFile, setVideoFile] = useState<File | null>(null)
26   const [status, setStatus] = useState<Status>('waiting')
27
28   const promptInputRef = useRef<HTMLTextAreaElement>(null)
29
30   function handleFileSelected(event: ChangeEvent<HTMLInputElement>) {
31     const { files } = event.currentTarget
32     if (!files) {
33       return
34     }
35     const selectedFile = files[0]
36     setVideoFile(selectedFile)
37   }
38
39   async function convertVideoToAudio(video: File) {
```

```
40      console.log('Converte started!')
41
42      const ffmpeg = await getFFmpeg()
43      await ffmpeg.writeFile('input.mp4', await fetchFile(video))
44
45      // ffmpeg.on('log', log => {
46      //   console.log(log)
47      // })
48
49      ffmpeg.on('progress', progress => {
50        console.log('Convert progress: ' + Math.round(progress.progress
              * 100))
51      })
52
53      await ffmpeg.exec([
54        '-i',
55        'input.mp4',
56        '-map',
57        '0:a',
58        '-b:a',
59        '20k',
60        '-acodec',
61        'libmp3lame',
62        'output.mp3'
63      ])
64
65      const data = await ffmpeg.readFile('output.mp3')
66
67      const audioFileBlob = new Blob([data], { type: 'audio/mpeg' })
68      const audioFile = new File([audioFileBlob], 'audio.mp3', {
69        type: 'audio/mpeg'
70      })
71
72      console.log('Convert finished!')
73
74      return audioFile
75    }
76
77    async function handleUploadVideo(event: FormEvent<HTMLFormElement>)
          {
78      event.preventDefault()
79      const prompt = promptInputRef.current?.value
80
81      if (!videoFile) {
82        return
83      }
84
```

```
85      setStatus('converting')
86
87      const audioFile = await convertVideoToAudio(videoFile)
88
89      const data = new FormData()
90      data.append('file', audioFile)
91
92      setStatus('uploading')
93
94      const response = await api.post('/videos', data)
95      const videoId = response.data.video.id
96
97      setStatus('generating')
98
99      await api.post(`/videos/${videoId}/transcription`, { prompt })
100
101     setStatus('success')
102
103     onVideoUploaded(videoId)
104   }
105
106   const previewURL = useMemo(() => {
107     if (!videoFile) {
108       return null
109     }
110     return URL.createObjectURL(videoFile)
111   }, [videoFile])
112
113   return (
114     <form onSubmit={handleUploadVideo} className="space-y-6">
115       <label htmlFor="video" className=" relative h-full flex flex-
              col items-center justify-center gap-4 bg-green-900/10 hover
              :bg-green-900/20 transition-colors border border-dashed
              border-green-900/50 rounded-md cursor-pointer aspect-video
              overflow-hidden">
116         {previewURL ? (
117           <video src={previewURL} controls={false} className="pointer
                -events-none absolute inset-0" />
118         ) : (
119           <>
120             <Waveform className="size-7" />
121             Selecionar um audio
122           </>
123         )}
124       </label>
125       <input type="file" id="video" accept="video/mp4" className="sr-
              only" onChange={handleFileSelected} />
```

```
126
127         <Separator />
128
129         <div className="space-y-2">
130           <Label htmlFor="transcription_prompt">Prompt de transcri  o
                </Label>
131           <Textarea
132             ref={promptInputRef}
133             disabled={status !== 'waiting'}
134             id="transcription_prompt"
135             className="min-h-20 leading-relaxed resize-none"
136             placeholder="Inclua palavras-chave mencionadas no video
                   separadas por (,)"
137           />
138         </div>
139
140         <Button
141           type="submit"
142           disabled={status !== 'waiting' || !videoFile}
143           data-success={status === 'success'}
144           className="w-full data-[success=true]:bg-green-700 select-
              none"
145         >
146           {status === 'waiting' ? (
147             <>
148               Carregar audio
149               <UploadSimple className="size-4 ml-2" weight="bold" />
150             </>
151           ) : statusMessages[status]}
152         </Button>
153       </form>
154     )
155 }
```

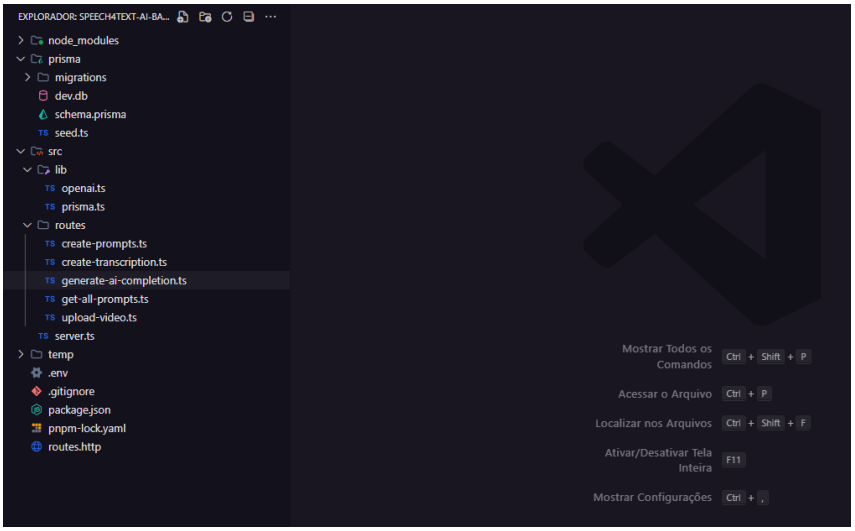
5 BACKEND FILES

Resumo do capítulo

Neste capitulo será apresentado todos os arquivos essenciais do projeto backend organizados e descritos. Este capítulo é crucial para a compreensão da estrutura do projeto, pois ele detalha os diferentes tipos de arquivos que compõem a interface do usuário, desde imagens e estilos até scripts e componentes.

5.1 Estrutura

Figura 5.1 – *Estrutura de pastas do Backend*



Fonte: PrintScreen: Izael Silva

5.2 Código fonte dos principais arquivos

5.2.1 Pasta ./prisma

5.2.1.1 schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "sqlite"
  url      = env("DATABASE_URL")
}
```

```
9
10 model Video {
11   id             String   @id @default(uuid())
12   name           String
13   path           String
14   transcription  String?
15   createdAt      DateTime @default(now())
16 }
17
18 model Prompt {
19   id       String @id @default(uuid())
20   title    String
21   template String
22 }
```

5.2.2 Pasta raiz ./src

5.2.2.1 server.ts

```
1 import { fastify } from 'fastify'
2 import fastifyCors from '@fastify/cors'
3 import { createPrompts } from './routes/create-prompts'
4 import { uploadVideoRoute } from './routes/upload-video'
5 import { getAllPromptsRoute } from './routes/get-all-prompts'
6 import { createTranscriptionRoute } from './routes/create-
    transcription'
7 import { generateAiCompletionRoute } from './routes/generate-ai-
    completion'
8
9 const app = fastify()
10
11 app.register(fastifyCors, {
12   origin: '*'
13 })
14
15 app.register(createPrompts)
16 app.register(uploadVideoRoute)
17 app.register(getAllPromptsRoute)
18 app.register(createTranscriptionRoute)
19 app.register(generateAiCompletionRoute)
20
21 app.listen(
22   { port: 3333 }
23 ).then(() => {
24   console.log('HTTP Server running on host: http://localhost:3333')
```

```
25 })
```

5.2.2.2 Sub pasta /routes

upload-video.ts

```
1  import { FastifyInstance } from "fastify";
2  import { fastifyMultipart } from '@fastify/multipart'
3  import path from "node:path";
4  import { randomUUID } from "node:crypto";
5  import { promisify } from "node:util";
6  import { pipeline } from "node:stream";
7  import fs from "node:fs";
8  import { prisma } from "../lib/prisma";
9
10 const pump = promisify(pipeline)
11
12 export async function uploadVideoRoute(app: FastifyInstance) {
13   app.register(fastifyMultipart, {
14     limits: {
15       fileSize: 1_048_576 * 25 //25mb
16     }
17   })
18
19   app.post('/videos', async (request, replay) => {
20     const data = await request.file()
21
22     if (!data) {
23       return replay.status(400).send({ error: 'Missing file input.'
           })
24     }
25
26     const extension = path.extname(data.filename)
27
28
29     if (extension !== '.mp3') {
30       return replay.status(400).send({ error: 'Invalid input type,
           please upload a MP3' })
31     }
32
33     const fileBaseName = path.basename(data.filename, extension)
34     const fileUploadName = `${fileBaseName}-${randomUUID()}${
         extension}`
35     const uploadDestination = path.resolve(__dirname, '../../temp',
         fileUploadName)
36
```

```
37      await pump(data.file, fs.createWriteStream(uploadDestination))
38
39      const video = await prisma.video.create({
40        data: {
41          name: data.filename,
42          path: uploadDestination
43        }
44      })
45
46      return { video }
47    })
48  }
```

get-all-prompts.ts

```
1  import { FastifyInstance } from "fastify";
2  import { prisma } from "../lib/prisma";
3
4  export async function getAllPromptsRoute(app: FastifyInstance) {
5    app.get('/prompts', async () => {
6      const prompts = await prisma.prompt.findMany()
7      return prompts
8    })
9  }
```

generate-ai-completion.ts

```
1  import { FastifyInstance } from 'fastify'
2  import { z } from 'zod';
3  import { prisma } from '../lib/prisma'
4  import { openai } from '../lib/openai'
5  import { streamToResponse, OpenAIStream } from 'ai'
6
7  export async function generateAiCompletionRoute(app: FastifyInstance)
       {
8    app.post('/ai/complete', async (req, reply) => {
9      const bodySchema = z.object({
10       videoId: z.string().uuid(),
11       prompt: z.string(),
12       temperature: z.number().min(0).max(1).default(0.5)
13     })
14
15     const { videoId, prompt, temperature } = bodySchema.parse(req.
           body)
16
17     const video = await prisma.video.findUniqueOrThrow({
18       where: {
19         id: videoId
20       }
```

```
    })

    if (!video.transcription) {
      return reply.status(400).send({ error: 'Video transcription was not generated yet.' })
    }

    const promptMessage = prompt.replace('{transcription}', video.transcription)

    // console.log(promptMessage)

    const response = await openai.chat.completions.create({
      model: "gpt-3.5-turbo-16k",
      temperature,
      messages: [{ role: 'user', content: promptMessage }],
      stream: true
    })

    const stream = OpenAIStream(response)

    streamToResponse(stream, reply.raw, {
      headers: {
        'Access-Control-Allow-Origin': '*',
        'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE, OPTIONS'
      }
    })
  })
}
```

create-transcription.ts

```
import { FastifyInstance } from 'fastify'
import { z } from 'zod';
import { prisma } from '../lib/prisma'
import { createReadStream } from 'node:fs' // Alterado de 'node:fs' para 'fs'
import { openai } from '../lib/openai'

export async function createTranscriptionRoute(app: FastifyInstance) {
  app.post('/videos/:videoId/transcription', async (request, reply) => { // Corrigido 'replay' para 'reply'
    const paramsSchema = z.object({
      videoId: z.string().uuid()
    })

```

```
    const { videoId } = paramsSchema.parse(request.params);

    const bodySchema = z.object({
      prompt: z.string()
    })
    const { prompt } = bodySchema.parse(request.body);

    const video = await prisma.video.findUniqueOrThrow({
      where: {
        id: videoId
      }
    })

    const videoPath = video.path;
    const audioReadStream = createReadStream(videoPath);

    try {
      const response = await openai.audio.transcriptions.create({
        file: audioReadStream,
        model: "whisper-1",
        language: 'pt',
        response_format: 'json',
        temperature: 0,
        prompt
      })

      const transcription = response.text

      await prisma.video.update({
        where: {
          id: videoId,
        },
        data: {
          transcription
        }
      })

      return { transcription }

    } catch (error) {
      return reply.status(400).send({ message: error })
    }
  })
}
```

create-prompts.ts

```
import { FastifyInstance } from 'fastify'
```

```
import { z } from 'zod'
import { prisma } from '../lib/prisma'

export async function createPrompts(app: FastifyInstance) {
  app.post('/prompts', async (request, replay) => {
    const bodySchema = z.object({
      title: z.string(),
      template: z.string()
    })
    const { title, template } = bodySchema.parse(request.body)

    const prompt = await prisma.prompt.create({
      data: {
        title,
        template
      }
    })

    return prompt
  })
}
```

5.2.2.3 Libs

prisma.ts

```
import { PrismaClient } from "@prisma/client"
export const prisma = new PrismaClient()
```

openai.ts

```
import { OpenAI } from "openai";
export const openai = new OpenAI({
  apiKey: process.env.OPENAI_KEY
})
```

hyperref

6 ANÁLISE DOS RESULTADOS

Os resultados do desenvolvimento indicam que a aplicação é capaz de processar vídeos no formato .mp4, com eficiência. O tempo de conversão e a qualidade dos vídeos processados foram satisfatórios, atendendo aos requisitos definidos. Nosso próximos passos é elevareste produto a outros patamares, até porque no momento o softwre só processa videos no formato .mp4, queremos diversificar, por exemplos para outros formatos, inclusive .mp3.

6.1 Avaliação

A aplicação mostrou-se eficiente, porém foram identificados alguns desafios, como a limitação de performance ao processar vídeos muito grandes diretamente no navegador.

6.2 Comparação

Os resultados obtidos estão alinhados com os objetivos propostos. A aplicação oferece uma interface intuitiva e funcional, permitindo o processamento de vídeos no navegador de forma eficiente.

6.3 Github do projeto

Conferir código do projeto no GitHub

7 CONCLUSÕES

7.1 Conclusões

O projeto atingiu seus objetivos principais, demonstrando que é possível criar uma aplicação web eficiente para processamento de vídeos. A integração de FFmpeg com React e TypeScript mostrou-se uma solução viável e eficaz.

7.2 Contribuições

Este trabalho contribui para a área de desenvolvimento web, mostrando como tecnologias modernas podem ser utilizadas para resolver problemas complexos de forma eficiente.

7.3 Trabalhos Futuros

Sugestões para trabalhos futuros incluem a otimização de performance para processamento de vídeos maiores e a adição de mais funcionalidades, como edição avançada e filtros de vídeo.

7.4 Curiosidade

Quando a faculdade, mais precisamente o professor do nosso projeto, nos deu as diversas opções, para realizar o trabalho, eu como lider do grupo propus fazer esta aplicação para suprir uma necessidade real que eu enfrentava.

Na igreja que eu frequento, eu gravo o audio dos ensinos dos preletores, para eu disponiblizar em podcast e em um blog da igreja.

Só que era muito trabalhoso, depois da gravação transcrever o audio e fazer um resumo, criar um titulo etc. e eu fazia isso tudo manual. Eu sei que há muitas ferramentas que automatiza isso, mas são todas pagas e muitas nem são baratas.

Pensando em tudo isso e juntando o util ao agradável, desenvolvemos esta aplicação que eu vou usar para me ajudar a pegar o audio que eu vou gravar, transcrever o oudio ou video, usando a API da OpenAi, e com esta transcrição a propria OpenAi gera os resumos e os titulos.

E no próprio software tem botãoes para postar tanto no Spotify em formato de audio, como no Youtube como video.

REFERÊNCIAS BIBLIOGRÁFICAS

BANKS, A.; PORCELLO, E. *Learning React: Functional Web Development with React and Redux*. [S.l.]: O'Reilly Media, 2017. Citado na pág. 11.

DAHL, R. Node.js: Event-driven i/o server-side javascript environment. 2009. Disponível em: https://nodejs.org/en/about/. Citado na pág. 15.

DOCS, P. Prisma. 2024. Disponível em: https://www.prisma.io/docs/. Citado na pág. 15.

DOCS, R. React. 2019. Disponível em: https://reactjs.org/docs/getting-started.html. Citado na pág. 16.

FREECODECAMP. Artigo sobre as vantagens do react: "why react is so popular for frontend development. 2012. Disponível em: https://www.freecodecamp.org/news/why-react-is-so-popular-for-frontend-development/. Citado na pág. 13.

FREEMAN, A. ***Pro TypeScript: Application-Scale JavaScript Development***. [S.l.]: Apress, 2019. Citado na pág. 11.

HEJLSBERG, A. Introducing typescript. 2012. Disponível em: https://devblogs.microsoft.com/typescript/announcing-typescript-0-8-1-release/. Citado na pág. 12.

OPENAI. Openai. 2020. Disponível em: https://openai.com/api/. Citado na pág. 16.

SHARKEY, C. *Mastering FFmpeg: Everything you need to know about audio and video conversion with FFmpeg*. [S.l.]: Packt Publishing, 2018. Citado na pág. 12.

SMITH, J. Handbrake: Open source video transcoder. *Journal of Open Source Software*, 2019. Citado na pág. 17.

WALT M., . O. S. V. D. *Learning Fastify: Build robust and lightning-fast web applications with Node.js and Fastify*. [S.l.]: Packt Publishing, 2020. Citado na pág. 15.