INF1629_2020.2/front-end/public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>Term Frequency Calculator</title>
```

Este trecho do código HTML insere e torna disponível a fonte Roboto da GoogleFonts.

PRÉ: fonte padrão.

PÓS: disponibiliza uma nova fonte, a fonte Roboto da GoogleFonts.

```
    <!-- Roboto font from GoogleFonts -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;400;500;700&display=swap"
rel="stylesheet">
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.
      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.
      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

INF1629_2020.2/front-end/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';
```

Este código atualiza a UI, criando o elemento React.StrictMode e passando ele pelo ReactDOM.render, que controla o conteúdo do container.
PRÉ: conteúdo do container pré-existente.
PÓS: atualiza o conteúdo do container com o novo elemento criado.

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

INF1629_2020.2/front-end/src/App.js

```
import React from 'react'
import  {Route, BrowserRouter as Router} from 'react-router-dom'

import Header from './components/Header'
import { GlobalStyles } from "./styles/GlobalStyles"

import Home from './pages/Home';
import Result from './pages/Result';
```

Este código é arquivo principal, atualmente está definindo as páginas (Routes) da aplicação.

PRÉ: definição padrão.

PÓS: definição específica para a aplicação Term Frequency Calculator.

```
export default function App() {
  return (
    <>
      <Header />
      <Router>
        <Route path="/" exact component={Home} />
        <Route path="/result" exact component={Result} />
      </Router>
      <GlobalStyles />
    </>
  )
}
```

INF1629_2020.2/front-end/src/styles/GlobalStyles.js

```
import { createGlobalStyle } from 'styled-components';
```

Este código CSS define a constante de estilo GlobalStyles, com margem 0, padding 0, box-sizing border-box (padding e borda estão incluídos na largura e na altura) e para todos os elementos, input e botão, font-family será utilizada a Roboto (importada da biblioteca da Google no arquivo html "INF1629_2020.2/front-end/public/index.html") e o body como background na cor #383838 cor #fff, que será utilizada para melhorar a estética dos elementos da aplicação.
PRÉ: estilo padrão.
PÓS: definição de estilos para as constantes Container, todos elementos, input, botão e body.

```
export const GlobalStyles = createGlobalStyle`
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  *, input, button {
    font-family: 'Roboto', sans-serif;
  }

  body {
    background-color: #383838;
    color: #fff;
  }
`
```

INF1629_2020.2/front-end/src/services/api.js

```
import axios from 'axios'
```

Este código define a base URL da aplicação como “http://localhost:3333” e a retorna.

PRÉ: base URL padrão.

PÓS: retorna uma base definida como “http://localhost:3333”.

```
const api = axios.create({
  baseURL: 'http://localhost:3333',
})

export default api
```

INF1629_2020.2/front-end/src/pages/Home/index.js

```
import React, { Component } from 'react'
import  { Link } from 'react-router-dom'

import api from '../../services/api'

import UploadContainer from '../../components/UploadContainer'
import Button from '../../components/Button'

import { Container, Disclaimer } from './styles'
```

Este código cria os componentes necessários para a página Home da aplicação Term Frequency Calculator.
Inicialmente são armazenadas as variáveis com valores padrões.
PRÉ: página padrão.
PÓS: página Home com os componentes essenciais à página Home da aplicação Term Frequency.

```
export default class Home extends Component {

  state = {
    document: null,
    stopWords: null,
    progress: 0,
  }
```

Atualiza o document com o selectedFile (arquivo contendo o documento para a contagem do Term Frequency) e imprime o state que contém o documento, as stopwords e o progresso.

```
  handleDocumentUpload = files => {
    const selectedFile = files[0]
    this.setState({
      document: selectedFile,
      stopWords: this.state.stopWords,
      progress: this.state.progress,
    })
    console.log(this.state)
  }
```

Atualiza o stopwords com o selectedFile (arquivo selecionado) e imprime o state que contém o documento, as stopwords e o progresso.

```
  handleStopWordsUpload = files => {
    const selectedFile = files[0]
    this.setState({
      document: this.state.document,
      stopWords: selectedFile,
      progress: this.state.progress,
    })
    console.log(this.state)
  }
```

```
  sendFiles = () => {
    const data = new FormData();
    data.append('documentFile', this.state.document, this.state.document.name)
    if(!!this.state.stopWords){
      data.append('stopWordsFile', this.state.stopWords, this.state.stopWords.name)
    }
    console.log(data)
    api.post('/tf', data, {
      onUploadProgress: e => {
        const progress = parseInt(Math.round((e.loaded * 100) / e.total))
        this.setState({
          document: this.state.document,
          stopWords: this.state.stopWords,
          progress: progress,
        })
      }
    }).then(response => {
      console.log(response)
      // Aqui vai entrar a lógica de exibir o resultado
    }).catch(() => {
      console.log('Error on upload files')
    })
  }
```

Atualiza o conteúdo do componente Container da Home com as variáveis atualizadas do state (document, stopwords e progress).

```
  render() {
    return (
      <>
        <Container>
          <Disclaimer>*Apenas documentos em português ou em inglês serão processados corretamente.</Disclaimer>
          <UploadContainer
            onUpload={this.handleDocumentUpload}
            selectedFile={this.state.document}
            description={<p>Arraste e solte seu <strong>documento (.txt)</strong></p>}
            subDescription='ou'
            buttonText='Selecione o arquivo'
            background='#404D3E'
            buttonColor='#BADDB4'
          />
          <UploadContainer
            onUpload={this.handleStopWordsUpload}
            selectedFile={this.state.stopWords}
            description={<p>Arraste e solte seu arquivo de <strong>stop-words (.txt)</strong></p>}
            subDescription='ou'
            buttonText='Selecione o arquivo'
            background='#4D3E3E'
```

```
            buttonColor='#DDB4B4'
          />
          {/* <Link to='/result'> */}
            <Button
              text='Calcular'
              isBig={true}
              backgroundColor='#ddd'
              disabled = {!this.state.document}
              handleClick={this.sendFiles}
            />
          {/* </Link> */}
        </Container>
      </>
    )
  }
}
```

INF1629_2020.2/front-end/src/pages/Home/styles.js

```
import styled from 'styled-components'
```

Este código CSS define a constante de estilo Container para a página Home, com largura 100%, altura calculada como (100viewport height - 100 pixels), altura mínima de 700 pixels, display flexível de itens, direção flexível dos itens é por colunas, alinhamento de itens centralizado e conteúdo de texto é space-around (itens são posicionados com espaço antes, entre e após as linhas); e do Disclaimer com e margem inferior de 25 pixels, que será utilizada pelo Home para melhorar a estética do elemento Container nesta parte da aplicação.
PRÉ: estilo padrão.
PÓS: definição de estilos para a constante Container da página Home.

```
export const Container = styled.div`
  width: 100%;
  height: calc(100vh - 100px);
  min-height: 700px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;

  * {
    margin-bottom: 25px;
  }
`

export const Disclaimer = styled.p`
  font-weight: 100;
`
```

INF1629_2020.2/front-end/src/pages/Result/index.js

```
import React from 'react'

import { Container } from './styles';
// import Button from '../../components/Button';
```

Este código exporta um container com o texto Result.

PRÉ: conteúdo padrão sem container do Result.

PÓS: retorna o container com o texto Result.

```
export default function Home() {
  return (
    <>
      <Container>
        <h1>Result</h1>
      </Container>
    </>
  )
}
```

INF1629_2020.2/front-end/src/pages/Result/styles.js

```
import styled from 'styled-components'
```

Este código CSS define a constante de estilo Container para a página Result, com largura 100%, altura calculada como (100viewport height - 100 pixels), display flexível, alinhamento de itens centralizado, conteúdo de texto space-around e padding de 30 pixels, que será utilizada pelo Result para melhorar a estética do elemento Container nesta parte da aplicação.

PRÉ: estilo padrão.

PÓS: definição de estilos para a constante Container da página Result.

```
export const Container = styled.div`
  width: 100%;
  height: calc(100vh - 100px);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
  padding: 30px;
```

INF1629_2020.2/front-end/src/components/Button/index.js

```
import React from 'react'

import { Container } from './styles'
```

Este código Javascript define a função do componente Botão da página passando como parâmetro o texto do botão, a cor de fundo #fff, define o botão como Big sendo falso, desabilita o HTML do botão, e passa a função de clicar do botão.
PRÉ: importação do componente Container do arquivo styles.js.
PÓS: definição da função do componente Botão da página HTML.

```
export default function Button({text, backgroundColor = '#fff', isBig = false, disabled = false, handleClick}) {
   return (
     <Container background={backgroundColor} isBig={isBig} disabled={disabled} onClick={handleClick}>
       {text}
     </Container>
   )
}
```

INF1629_2020.2/front-end/src/components/Button/styles.js

```
import styled, { css } from "styled-components";
```

Este código CSS define a constante de estilo Container que será utilizada pelo componente Botão para melhorar a estética na aplicação.

PRÉ: estilo padrão.

PÓS: definição de estilos para a constante Container do componente Botão.

```
export const Container = styled.button`
  border: none;
  background: ${props => props.background};
  color: black;
  font-weight: 100;
  cursor: pointer;

  &:hover{
    transition-property: opacity letter-spacing;
    transition-duration: 0.5s;
    opacity: 0.7;
    ${props =>
    props.isBig ?
      css`
      letter-spacing: 4px;
    `  : css`
      letter-spacing: 1px;
    `
    }
  }
  ${props =>
    props.isBig ?
      css`
        font-size: 22px;
        padding: 20px 50px;
        border-radius: 30px;
        letter-spacing: 3px;
        width: 50%;
        min-height: 62px;
        width: 50%;
        min-width: 800px;
        @media(max-width: 841px) {
          min-width: 0;
          width: 100%;
        }
    `  : css`
        font-size: 16px;
        padding: 5px 50px;
        border-radius: 20px;
        min-height: 26px;
    `
    }
`
```

INF1629_2020.2/front-end/src/components/Header/index.js

```
import React from 'react'

import { Container } from './styles'
```

Este código Javascript define o cabeçalho da página web, com imagem e descrição “Term Frequency Calculator”.
PRÉ: estilo padrão de cabeçalho.
PÓS: definição do cabeçalho da página HTML.

```
export default function Header() {
  return (
    <Container>
      <img src='/TFCLogo.png' alt='Term Frequency Calculator'/>
    </Container>
  )
}
```

INF1629_2020.2/front-end/src/components/Header/styles.js

```
import styled from 'styled-components'
```

Este código CSS define a constante de estilo Container para o Cabeçalho, com largura 100%, altura 100 pixels, display flexível, background na cor #2B2B2B, alinhamento de itens centralizado e conteúdo centralizado, que será utilizada pelo Cabeçalho para melhorar a estética do elemento Container nesta parte da aplicação.
PRÉ: estilo padrão.
PÓS: definição de estilos para a constante Container.

```
export const Container = styled.div`
  width: 100%;
  height: 100px;
  display: flex;
  background-color: #2B2B2B;
  align-items: center;
  justify-content: center;
```

INF1629_2020.2/front-end/src/components/UploadContainer/index.js

```
import React, { Component } from 'react'
import Dropzone from 'react-dropzone'

import Button from '../Button'
import { Container, Description, SubDescription } from './styles'
```

Este código define o componente de carregamento dos arquivos da aplicação Term Frequency Calculator. Define suas imagens, descrições e subdescrições.
PRÉ: conteúdo padrão.
PÓS: componente do container de carregamento de arquivos definido.

```
//https://www.youtube.com/watch?v=G5UZmvkLWSQ
export default class UploadContainer extends Component {


  renderDragMessage = (isDragActive, selectedFile) => {

    if (!isDragActive) {
      return (
        <>
          <Description>{this.props.description}</Description>
          <SubDescription>{this.props.subDescription}</SubDescription>
          <Button text={this.props.buttonText} backgroundColor={this.props.buttonColor}/>
          {!!selectedFile ? <SubDescription>Arquivo selecionado:
{selectedFile.name}</SubDescription> : ''}
        </>
      )
    }


    return <Description>Solte o arquivo aqui.</Description>


  }


  render() {
    const { onUpload } = this.props


    return(
      <Dropzone accept='text/plain' onDropAccepted={onUpload}>
        {({getRootProps, getInputProps, isDragActive}) => (
          <Container
            background={this.props.background}
            {...getRootProps()}
            isDragActive={isDragActive}
```

```
          >
          <input {...getInputProps()}/>
          <img src='/uploadIcon.png' alt='Upload Icon' />
          {this.renderDragMessage(isDragActive, this.props.selectedFile)}
        </Container>
      )}
    </Dropzone>
  )
}
            }
```

INF1629_2020.2/front-end/src/components/UploadContainer/styles.js

```
import styled, { css } from "styled-components";
```

Este código CSS define constantes de estilo que serão utilizadas pelo arquivo HTML para melhorar a estética da aplicação.
PRÉ: estilo padrão.
PÓS: definição de estilos para as constantes dragActive, Container, Description e SubDescription.

```
const dragActive = css`
  border-style: solid;
  border-width: 1px;
  border-color: #00C8FF;
`

export const Container = styled.div`
  width: 50%;
  min-width: 800px;
  height: 100%;
  border-radius: 30px;
  overflow: hidden;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-evenly;
  padding: 60px;
  background: ${props => props.background};

  ${props => props.isDragActive && dragActive}

  @media(max-width: 841px) {
    min-width: 0;
    width: 100%;
  }
`

export const Description = styled.p`
  font-size: 20px;
  font-weight: 400;
  text-align: center;
`

export const SubDescription = styled.p`
  font-size: 20px;
  font-weight: 100;
  text-align: center;
`
```

INF1629_2020.2/back-end/src/app.js

```
import express from 'express';
import cors from 'cors';
import Youch from 'youch';
import path from 'path';

import routes from './routes';
```

Este código cria a classe App e chama os métodos middlewares, routes e no final o exceptionHandler para lidar com erros durante o desenvolvimento.
PRÉ: importar express da aplicação 'express', importar cors de 'cors', Youch de 'youch', path de 'path' e routes de './routes' (INF1629_2020.2/back-end/src/routes.js). É necessário que a classe routes.js tenha sido previamente criada.
PÓS: construção da classe App, exportando como padrão um objeto da aplicação express, App.server.

```
class App {
    constructor() {
      this.server = express();
      this.middlewares();
      this.routes();
      this.exceptionHandler();
    }

    middlewares() {
      this.server.use(express.json());
      this.server.use(cors());
      this.server.use(
        '/tf',
        express.static(path.resolve(__dirname, '..', 'tmp', 'upload'))
      );
    }

    routes() {
      this.server.use(routes);
    }

    exceptionHandler() {
      this.server.use(async (err, req, res, next) => {
        if (process.env.NODE_ENV === 'development') {
          const errors = await new Youch(err, req).toJSON();
          return res.status(500).json(errors);
        }
        return res.status(500).json({ error: 'Internal Server Error' });
      });
    }
  }
  export default new App().server;
```

INF1629_2020.2/back-end/src/routes.js

Este código importa o objeto de aplicação Express, utiliza isso para obter o objeto Router e fazer uma requisição POST de forma assíncrona. Por último o modulo exporta o objeto Router, routes.
PRÉ: importar Router da aplicação 'express'.
PÓS: exportar o objeto Router criado, routes.

```
import {Router} from 'express';

const routes = new Router();

routes.post('/tf', async (req, res) => {

});

export default routes;
```

INF1629_2020.2/back-end/src/server.js

```
Este código está ouvindo as conexões pela porta 3333.
PRÉ: importar app de './app (INF1629_2020.2/back-end/src/app.js). É necessário que a class app.js tenha
sido previamente criada.
PÓS: o servidor aceitará requisições feitas na porta 3333.

import app from './app';

app.listen(3333);

console.log("Backend iniciado!");
```