Documentação do Projeto Desemburaca Teresina - Frontend

1. Sobre o Projeto

O Desemburaca Teresina é uma aplicação web que permite ao cidadão denunciar buracos nas vias públicas da cidade de Teresina, utilizando imagens com dados de localização GPS. O sistema foi desenvolvido como parte de um projeto acadêmico no Centro Universitário iCEV, com a finalidade de aproximar a população da gestão urbana e contribuir para uma cidade mais segura e funcional.

2. Desenvolvedores

- Talyson Machado Barros (Frontend)
- Augusto César A. M. Neto (Frontend / Backend)
- Ygor Jivago Leal Félix (Frontend / Backend)
- Vinicius Azevedo da Fonseca (Backend)

3. Tecnologias Utilizadas

- HTML, CSS, JavaScript
- Turf.js (verificação geográfica – Se esta dentro de teresina)
- EXIF.js (extração de metadados de imagem)
- LocationIQ (reversão geografica)
- IndexedDB (armazenamento offline)
- FormSubmit (suporte via e-mail)
- Google Drive API, PostgreSQL - Neon (via backend) {API próprias hospedadas pelo Railway }
- Repositorio do Back - https://github.com/ditimon01/Projeto_Desemburaca_Teresina_API

4. Funcionalidades

- Envio de denúncia com imagem contendo dados GPS
- Armazenamento offline com envio automático posterior
- Validação se a foto foi tirada em Teresina
- Exemplo de formulário preenchido e instruções de uso

- Campo de Envio de email para suporte

- Responsividade pensada para dispositivos moveis

5. Estrutura dos Arquivos

- `index.html`: Redireciona para a página principal.
- `home`: Pagina Inicial sobre a proposta do site.
- `ajuda`: Pagina com principais duvidas que podem surgir e um campo para enviar e-mails ao sup.
- `serv` : Pagina que representa os serviços do site e mostra as denucias registradas
- `sobre`: Pagina sobre os integrasntes do grupo e sobre a instituição
- `assets`: Pasta com as imagens do site
- `formulario`: Formulário de denuncia com validação de imagem e GPS.
    - `FuncoesPrincipais.js`: Funções de envio, verificação de GPS e outras verificações e armazenamento offline
    - `TeresinaPontos.js`: GeoJSON contendo os limites do município de Teresina.

6. Detalhe do Codigo

```
export function PreparandoEnvioOFF(){···
};

export function tentarEnviar(){···
};

export function LeituraDefoto(Entrada){···
};

export function TransformaCordenada(cordenadas, hemisferio,) {···
};

export function resetTodos() {···
};

export async function verificaTeresina(longitude, latitude) {···
};

export function loopDeAnim(contador,texto){···
};

export async function RevesaoGeografica(lon, lat){···
};

export async function DriveUploader(Foto){···
};

export async function PostBancoDeDados(data,categoria,observacao,imagem,lon,lat,rua,bairro) {···
};

function normalizarDataParaPostgres(dataString) {···
}
```

-PreparandoEnvioOFF-

Função para que salva todos os dados necesario para a denuncia, apenas se a internet cair, e segura esses dados na aplicação usando a o IndexedDB (divido em 2 partes na entrada do banco

de dados local : Dados para os dados brutos, e Foto para o pnj/png da foto que sera enviada ao drive

-tentarEnviar-

Função usada para enviar os dados quando a internet voltar, depois que identificar que ainternet voltou na pagina (EventLissener de online) o os dados salvos pela função PreparandoEnvioOFF são usados como entrada para as funçoes de envio das informçoes ao back end e consegutemente ao drive e o banco de dados (DriveUploader, PostBancodeDados e ReversãoGeografica) e logo em seguida apaga os dados salvos no IndexedDB

-LeituraDeFoto-

Função que utiliza a biblioteca EXIF.js para extrarir todos os metadados da foto necessario a aplicação, sendo esses nome,data e GPS

-TransformaCordenadas-

Função para pegar as cordendas da foto e formatalas da forma correta, colocando – ou + antes depedendo do hemisferio que ela foi tirada (no caso sempre vai ser negativo, mas é bom fazer pra todos os casos)

-ResetTodos-

Existem 2 opçoes de foto na aplicação, essa função é usada para quando você usar uma desas opçoes a outra ser limpa e as variaveis usadas pelo programa que são usadas pelas 2 opçoes limpas, evitando o envio de 2 fotos e possiveis erros

-VerificaTeresina-

Função que verifica se os dados de GPS estão dentro de Teresina, ela “roda” uma lista gigante de cordenadas (TeresinaPontos.js) e verifica se a cordenada esta dentro (ou seja dentro de Teresina

-LoopDeAnim-

Função que faz o botão enquanto esta enviando o form ou esperando conceção com o salve offline criar uma simples animação (usando mudanças do tescto do botão) para dar um feedback visual de que a aplicação esta trabalhando

-ReversãoGeografica-

Função que da fetch na API de revesão geografica (, ela é usada para devolver um json (resposta)

que deste é extraido a rua e o bairro de onde a foto foi tirada,(entrada necessaria para o banco de dados)

- Drive Uploader -PostBancoDeDados- normalizarDataParaPostgress-

Funçoes que dão post para o back processar para as APIs, a Drive envia a foto para o drive configurado a aplicação, o BancoDeDados, para o banco de dados na PostGress na platarforma Neon , e o normaliza é uma função para tratar a data da foto, pois quando usada direto da erro pois é formatada de forma diferente do que o banco de dados aceita

```
<script src="https://cdn.jsdelivr.net/npm/@turf/turf@6/turf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/exif-js"></script>
<script src="TeresinaPontos.js"></script>
<script type="module">

import {PreparandoEnvioOFF,tentarEnviar,LeituraDefoto,TransformaCordenada,resetTodos,verificaTeresina,RevesaoGeografica, DriveUploader,PostBancoDeDados,loopDeAnim} from './FuncoesPrincipais.js

window.esperandoConc =  localStorage.getItem('SerReiniciar')
window.lat2 = null;
window.lon2 = null;
window.Data = null;
window.Foto = null;
window.Rua = null;
window.Bairro = null;
window.timeoutAnimacao = null;

console.log(localStorage.getItem('SerReiniciar'))
console.log(window.esperandoConc)

/*  script para verificar se a imagem tem tudo certinho, data, gps e depois se internet ta on ne, ···

document.getElementById('form-denuncia').addEventListener('submit', (Entrada) => {  ···

document.getElementById('photo').addEventListener('change', (Entrada) => {···

document.getElementById('camera').addEventListener('change', (Entrada) => {···

document.getElementById('fecharenvio').addEventListener('click',() =>{···

document.getElementById('fecharimagem').addEventListener('click',() =>{···

window.addEventListener('online', () => {···
window.addEventListener('offline', () => {···
window.addEventListener('onload', () =>{···
document.addEventListener('DOMContentLoaded', () => {···
```

-As Verificações da Foto:

Para ser enviado um formulario de denuncia o html verifica se todos so campos estão preenchidos, se alguma foto ja foi colocada (se tentar enviar sem alguma foto ele dispara um popup pedindo uma) e o javascript verifica os dados da foto estão de acordo com o necessario, desativando o botão de envio caso a foto não apresente os dados necesario (ou esteja fora de teresina) (EventLissener (photo e camera)) e quando a foto é finalemnte enviada a pagina dispara um novo popup confirmando o envio.

-O banco de dados do Serviço:

Como ja mencionado, na pagina de serviços é possivel ver as denuncias ja feitas, para fazer isso foi

usado um fetch no mesmo endpoint usado na função PostBancodeDaDos (so que ao inves de ser um post é um get) que devolve uma resposta json para cada linha do banco , e logo em sequida usa-se cada campo do json para completar a tabela

```
<script>
fetch('https://projetodesemburacateresinaapi-production-1abf.up.railway.app/registro')
  .then(response => response.json())
  .then(data => {
    const tbody = document.querySelector("#tabelaDenuncias tbody");
    data.forEach(registro => {
      const linha = `
        <tr>
          <td>${registro.fid}</td>
          <td>${registro.data}</td>
          <td>${registro.categoria}</td>
          <td>${registro.status}</td>
          <td>${registro.observacao}</td>
          <td>${registro.imagem}</td>
          <td>${registro.rua}</td>
          <td>${registro.bairro}</td>
          <td>${registro.longitude}</td>
          <td>${registro.latitude}</td>
        </tr>
      `;
      tbody.innerHTML += linha;
    });
  });
</script>
</body>
<footer class="footer">
    <div class="footer-content">
        <div class="footer-left">
            <p>&copy; 2025 Prefeitura Municipal de Tereseina - Desemburaca Teresina. Todos os direitos reservados.</p>
        </div>
    </div>
</footer>
```

- O envio de email :

Através API de formSubmit, quando o usuario que enviar uma mensgem, é necesario colocar o emiail (para poder responder – sem denuncias anonimas aqui) e a mensagem que sera enviada

```
    <div class="localdaImagem">
        <form id="formemai" action="https://formsubmit.co/34094d6d84f766c6f192c602cb3a4cf0" method="POST" target="formFrame">
            <label for="mensagem">Mais alguma dúvida? Mande uma mensagem ao suporte:</label><br>
            <input type="email" name="email" id="campodeemail" required placeholder="Digite aqui seu emai, para resposta">
            <textarea id="mensagem" name="mensagem" rows="5" cols="55" required></textarea><br><br>

            <input type="text" name="_honey" style="display:none">
            <input type="hidden" name="_captcha" value="false">

            <button type="submit" id="email" class="Bemail">Enviar Email</button>
        </form>

        <iframe name="formFrame" style="display:none;"></iframe>
    </div>
</body>
```