GESTION DE INTEGRACION DEL PROYECTO

“ADMINISTRACION DE PROYECTOS DE TI”

INGENIERÍA EN

DESARROLLO Y GESTION DE SOTFWARE

PRESENTA:

URIEL MÁRQUEZ RESÉNDIZ

CARLOS DANIEL VILLA CASAS

DOCENTE

Ing. Maribel Trejo Reséndiz

JALPAN DE SERRA, QRO.

MAYO-AGOSTO DE 2024

Índex

Descripción Detallada del Código

Métodos:

addEventListener('submit', async function(event)):

- Función asociada: Se ejecuta cuando se envía el formulario de carga de imagen.
- Responsabilidad: Previene el comportamiento predeterminado del formulario, gestiona la carga del archivo de imagen, el procesamiento de la imagen con TensorFlow.js, la obtención de predicciones y la generación de un documento PDF con el resultado.

FileReader.onload:

- Función asociada: Se dispara cuando el archivo de imagen ha sido leído por FileReader.
- Responsabilidad: Asigna la imagen leída al elemento HTML <img> y procesa la imagen para su análisis con el modelo preentrenado.

Image.onload:

- Función asociada: Se dispara cuando la imagen ha sido cargada.
- Responsabilidad: Preprocesa la imagen, realiza las predicciones y muestra el diagnóstico.

jsPDF().text():

- Función asociada: Añade texto al documento PDF.
- Responsabilidad: Escribe el diagnóstico en el PDF.

jsPDF().save():

- Función asociada: Guarda el documento PDF.
- Responsabilidad: Descarga el PDF con el nombre 'diagnostico.pdf'.

Atributos:

const fileInput = document.getElementById('file-input'):

- Descripción: Referencia al input de tipo archivo en el formulario.
- Uso: Acceder al archivo de imagen seleccionado por el usuario.

const file = fileInput.files[0]:

- Descripción: Almacena el primer archivo seleccionado en el input.
- Uso: Para leer y procesar la imagen seleccionada.

const imgElement = document.getElementById('uploaded-image'):

- Descripción: Referencia al elemento <img> donde se mostrará la imagen cargada.
- Uso: Para visualizar la imagen seleccionada por el usuario.

const tensor:

- Descripción: Almacena el tensor generado a partir de la imagen procesada.
- Uso: Para alimentar el modelo de TensorFlow y generar predicciones.

const predictions:

- Descripción: Almacena el resultado de las predicciones generadas por el modelo.
- Uso: Determinar si la imagen presenta signos de enfermedad o no.

const diagnosis:

- Descripción: Almacena el diagnóstico basado en las predicciones.
- Uso: Para mostrar el diagnóstico en la interfaz y en el documento PDF.

Variables:

file:

- Tipo: Objeto File.
- Descripción: Representa el archivo de imagen seleccionado por el usuario.

reader:

Tipo: Objeto FileReader.

- Descripción: Permite leer el contenido del archivo seleccionado.

imgElement:

Tipo: Objeto HTMLImageElement.

- Descripción: Representa el elemento de imagen donde se mostrará la imagen cargada.

model:

Tipo: Objeto tf.LayersModel.

- Descripción: Representa el modelo preentrenado cargado desde un archivo JSON.

tensor:

Tipo: Objeto tf.Tensor.

- Descripción: Representa el tensor de la imagen que se utiliza para hacer predicciones.

doc:

Tipo: Objeto jsPDF.

- Descripción: Representa el documento PDF que se genera para guardar el diagnóstico.

Código de la aplicación

INDEX

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Diagnóstico Médico con IA</title>
    <link rel="stylesheet" href="CSS.css">
</head>
<body>
    <div class="container">
        <h1>DLEscaner</h1>
        <form id="upload-form">
            <input type="file" id="file-input" accept="image/*">
            <button type="submit">Subir Imagen</button>
        </form>
        <div id="screen">
            <h2>Pantalla de Diagnóstico</h2>
            <img id="uploaded-image" src="" alt="Imagen subida">
        </div>
        <div id="result">
            <h2>Resultado del Diagnóstico</h2>
            <p id="diagnosis">Sube una imagen para obtener el
diagnóstico.</p>
        </div>
        <div class="buttons">
            <script>
                document.getElementById('upload-
form').addEventListener('submit', async function(event) {
                    document.getElementById('upload-
form').addEventListener('submit', async function(event) {
    event.preventDefault();
    const fileInput = document.getElementById('file-input');
    const file = fileInput.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = async function(e) {
            const imgElement = document.getElementById('uploaded-image');
            imgElement.src = e.target.result;

            // Cargar el modelo preentrenado
            const model = await tf.loadLayersModel('model/model.json');
```

```
            // Preprocesar la imagen
            const img = new Image();
            img.src = e.target.result;
            img.onload = async () => {
                const tensor = tf.browser.fromPixels(img)
                    .resizeNearestNeighbor([224, 224]) // Tamaño que espera
el modelo
                    .toFloat()
                    .expandDims();

                // Hacer predicciones
                const predictions = await model.predict(tensor).data();

                // Mostrar el resultado
                const diagnosis = predictions[0] > 0.5 ? 'Enfermedad
detectada' : 'No se detecta enfermedad';
                document.getElementById('diagnosis').innerText =
`Diagnóstico: ${diagnosis}`;

                // Crear y descargar el documento PDF
                const doc = new jsPDF();
                doc.text("Resultado del Análisis de Imagen", 10, 10);
                doc.text(`Diagnóstico: ${diagnosis}`, 10, 20);
                doc.save('diagnostico.pdf');
            };
        };
        reader.readAsDataURL(file);
    }
});
                });
            </script>
        </div>
    </div>

    <!-- Load TensorFlow.js. This is required to use MobileNet. -->
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.1"> </script>

<!-- Load the MobileNet model. -->
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-
models/mobilenet@1.0.0"> </script>

<!-- Replace this with your image. Make sure CORS settings allow reading the
image! -->
```

```
<!-- Place your code in the script tag below. You can also use an external
.js file -->
<script>
  // Notice there is no 'import' statement. 'mobilenet' and 'tf' is
  // available on the index-page because of the script tag above.

  const img = document.getElementById('img');

  // Load the model.
  mobilenet.load().then(model => {
    // Classify the image.
    model.classify(img).then(predictions => {
      console.log('Predictions: ');
      console.log(predictions);
    });
  });
</script>
    <script src="scrip.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.4.0/jspdf.umd.min.js"></
script>
    <script src="JS.js"></script>
    <script src="TF.js"></script>
    <script src="UF.js"></script>
</body>
</html>
```

CSS

```css
body {
    font-family: 'Arial', sans-serif;
    background-color: #0f172a;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: #ffffff;
}

.container {
    background-color: #1e293b;
    padding: 2rem;
    border-radius: 15px;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    text-align: center;
    max-width: 800px;
    width: 100%;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    height: 90vh;
}

h1 {
    color: #0ea5e9;
    font-size: 2.5rem;
    margin-bottom: 1.5rem;
}

form {
    margin-bottom: 1.5rem;
}

input[type="file"] {
    display: block;
    margin: 0 auto 1rem auto;
    padding: 0.5rem;
    border-radius: 10px;
    border: 2px solid #0ea5e9;
    background-color: #1e293b;
    color: #ffffff;
}
```

```css
button {
    background-color: #0ea5e9;
    color: #ffffff;
    border: none;
    padding: 0.5rem 1rem;
    border-radius: 10px;
    cursor: pointer;
    font-size: 1rem;
    margin: 0.5rem;
}

button:hover {
    background-color: #0284c7;
}

#screen {
    background-color: #1e293b;
    padding: 1rem;
    border-radius: 10px;
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3);
    margin-bottom: 1.5rem;
}

#screen h2 {
    color: #0ea5e9;
    margin-bottom: 1rem;
}

#uploaded-image {
    max-width: 100%;
    max-height: 300px;
    border-radius: 10px;
    border: 2px solid #0ea5e9;
}

#result {
    background-color: #334155;
    padding: 1rem;
    border-radius: 10px;
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3);
    margin-bottom: 1.5rem;
}

#diagnosis {
```

```
    color: #94a3b8;
}

.buttons {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.buttons button {
    width: 200px;
}
```

JS

```
document.getElementById('upload-form').addEventListener('submit', async
function(event) {
    event.preventDefault();
    const fileInput = document.getElementById('file-input');
    const file = fileInput.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = async function(e) {
            const imgElement = document.getElementById('uploaded-image');
            imgElement.src = e.target.result;

            // Cargar el modelo preentrenado
            const model = await tf.loadLayersModel('model/model.json');

            // Preprocesar la imagen
            const img = new Image();
            img.src = e.target.result;
            img.onload = async () => {
                const tensor = tf.browser.fromPixels(img)
                    .resizeNearestNeighbor([224, 224]) // Tamaño que espera
el modelo
                    .toFloat()
                    .expandDims();

                // Hacer predicciones
                const predictions = await model.predict(tensor).data();

                // Mostrar el resultado
                const diagnosis = predictions[0] > 0.5 ? 'Enfermedad
detectada' : 'No se detecta enfermedad';
```

```
                document.getElementById('diagnosis').innerText =
`Diagnóstico: ${diagnosis}`;

                // Crear y descargar el documento PDF
                const doc = new jsPDF();
                doc.text("Resultado del Análisis de Imagen", 10, 10);
                doc.text(`Diagnóstico: ${diagnosis}`, 10, 20);
                doc.save('diagnostico.pdf');
            };
        };
        reader.readAsDataURL(file);
    }
});
```

JSON

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit:
https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "chrome",
            "request": "launch",
            "name": "Launch Chrome against localhost",
            "url": "http://localhost/DEEP LEARNING/",
            "webRoot": "${workspaceFolder}"
        }
    ]
}
```

TENSORFLOOW

```
document.getElementById('upload-form').addEventListener('submit', async
function(event) {
    event.preventDefault();
    const fileInput = document.getElementById('file-input');
    const file = fileInput.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = async function(e) {
            const imgElement = document.getElementById('uploaded-image');
            imgElement.src = e.target.result;
```

```
            // Cargar el modelo preentrenado
            const model = await tf.loadLayersModel('model/model.json');

            // Preprocesar la imagen
            const img = new Image();
            img.src = e.target.result;
            img.onload = async () => {
                const tensor = tf.browser.fromPixels(img)
                    .resizeNearestNeighbor([224, 224]) // Tamaño que espera
el modelo
                    .toFloat()
                    .expandDims();

                // Hacer predicciones
                const predictions = await model.predict(tensor).data();

                // Mostrar el resultado
                const diagnosis = predictions[0] > 0.5 ? 'Enfermedad
detectada' : 'No se detecta enfermedad';
                document.getElementById('diagnosis').innerText =
`Diagnóstico: ${diagnosis}`;

                // Guardar la predicción en una variable
                window.diagnosisResult = `Diagnóstico: ${diagnosis}`;

                // Generar PDF con jsPDF
                const doc = new jsPDF();
                doc.text("Resultado del Análisis de Imagen", 10, 10);
                doc.text(`Diagnóstico: ${diagnosis}`, 10, 20);
                doc.save("diagnostico.pdf");
            };
        };
        reader.readAsDataURL(file);
    }
});
```

UF

```
document.getElementById('upload-form').addEventListener('submit', async
function(event) {
    event.preventDefault();
    const fileInput = document.getElementById('file-input');
    const file = fileInput.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = async function(e) {
            const imgElement = document.getElementById('uploaded-image');
            imgElement.src = e.target.result;

            // Cargar el modelo preentrenado
            const model = await tf.loadLayersModel('model/model.json');

            // Preprocesar la imagen
            const img = new Image();
            img.src = e.target.result;
            img.onload = async () => {
                const tensor = tf.browser.fromPixels(img)
                    .resizeNearestNeighbor([224, 224]) // Tamaño que espera
el modelo
                    .toFloat()
                    .expandDims();

                // Hacer predicciones
                const predictions = await model.predict(tensor).data();

                // Mostrar el resultado
                const diagnosis = predictions[0] > 0.5 ? 'Enfermedad
detectada' : 'No se detecta enfermedad';
                document.getElementById('diagnosis').innerText =
`Diagnóstico: ${diagnosis}`;

                // Guardar la predicción en una variable
                window.diagnosisResult = `Diagnóstico: ${diagnosis}`;
            };
        };
        reader.readAsDataURL(file);
    }
});

document.getElementById('generate-doc').addEventListener('click', function()
{
    const { jsPDF } = window.jspdf;
```

```
    const doc = new jsPDF();

    doc.text("Resultado del Diagnóstico", 10, 10);
    doc.text(window.diagnosisResult, 10, 20);
    doc.save("diagnostico.pdf");
});
```