Informe sobre React y Minería de Datos

Descripción del Proyecto

En el contexto de una asignatura relacionada con minería de datos, se nos asignó una tarea que consistía en trabajar con un archivo comprimido (.rar) que contenía 39 archivos CSV. Estos archivos incluían datos sobre pozos registrados durante un periodo de cuatro años, con las siguientes variables:

- **date**: Fecha y hora del registro.
- **level**: Nivel de agua.
- **temperature**: Temperatura registrada.

Los datos se recolectaron cada 15 minutos, lo que generó un volumen significativo de información. La tarea consistía en graficar los datos, pero debido a la cantidad, resultaba difícil interpretarlos. Por ello, se nos solicitó encontrar una forma de reducir la cantidad de datos para facilitar su visualización y análisis.

Solución Implementada

Para resolver este problema, desarrollé una aplicación web utilizando React. La aplicación permite procesar los archivos CSV, reducir la cantidad de datos seleccionando uno cada dos meses, y visualizar los resultados de manera más clara.

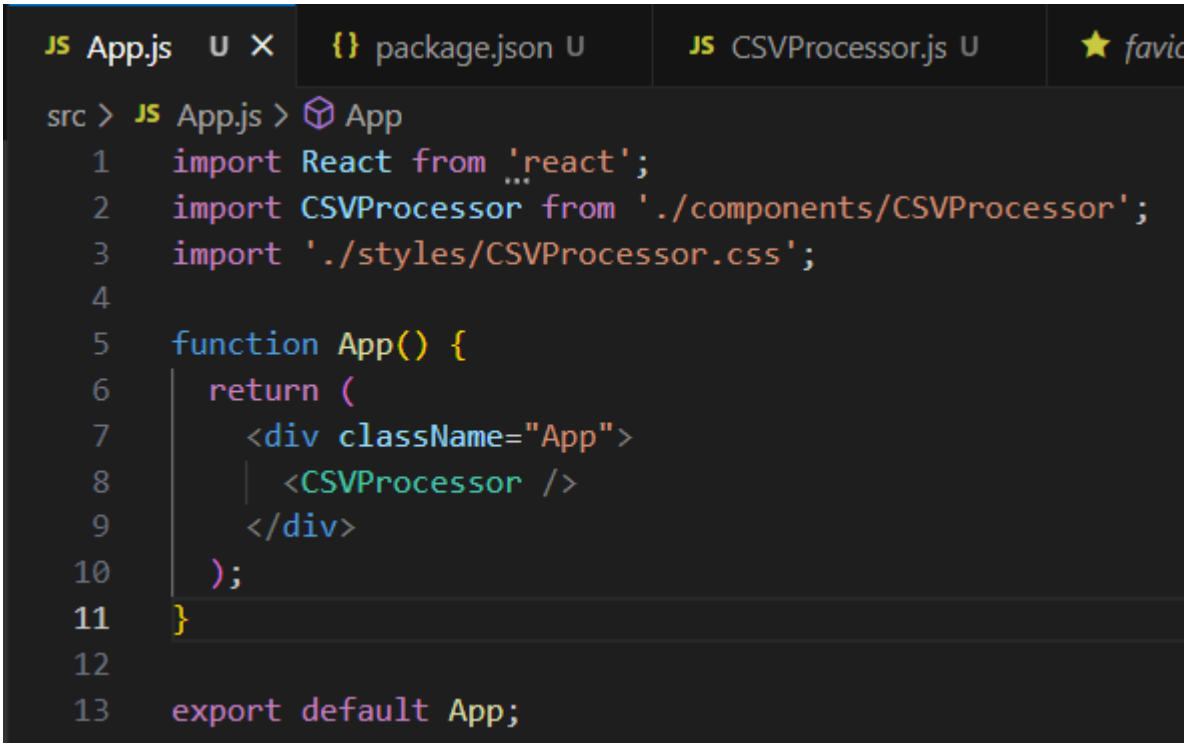
Estructura del Proyecto

El proyecto se organizó de la siguiente manera:

```
csv-processor/
├── package.json
├── public/
│   └── index.html
└── src/
    ├── App.js
    ├── components/
    │   └── CSVProcessor.js
    ├── styles/
    │   └── CSVProcessor.css
    ├── utils/
    │   └── csvProcessor.js
    └── index.js
```

Archivos del Proyecto

App.js



components/CSVProcessor.js

CSVProcessor.js U

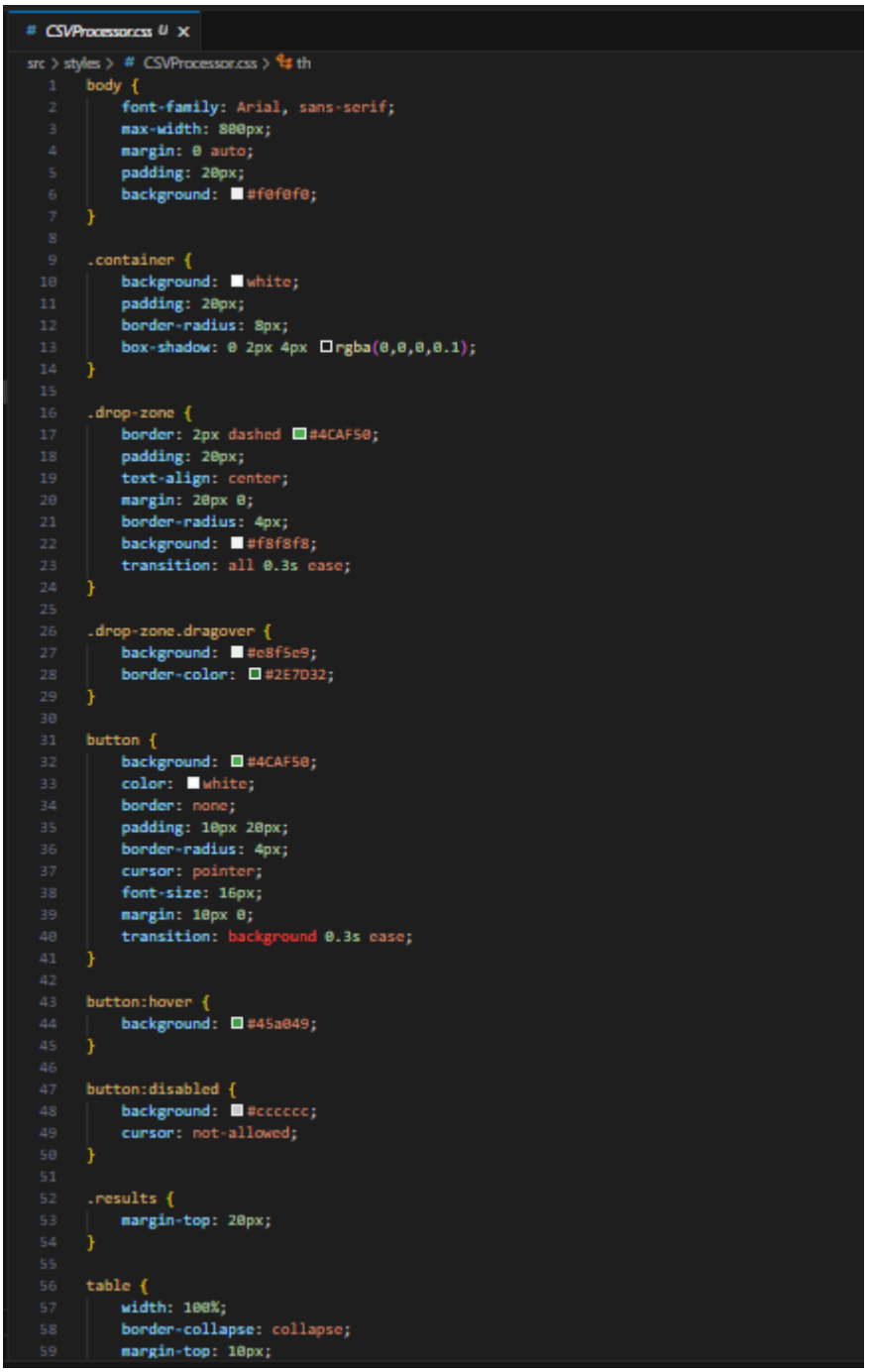
src > components > CSVProcessor.js > ...

```
import React, { useState } from 'react';
import JSZip from 'jszip';
import { saveAs } from 'file-saver';
import { processCSV } from '../utils/csvProcessor';

function CSVProcessor() {
    const [archiveFiles, setArchiveFiles] = useState(null);
    const [processingResults, setProcessingResults] = useState(null);
    const [dropZoneActive, setDropZoneActive] = useState(false);

    const handleDragOver = (e) => {
        e.preventDefault();
        setDropZoneActive(true);
    };

    const handleDragLeave = () => {
        setDropZoneActive(false);
    };

    const handleDrop = async (e) => {
        e.preventDefault();
        setDropZoneActive(false);
        const file = e.dataTransfer.files[0];
        await processArchiveFile(file);
    };

    const handleFileInput = async (e) => {
        const file = e.target.files[0];
        await processArchiveFile(file);
    };

    const processArchiveFile = async (file) => {
        if (file) {
            try {
                const zip = new JSZip();
                const zipContents = await zip.loadAsync(file);
                setArchiveFiles(zipContents);
            } catch (error) {
                console.error('Error processing file:', error);
                alert('Error processing file. Ensure it is a valid ZIP file.');
            }
        }
    };

    const processAndDownload = async () => {
        if (!archiveFiles) return;

        const zip = new JSZip();
        const results = {};

        // Process each CSV file
        for (const [filename, fileObj] of Object.entries(archiveFiles.files)) {
            if (filename.toLowerCase().endsWith('.csv')) {
                const content = await fileObj.async('text');
                const averages = await processCSV(content);
```

```
                // Create new CSV content with averages
                const newContent = 'level,temperature\n' +
                                   `${averages.level},${averages.temperature}`;
                // Add to new ZIP
                zip.file(filename, newContent);
                results[filename] = averages;
              }
            }
            // Update processing results for display
            setProcessingResults(results);
            // Generate and download ZIP
            const content = await zip.generateAsync({type: 'blob'});
            saveAs(content, 'csv_averages.zip');
        };
        return (
            <div className="container">
                <h1>CSV Processor</h1>
                <div
                    className={`drop-zone ${dropZoneActive ? 'dragover' : ''}`}
                    onDragOver={handleDragOver}
                    onDragLeave={handleDragLeave}
                    onDrop={handleDrop}
                >
                    <p>Drag and drop your ZIP file with CSVs here or</p>
                    <input
                        type="file"
                        id="fileInput"
                        accept=".zip"
                        onChange={handleFileInput}
                    />
                </div>
                <button
                    onClick={processAndDownload}
                    disabled={!archiveFiles}
                >
                    Process and Download
                </button>
```

```
                </button>
                {processingResults && (
                    <div className="results">
                        <h2>Processing Results:</h2>
                        <table>
                            <thead>
                                <tr>
                                    <th>File</th>
                                    <th>Avg Level</th>
                                    <th>Avg Temperature</th>
                                </tr>
                            </thead>
                            <tbody>
                                {Object.entries(processingResults).map(([filename, values]) => (
                                    <tr key={filename}>
                                        <td>{filename}</td>
                                        <td>{values.level !== 'N/A' ? values.level.toFixed(2) : 'N/A'}</td>
                                        <td>{values.temperature !== 'N/A' ? values.temperature.toFixed(2) : 'N/A'}</td>
                                    </tr>
                                ))}
                            </tbody>
                        </table>
                    </div>
                )}
            </div>
        );
    }

    export default CSVProcessor;
```

styles/CSVProcessor.css



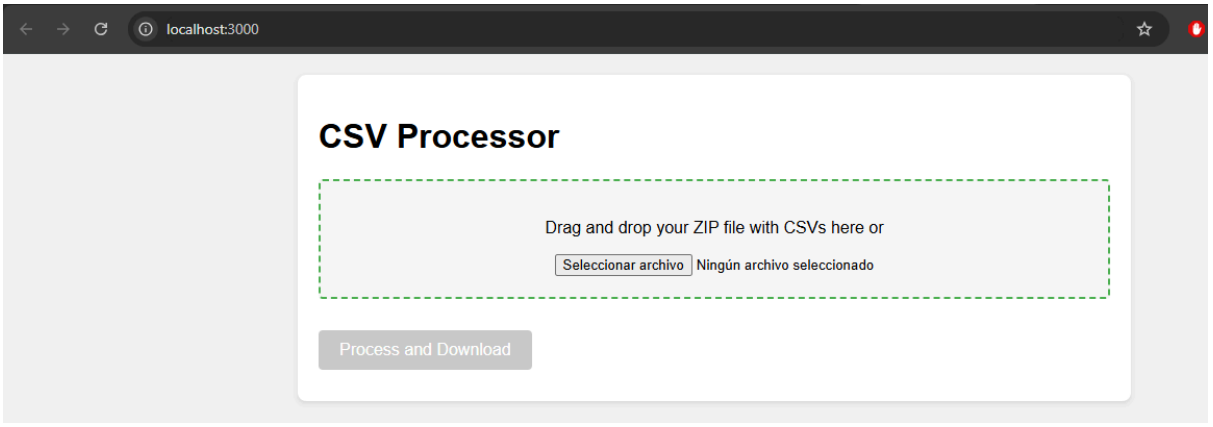
```
# CSVProcessor.css U
src > styles > # CSVProcessor.css > th
1   body {
2       font-family: Arial, sans-serif;
3       max-width: 800px;
4       margin: 0 auto;
5       padding: 20px;
6       background: #f0f0f0;
7   }
8
9   .container {
10      background: white;
11      padding: 20px;
12      border-radius: 8px;
13      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
14  }
15
16  .drop-zone {
17      border: 2px dashed #4CAF50;
18      padding: 20px;
19      text-align: center;
20      margin: 20px 0;
21      border-radius: 4px;
22      background: #f8f8f8;
23      transition: all 0.3s ease;
24  }
25
26  .drop-zone.dragover {
27      background: #e8f5e9;
28      border-color: #2E7D32;
29  }
30
31  button {
32      background: #4CAF50;
33      color: white;
34      border: none;
35      padding: 10px 20px;
36      border-radius: 4px;
37      cursor: pointer;
38      font-size: 16px;
39      margin: 10px 0;
40      transition: background 0.3s ease;
41  }
42
43  button:hover {
44      background: #45a049;
45  }
46
47  button:disabled {
48      background: #cccccc;
49      cursor: not-allowed;
50  }
51
52  .results {
53      margin-top: 20px;
54  }
55
56  table {
57      width: 100%;
58      border-collapse: collapse;
59      margin-top: 10px;
```

utils/csvProcessor.js

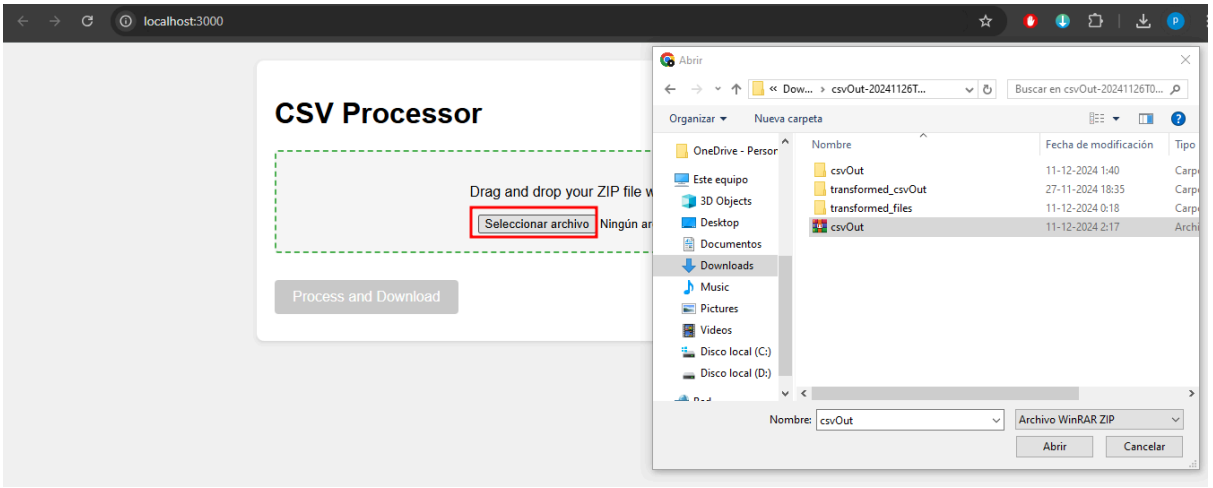
index.js

```
src > JS index.js > ...
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import App from './App';
4
5   const root = ReactDOM.createRoot(document.getElementById('root'));
6   root.render(
7     <React.StrictMode>
8       <App />
9     </React.StrictMode>
10  );
```

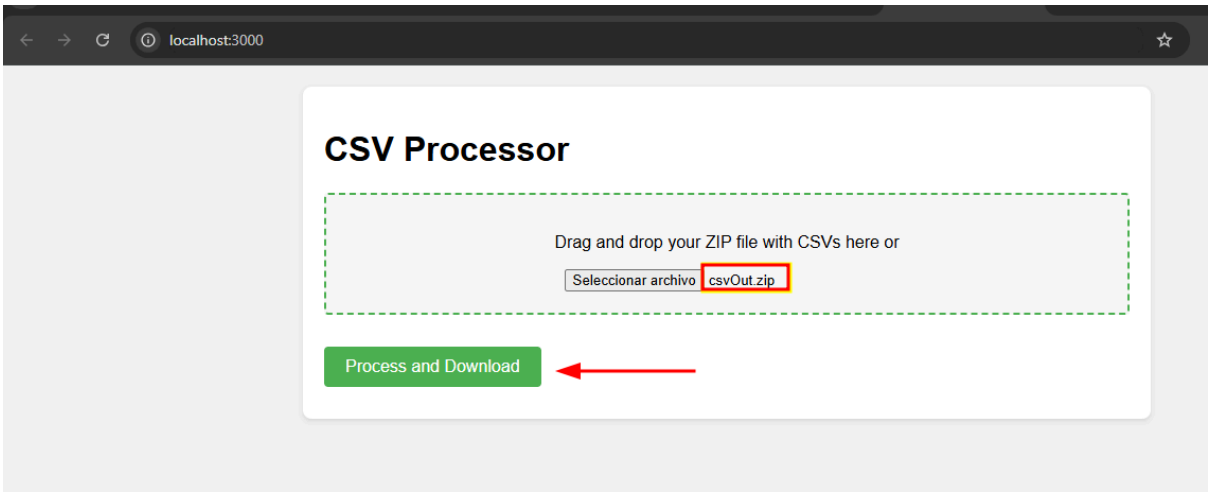
Resultados



Al interactuar con la opción "Seleccionar archivo" o al arrastrar un archivo, será necesario cargar un archivo en formato RAR que contenga las características **date**, **level** y **temperature**.



Una vez seleccionado el archivo, su nombre aparecerá junto al botón "Seleccionar archivo", y se habilitará el botón "Procesar y descargar".



Tras procesar el archivo, se generará una tabla que mostrará el promedio correspondiente a cada pozo. Además, se abrirá la ubicación donde se guardará el archivo descargado con los resultados de los promedios calculados.