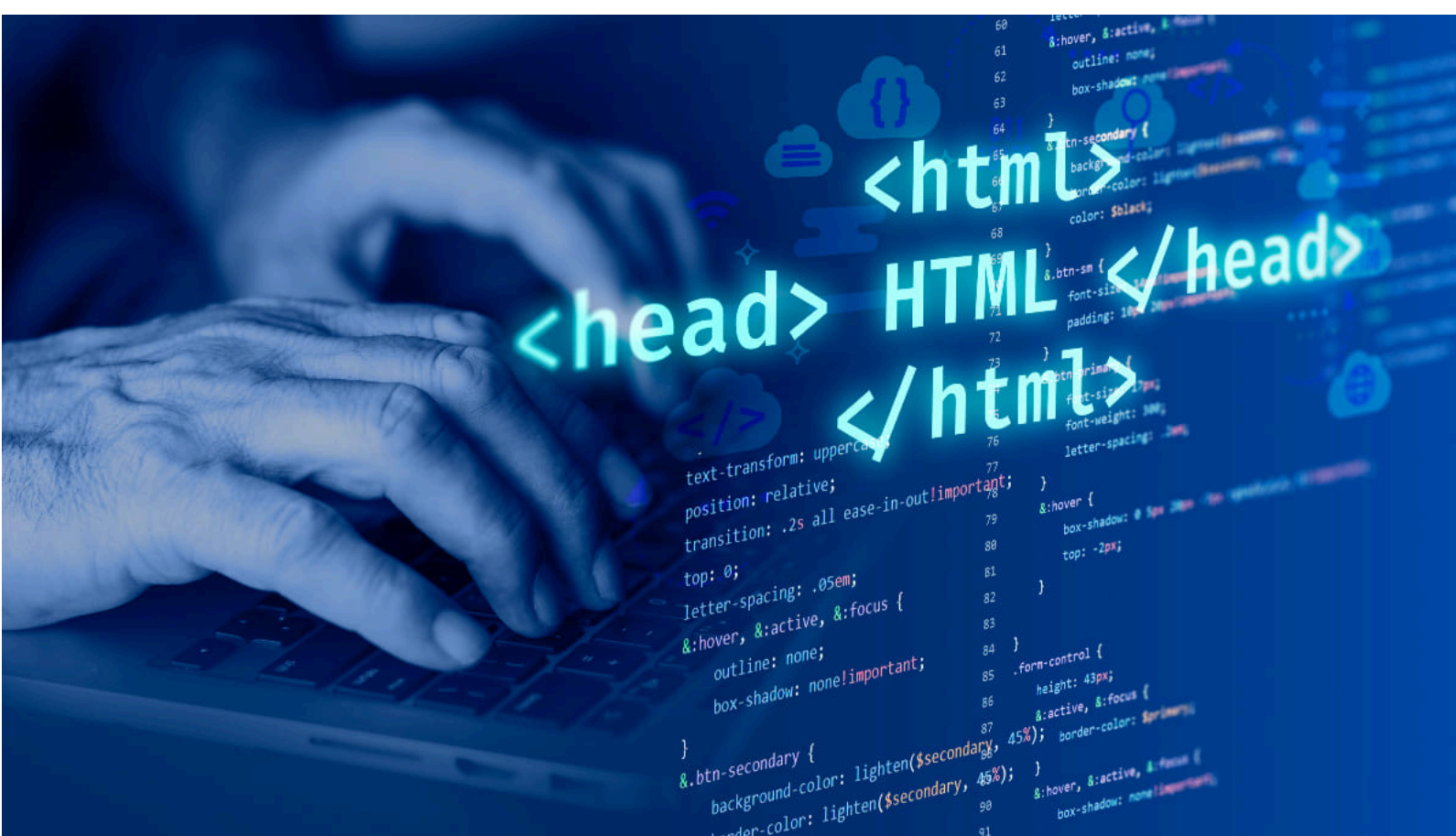


1er hito 3er Trimestre

Adrian Cabrera Caceres
30 Abril Fecha de Entrega



INDICE

FASE 1:Desarrollo de Calculadora:	3
Investigación y Planificación:	3
Desarrollo y Implementación:	3
Fase 2: Integración de Datos Meteorológicos:	4
Selección de la API:	4
Desarrollo e Integración:	5
Personalización y Diseño de la Interfaz:	5
Pruebas y Depuración:	5
Investigación sobre Tecnologías Utilizadas:	6
HTML :	6
CSS :	6
JavaScript:	6
API :	6

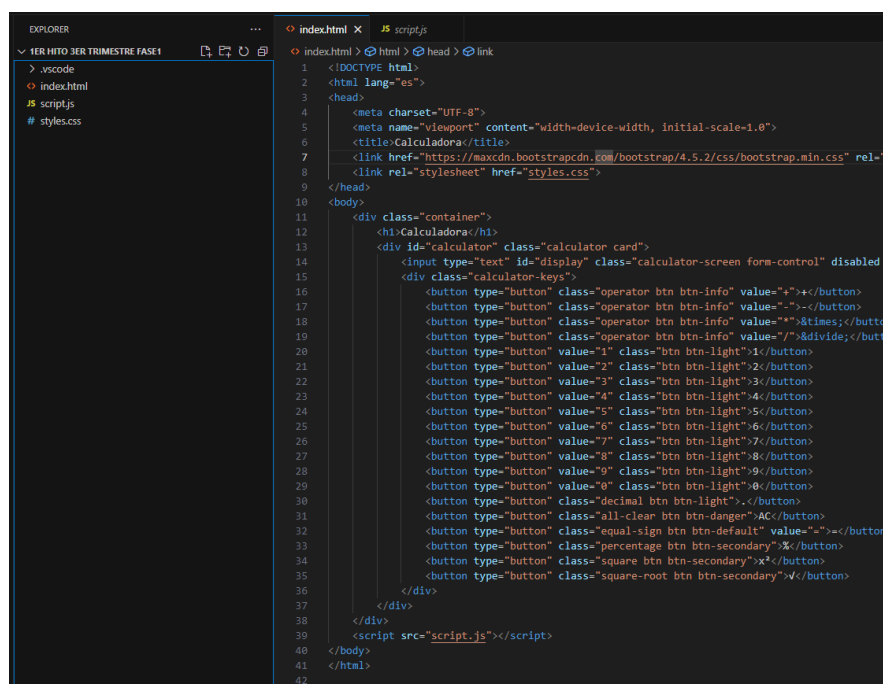
FASE 1: Desarrollo de Calculadora:

Investigación y Planificación:

Antes de comenzar con la implementación, nos embarcamos en una investigación exhaustiva sobre cómo construir una calculadora desde cero. Exploramos recursos en línea, tutoriales y documentación para comprender los fundamentos del desarrollo web y cómo aplicarlos para crear nuestra propia calculadora. Durante esta etapa, también delineamos un plan detallado que especificaba las funcionalidades que queríamos incluir, como la suma, resta, multiplicación y división, así como la apariencia y la experiencia del usuario que deseábamos lograr. Además, decidimos agregar funcionalidades adicionales como el cálculo del porcentaje, la raíz cuadrada y el elevado al cuadrado para mejorar la utilidad y versatilidad de nuestra calculadora.

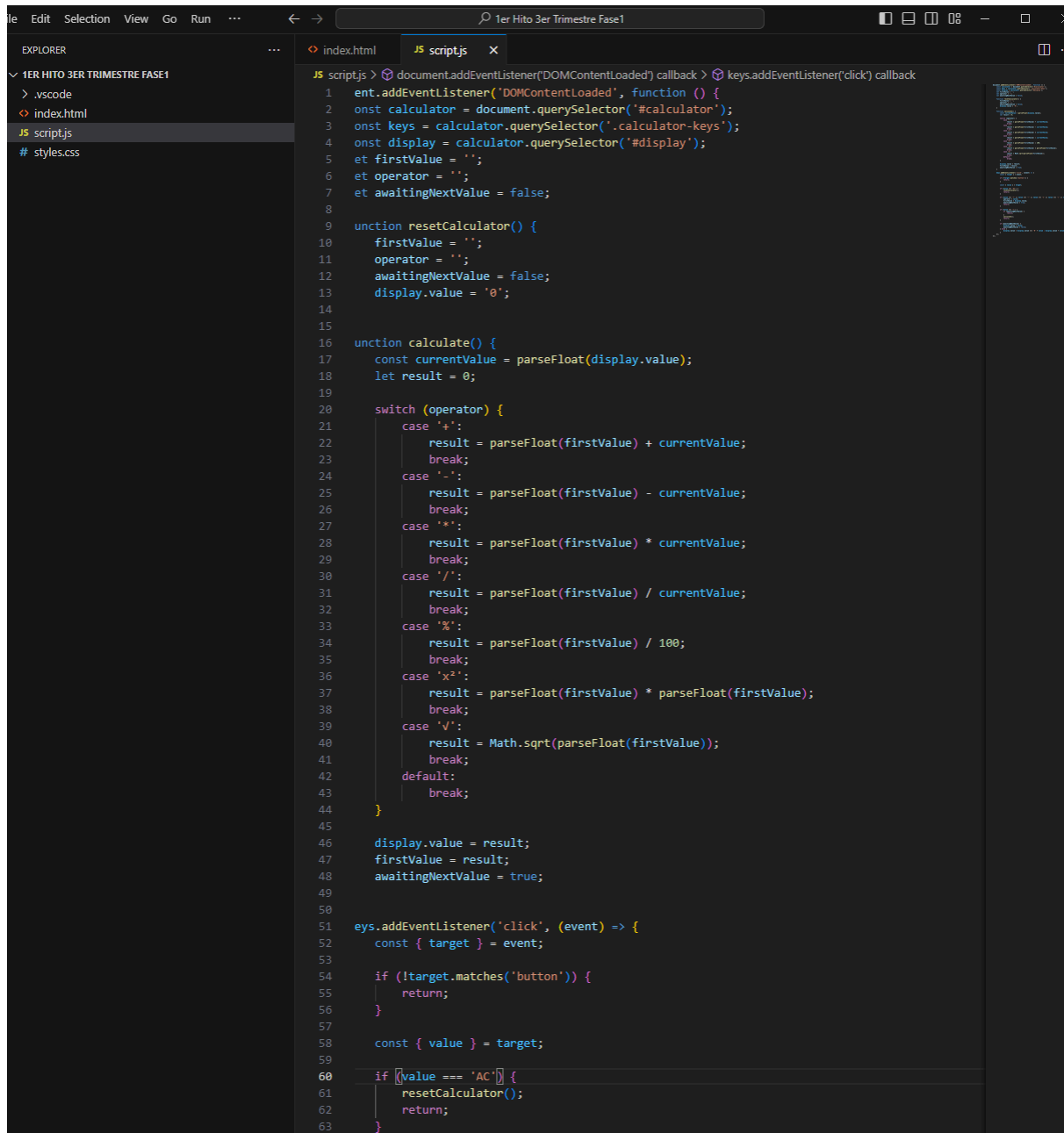
Desarrollo y Implementación:

Con nuestro plan en su lugar, comenzamos a escribir el código para nuestra calculadora. Utilizamos HTML para crear la estructura básica de la calculadora, incluidos los botones numéricos y de operación. Luego, aplicamos estilos CSS para darle un aspecto atractivo y fácil de usar, asegurándonos de que la calculadora fuera visualmente agradable y responsive en diferentes dispositivos. Finalmente, utilizamos JavaScript para agregar la funcionalidad de cálculo a la calculadora, permitiendo a los usuarios realizar operaciones matemáticas en tiempo real y ver los resultados de manera dinámica en la pantalla de la aplicación implantada.



```
EXPLORER
  1ER HITO 3ER TRIMESTRE FASE1
    .vscode
    index.html
    script.js
    styles.css

index.html X JS script.js
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Calculadora</title>
7   <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
8   <link rel="stylesheet" href="styles.css">
9 </head>
10 <body>
11   <div class="container">
12     <h1>Calculadora</h1>
13     <div id="calculator" class="calculator card">
14       <input type="text" id="display" class="calculator-screen form-control" disabled="">
15       <div class="calculator-keys">
16         <button type="button" class="operator btn btn-info" value="+"/>+</button>
17         <button type="button" class="operator btn btn-info" value="-"/>-</button>
18         <button type="button" class="operator btn btn-info" value="*"/>*</button>
19         <button type="button" class="operator btn btn-info" value="/">/</button>
20         <button type="button" value="1" class="btn btn-light">1</button>
21         <button type="button" value="2" class="btn btn-light">2</button>
22         <button type="button" value="3" class="btn btn-light">3</button>
23         <button type="button" value="4" class="btn btn-light">4</button>
24         <button type="button" value="5" class="btn btn-light">5</button>
25         <button type="button" value="6" class="btn btn-light">6</button>
26         <button type="button" value="7" class="btn btn-light">7</button>
27         <button type="button" value="8" class="btn btn-light">8</button>
28         <button type="button" value="9" class="btn btn-light">9</button>
29         <button type="button" value="0" class="btn btn-light">0</button>
30         <button type="button" class="decimal btn btn-light">.</button>
31         <button type="button" class="all-clear btn btn-danger">AC</button>
32         <button type="button" class="equal-sign btn btn-default" value="=">=</button>
33         <button type="button" class="percentage btn btn-secondary">%</button>
34         <button type="button" class="square btn btn-secondary">x</button>
35         <button type="button" class="square-root btn btn-secondary">√</button>
36       </div>
37     </div>
38   </div>
39   <script src="script.js"></script>
40 </body>
41 </html>
42
```

A screenshot of a Visual Studio Code editor window. The Explorer sidebar on the left shows a project named '1ER HITO 3ER TRIMESTRE FASE1' with files '.vscode', 'index.html', 'script.js', and 'styles.css'. The main editor area is open to 'script.js', which contains a JavaScript calculator application. The code includes DOMContentLoaded and click event listeners, variables for calculator state, and functions for resetting and calculating the result based on a switch statement for different operators. The status bar at the bottom indicates line 63, column 1.

```
1  ent.addEventListener('DOMContentLoaded', function () {
2    const calculator = document.querySelector('#calculator');
3    const keys = calculator.querySelector('.calculator-keys');
4    const display = calculator.querySelector('#display');
5    let firstValue = '';
6    let operator = '';
7    let awaitingNextValue = false;
8
9    function resetCalculator() {
10      firstValue = '';
11      operator = '';
12      awaitingNextValue = false;
13      display.value = '0';
14    }
15
16    function calculate() {
17      const currentValue = parseFloat(display.value);
18      let result = 0;
19
20      switch (operator) {
21        case '+':
22          result = parseFloat(firstValue) + currentValue;
23          break;
24        case '-':
25          result = parseFloat(firstValue) - currentValue;
26          break;
27        case '*':
28          result = parseFloat(firstValue) * currentValue;
29          break;
30        case '/':
31          result = parseFloat(firstValue) / currentValue;
32          break;
33        case '%':
34          result = parseFloat(firstValue) / 100;
35          break;
36        case 'x²':
37          result = parseFloat(firstValue) * parseFloat(firstValue);
38          break;
39        case '√':
40          result = Math.sqrt(parseFloat(firstValue));
41          break;
42        default:
43          break;
44      }
45
46      display.value = result;
47      firstValue = result;
48      awaitingNextValue = true;
49    }
50
51    keys.addEventListener('click', (event) => {
52      const { target } = event;
53
54      if (!target.matches('button')) {
55        return;
56      }
57
58      const { value } = target;
59
60      if (value === 'AC') {
61        resetCalculator();
62        return;
63      }
64    });
65  });
```

Fase 2: Integración de Datos Meteorológicos

Selección de la API:

seleccionamos una API que cumplía con nuestros requisitos y nos proporcionaba la información meteorológica necesaria para nuestra aplicación.

Desarrollo e Integración:

Una vez seleccionada la API, procedimos a integrarla en nuestra aplicación web. Utilizamos JavaScript y la función `fetch()` para realizar solicitudes a la API y obtener los datos del tiempo para diferentes regiones. Luego, procesamos estos datos y los mostramos de manera significativa en nuestra interfaz de usuario. Creamos tarjetas de información meteorológica para cada región, que incluían detalles como la temperatura, la humedad, la velocidad del viento y las condiciones climáticas actuales. (esto no me sale)

Personalización y Diseño de la Interfaz:

Además de mostrar los datos del tiempo, también nos aseguramos de que la interfaz de usuario fuera atractiva y fácil de usar. Aplicamos estilos CSS para personalizar el aspecto de las tarjetas de información meteorológica y hacerlas visualmente atractivas. Utilizamos colores y elementos relacionados con el clima para crear una experiencia cohesiva y temática para los usuarios.

Pruebas y Depuración:

He intentado arreglarlo pero no ay forma me pasa con todas las apis no se me muestra la información meteorológica

```
1 document.addEventListener('DOMContentLoaded', function() {
2   const weatherInfoDiv = document.getElementById('weather-info');
3
4   // Realizar la petición Fetch al API
5   fetch('https://www.el-tiempo.net/api/json/v2/provincias')
6     .then(response => response.json())
7     .then(data => {
8       if (data && data.provincias) {
9         // Manejar los datos obtenidos
10        const provinces = data.provincias;
11        provinces.forEach(province => {
12          console.log(province); // Imprimir datos de cada provincia en la consola
13          const provinceName = province.NOMBRE_PROVINCIA;
14          const provinceWeather = province.info;
15
16          const weatherCard = document.createElement('div');
17          weatherCard.classList.add('weather-card');
18
19          const title = document.createElement('h2');
20          title.textContent = `Tiempo en ${provinceName}`;
21          weatherCard.appendChild(title);
22
23          if (provinceWeather && provinceWeather.prediccion) {
24            const weatherPredictions = provinceWeather.prediccion;
25            const weatherDetails = document.createElement('div');
26            weatherDetails.classList.add('weather-details');
27
28            // Iterar sobre las predicciones meteorológicas
29            for (const [date, prediction] of Object.entries(weatherPredictions)) {
30              const detail = document.createElement('div');
31              detail.innerHTML = `<strong>${date}</strong>: ${prediction}`;
32              weatherDetails.appendChild(detail);
33            }
34
35            weatherCard.appendChild(weatherDetails);
36          } else {
37            const errorMessage = document.createElement('p');
38            errorMessage.textContent = 'No hay datos disponibles para este lugar.';
39            weatherCard.appendChild(errorMessage);
40          }
41
42          weatherInfoDiv.appendChild(weatherCard);
43        });
44      } else {
45        throw new Error('La propiedad "provincias" no está definida en los datos recibidos');
46      }
47    })
48    .catch(error => {
49      console.error('Error al obtener los datos del tiempo:', error);
50      weatherInfoDiv.innerHTML = `<p>Ocurrió un error al obtener los datos del tiempo.</p>`;
51    });
52 });
```

Investigación sobre Tecnologías Utilizadas

En este apartado, profundizaremos en las tecnologías fundamentales utilizadas en el desarrollo de nuestro proyecto, así como en los conceptos clave detrás de ellas. Exploraremos cómo HTML, CSS, JavaScript y el uso de API contribuyeron a la creación de nuestra aplicación web de calculadora y la integración de datos meteorológicos en tiempo real.

HTML :

HTML es el lenguaje estándar utilizado para crear y estructurar contenido en la web. En nuestro proyecto, utilizamos HTML para construir la estructura básica de nuestra aplicación web de calculadora y para definir los elementos y componentes que componen la interfaz de usuario. Utilizamos etiquetas HTML como `<div>`, `<input>` y `<button>` para crear la disposición y funcionalidad de la calculadora, permitiendo a los usuarios interactuar con ella de manera intuitiva.

CSS :

CSS es un lenguaje de estilo utilizado para definir el aspecto y el diseño de las páginas web. En nuestro proyecto, aplicamos estilos CSS para personalizar la apariencia de nuestra calculadora y hacerla visualmente atractiva y responsive en diferentes dispositivos. Utilizamos reglas CSS para controlar propiedades como el color, la tipografía, el espaciado y el diseño de los elementos HTML, creando una interfaz de usuario cohesiva y agradable.

JavaScript:

JavaScript es un lenguaje de programación de alto nivel utilizado para agregar interactividad y dinamismo a las páginas web. En nuestro proyecto, utilizamos JavaScript para agregar la funcionalidad de cálculo a nuestra calculadora, permitiendo a los usuarios realizar operaciones matemáticas en tiempo real y ver los resultados de manera dinámica en la pantalla. También utilizamos JavaScript para integrar datos meteorológicos en tiempo real en nuestra aplicación, realizando solicitudes a una API y procesando los datos devueltos para mostrarlos de manera significativa en nuestra interfaz de usuario.

API :

Una API es un conjunto de reglas y protocolos que permite que diferentes sistemas de software se comuniquen entre sí. En nuestro proyecto, utilizamos una API de tiempo para

obtener datos actualizados sobre el clima de diferentes regiones y los integramos en nuestra aplicación web. Utilizamos la función `fetch()` en JavaScript para realizar solicitudes a la API y obtener los datos del tiempo para diferentes ubicaciones. Luego, procesamos estos datos y los mostramos de manera significativa en nuestra interfaz de usuario, brindando a los usuarios una experiencia más completa y útil.