

## Taller de Diagnóstico Inicial

Melanny Yilyan Guate

Asignatura: Arquitectura y Diseño del Software

Fecha: 27 de octubre 2025

Tema: Proceso de Evaluación, Documentación y Herramientas de Calidad.

Objetivo de la Clase: Realizar un diagnóstico de calidad de software. Aprender a usar herramientas profesionales para analizar, corregir y documentar un proyecto de código, asegurando no solo su funcionamiento sino también su comprensión. Además, investiga el propósito de las herramientas que utilizas.

Plan A: Flujo de Trabajo Integrado (Si se pueden instalar extensiones)

### **Investigación de Herramientas / Tools Investigation**

#### **ESLint**

[Español]:

ESLint es una herramienta que analiza el código JavaScript para detectar errores, malas prácticas y problemas de estilo.

Ayuda a mantener un código limpio, uniforme y sin errores lógicos o sintácticos.

[English]:

ESLint is a tool that analyzes JavaScript code to detect errors, bad practices, and style issues.

It helps keep the code clean, consistent, and free of logical or syntax errors.

#### **Prettier**

[Español]:

Prettier es un formateador automático de código que corrige la indentación, espacios, comillas y otros detalles de estilo.

Su objetivo es hacer que el código sea más legible y consistente sin cambiar su funcionalidad.

[English]:

Prettier is an automatic code formatter that adjusts indentation, spacing, quotes, and other style details.

Its goal is to make code more readable and consistent without altering its functionality.

Paso 1: Preparación del Entorno Evidencia Visual / Visual Evidence:

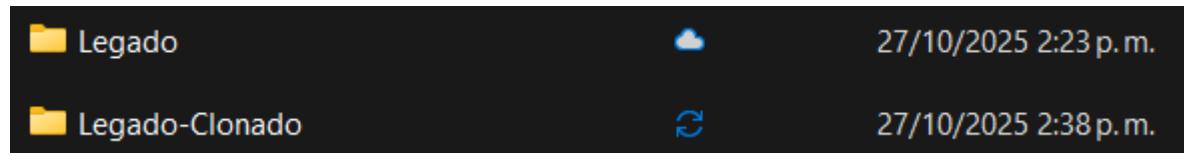
## 1. Clonar el Proyecto: Clona el repositorio del proyecto "Legado" desde GitHub

```
C:\Users\yilgr\OneDrive\Desktop>git clone https://github.com/yilgr19/Legado.git Legado-Clonado
Cloning into 'Legado-Clonado'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.

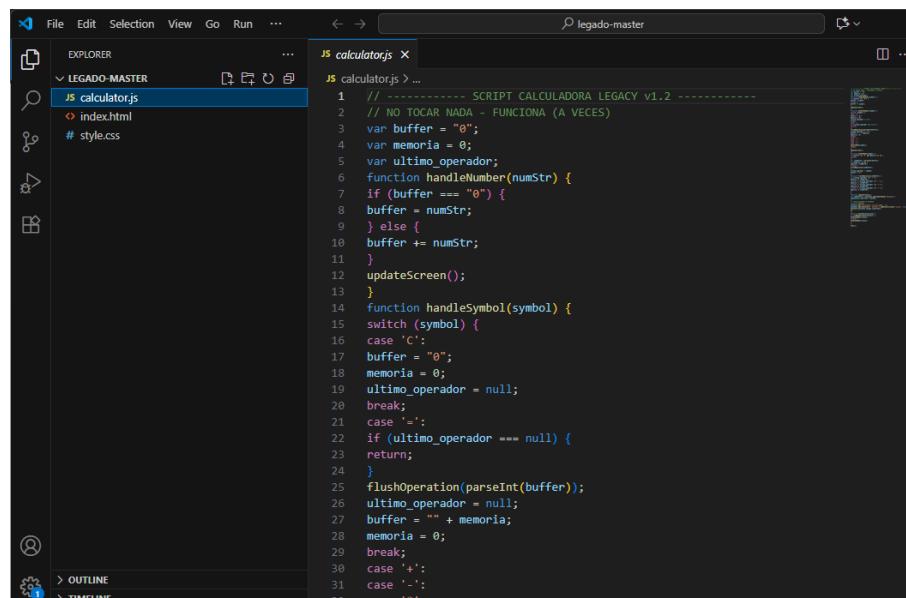
C:\Users\yilgr\OneDrive\Desktop>cd Legado-Clonado

C:\Users\yilgr\OneDrive\Desktop\Legado-Clonado>code .

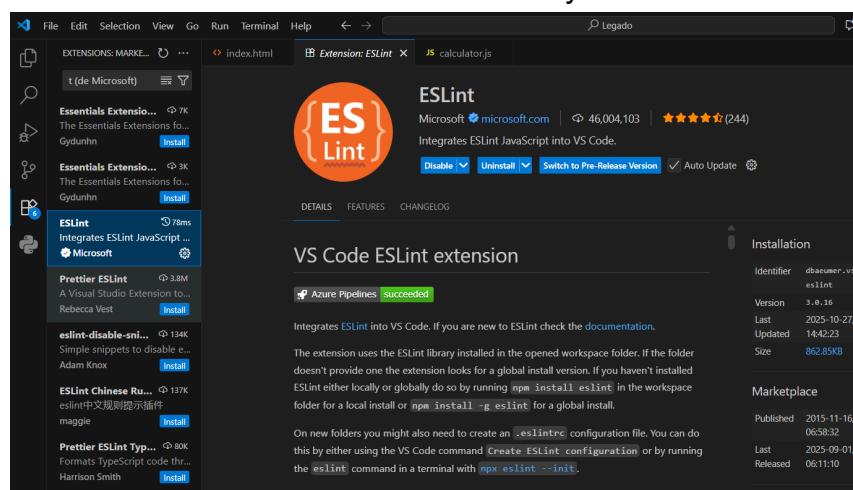
C:\Users\yilgr\OneDrive\Desktop\Legado-Clonado>
```

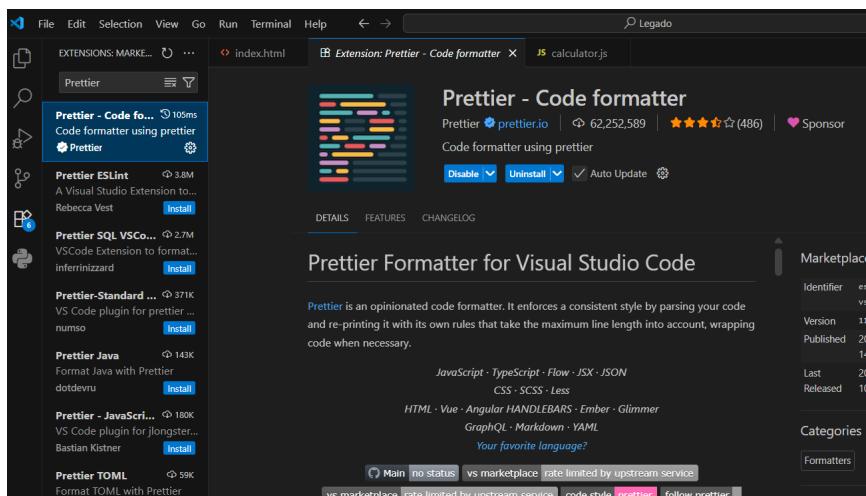


## 2. Abrir en VS Code y Navegar: Abre la carpeta en VS Code y explora los archivos.



## 3. Instalar Extensiones: Instalar ESLint y Prettier - Code formatter.





## Paso 2: Análisis y Diagnóstico

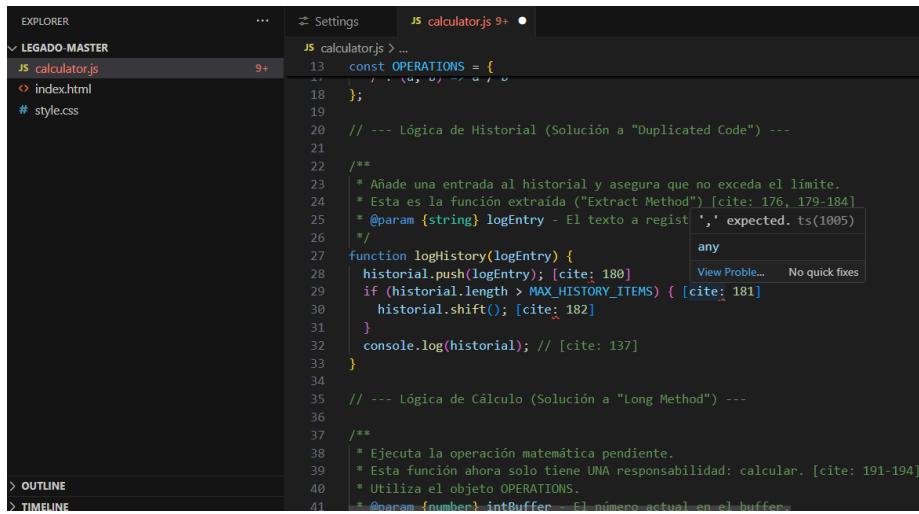
1. Ejecutar y Documentar "Antes": Abre index.html en el navegador. Toma una captura de pantalla del archivo calculator.js mostrando su estado original desordenado.

```

js calculator.js > ...
1 // MODULO DE CALCULADORA v2 - REFACTORIZADO
2
3 // --- Variables y Constantes Globales ---
4 var buffer = "0";
5 var memoria = 0;
6 var ultimo_operador;
7 var historial = [];
8
9 // Solución a "Magic Number"
10 const MAX_HISTORY_ITEMS = 5;
11
12 // Solución a "if-else gigante" (Replace Conditional with Strategy) [cite: 198, 200]
13 const OPERATIONS = {
14   '+': (a, b) => a + b,
15   '-': (a, b) => a - b,
16   '*': (a, b) => a * b,
17   '/': (a, b) => a / b
18 };
19
20 // --- Lógica de Historial (Solución a "Duplicated Code") ---
21
22 /**
23  * Añade una entrada al historial y asegura que no exceda el límite.
24  * Esta es la función extraída ("Extract Method") [cite: 176, 179-184]
25  * @param {string} logEntry - El texto a registrar.
26 */

```

2. Diagnóstico con ESLint: Configura ESLint. Dedica tiempo a analizar las advertencias que aparecen. Pasa el mouse sobre ellas para entender los problemas reportados.



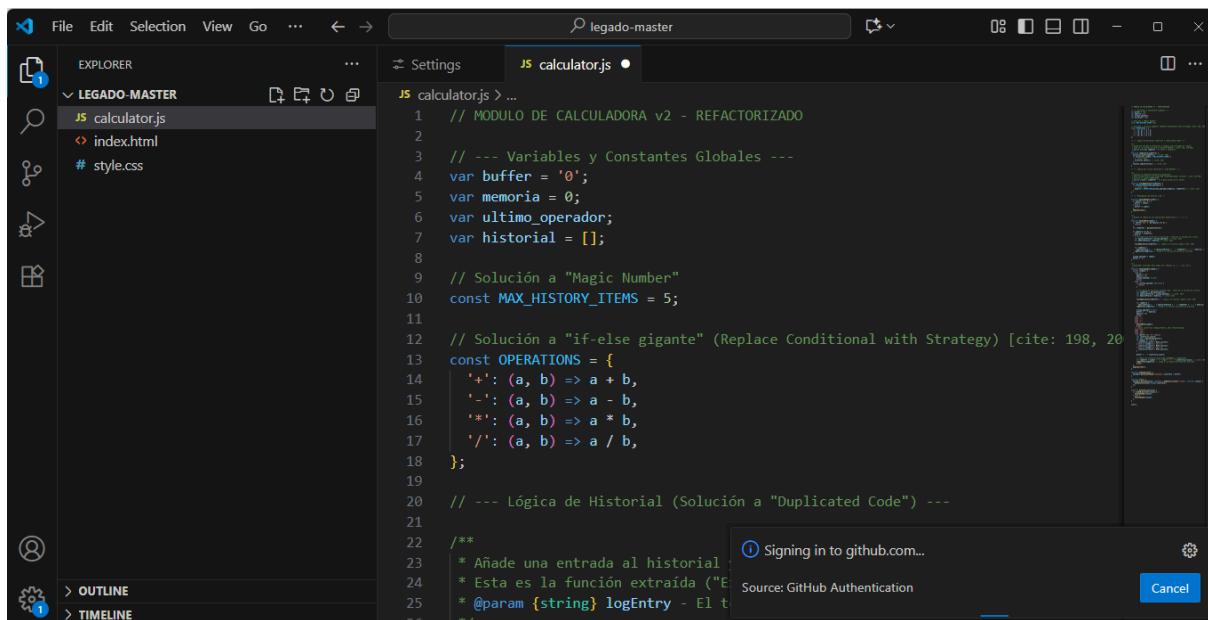
A screenshot of the Visual Studio Code interface. The left sidebar shows the 'EXPLORER' view with files: 'LEGADO-MASTER' (calculator.js, index.html, style.css), 'OUTLINE', and 'TIMELINE'. The main editor tab is 'calculator.js' with the file content visible. A tooltip is shown over a line of code, providing information about a ESLint error: 'any' at [cite: 181]. The status bar at the bottom shows the path 'legado-master'.

```
JS calculator.js > ...
13 const OPERATIONS = {
14   '+': (a, b) => a + b,
15   '-': (a, b) => a - b,
16   '*': (a, b) => a * b,
17   '/': (a, b) => a / b,
18 };
19
20 // --- Lógica de Historial (Solución a "Duplicated Code") ---
21
22 /**
23  * Añade una entrada al historial y asegura que no excede el límite.
24  * Esta es la función extraída ("Extract Method") [cite: 176, 179-184]
25  * @param {string} logEntry - El texto a registrar, 'expected' ts(1005)
26  */
27 function logHistory(logEntry) {
28   historial.push(logEntry); [cite: 180]
29   if (historial.length > MAX_HISTORY_ITEMS) { [cite: 181]
30     historial.shift(); [cite: 182]
31   }
32   console.log(historial); // [cite: 137]
33 }
34
35 // --- Lógica de Cálculo (Solución a "Long Method") ---
36
37 /**
38  * Ejecuta la operación matemática pendiente.
39  * Esta función ahora solo tiene UNA responsabilidad: calcular. [cite: 191-194]
40  * Utiliza el objeto OPERATIONS.
41  * @param {number} intBuffer - El número actual en el buffer

```

### Paso 3: Corrección Automática

1. Formateo con Prettier: En el archivo calculator.js, presiona Alt + Shift + F para corregir automáticamente el estilo del código

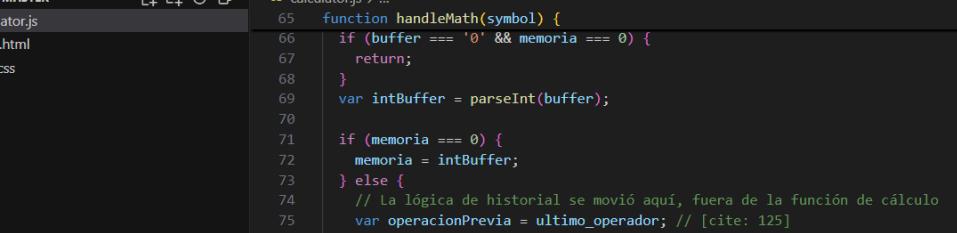


A screenshot of the Visual Studio Code interface. The left sidebar shows the 'EXPLORER' view with files: 'LEGADO-MASTER' (calculator.js, index.html, style.css). The main editor tab is 'calculator.js' with the file content visible. A tooltip is shown over a line of code, providing information about a ESLint error: 'any' at [cite: 181]. A GitHub authentication dialog is open in the bottom right corner, with the message 'Signing in to github.com...' and 'Source: GitHub Authentication'. The status bar at the bottom shows the path 'legado-master'.

```
JS calculator.js > ...
1 // MODULO DE CALCULADORA v2 - REFACTORIZADO
2
3 // --- Variables y Constantes Globales ---
4 var buffer = '0';
5 var memoria = 0;
6 var ultimo_operador;
7 var historial = [];
8
9 // Solución a "Magic Number"
10 const MAX_HISTORY_ITEMS = 5;
11
12 // Solución a "if-else gigante" (Replace Conditional with Strategy) [cite: 198, 200]
13 const OPERATIONS = {
14   '+': (a, b) => a + b,
15   '-': (a, b) => a - b,
16   '*': (a, b) => a * b,
17   '/': (a, b) => a / b,
18 };
19
20 // --- Lógica de Historial (Solución a "Duplicated Code") ---
21
22 /**
23  * Añade una entrada al historial.
24  * Esta es la función extraída ("Extract Method") [cite: 176, 179-184]
25  * @param {string} logEntry - El texto a registrar, 'expected' ts(1005)
26  */
27 function logHistory(logEntry) {
28   historial.push(logEntry); [cite: 180]
29   if (historial.length > MAX_HISTORY_ITEMS) { [cite: 181]
30     historial.shift(); [cite: 182]
31   }
32   console.log(historial); // [cite: 137]
33 }
34
35 // --- Lógica de Cálculo (Solución a "Long Method") ---
36
37 /**
38  * Ejecuta la operación matemática pendiente.
39  * Esta función ahora solo tiene UNA responsabilidad: calcular. [cite: 191-194]
40  * Utiliza el objeto OPERATIONS.
41  * @param {number} intBuffer - El número actual en el buffer

```

2. Documentar el "Después": Toma una captura del mismo archivo, ahora limpio y formateado.



```
function handleMath(symbol) {
    if (buffer === '0' && memoria === 0) {
        return;
    }
    var intBuffer = parseInt(buffer);

    if (memoria === 0) {
        memoria = intBuffer;
    } else {
        // La lógica de historial se movió aquí, fuera de la función de cálculo
        var operacionPrevia = ultimo_operador; // [cite: 125]
        var memoriaPrevia = memoria; // [cite: 126]

        flushOperation(intBuffer); // Llama a la función limpia [cite: 191]

        var logEntry =
            memoriaPrevia + ' ' + operacionPrevia + ' ' + intBuffer + ' = ' + memoria; /
        logHistory(logEntry); // Llama a la función de historial extraída
    }

    ultimo_operador = symbol;
    buffer = '0';
}
```

#### Paso 4: Documentación Bilingüe del Código

1. Añadir comentarios: Ahora que el código está funcional y limpio, tu tarea es documentarlo. En el archivo calculator.js, añade un comentario línea por línea explicando qué hace.

## Análisis de Hallazgos / Findings Analysis

Nº	Aspecto Analizado	Descripción (Español)	Description (English)
1	<b>Estructura modular</b>	El código está dividido en secciones lógicas: variables globales, operaciones, historial y UI. Esto mejora la legibilidad y el mantenimiento.	The code is organized into logical sections: globals, operations, history, and UI, improving readability and maintainability.
2	<b>Uso de constantes</b>	Se define MAX_HISTORY_ITEMS para evitar números mágicos. Buen ejemplo de refactorización.	Defines MAX_HISTORY_ITEMS to avoid magic numbers — a good refactoring practice.
3	<b>Patrón Strategy aplicado</b>	El objeto OPERATIONS reemplaza condicionales múltiples para las operaciones básicas.	The OPERATIONS object replaces multiple conditional statements, implementing the Strategy pattern.
4	<b>Control del historial</b>	La función logHistory() evita duplicación de código y controla el tamaño máximo del historial.	The logHistory() function prevents duplicated logic and limits the history length.
5	<b>Responsabilidad única</b>	Cada función realiza una tarea específica (ej. flushOperation() solo calcula). Cumple con el principio SRP.	Each function has a single purpose (e.g., flushOperation() only performs calculations), following SRP (Single Responsibility Principle).
6	<b>Interacción con la UI</b>	updateScreen() y init() manejan el DOM sin mezclar la lógica de negocio.	updateScreen() and init() handle DOM operations separately from business logic.
7	<b>Manejo de eventos</b>	El evento click en .buttons delega correctamente el manejo según el tipo de entrada (número o símbolo).	Button click events are properly delegated depending on input type (number or symbol).
8	<b>Código reutilizable</b>	Las funciones científicas (sin, cos, tan) fueron integradas de forma clara con una sola función de registro.	Scientific functions are integrated cleanly with a single history logging mechanism.
9	<b>Legibilidad y comentarios</b>	Excelente uso de comentarios bilingües y consistentes, ideal para trabajo colaborativo o educativo.	Excellent bilingual commenting, useful for collaboration and teaching.

10	<b>Posibles mejoras</b>	Se podría mejorar el manejo de errores (división por 0, valores no válidos) y usar let/const en lugar de var para modernizar el código.	Could improve error handling (e.g., division by zero) and replace var with let/const for modern JavaScript standards.
----	-------------------------	---	---

## Repositorio de GitHub / GitHub Repository

[Español]:

El proyecto final se encuentra alojado en el siguiente enlace:

<https://github.com/RobinsonGo09/legado>