TECNOLÓGICO®
SUPERIOR DE LA SIERRA NEGRA
DE AJALPAN

PORTAFOLIO DE EVIDENCIAS ESTRUCTURA DE DATOS

Nombre de la alumna: Andrea Vazquez Sanchez

Semestre: 3

Carrera: Sistemas computacionales

Tabla de contenido

Semana 1……

Semana 2……

Semana 3……..

Semana 4……..

Semana 5…….

Semana 6……

Semana 7…….

Semana 8…….

Semana 9…….

Semana 10…….

Semana 11……..

Semana 12…….

Semana 13…….

Semana 14…….

Semana 15……..

Semana 16……

Semana 1

HTML5 — Lenguaje de marcado para estructurar contenido web. :content Reference[oaicite:3] {i ndex =3}

- **CSS** (si está enlazado desde el HTML) — Para el estilo visual.

- **JavaScript** (si existe código enlazado o inline) — Para lógica de interacción

Declaración `<!DOCTYPE html>` — Establece que se usará HTML5. :contentReference[oaicite:7]{index=7}

2. **Etiqueta `<html>`** con el atributo de idioma (`lang="es"`).

3. **Sección `<head>`** con:

   - `<meta charset="UTF-8">` — Configuración de codificación.

   - `<meta name="viewport">` — Para responsividad en dispositivos móviles.

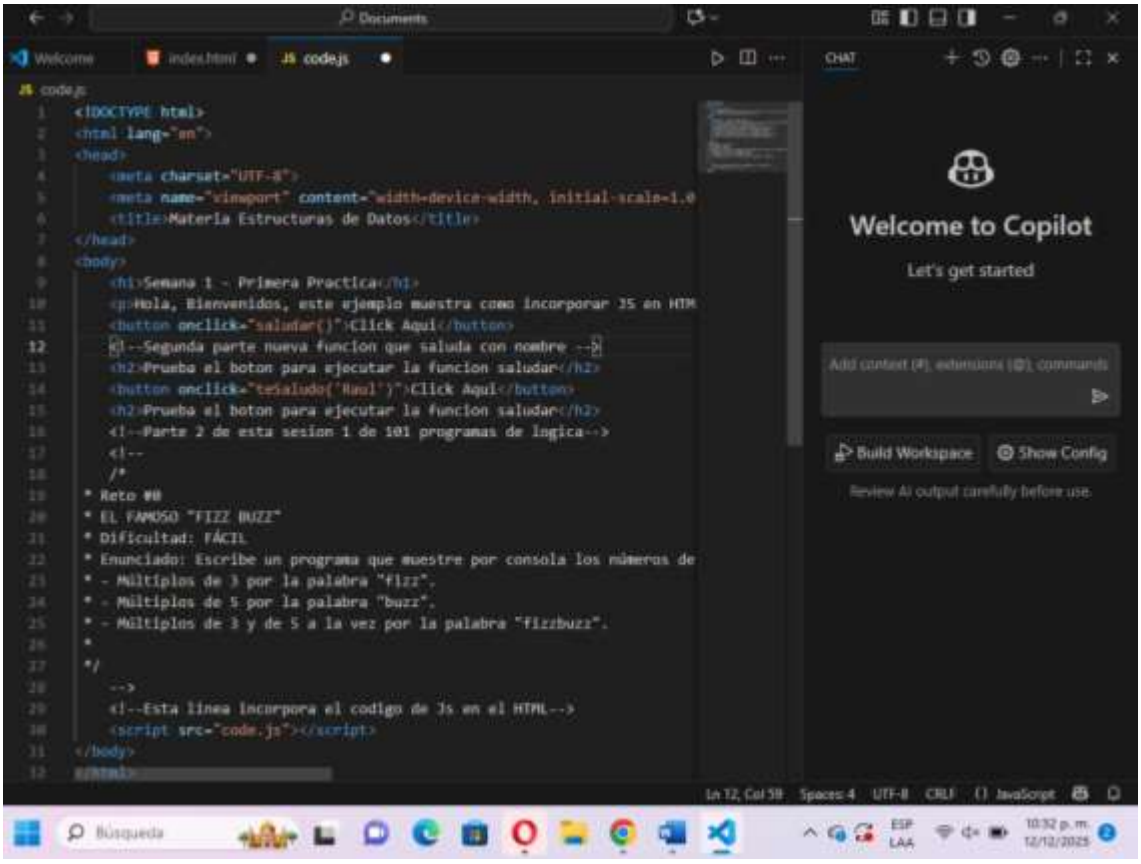
   - `<title>` — Título de la página. :contentReference[oaicite:8]{index=8}

4. **Sección `<body>`** con el contenido visible:

   - Encabezados `<h1>`, `<h2>`, etc.

   - Párrafos `<p>`.

- **Estructura didáctica clara:** El uso de HTML en este contexto educativo facilita la comprensión de conceptos básicos de desarrollo web. :contentReference[oaicite:10]{index=10}



Semana 2

HTML inicia con la etiqueta <!DOCTYPE html> y termina con la etiqueta </html>

tipos de etiqueta

1 es aquella que tiene un inicio y un fin puede contener mas etiquetas dentro

2 es aquella que solo indica incio y fin en una sola etiqueta ejemplo <nombre/>

HTML5 – Para la estructura de las páginas web educativas.

- **CSS** – Si está enlazado o incluido, para el diseño visual de las páginas.

- **JavaScript** – Si hay interacción o lógica dinámica en las páginas.

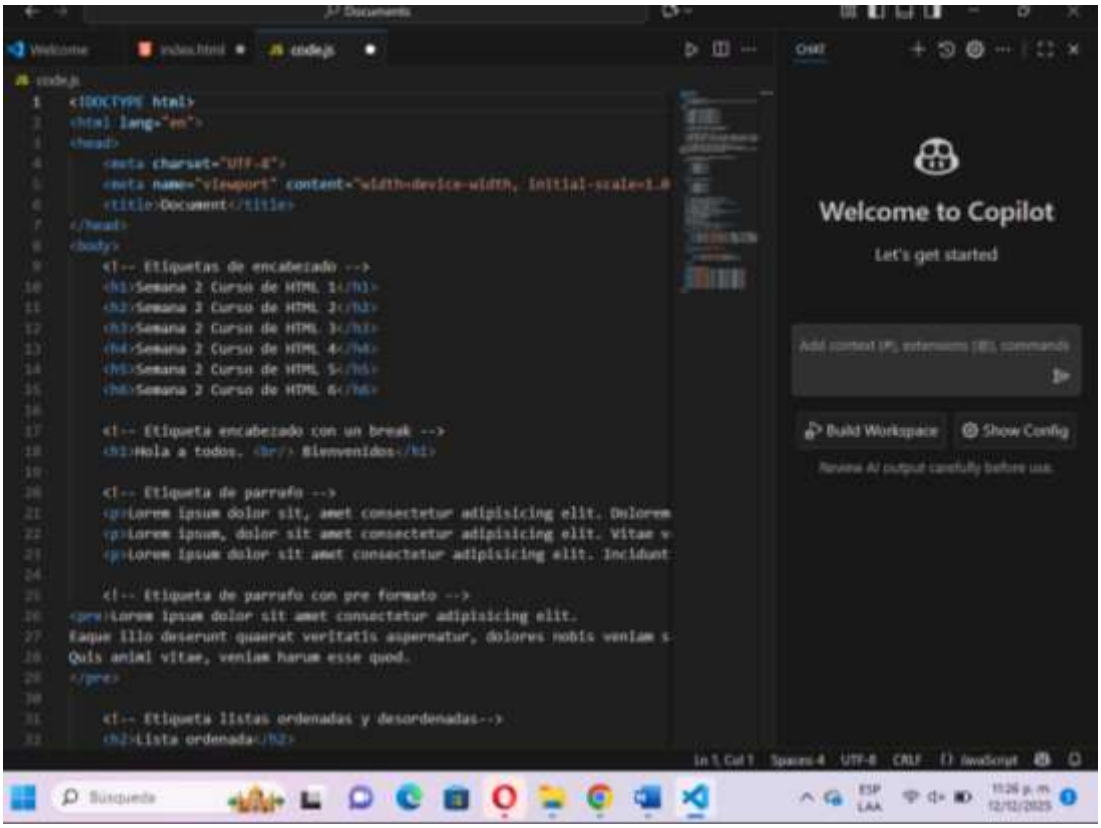
El archivo `acerca.html` (que literalmente significa *“Acerca de”* o *“About”*) suele usarse para:

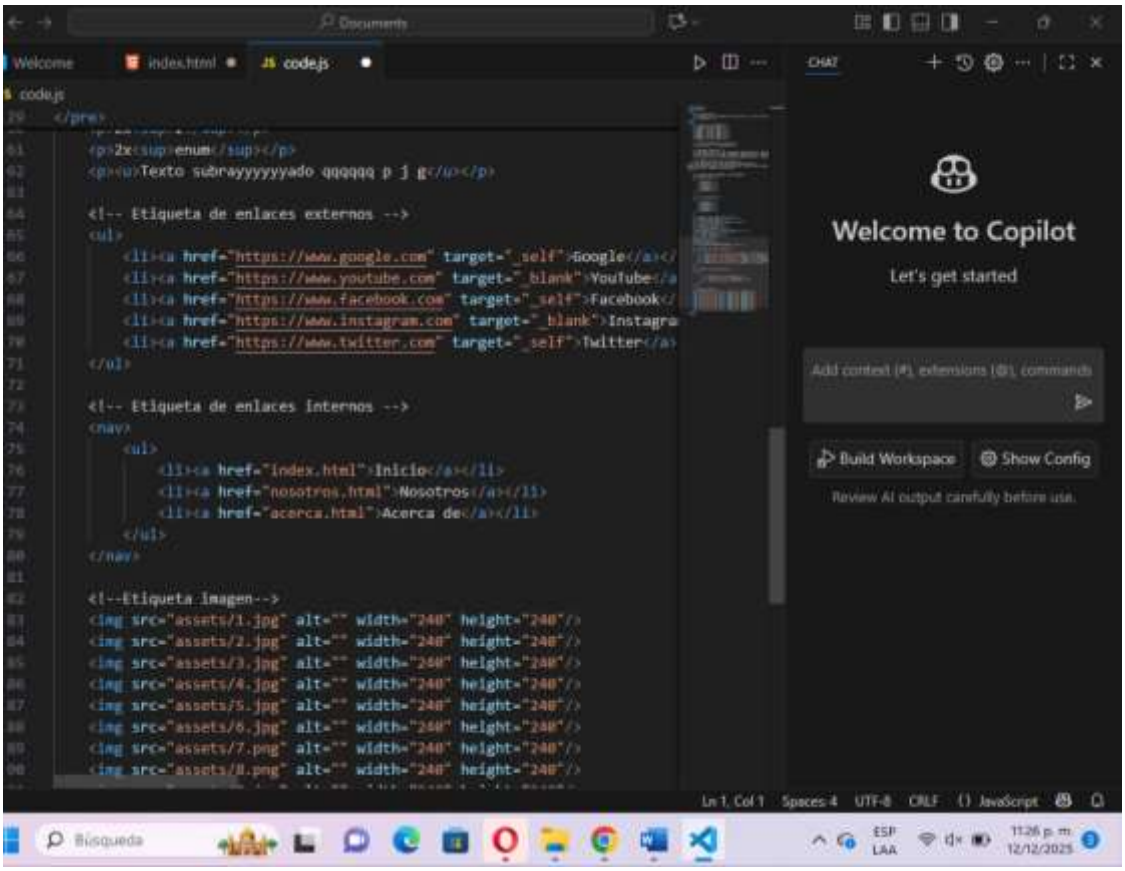
- Presentar información general sobre la **semana 2** del curso.

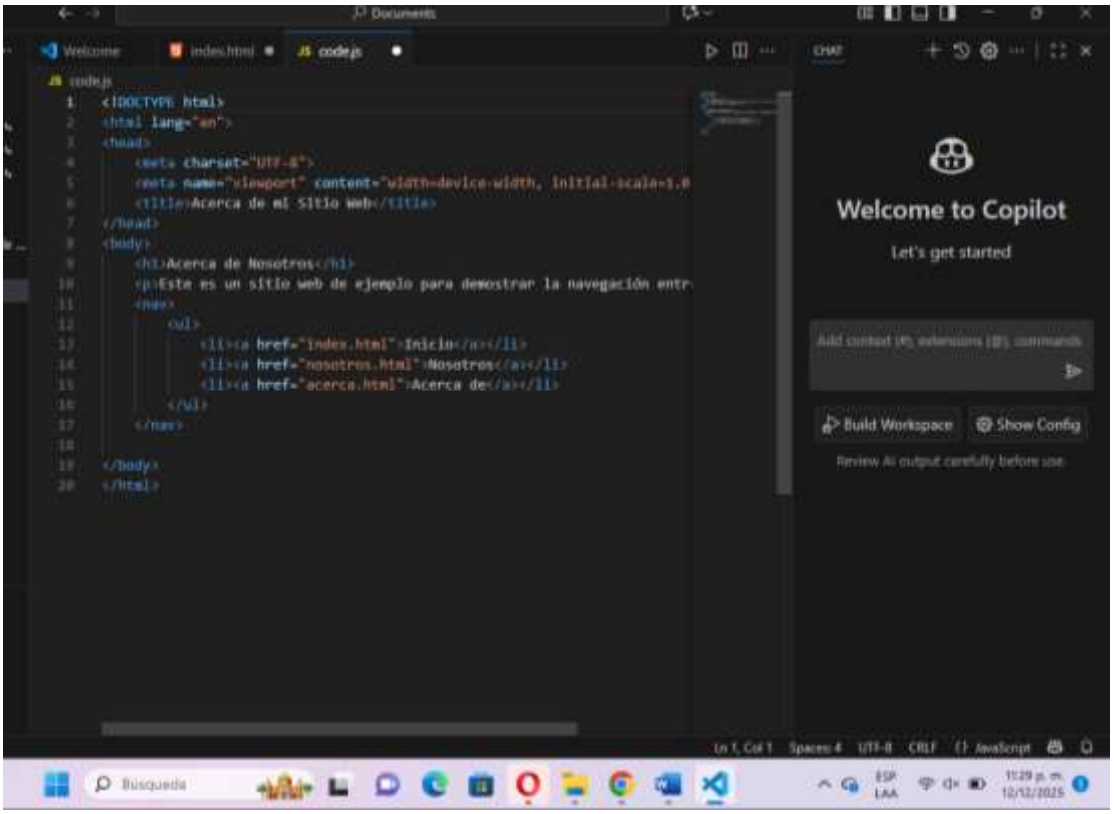
- Explicar el **objetivo pedagógico** de esa lección.

- Indicar qué **temas se van a cubrir** y por qué son relevantes.

- Dar contexto al alumno antes de ver ejemplos o ejercicios prácticos.




Welcome
index.html
code.js
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Búsqueda


Welcome
index.html
code.js
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Búsqueda

Semana 3

El objetivo principal del repositorio es **facilitar la comprensión teórica y práctica** de las estructuras de datos mediante:

- Implementaciones funcionales en **JavaScript**.

- Visualización de estructuras dentro de un entorno web.

- Organización modular del contenido para su uso en un curso semestral.

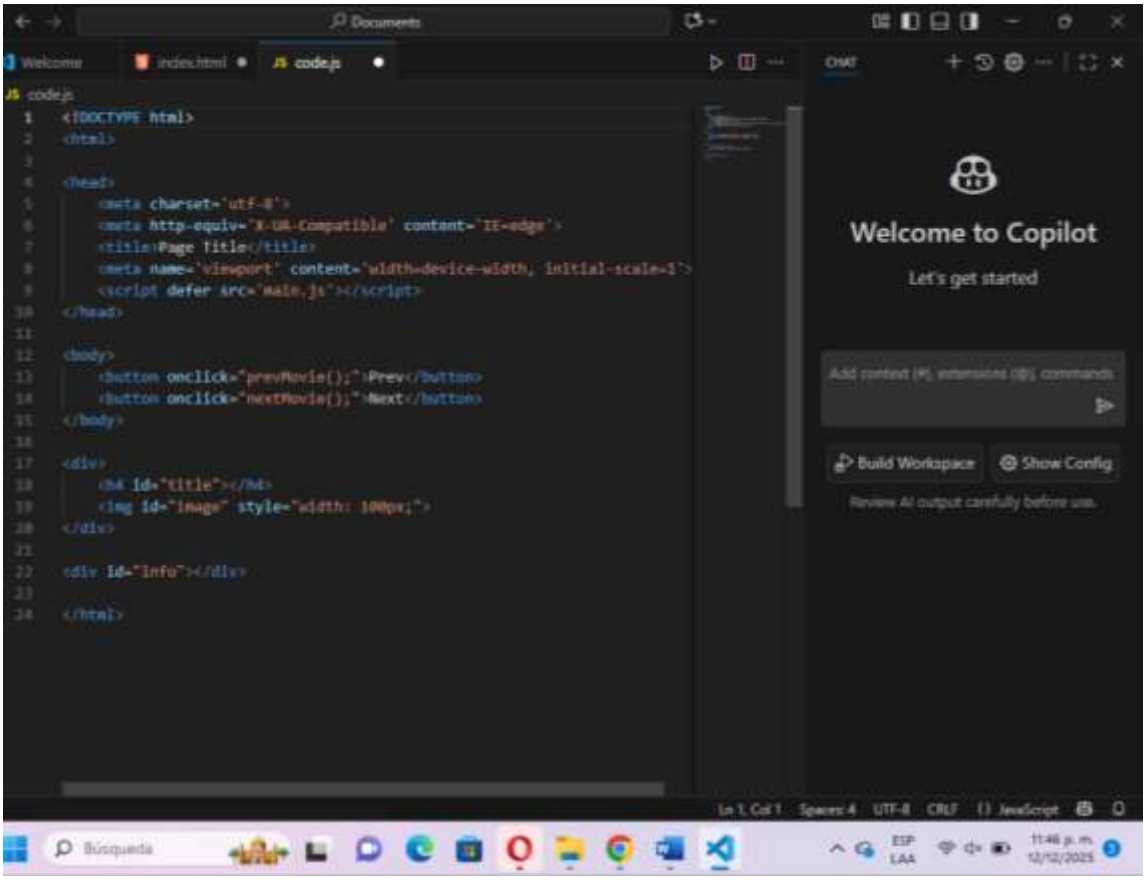
Este enfoque busca fortalecer las habilidades de análisis, diseño e implementación de estructuras de datos

graphs/

Subcarpeta especializada en el manejo y construcción de grafos.

- **buildGraph.js**

Archivo JavaScript encargado de crear o construir la estructura de un grafo



Semana 4

El código analizado presenta una correcta aplicación de conceptos básicos de programación en JavaScript, como:

Declaración de constantes.

Uso de funciones.

Manejo de parámetros.

Implementación de ciclos for.

Operaciones aritméticas básicas.

Estas funciones son apropiadas para un nivel introductorio o intermedio universitario y sientan las bases para el desarrollo de programas más complejos.

number: Constante de tipo entero utilizada como valor base para operaciones aritméticas.

decimal: Constante de tipo decimal (float), declarada como ejemplo de números con punto flotante.

legibleNumber: Constante entera escrita usando separadores (_) para mejorar la legibilidad del número.

Análisis

El uso de constantes garantiza que los valores no cambien durante la ejecución del programa. El separador numérico mejora la claridad al trabajar con cantidades grandes.

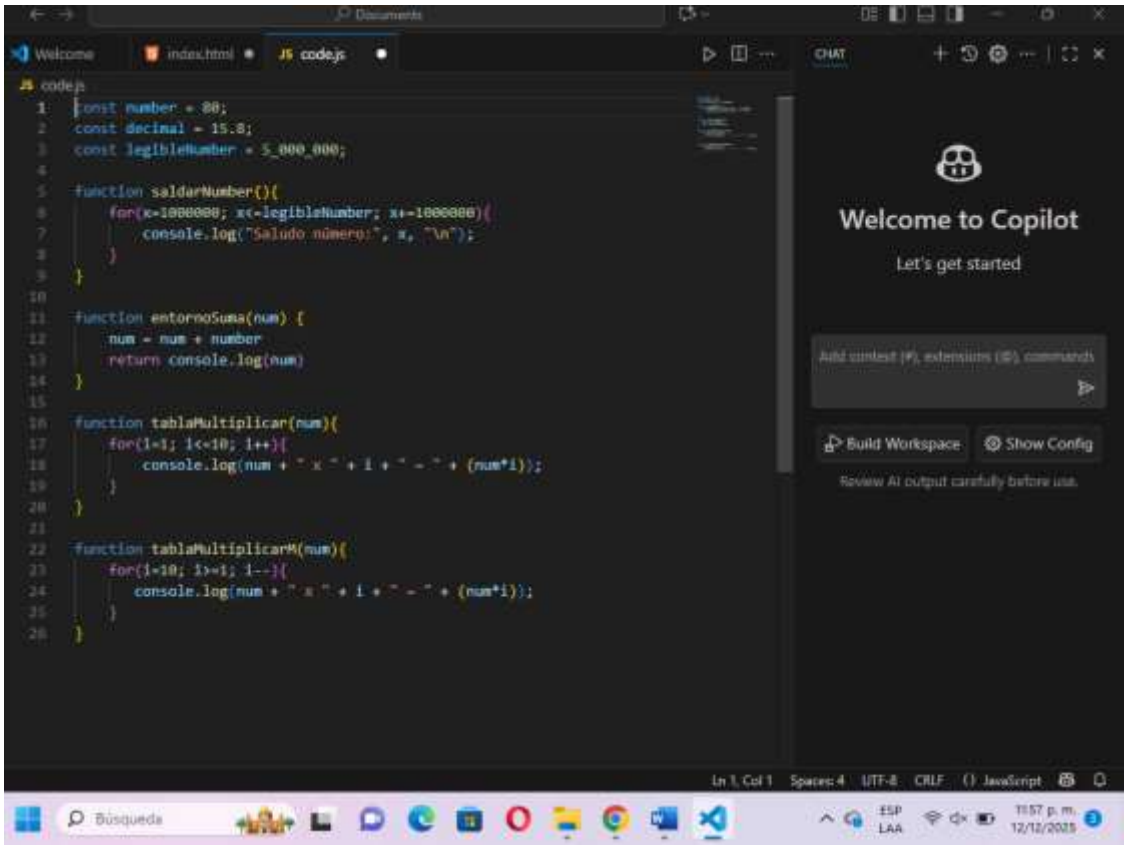
Objetivo

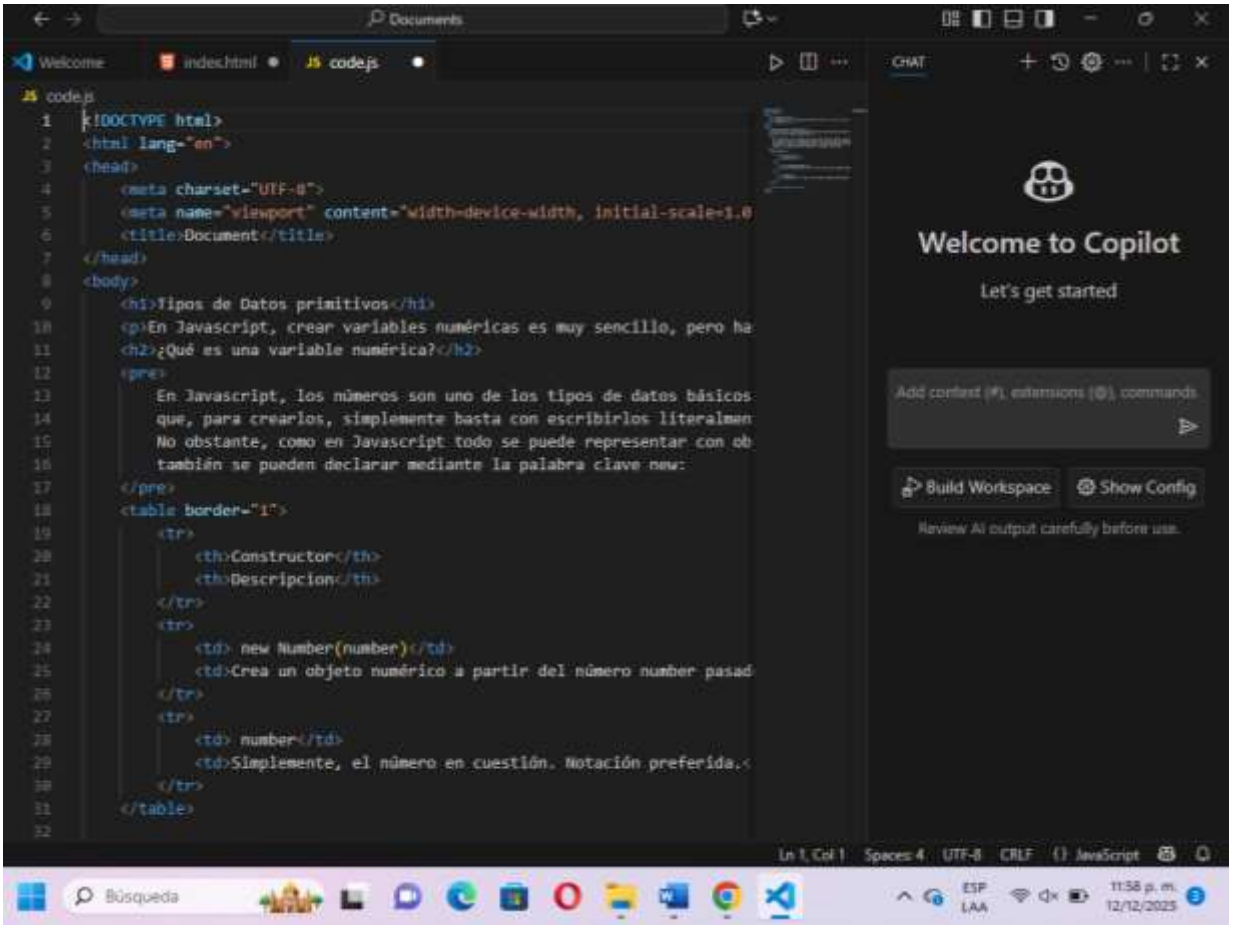
Ilustrar el paso de parámetros a una función.

Mostrar la interacción entre variables locales y globales.

Análisis

La función modifica el valor local del parámetro y muestra el resultado en consola. Aunque utiliza return, el valor retornado es el resultado del console.log, lo cual no es estrictamente necesario, pero es funcional para fines demostrativos.






```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    <title>Document</title>
</head>
<body>
    <h1>Tipos de Datos primitivos</h1>
    <p>En Javascript, crear variables numéricas es muy sencillo, pero ha
    <h2>¿Qué es una variable numérica?</h2>
    <pre>
        En Javascript, los números son uno de los tipos de datos básicos
        que, para crearlos, simplemente basta con escribirlos literalmen
        No obstante, como en Javascript todo se puede representar con ob
        también se pueden declarar mediante la palabra clave new:
    </pre>
    <table border="1">
        <tr>
            <th>Constructor</th>
            <th>Descripcion</th>
        </tr>
        <tr>
            <td> new Number(number)</td>
            <td>Crea un objeto numérico a partir del número number pasad
        </tr>
        <tr>
            <td> number</td>
            <td>Simplemente, el número en cuestión. Notación preferida.<
        </tr>
    </table>
```

Semana 5

El código en main.js está diseñado para:

Presentar ejemplos de declaración y uso de funciones en JavaScript.

Demostrar cómo recibir parámetros, procesar valores y devolver resultados.

Mostrar estructuras de control internas como condicionales y bucles dentro de funciones.

Reforzar buenas prácticas de programación al dividir procedimientos en funciones reutilizables.

MDN Web Docs

3. Lenguaje y Paradigmas Utilizados

El código se basa en:

JavaScript (ES5/ES6) como lenguaje principal.

Paradigma estructurado e imperativo, que usa funciones como bloques de lógica.

MDN Web Docs

En JavaScript, las funciones son ciertas entidades de primera clase, lo que significa que pueden ser pasadas, retornadas, e incluso almacenadas como valores

Parámetros y Argumentos

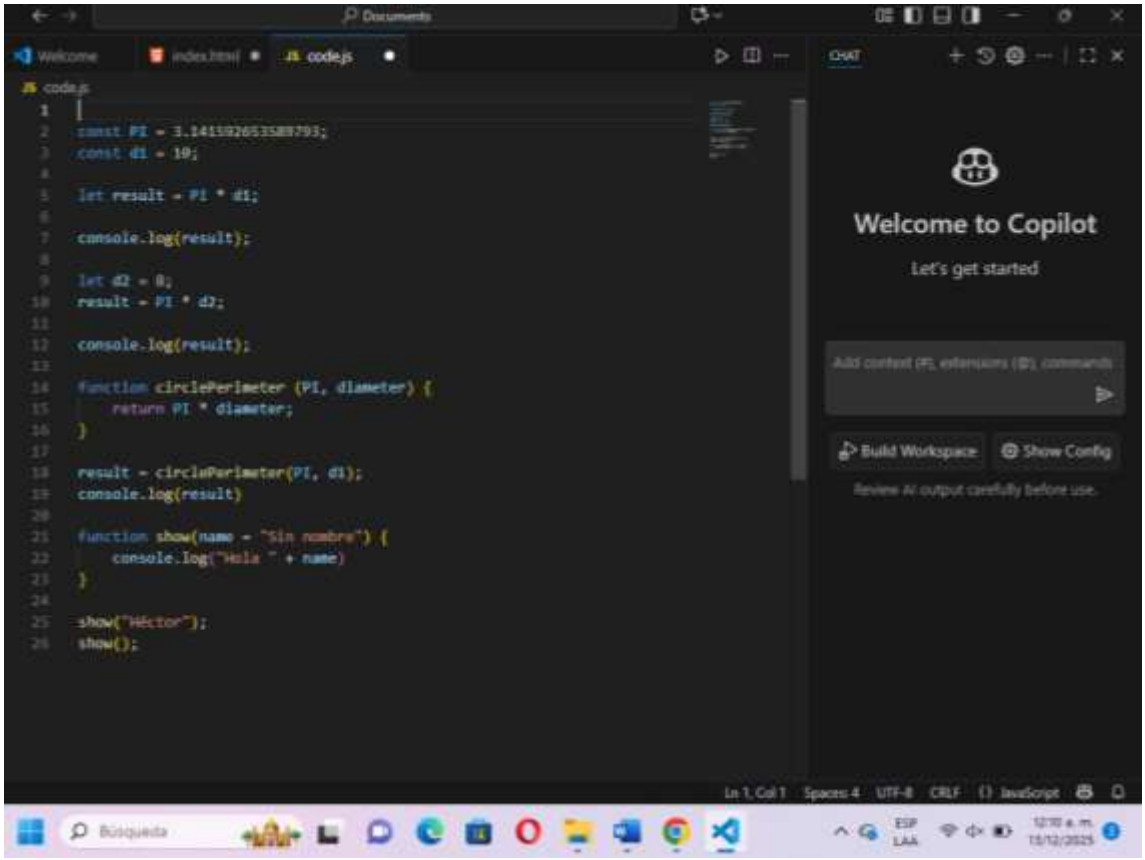
El código usualmente define funciones que reciban valores desde otras partes del programa para procesarlos.

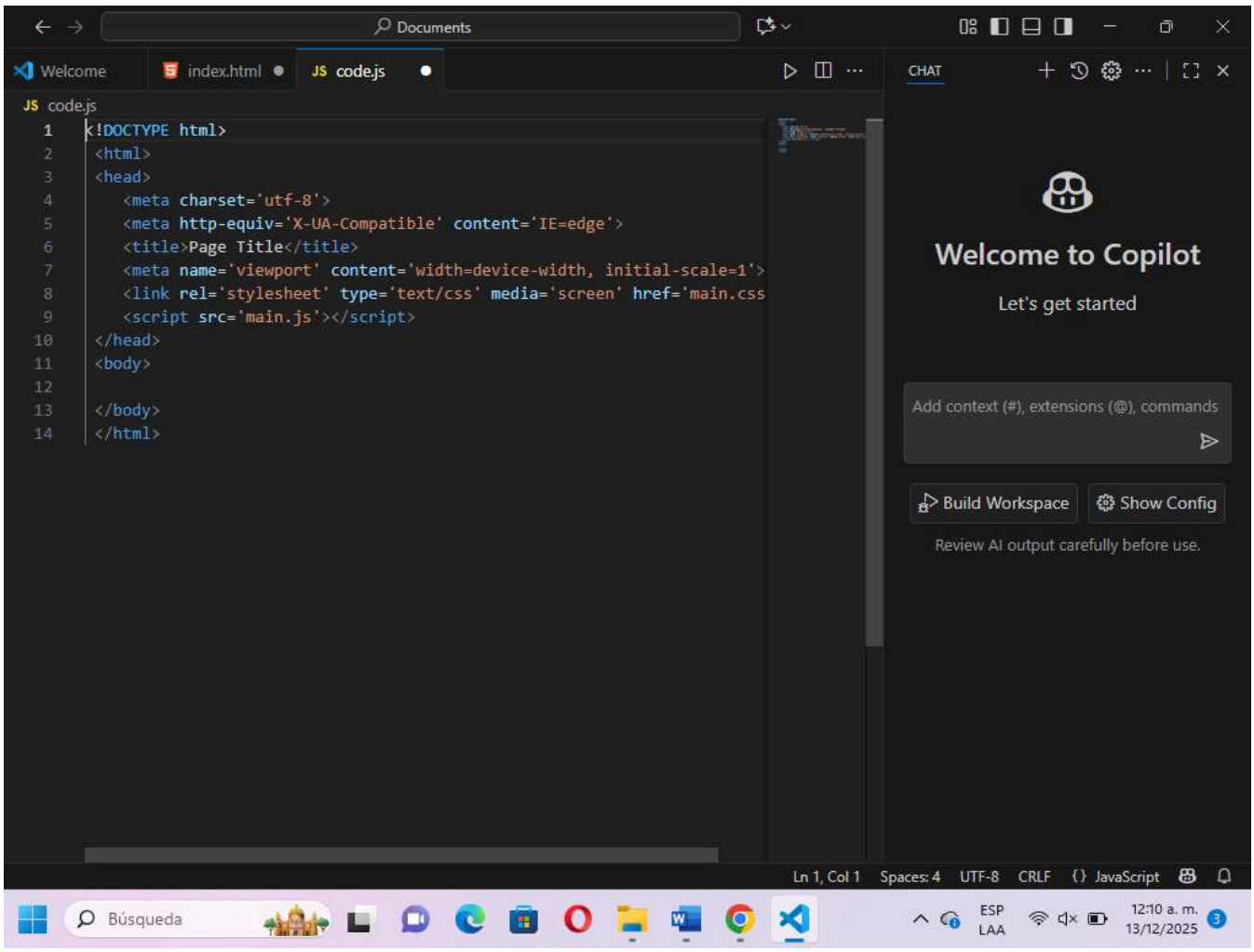
- Retorno de Valores

Se espera que varias funciones devuelvan resultados usando return, que es la forma estándar de pasar datos de regreso al contexto que las invocó.

- Invocación de Funciones

Las funciones definidas son llamadas varias veces desde distintos puntos del main.js para demostrar cómo operan con diferentes datos.




Documents
Welcome
index.html
code.js
code.js
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8'>
<meta http-equiv='X-UA-Compatible' content='IE=edge'>
<title>Page Title</title>
<meta name='viewport' content='width=device-width, initial-scale=1'>
<link rel='stylesheet' type='text/css' media='screen' href='main.css
<script src='main.js'></script>
</head>
<body>
</body>
</html>
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Búsqueda
ESP
LAA
12:10 a. m.
13/12/2025

Semana 6

El objetivo principal del código contenido en la Semana 6 es:

Consolidar el uso de funciones como bloques reutilizables de código.

Aplicar estructuras de control como ciclos y condicionales.

Fortalecer la lógica algorítmica del estudiante mediante ejercicios prácticos.

Preparar al alumno para estructuras de datos más complejas en semanas posteriores.

Lenguaje y Tecnologías Utilizadas

JavaScript

Lenguaje principal utilizado para implementar la lógica del programa y las funciones.

HTML (si aplica)

Usado como medio de ejecución y visualización del código JavaScript en el navegador.

CSS (si aplica)

Empleado únicamente con fines visuales y de presentación.

El enfoque principal es la lógica del código, no el diseño visual.

El código de la Semana 6 está diseñado para demostrar:

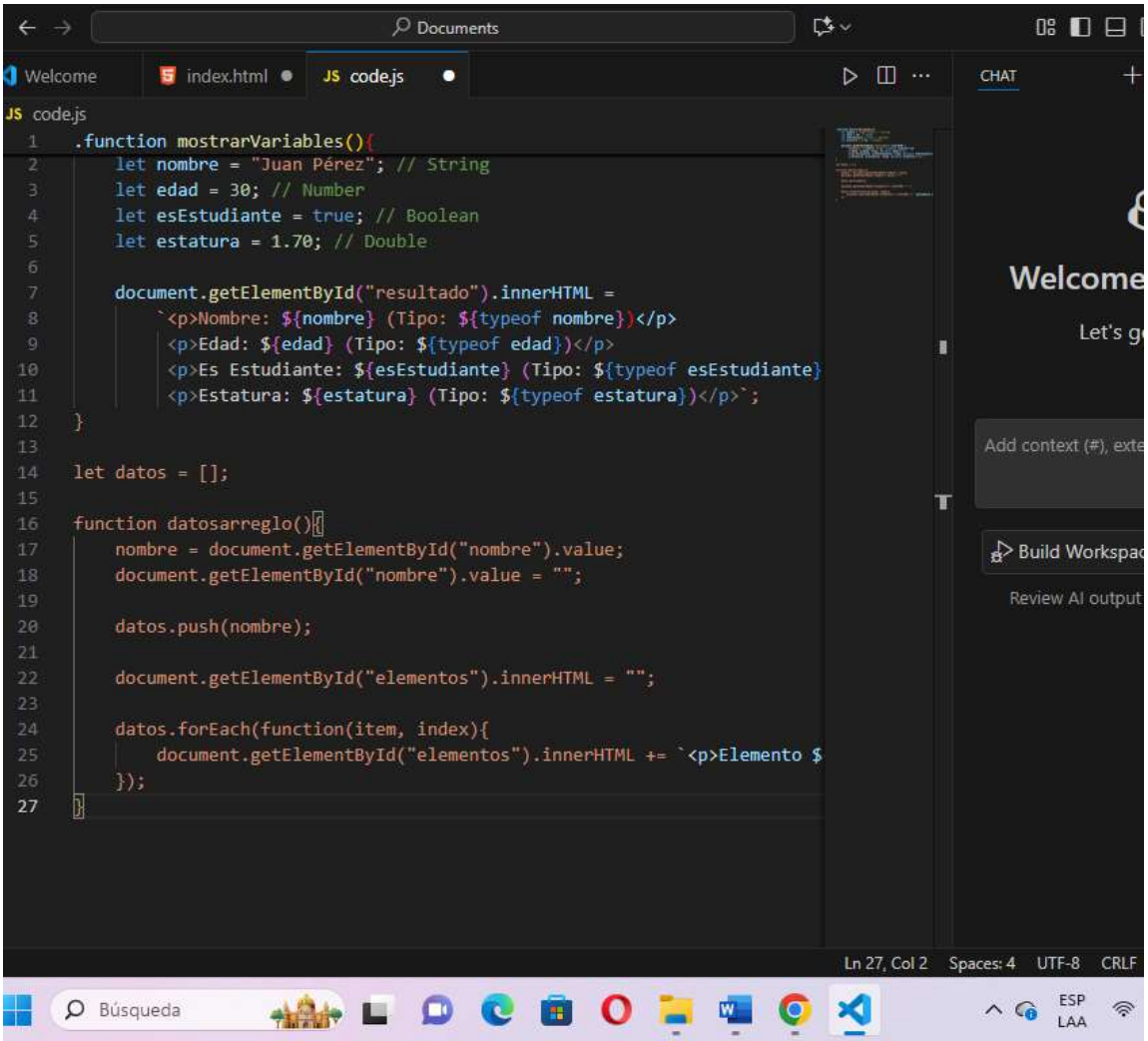
Definición y uso de funciones personalizadas.

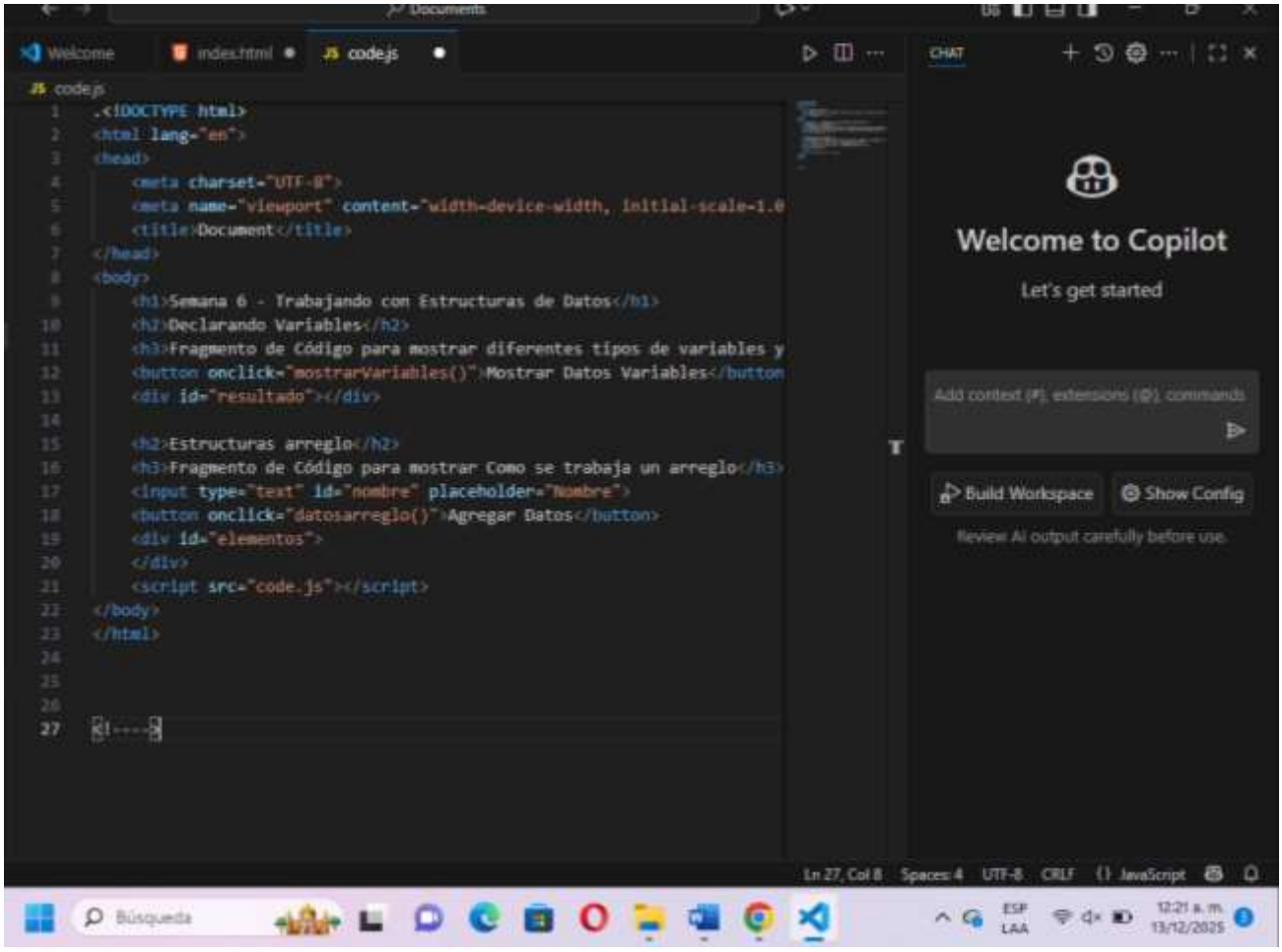
Paso de parámetros a funciones.

Uso de variables locales y globales.

Implementación de ciclos (for, while) para repetir procesos.

Uso de operaciones aritméticas y lógicas.




Welcome
index.html
code.js
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 27, Col 8
Spaces: 4
UTF-8
CRLF
JavaScript
Búsqueda
ESP
LAA
12:21 a. m.
13/12/2025

Semana 7

Objetivo de la Semana 7

El objetivo principal del código desarrollado en la Semana 7 es:

Aplicar correctamente funciones para modularizar el programa.

Utilizar estructuras repetitivas y condicionales para resolver problemas.

Mejorar la capacidad de análisis lógico del estudiante.

Fortalecer la transición entre programación básica y estructuras de datos

Lenguaje y Tecnologías Utilizadas

JavaScript

Lenguaje principal e

mpleado para la implementación de la lógica del programa.

HTML (si aplica)

Utilizado como entorno de ejecución para el código JavaScript en el navegador.

CSS (si aplica)

Empleado únicamente para mejorar la presentación visual.

El código contenido en la Semana 7 está diseñado para:

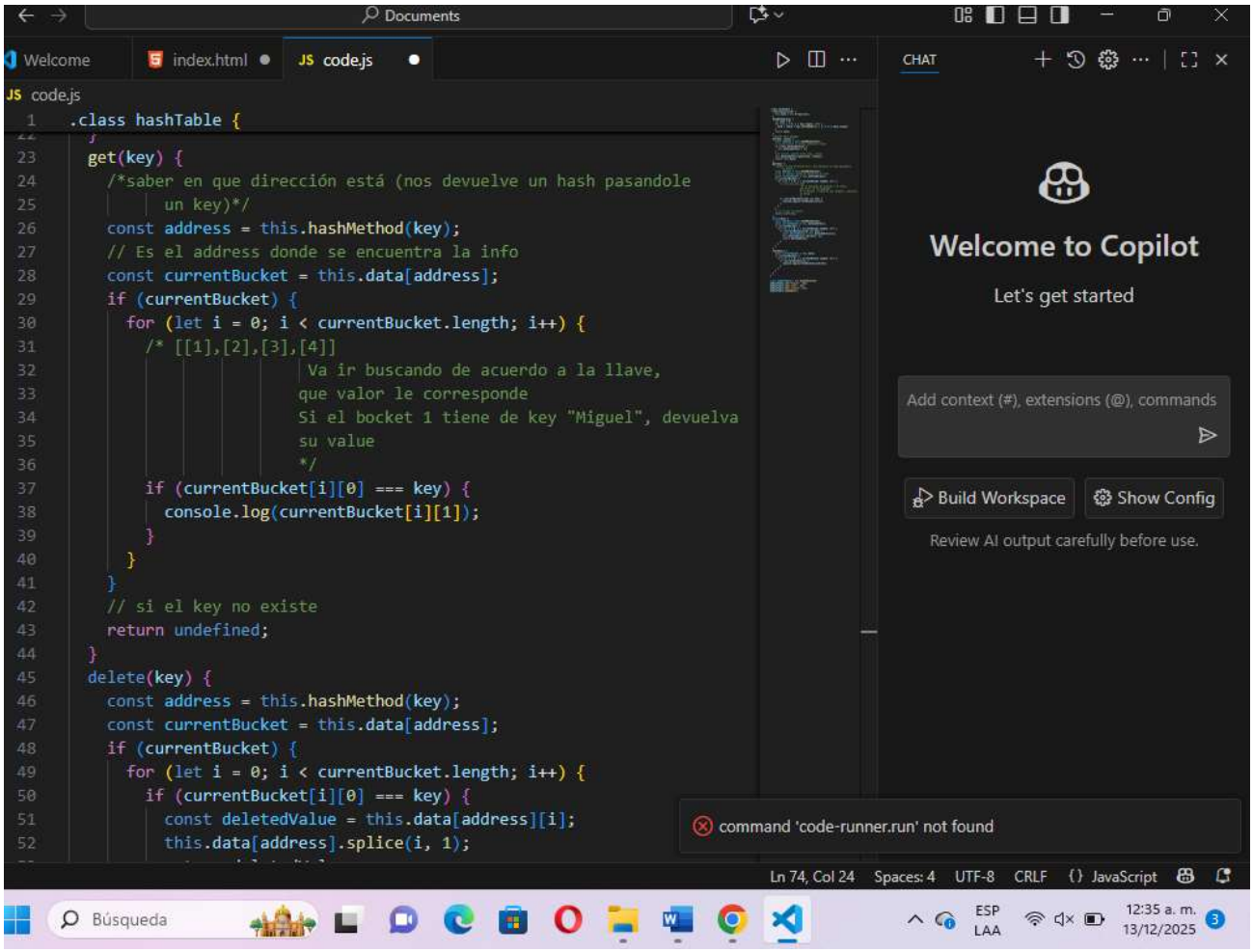
Ejecutar procesos lógicos mediante funciones.

Manipular valores numéricos o datos de entrada.

Repetir operaciones mediante ciclos.

Mostrar resultados a través de la consola o la interfaz web.

Cada función cumple un propósito específico, lo que permite dividir el problema general en tareas más pequeñas y comprensibles.



Semana 8

El objetivo principal del archivo addNode.js es:

Implementar la operación de agregar un nodo a una lista enlazada.

Comprender el uso de referencias (next) entre nodos.

Aplicar programación estructurada mediante funciones.

Reforzar el concepto de estructuras de datos dinámicas.

3. Lenguaje y Tecnologías Utilizadas

JavaScript

Lenguaje utilizado para la implementación de la lista enlazada y la manipulación de nodos.

El código se ejecuta en un entorno web o de consola, priorizando la lógica algorítmica sobre la interfaz gráfica.

Operación principal

La función addNode permite añadir dinámicamente elementos, evitando el uso de arreglos de tamaño fijo.

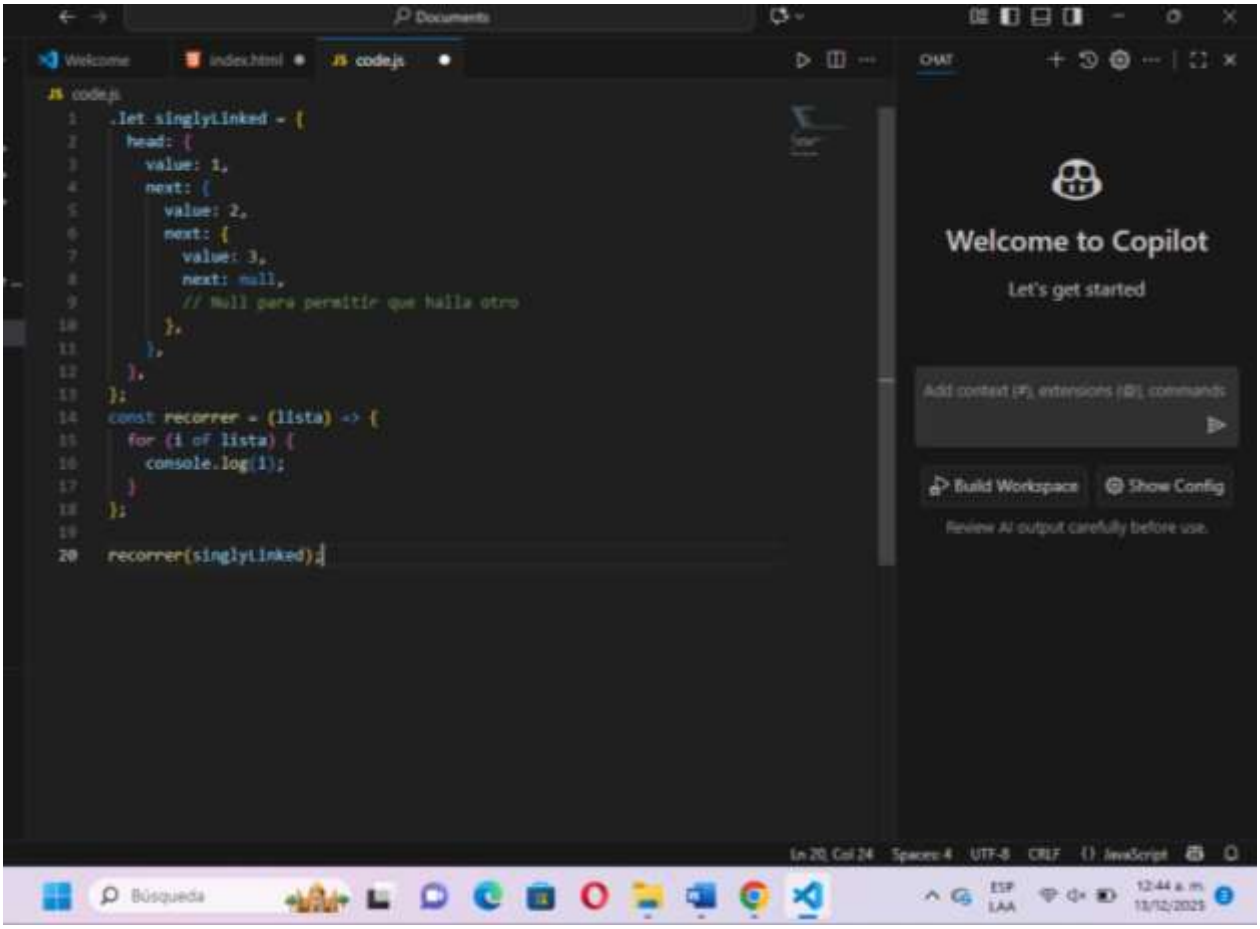
6.2 Uso de referencias

Cada nodo mantiene una referencia al siguiente (next), lo que permite recorrer la lista sin necesidad de índices.

6.3 Complejidad

Tiempo: O(n), si la inserción se realiza al final de la lista.

Espacio: O(1), ya que solo se agrega un nuevo nodo.



Semana 9 – Objetivo

length: almacena la cantidad de elementos.

data: es un objeto donde se guardan los valores por índice

También hay varias pruebas comentadas donde se muestran ejemplos de cómo:

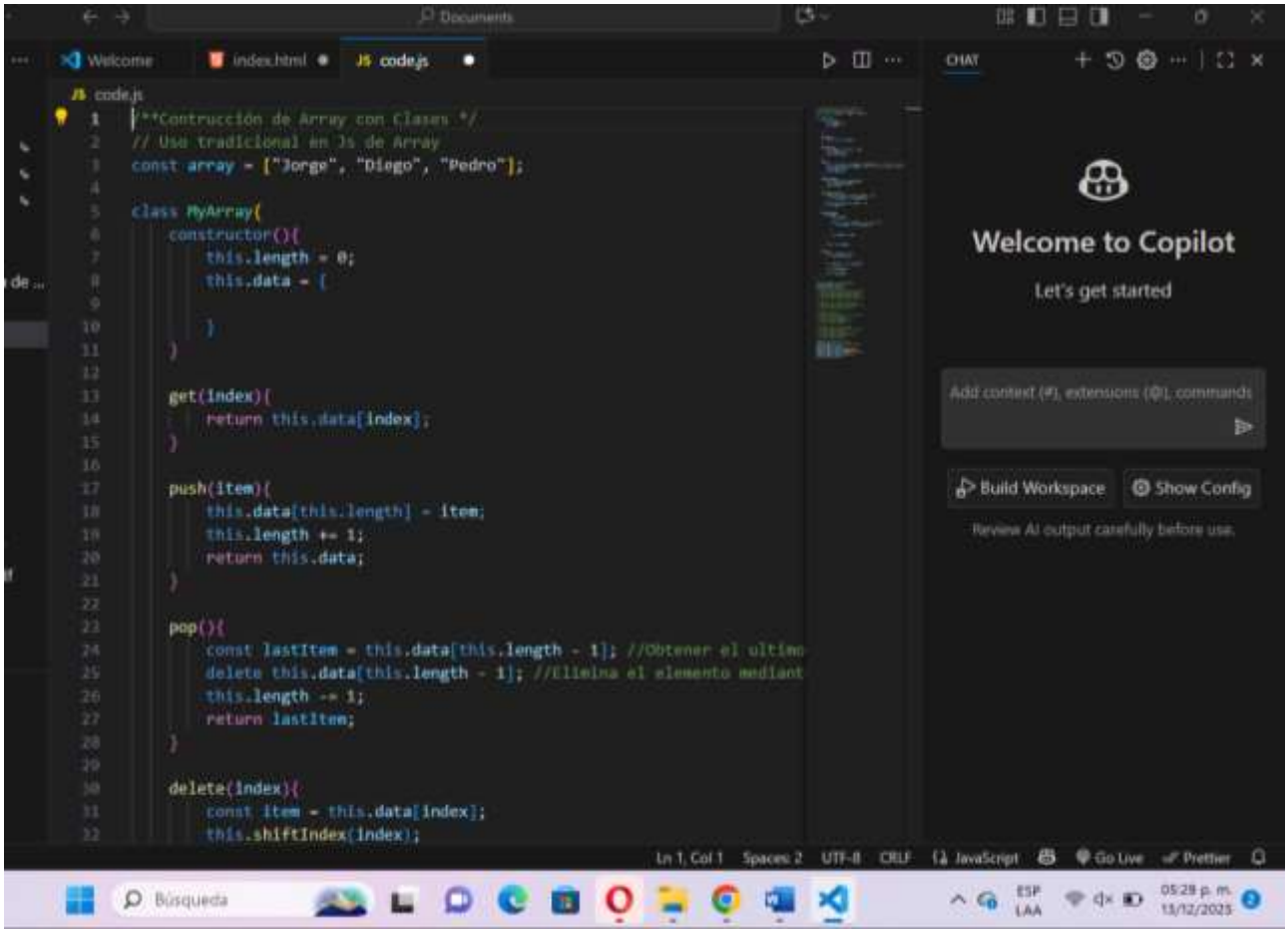
agregar con push(),

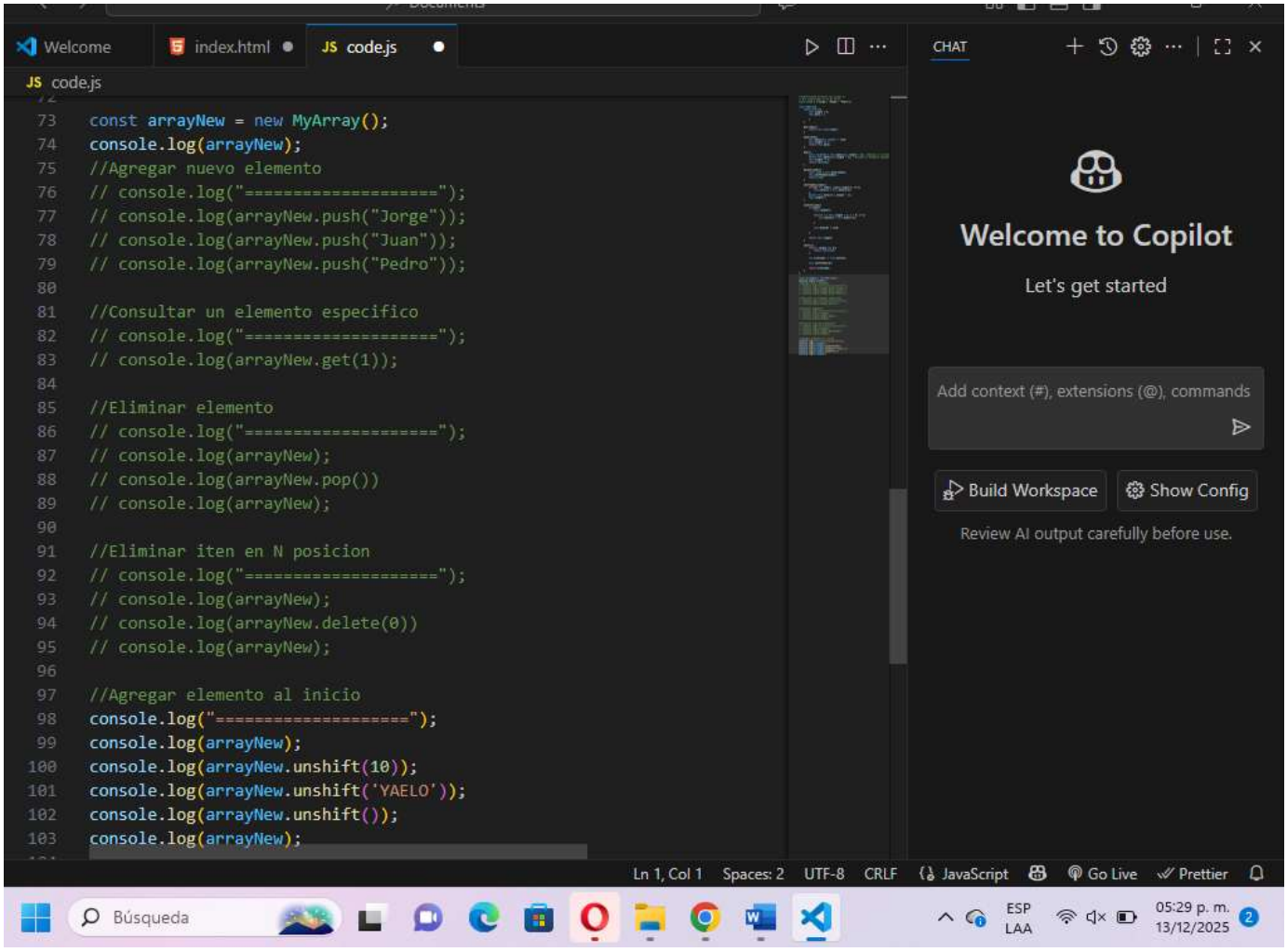
consultar con get(),

eliminar con pop(),

eliminar en una posición con delete(),agregar al inicio con unshift().

Las líneas con unshift() se ejecutan y despliegan resultados.




Welcome
index.html
code.js
JS code.js
const arrayNew = new MyArray();
console.log(arrayNew);
//Agregar nuevo elemento
// console.log("====================");
// console.log(arrayNew.push("Jorge"));
// console.log(arrayNew.push("Juan"));
// console.log(arrayNew.push("Pedro"));
//Consultar un elemento especifico
// console.log("====================");
// console.log(arrayNew.get(1));
//Eliminar elemento
// console.log("====================");
// console.log(arrayNew);
// console.log(arrayNew.pop())
// console.log(arrayNew);
//Eliminar iten en N posicion
// console.log("====================");
// console.log(arrayNew);
// console.log(arrayNew.delete(0))
// console.log(arrayNew);
//Agregar elemento al inicio
console.log("====================");
console.log(arrayNew);
console.log(arrayNew.unshift(10));
console.log(arrayNew.unshift('YAELO'));
console.log(arrayNew.unshift());
console.log(arrayNew);
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live
Prettier
Búsqueda
ESP
LAA
05:29 p. m.
13/12/2025

Semana 10 – Reporte

```
const array = ["Jorge", "Diego", "Pedro"];
```

Aquí se muestra la forma estándar de declarar un arreglo en JavaScript, el cual contiene tres elementos tipo cadena. Este ejemplo sirve como referencia para comparar con la implementación personalizada.

```
constructor(){
    this.length = 0;
    this.data = {};
}
```

length: almacena el número de elementos en el arreglo.

data: objeto que guarda los valores utilizando índices como claves.

push() para agregar elementos

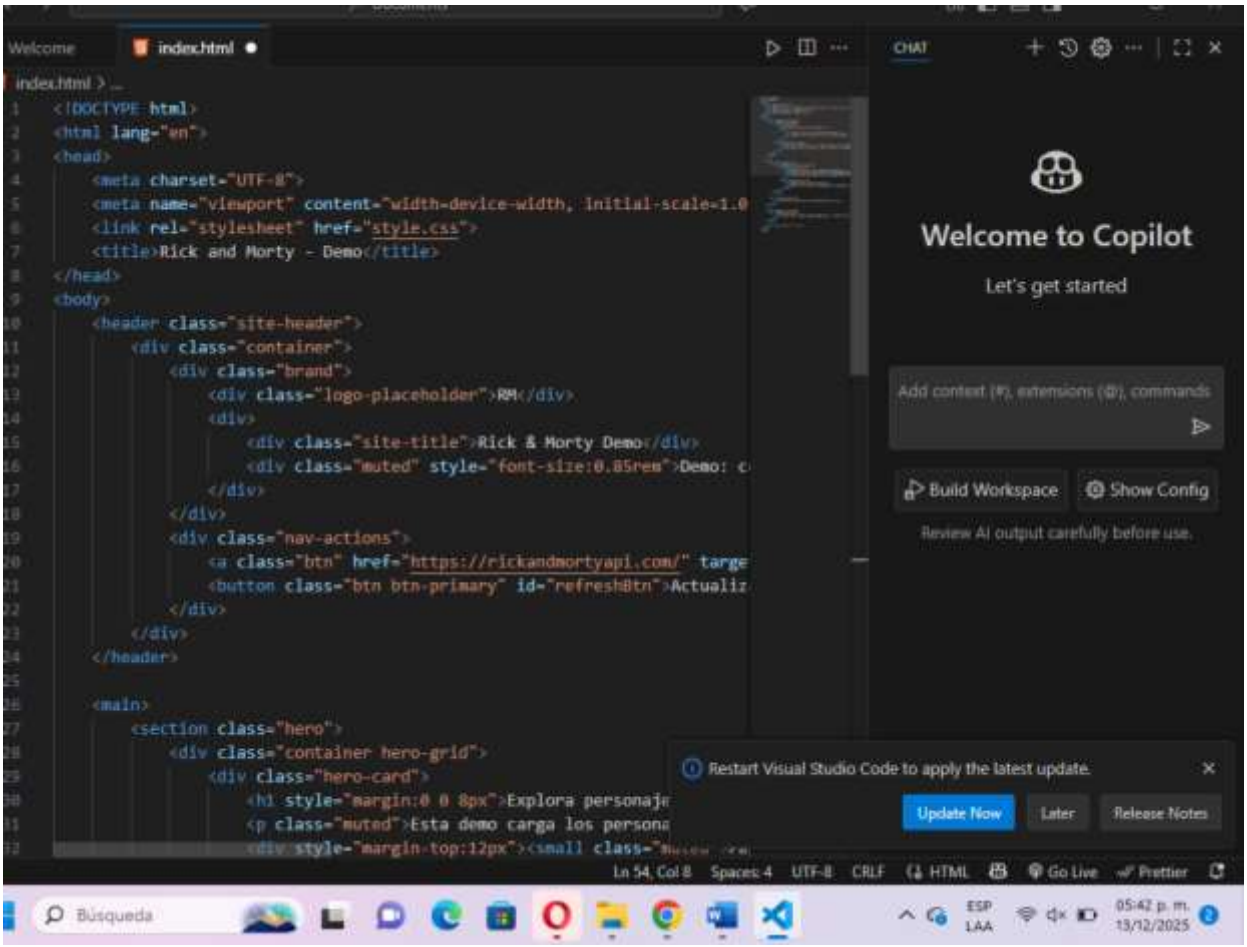
get() para consultar

pop() para eliminar el último

delete() para eliminar por índice

unshift() para insertar al inicio

Las pruebas se muestran mediante console.log().


Welcome
index.html
index.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0
<link rel="stylesheet" href="style.css">
<title>Rick and Morty - Demo</title>
</head>
<body>
<header class="site-header">

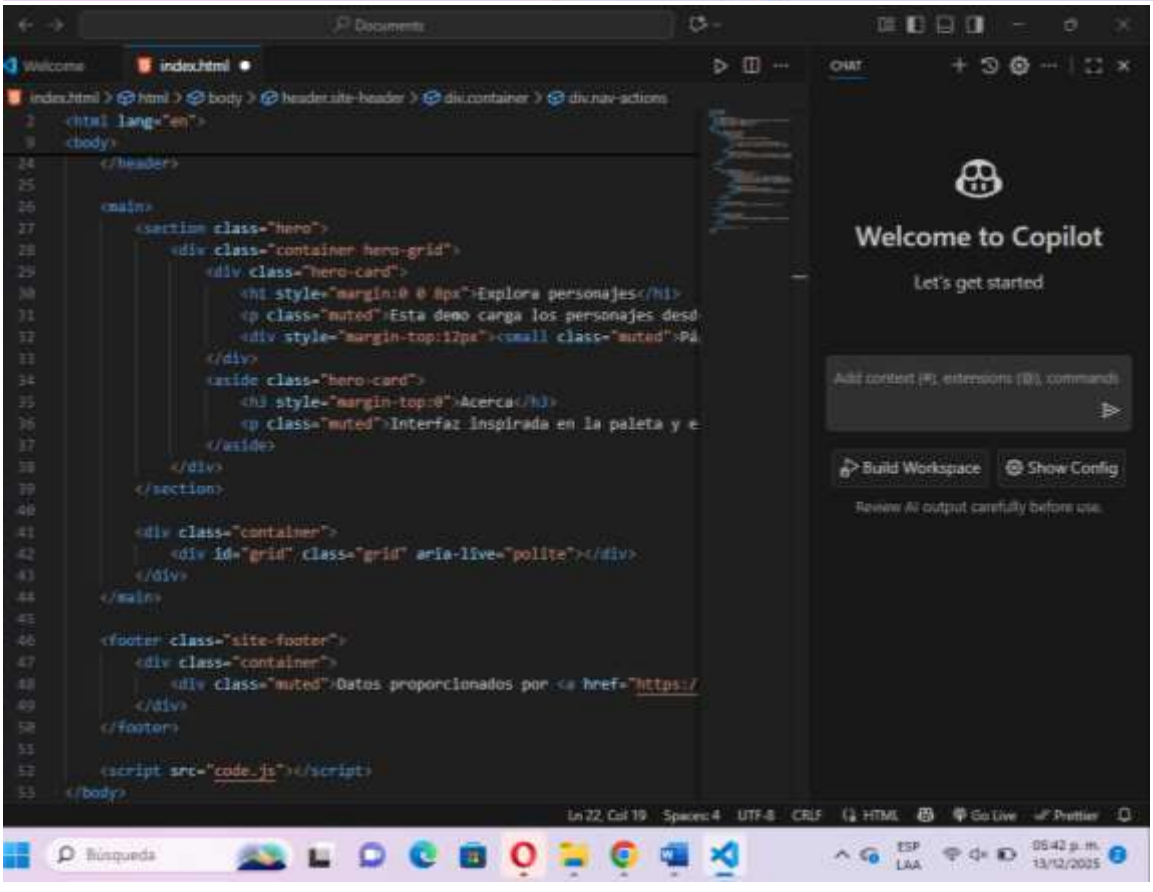
RM

Rick & Morty Demo
Demo: c

<a class="btn" href="https://rickandmortyapi.com/" targe
<button class="btn btn-primary" id="refreshBtn">Actualiz

</header>
<main>
<section class="hero">

<h1 style="margin:0 0 8px">Explora personaje
<p class="muted">Esta demo carga los persona
<small class="
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Restart Visual Studio Code to apply the latest update.
Update Now
Later
Release Notes
Ln 54, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
Búsqueda
ESP
LAA
05:42 p. m.
13/12/2025


Documents
Welcome
index.html
index.html > html > body > header.site-header > div.container > div.nav-actions
<html lang="en">
<body>
</header>
<main>
<section class="hero">

<h1 style="margin:0 0 8px">Explora personajes</h1>
<p class="muted">Esta demo carga los personajes desd
<small class="muted">Pá

<aside class="hero-card">
<h3 style="margin-top:0">Acerca</h3>
<p class="muted">Interfaz inspirada en la paleta y e
</aside>

</section>

</main>
<footer class="site-footer">

Datos proporcionados por <a href="https:/

</footer>
<script src="code.js"></script>
</body>
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 22, Col 19
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
Búsqueda
ESP
LAA
05:42 p. m.
13/12/2025

Semana 11 – Reporte

Funciones principales:

meta charset="UTF-8": Permite el uso de caracteres especiales.

meta viewport: Hace que el sitio sea adaptable a dispositivos móviles.

link rel="stylesheet": Importa el archivo style.css, encargado del diseño visual.

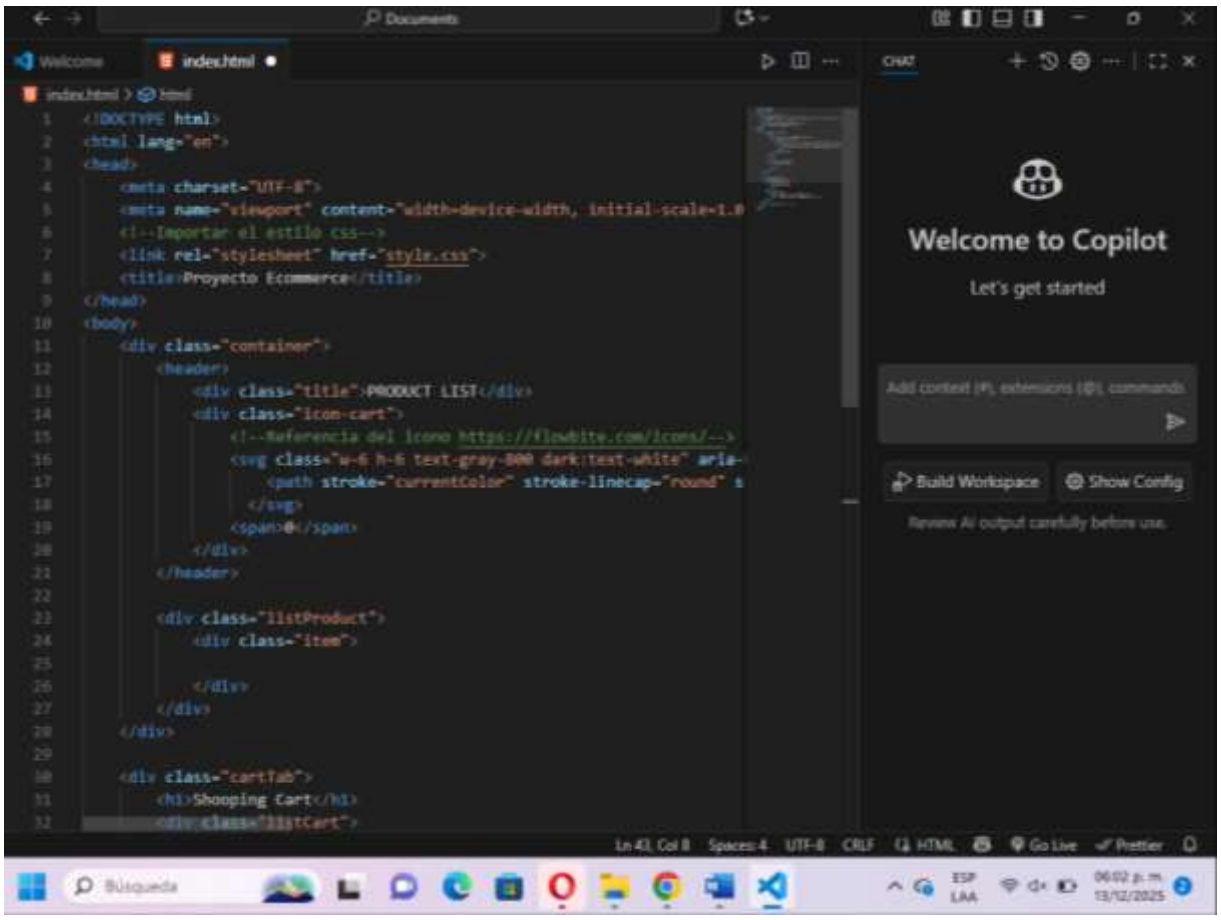
title: Define el título que aparece en la pestaña del navegador.

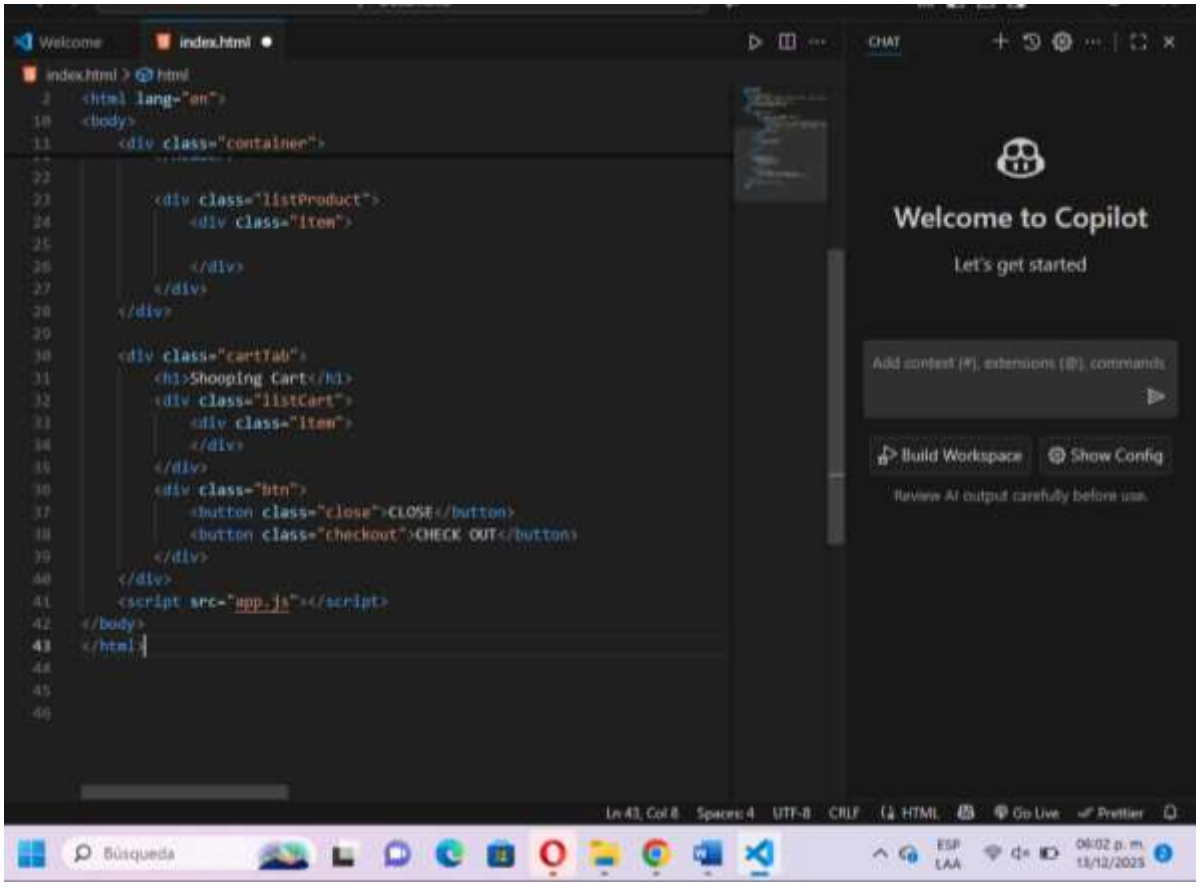
Elementos:

Título "PRODUCT LIST": Indica la sección donde se mostrarán los productos.

Icono del carrito: Representado mediante un SVG, tomado de Flowbite Icons.

Contador (0): Muestra la cantidad de productos agregados al carrito.


Welcome
index.html
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 43, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
Búsqueda


Welcome
index.html
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 43, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
Búsqueda

Semana 12 – Reporte

El presente código CSS tiene como finalidad definir el diseño visual, la estructura y el comportamiento responsivo de una aplicación web. El estilo está orientado a mostrar elementos tipo tarjetas (cards), botones y listas, haciendo uso de variables CSS, flexbox, grid layout y media queries para lograr una interfaz moderna, ordenada y adaptable a distintos tamaños de pantalla.

Se realiza un reset de estilos, eliminando márgenes y rellenos por defecto, y estableciendo:

box-sizing: border-box para un mejor control del tamaño de los elementos

Fuente Rubik como tipografía principal

🥎 Estilos generales de elementos

ul: elimina viñetas de listas

button: elimina bordes y fondos por defecto

body: establece altura mínima completa y color de fondo gris

Información interna del Pokémon

Se definen estilos para:

Imagen (.pokemon-imagen)

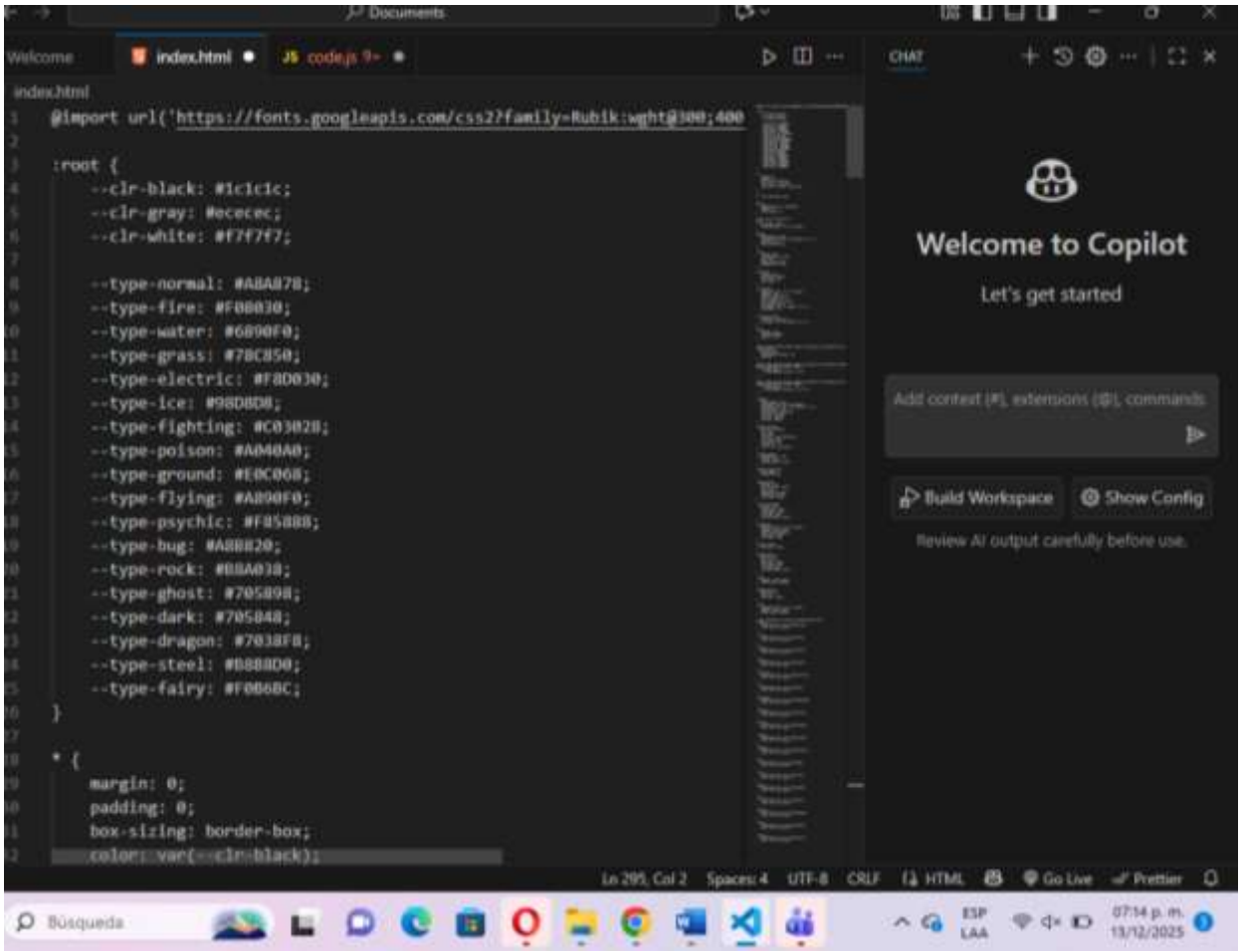
Identificador (.pokemon-id)

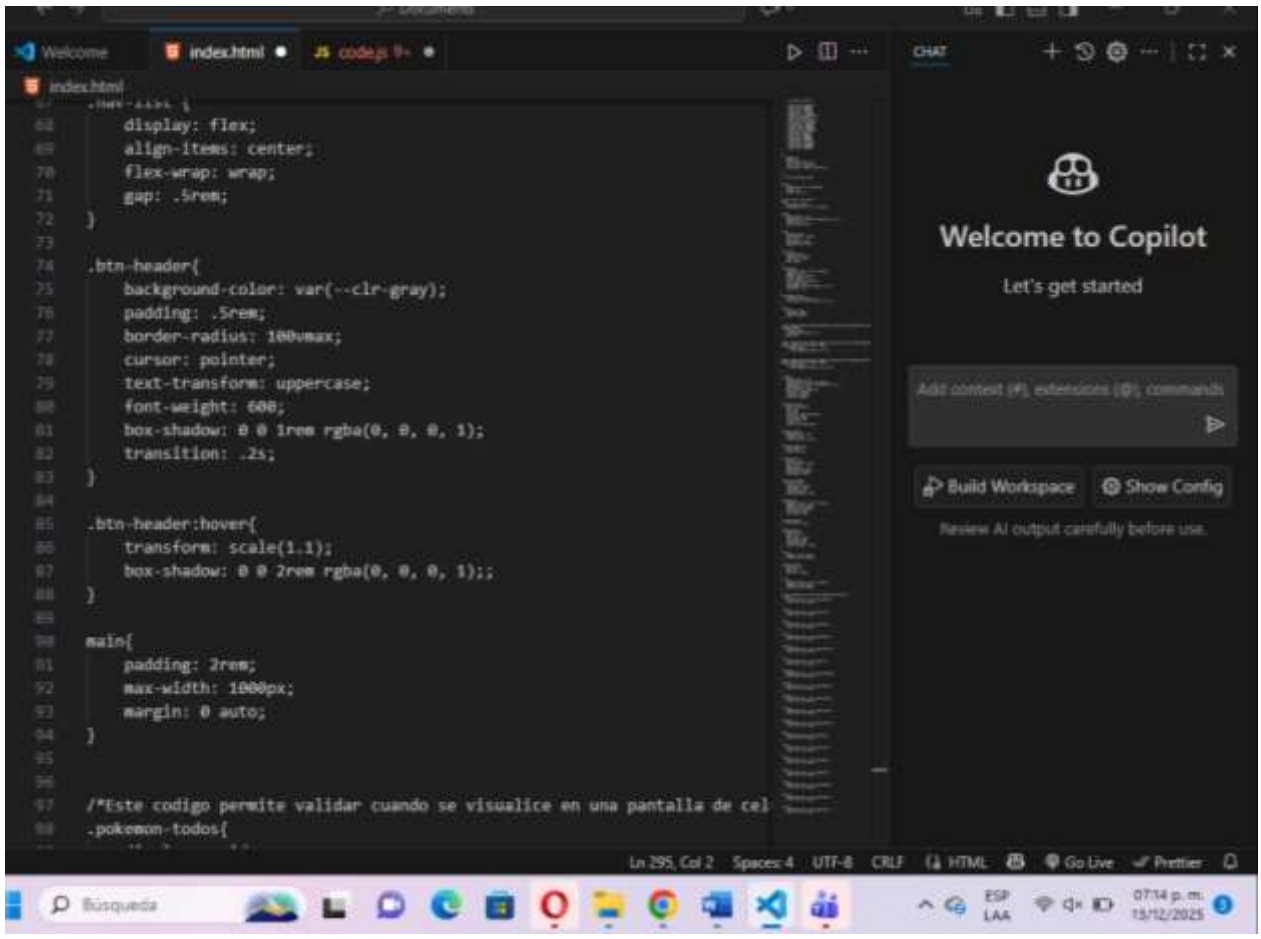
Nombre (.pokemon-nombre)

Tipos (.pokemon-tipos)

Estadísticas (.pokemon-stats)

Todo se organiza usando Flexbox para centrar y distribuir el contenido correctamente.




Welcome
index.html
code.js 9+
index.html
display: flex;
align-items: center;
flex-wrap: wrap;
gap: .5rem;
}
.btn-header{
background-color: var(--clr-gray);
padding: .5rem;
border-radius: 100vmax;
cursor: pointer;
text-transform: uppercase;
font-weight: 600;
box-shadow: 0 0 1rem rgba(0, 0, 0, 1);
transition: .2s;
}
.btn-header:hover{
transform: scale(1.1);
box-shadow: 0 0 2rem rgba(0, 0, 0, 1);;
}
main{
padding: 2rem;
max-width: 1000px;
margin: 0 auto;
}
/*Este codigo permite validar cuando se visualice en una pantalla de cel
.pokemon-todos{
Ln 295, Col 2
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Búsqueda
ESP
LAA
07:14 p. m.
15/12/2025

Semana 13 – Reporte

EduTrack - Sistema de Asistencia y Participación

Descripción general

EduTrack es un sistema completo de gestión de asistencia y participación estudiantil diseñado para instituciones educativas. Ofrece una interfaz moderna y eficiente que permite a los docentes registrar, monitorear y analizar el rendimiento académico de sus estudiantes.

Características principales

📊 Gestión de Asistencia

Registro rápido de asistencia por clase y fecha

Marcación masiva (todos presentes/ausentes)

Registro de llegadas tardías

Historia completa de asistencia por estudiante.

💬 Seguimiento de Participación

Evaluación de participación con sistema de puntuación (1-10)

Categorización por tipo de participación (respuesta, pregunta, participación activa, presentación)

Notas descriptivas para cada registro

Análisis de patrones de participación

📈 Reportes y Análisis

Dashboard con estadísticas en tiempo real

Informes detallados de asistencia y participación

Análisis individual de estudiantes

Vista general por clase

Exportación de datos en formato JSON

🎨 Interfaz moderna

Diseño responsivo y accesible

Animaciones suaves y microinteracciones.

Panel de control intuitivo

Notificaciones en tiempo real

Arquitectura del Sistema

Tecnologías utilizadas

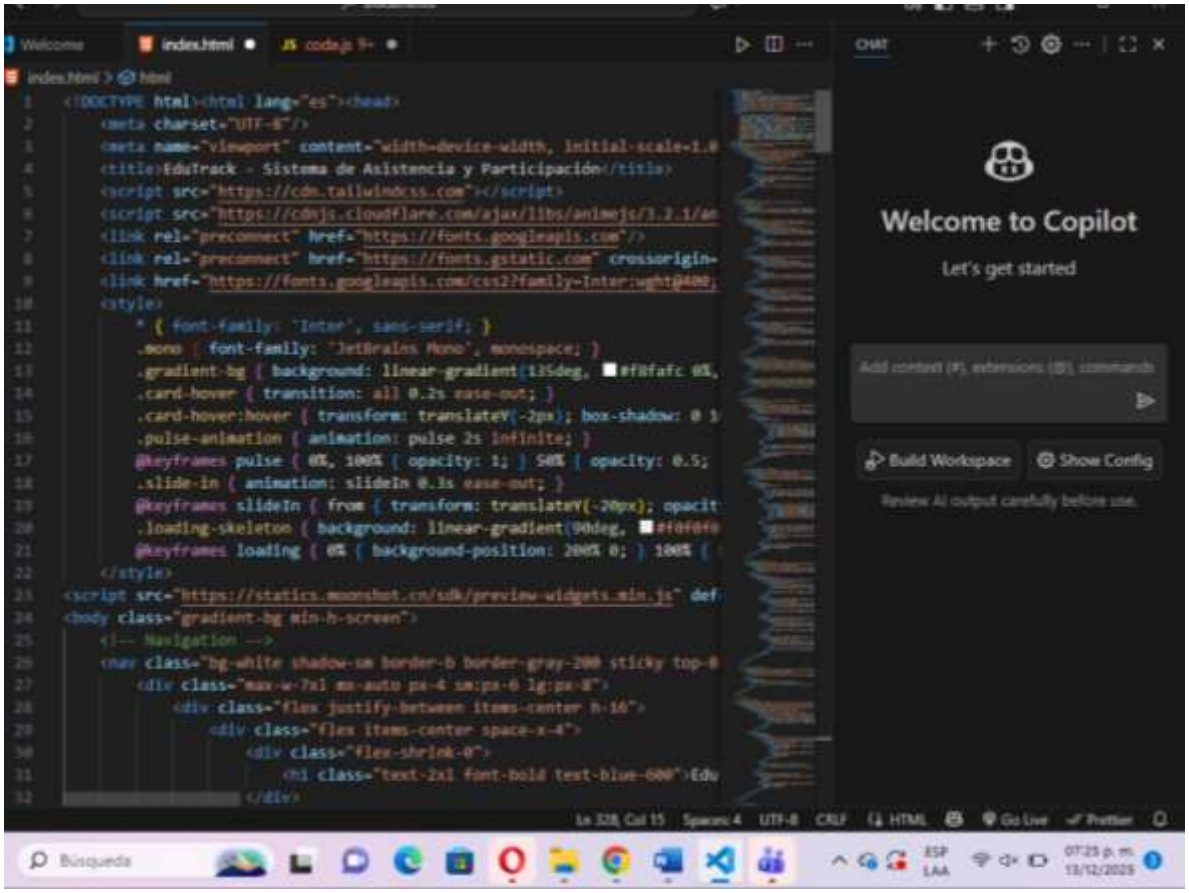
Interfaz : HTML5, CSS3, JavaScript ES6+

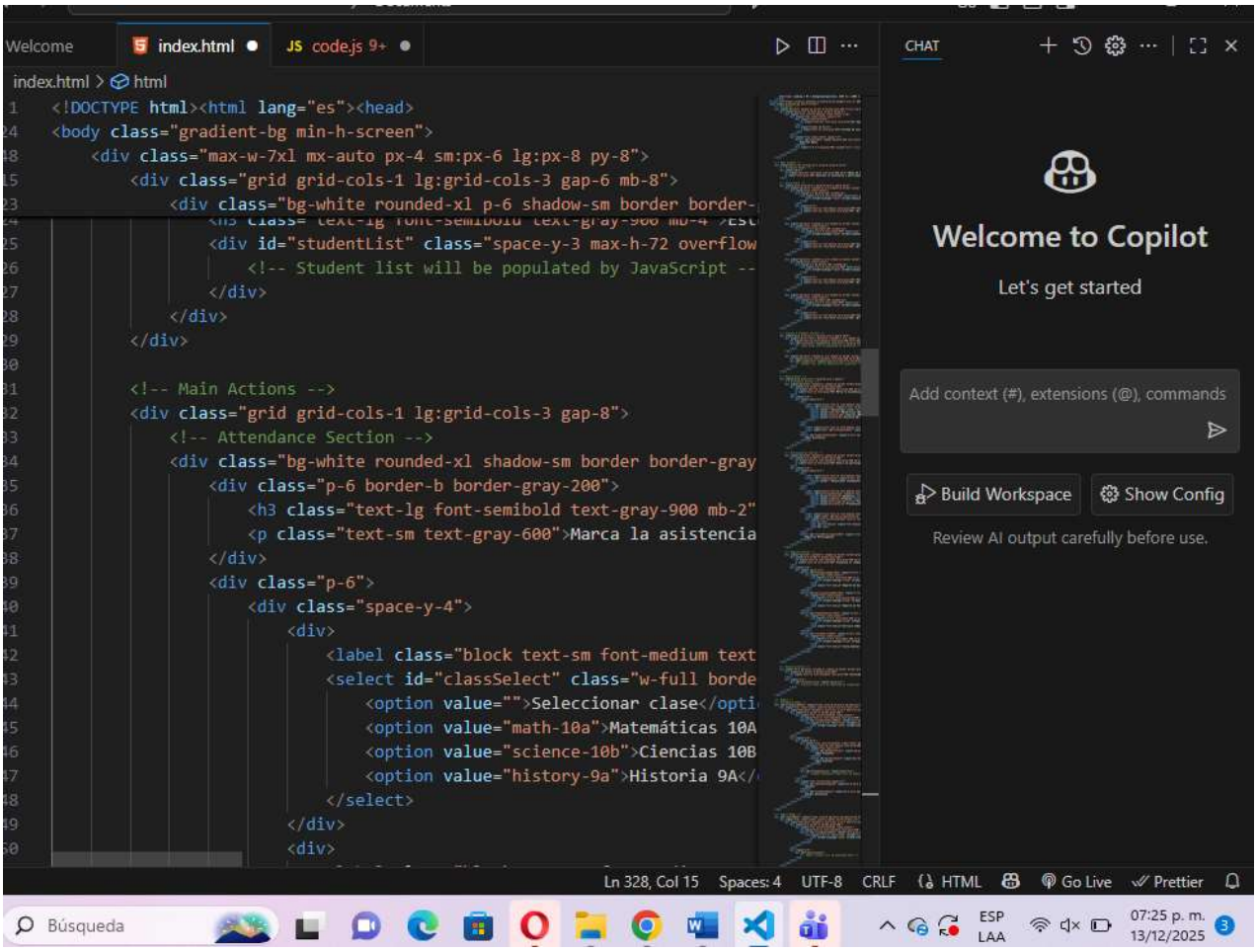
Marco CSS : Tailwind CSS

Animaciones : Anime.js

Almacenamiento : LocalStorage (base de datos local)

Tipografías : Inter (texto principal), JetBrains Mono (datos)




Welcome
index.html
code.js 9+
index.html > html
<!DOCTYPE html><html lang="es"><head>
<body class="gradient-bg min-h-screen">

<div class="bg-white rounded-xl p-6 shadow-sm border border-
<div id="studentList" class="space-y-3 max-h-72 overflow
<!-- Student list will be populated by JavaScript --

<!-- Main Actions -->

<!-- Attendance Section -->
<div class="bg-white rounded-xl shadow-sm border border-gray

<h3 class="text-lg font-semibold text-gray-900 mb-2"
<p class="text-sm text-gray-600">Marca la asistencia

<label class="block text-sm font-medium text
<select id="classSelect" class="w-full borde
<option value="">Seleccionar clase</opti
<option value="math-10a">Matemáticas 10A
<option value="science-10b">Ciencias 10B
<option value="history-9a">Historia 9A</
</select>

CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Ln 328, Col 15
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
Búsqueda
ESP
LAA
07:25 p. m.
13/12/2025

Semana 14 – Reporte

Registro de los datos

Tener un arroglo, dentro de ese arreglo una estructura los datos que solicita son una id, título, descripción, completada.

El presente código HTML corresponde a una aplicación web tipo ToDo, cuyo objetivo principal es permitir al usuario registrar, consultar, editar y eliminar tareas, utilizando almacenamiento local del navegador. El archivo define la estructura base de la interfaz, mientras que el diseño y la lógica se implementan mediante archivos externos de CSS y JavaScript.

Función de los elementos:

meta charset="UTF-8": Permite el uso de caracteres especiales.

meta viewport: Garantiza un diseño responsivo.

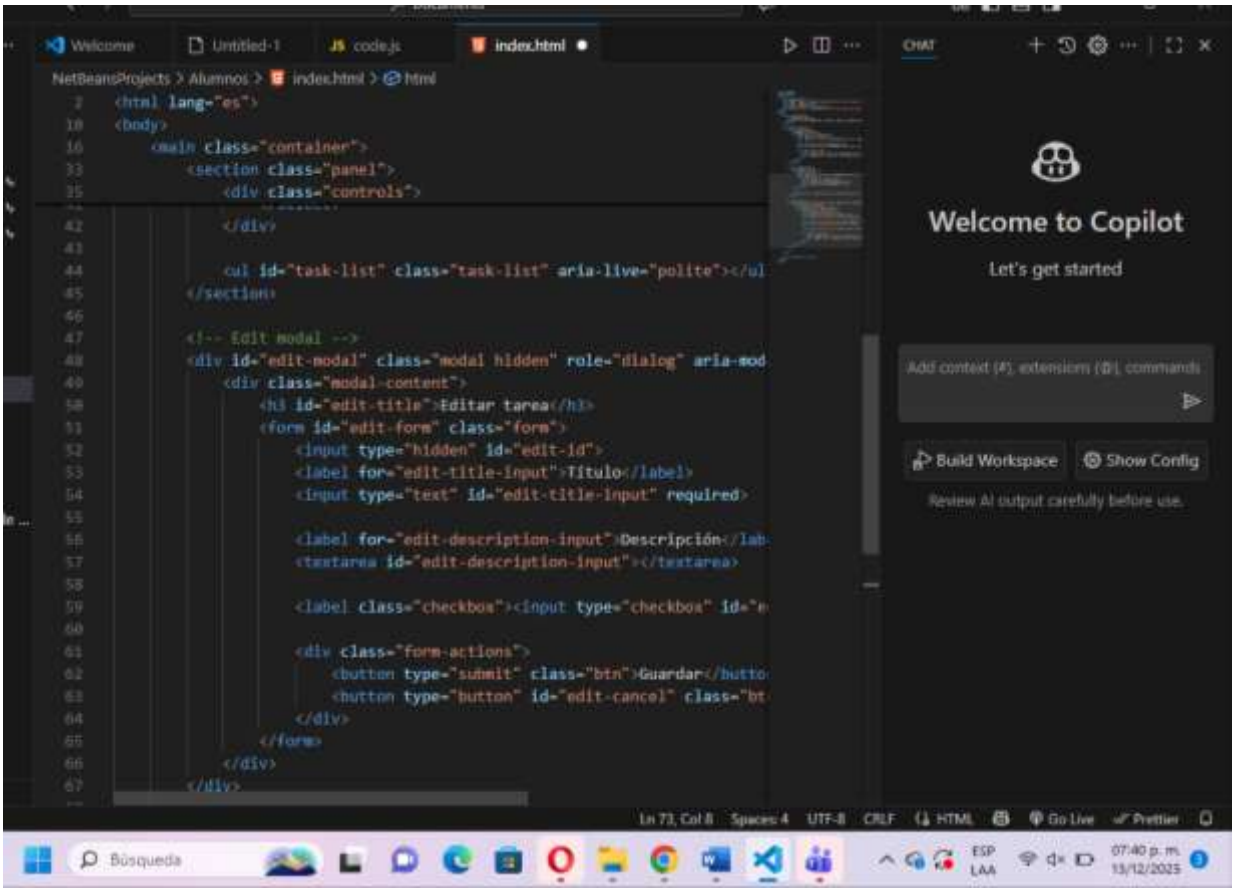
title: Define el nombre de la página en el navegador.

link stylesheet: Conecta el archivo CSS para el diseño visual.

meta description: Describe brevemente la funcionalidad de la aplicación.

El archivo app.js se encarga de:

- Gestionar eventos del formulario
- Almacenar tareas en localStorage
- Mostrar, editar y eliminar tareas
- Controlar el modal de edición


Welcome
Untitled-1
code.js
index.html
NetBeansProjects > Alumnos > index.html > html
<html lang="es">
<body>
<main class="contalner">
<section class="panel">

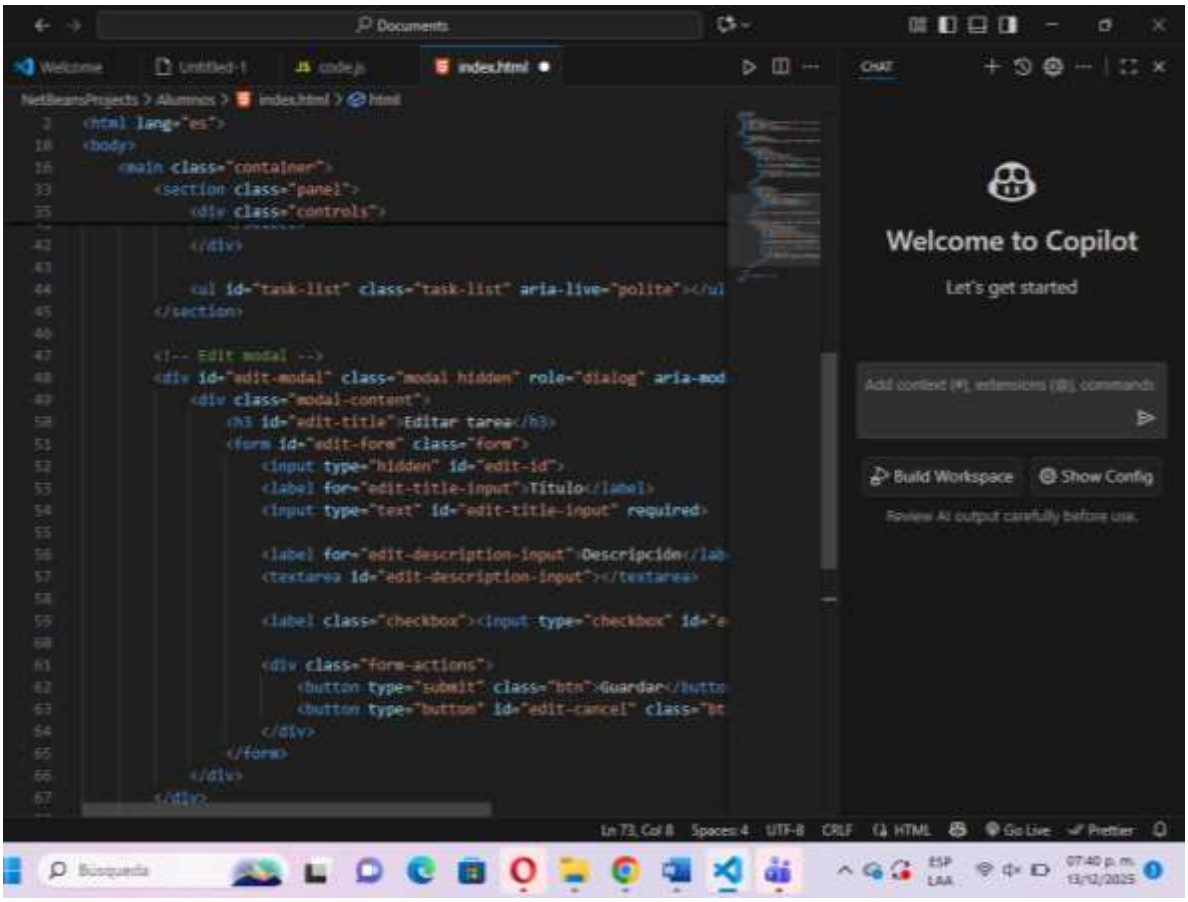
<ul id="task-list" class="task-list" aria-live="polite"></ul
</section>
<!-- Edit modal -->
<div id="edit-modal" class="modal hidden" role="dialog" aria-mod

<h3 id="edit-title">Editar tarea</h3>
<form id="edit-form" class="form">
<input type="hidden" id="edit-id">
<label for="edit-title-input">Título</label>
<input type="text" id="edit-title-input" required>
<label for="edit-description-input">Descripción</lab
<textarea id="edit-description-input"></textarea>
<label class="checkbox"><input type="checkbox" id="e

<button type="submit" class="btn">Guardar</butto
<button type="button" id="edit-cancel" class="bt

</form>

Ln 73, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Búsqueda
ESP
LAA
07:40 p. m.
13/12/2025


Documents
Welcome
Untitled-1
code.js
index.html
NetBeansProjects > Alumnos > index.html > html
<html lang="es">
<body>
<main class="contalner">
<section class="panel">

<ul id="task-list" class="task-list" aria-live="polite"></ul
</section>
<!-- Edit modal -->
<div id="edit-modal" class="modal hidden" role="dialog" aria-mod

<h3 id="edit-title">Editar tarea</h3>
<form id="edit-form" class="form">
<input type="hidden" id="edit-id">
<label for="edit-title-input">Título</label>
<input type="text" id="edit-title-input" required>
<label for="edit-description-input">Descripción</lab
<textarea id="edit-description-input"></textarea>
<label class="checkbox"><input type="checkbox" id="e

<button type="submit" class="btn">Guardar</butto
<button type="button" id="edit-cancel" class="bt

</form>

Ln 73, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Go Live
Prettier
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), commands
Build Workspace
Show Config
Review AI output carefully before use.
Búsqueda
ESP
LAA
07:40 p. m.
13/12/2025

Semana 15 – Reporte

El presente código tiene como finalidad implementar una estructura de datos tipo Tabla Hash (Hash Table) utilizando clases en JavaScript. Esta estructura permite almacenar y recuperar datos mediante pares clave–valor, ofreciendo un acceso eficiente a la información a través de una función hash.

El programa define una clase llamada HashTable, la cual:

Utiliza un arreglo como estructura base.

Emplea una función hash para convertir claves en índices.

Implementa métodos para insertar, obtener, eliminar y listar claves.

```
hashMethod(key) {
    let hash = 0;
    for (let i = 0; i < key.length; i++) {
        hash = (hash + key.charCodeAt(i) * i) % this.data.length;
    }
    return hash;
}
```

Este método:

Convierte una clave tipo string en un índice numérico.

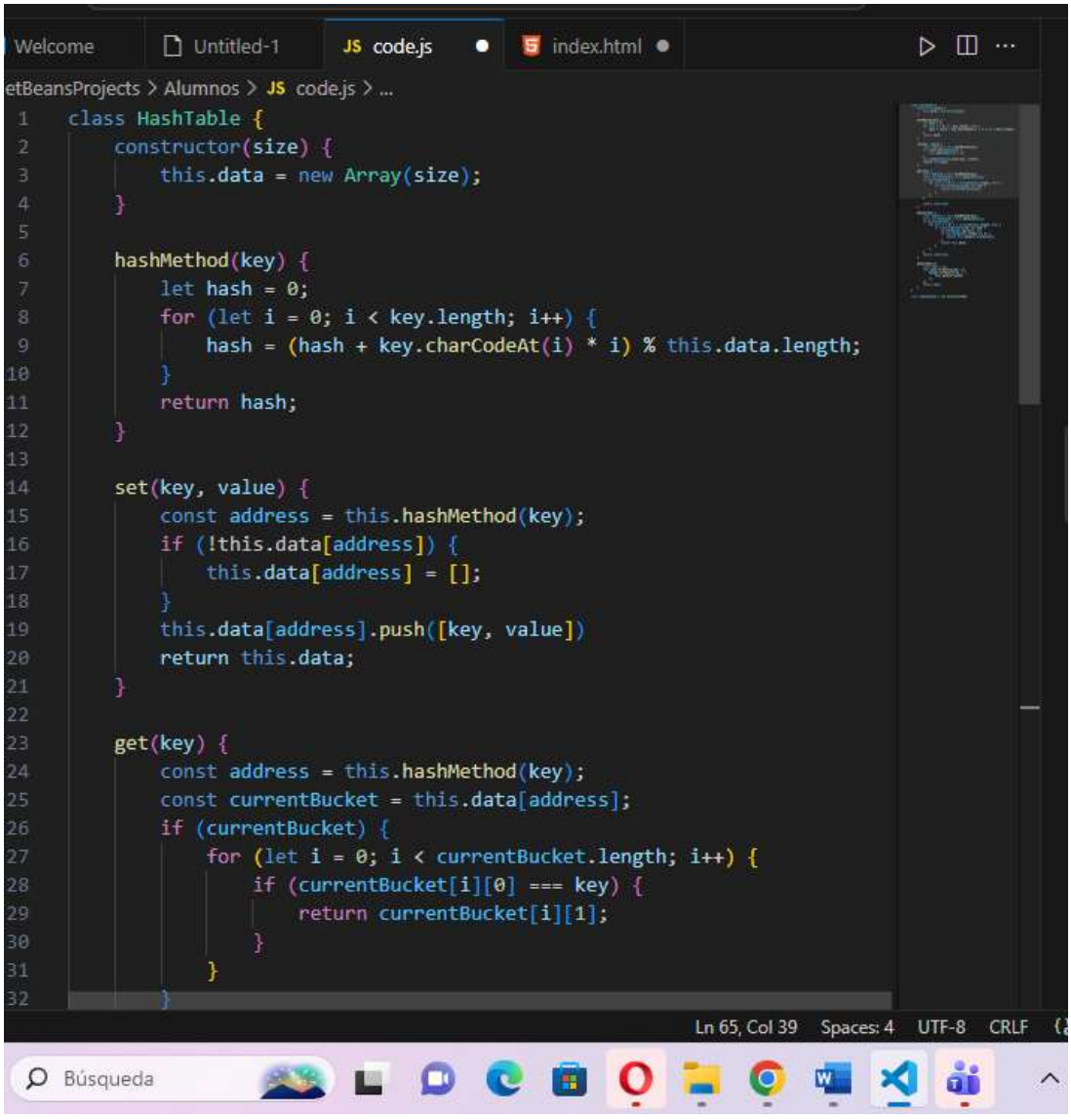
Utiliza el valor ASCII de cada carácter.

Aplica el operador módulo para asegurar que el índice esté dentro del tamaño del arreglo.

```
set(key, value) {
    const address = this.hashMethod(key);
    if (!this.data[address]) {
        this.data[address] = [];
    }
    this.data[address].push([key, value]);
    return this.data;
}
```

Permite insertar un par clave–valor en la tabla hash.

Si ya existe información en esa posición, se utiliza un arreglo interno (bucket) para manejar colisiones mediante separate chaining.



Semana 16 – Reporte

El presente código HTML corresponde a la estructura base de una aplicación web que tiene como finalidad consumir y mostrar información de la API de Pokémon, permitiendo filtrar los resultados por tipo. El archivo define la organización visual de la página y se apoya en un archivo CSS externo para el diseño.

Función de los elementos:

meta charset="UTF-8": Permite la correcta visualización de caracteres especiales.

meta viewport: Asegura un diseño responsivo para dispositivos móviles.

title: Define el título de la página en el navegador.

link rel="stylesheet": Conecta el archivo index.css, encargado del diseño visual.

```
<main>
    <h1>Hola como estas</h1>
    <p>Hola como estas</p>
</main>
```

Esta sección representa el contenido principal de la página. Actualmente contiene texto de ejemplo, el cual puede ser reemplazado por:

Tarjetas de Pokémon

Información obtenida dinámicamente desde la API

Resultados del filtrado por tipo

Integración con CSS y JavaScript

Aunque el código presentado solo incluye HTML y CSS:

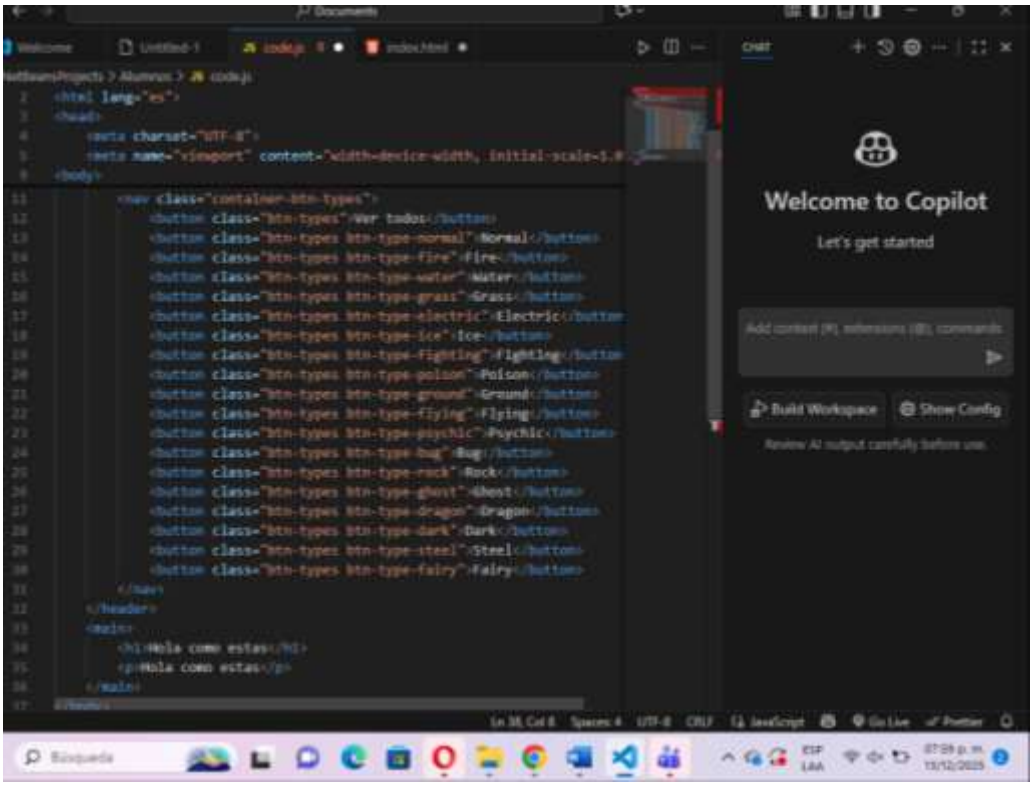
El archivo index.css se encarga del diseño visual.

Posteriormente puede integrarse un archivo JavaScript para:

Consumir la API de Pokémon

Mostrar los datos en pantalla

Filtrar resultados según el botón seleccionado



INSIGNIA

https://developers.google.com/profile/u/104498823974458677367?utm_source=web.dev