



Universidad Tecmilenio

Campus Las Torres

Materia: Desarrollo FullStack

Actividad 2 – Aplicación Interactiva de Gestión de Tareas

Alumno: Patricio Calvo

Matricula: 07097795

Enero / 2026

Link de Github:

Funcionamiento del código HTML

Este código HTML crea la estructura básica de una aplicación web interactiva para la gestión de tareas, permitiendo al usuario agregar, visualizar y administrar tareas directamente desde el navegador.

Primero, la línea `<!DOCTYPE html>` indica al navegador que se trata de un documento HTML5.

La etiqueta `<html lang="es">` especifica que el idioma del contenido es español.

Dentro de la sección `<head>` se incluyen configuraciones importantes como:
La codificación de caracteres UTF-8 para soportar caracteres especiales.
El título de la página, que se muestra en la pestaña del navegador.
El enlace al archivo `style.css`, encargado de definir la apariencia visual de la aplicación.

```
↳ index.html > ...
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4       <meta charset="UTF-8">
5       <title>Gestor de Tareas</title>
6       <link rel="stylesheet" href="style.css">
7   </head>
```

En el **<body>** se encuentran los elementos visibles de la aplicación:

- Un título principal que identifica la aplicación de gestión de tareas.
- Un campo de entrada (**input**) donde el usuario puede escribir una nueva tarea.
- Un botón que permite agregar la tarea a la lista.
- Un párrafo utilizado para mostrar mensajes de validación o error.
- Una lista no ordenada (**ul**) donde se muestran dinámicamente las tareas registradas.

```
8  <body>
9
10 <h1>Gestión de Tareas</h1>
11
12 <div class="panel">
13   <input type="text" id="tareaInput" placeholder="Escribe una tarea">
14   <button id="btnAgregar">Agregar</button>
15 </div>
16
17 <p id="mensaje"></p>
18
19 <ul id="listaTareas"></ul>
20
21 <script src="script.js"></script>
22 </body>
23 </html>
```

La estructura HTML mantiene una separación clara entre contenido y comportamiento, permitiendo que la interacción se maneje completamente desde JavaScript.

Funcionamiento del código JavaScript

El archivo JavaScript contiene la lógica principal de la aplicación y se encarga de la manipulación del **DOM**, el control de eventos y la gestión de las tareas mediante programación orientada a objetos.

Se utilizan las palabras reservadas **const** para declarar variables, cumpliendo con las prácticas modernas de **ES6**.

```
JS script.js > ...
1  console.log("JavaScript conectado correctamente");
2
3  const input = document.getElementById("tareaInput");
4  const boton = document.getElementById("btnAgregar");
5  const lista = document.getElementById("listaTareas");
6  const mensaje = document.getElementById("mensaje");
7
```

Se define la clase **Tarea**, la cual representa cada tarea creada por el usuario. Esta clase contiene propiedades como un identificador único, el nombre de la tarea y su estado (completa o incompleta), así como métodos para cambiar su estado y editar su contenido.

```
9   class Tarea {
10     constructor(nombre, completa = false) {
11       this.id = Date.now();
12       this.nombre = nombre;
13       this.completa = completa;
14     }
15
16     toggleEstado() {
17       this.completa = !this.completa;
18     }
19
20     editar(nuevoNombre) {
21       this.nombre = nuevoNombre;
22     }
23 }
```

La clase **GestorDeTareas** se encarga de administrar la colección de tareas. Incluye métodos para agregar, editar, eliminar tareas y cambiar su estado, así como funciones para guardar y recuperar la información utilizando **LocalStorage**.

```
26  class GestorDeTareas {
27    constructor() {
28      this.tareas = [];
29    }
30
31    agregar(nombre) {
32      const nueva = new Tarea(nombre);
33      this.tareas.push(nueva);
34      this.guardar();
35    }
36
37    eliminar(id) {
38      this.tareas = this.tareas.filter(t => t.id !== id);
39      this.guardar();
40    }
41
42    editar(id, nuevoNombre) {
43      const tarea = this.tareas.find(t => t.id === id);
44      if (tarea) tarea.editar(nuevoNombre);
45      this.guardar();
46    }
47
48    toggleEstado(id) {
49      const tarea = this.tareas.find(t => t.id === id);
50      if (tarea) tarea.toggleEstado();
51      this.guardar();
52    }
53
54    guardar() {
55      localStorage.setItem("tareas", JSON.stringify(this.tareas));
56    }
57
58    cargar() {
59      const datos = JSON.parse(localStorage.getItem("tareas"));
60      if (datos) this.tareas = datos;
61    }
62  }
63
64  const gestor = new GestorDeTareas();
65  gestor.cargar();
66  renderLista();
```

La función **renderLista** se encarga de actualizar visualmente la lista de tareas en pantalla, recorriendo el arreglo de tareas con **forEach** y generando los elementos HTML de forma dinámica.

```
JS script.js > ...
...
64 const gestor = new GestorDeTareas();
65 gestor.cargar();
66 renderLista();
67
68
69 function renderLista() {
70     lista.innerHTML = "";
71
72     gestor.tareas.forEach(tarea => {
73         const li = document.createElement("li");
74         li.textContent = tarea.nombre;
75
76         if (tarea.completa) {
77             li.classList.add("completa");
78         }
79
80
81         li.addEventListener("click", () => {
82             gestor.toggleEstado(tarea.id);
83             renderLista();
84         });
85
86
87         li.addEventListener("dblclick", () => {
88             gestor.eliminar(tarea.id);
89             renderLista();
90         });
91
92
93         li.addEventListener("contextmenu", (e) => {
94             e.preventDefault();
95             const nuevo = prompt("Editar tarea:", tarea.nombre);
96             if (nuevo) {
97                 gestor.editar(tarea.id, nuevo);
98                 renderLista();
99             }
100        });
101    });
102}
```

La aplicación utiliza eventos del mouse para interactuar con las tareas:

Un clic cambia el estado de la tarea.

```
81 |     li.addEventListener("click", () => {  
82 |         gestor.toggleEstado(tarea.id);  
83 |         renderLista();  
84 |     });
```

Un doble clic elimina la tarea seleccionada.

```
87 v     li.addEventListener("dblclick", () => {  
88 |         gestor.eliminar(tarea.id);  
89 |         renderLista();  
90 |     });
```

El clic derecho permite editar el contenido de la tarea.

```
93 v     li.addEventListener("contextmenu", (e) => {  
94 |     e.preventDefault();  
95 |     const nuevo = prompt("Editar tarea:", tarea.nombre);  
96 v     if (nuevo) {  
97 |         gestor.editar(tarea.id, nuevo);  
98 |         renderLista();  
99 |     }  
100 | );  
101 |     lista.appendChild(li);  
102 | );  
103 | };  
104 }
```

Además, se implementa una validación para evitar que el usuario agregue tareas vacías, mostrando un mensaje de error cuando es necesario.

```
107 v boton.addEventListener("click", () => {  
108 |     const texto = input.value.trim();  
109 |  
110 v     if (texto === "") {  
111 |         mensaje.textContent = "No puedes agregar tareas vacías.";  
112 |         mensaje.classList.add("mensaje-error");  
113 |         return;  
114 |     }  
115 |  
116 |     mensaje.textContent = "";  
117 |     mensaje.classList.remove("mensaje-error");  
118 |  
119 |     gestor.agregar(texto);  
120 |     input.value = "";  
121 |     renderLista();  
122 |});
```

Funcionamiento del código CSS

El archivo CSS define el estilo visual de la aplicación, manteniendo un diseño simple, claro y funcional.

La regla **body** establece una fuente estándar, un espaciado general y una presentación limpia.

```
# style.css > ...
1  ↴ body {
2    |   font-family: Arial, sans-serif;
3    |   padding: 20px;
4  }
```

Se aplican estilos a la lista y a los elementos **li** para mejorar la legibilidad y facilitar la interacción del usuario.

```
6  .panel {
7  |   margin-bottom: 10px;
8  }
9
10 ul {
11 |   list-style: none;
12 |   padding: 0;
13 }
14
15 li {
16 |   padding: 6px;
17 |   cursor: pointer;
18 |   border-bottom: 1px solid ■#ccc;
19 }
```

La clase **.completa** se utiliza para identificar visualmente las tareas completadas, aplicando un cambio de color y estilo de texto.

```
21 .completa {
22 |   font-weight: bold;
23 |   color: ■green;
24 }
25
```

La clase **.mensaje-error** se encarga de mostrar los mensajes de validación en color rojo para alertar al usuario.

```
26 ▶ .mensaje-error {
27 |   color: ■red;
28 }
```

Vista de la Aplicación Web

Gestión de Tareas

Agregar Tareas

Gestión de Tareas

Tarea 1

Tarea 2

Interacción 1 clic (Completar Tarea)

Gestión de Tareas

Tarea 1

Tarea 2

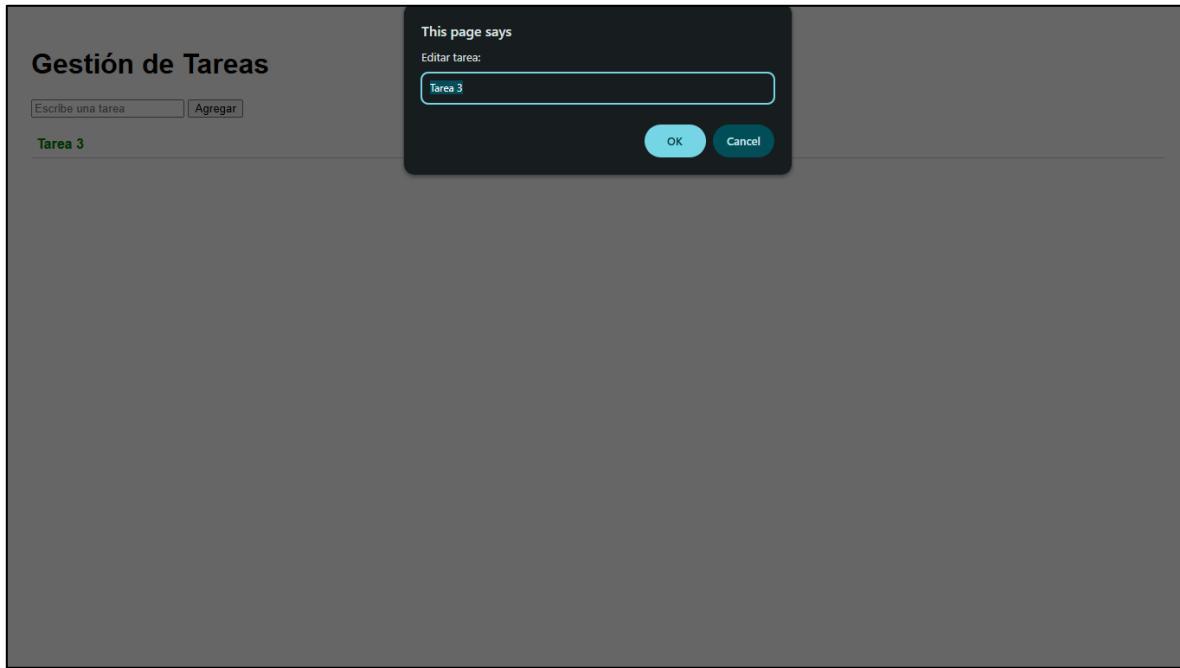
Tarea 3

Interacción 2 clics (Eliminar Tarea)

Gestión de Tareas

Tarea 3

Interacción clic Derecho (Editar Tarea)



Vista en Dispositivo Móvil

