PROYECTO: LISTA DE TAREAS INTERACTIVA

DESCRIPCIÓN:

Crea una aplicación web sencilla de lista de tareas (To-Do List) donde los usuarios puedan añadir, eliminar y marcar tareas como completadas.

FUNCIONALIDADES BÁSICAS:

1. **Añadir Tarea**: Un campo de entrada y un botón para añadir nuevas tareas a la lista.
2. **Eliminar Tarea**: Un botón junto a cada tarea para eliminarla de la lista.
3. **Marcar como Completada**: Al hacer clic en una tarea, esta se marca como completada (puedes cambiar su estilo para indicar que está completada).

FUNCIONALIDADES AVANZADAS (PARA AÑADIR MÁS ADELANTE):

1. **Editar Tarea**: Permitir a los usuarios editar el texto de una tarea existente.
2. **Filtros**: Añadir filtros para mostrar todas las tareas, solo las completadas o solo las pendientes.
3. **Persistencia de Datos**: Usar localStorage para guardar las tareas y que no se pierdan al recargar la página.

ESTRUCTURA BÁSICA DEL PROYECTO:

- **HTML**: Estructura básica de la página con un campo de entrada, un botón y una lista para mostrar las tareas.
- **CSS**: Estilos para la lista de tareas, incluyendo diferentes estilos para tareas completadas.
- **JavaScript**: Lógica para añadir, eliminar y marcar tareas como completadas.

HTML (INDEX.HTML)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Lista de Tareas</title>
    <link rel="stylesheet" href="styles.css"> <!-- Enlace al
archivo CSS para estilos -->
</head>
<body>
    <div class="container">
        <h1>Lista de Tareas</h1> <!-- Título de la aplicación -->

        <!-- Campo de entrada para añadir una nueva tarea -->
        <input type="text" id="new-task" placeholder="Añadir nueva
tarea">

        <!-- Campo de selección para elegir la prioridad de la
tarea -->
        <select id="priority">
            <option value="low">Baja</option>
            <option value="medium">Media</option>
            <option value="high">Alta</option>
        </select>

        <!-- Campo de entrada para seleccionar la fecha de
vencimiento de la tarea -->
        <input type="date" id="due-date">

        <!-- Botón para añadir la tarea a la lista -->
        <button id="add-task">Añadir</button>

        <!-- Contenedor para los botones de filtro y ordenación -->
        <div class="filters">
            <!-- Botones para filtrar las tareas -->
            <button id="all-filter">Todas</button>
            <button id="completed-filter">Completadas</button>
            <button id="pending-filter">Pendientes</button>

            <!-- Botones para ordenar las tareas -->
            <button id="sort-priority">Prioridad</button>
            <button id="sort-due-date">Fecha de
Vencimiento</button>
        </div>

        <!-- Lista desordenada donde se mostrarán las tareas -->
        <ul id="task-list"></ul>
    </div>
    <script src="script.js"></script> <!-- Enlace al archivo
JavaScript para la lógica de la aplicación -->
</body>
</html>
```

Explicación de los Elementos:

- **Campos de Entrada**:
    - <input type="text" id="new-task" placeholder="Añadir nueva tarea">: Campo para escribir el texto de la nueva tarea.
    - <select id="priority">: Campo de selección para elegir la prioridad de la tarea (baja, media, alta).
    - <input type="date" id="due-date">: Campo para seleccionar la fecha de vencimiento de la tarea.
- **Botones**:
    - <button id="add-task">Añadir</button>: Botón para añadir la tarea a la lista.
    - : Contenedor para los botones de filtro y ordenación.
    - <button id="all-filter">Todas</button>: Muestra todas las tareas.
    - <button id="completed-filter">Completadas</button>: Muestra solo las tareas completadas.
    - <button id="pending-filter">Pendientes</button>: Muestra solo las tareas pendientes.
    - <button id="sort-priority">Prioridad</button>: Ordena las tareas por prioridad.
    - <button id="sort-due-date">Fecha de Vencimiento</button>: Ordena las tareas por fecha de vencimiento.
- **Lista de Tareas**:
    - <ul id="task-list">: Lista desordenada donde se mostrarán las tareas.

CSS (STYLES.CSS)

```
body {
    font-family: Arial, sans-serif; /* Fuente predeterminada para el texto */
    background-color: #f4f4f4; /* Color de fondo claro para la página */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh; /* Altura completa de la ventana */
    margin: 0; /* Elimina el margen predeterminado del cuerpo */
    transition: background-color 0.3s ease; /* Transición suave para cambios en el color de fondo */
}
```

```
.container {
    background-color: #fff; /* Color de fondo blanco para el contenedor */
    padding: 20px; /* Espaciado interno del contenedor */
    border-radius: 5px; /* Bordes redondeados */
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Sombra suave para dar profundidad */
    text-align: center; /* Centra el texto dentro del contenedor */
    transition: background-color 0.3s ease, color 0.3s ease; /* Transiciones suaves para cambios de color */
}

#new-task {
    width: 70%; /* Ancho del campo de entrada */
    padding: 10px; /* Espaciado interno del campo */
    margin-right: 10px; /* Espacio a la derecha del campo */
    border: 1px solid #ccc; /* Borde del campo */
    border-radius: 5px; /* Bordes redondeados */
}

#add-task, .filters button {
    padding: 10px 20px; /* Espaciado interno de los botones */
    border: none; /* Elimina el borde predeterminado */
    border-radius: 5px; /* Bordes redondeados */
    cursor: pointer; /* Cambia el cursor al pasar sobre el botón */
}

#add-task {
    background-color: #28a745; /* Color de fondo verde para el botón de añadir */
    color: #fff; /* Color de texto blanco */
}

#add-task:hover {
    background-color: #218838; /* Color de fondo más oscuro al pasar el ratón */
}

.filters {
    margin: 20px 0; /* Espaciado superior e inferior del contenedor de filtros */
```

```
}

.filters button {
    margin-right: 10px; /* Espacio a la derecha de cada botón */
    background-color: #007bff; /* Color de fondo azul para los botones de filtro */
    color: #fff; /* Color de texto blanco */
}

.filters button:hover {
    background-color: #0056b3; /* Color de fondo más oscuro al pasar el ratón */
}

#task-list {
    list-style-type: none; /* Elimina los puntos de la lista */
    padding: 0; /* Elimina el espaciado interno predeterminado */
}

.task {
    display: flex; /* Usa flexbox para alinear elementos */
    justify-content: space-between; /* Espacia los elementos dentro de la tarea */
    padding: 10px; /* Espaciado interno de cada tarea */
    border-bottom: 1px solid #ccc; /* Borde inferior de cada tarea */
}

.task.completed {
    text-decoration: line-through; /* Tacha el texto de las tareas completadas */
    color: #888; /* Color de texto más claro */
}

.task button {
    background-color: #dc3545; /* Color de fondo rojo para los botones de eliminar */
    color: #fff; /* Color de texto blanco */
    border: none; /* Elimina el borde predeterminado */
    border-radius: 5px; /* Bordes redondeados */
    cursor: pointer; /* Cambia el cursor al pasar sobre el botón */
```

```
}
```

Vamos a dividir el desarrollo en pasos para aprender y entender cada parte del código. Vamos a desglosar el JavaScript en varios pasos para que puedas ir añadiendo funcionalidades de manera progresiva.

PASO 1: AÑADIR UNA NUEVA TAREA

Primero, vamos a implementar la funcionalidad básica para añadir una nueva tarea a la lista.

- **Elementos Añadidos al html**:
    - <input type="text" id="new-task" placeholder="Añadir nueva tarea">: Campo de entrada para escribir la nueva tarea.
    - <button id="add-task">Añadir</button>: Botón para añadir la tarea a la lista.
    - <ul id="task-list">: Lista desordenada para mostrar las tareas.

```
document.getElementById('add-task').addEventListener('click',
function() {
    const taskInput = document.getElementById('new-task');
    const taskText = taskInput.value.trim();

    if (taskText !== '') {
        const taskList = document.getElementById('task-list');
        const listItem = document.createElement('li');
        listItem.className = 'task';
        listItem.textContent = taskText;
        taskList.appendChild(listItem);
        taskInput.value = '';
    }
});
```

PASO 2: ELIMINAR UNA TAREA

Ahora, añadamos la funcionalidad para eliminar una tarea de la lista.

- **Elementos Añadidos al html**:
    - Dentro de cada <li> (elemento de la lista), se añade un botón <button onclick="removeTask(this)">Eliminar</button> para eliminar la tarea.

```
function removeTask(button) {
    const listItem = button.parentElement;
    listItem.remove();
}

document.getElementById('add-task').addEventListener('click',
function() {
    const taskInput = document.getElementById('new-task');
```

```
    const taskText = taskInput.value.trim();

    if (taskText !== '') {
        const taskList = document.getElementById('task-list');
        const listItem = document.createElement('li');
        listItem.className = 'task';
        listItem.innerHTML = `${taskText} <button
onclick="removeTask(this)">Eliminar</button>`;
        taskList.appendChild(listItem);
        taskInput.value = '';
    }
});
```

PASO 3: MARCAR UNA TAREA COMO COMPLETADA

Vamos a añadir la funcionalidad para marcar una tarea como completada al hacer clic en ella.

- **Elementos Modificados**:
    - No se añadieron nuevos elementos HTML, pero se modificó el comportamiento de los elementos <li> para que respondan a los clics y marquen la tarea como completada.

```
document.getElementById('add-task').addEventListener('click',
function() {
    const taskInput = document.getElementById('new-task');
    const taskText = taskInput.value.trim();

    if (taskText !== '') {
        const taskList = document.getElementById('task-list');
        const listItem = document.createElement('li');
        listItem.className = 'task';
        listItem.innerHTML = `${taskText} <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
            if (event.target.tagName === 'LI') {
                listItem.classList.toggle('completed');
            }
        });
        taskList.appendChild(listItem);
        taskInput.value = '';
    }
});
```

PASO 4: PERSISTENCIA DE DATOS CON LOCALSTORAGE

Ahora, vamos a añadir la funcionalidad para guardar las tareas en localStorage para que no se pierdan al recargar la página.

- **Elementos Modificados**:
    - No se añadieron nuevos elementos HTML, pero se implementó la lógica en JavaScript para guardar y cargar tareas desde localStorage.

```
function loadTasks() {
    const taskList = document.getElementById('task-list');
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.forEach(taskText => {
        const listItem = document.createElement('li');
        listItem.className = 'task';
        listItem.innerHTML = `${taskText} <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
            if (event.target.tagName === 'LI') {
                listItem.classList.toggle('completed');
                saveTasks();
            }
        });
        taskList.appendChild(listItem);
    });
}

function saveTasks() {
    const taskList = document.getElementById('task-list');
    const tasks = [];
    taskList.querySelectorAll('.task').forEach(listItem => {
        tasks.push(listItem.textContent.replace('Eliminar',
'').trim());
    });
    localStorage.setItem('tasks', JSON.stringify(tasks));
}

document.getElementById('add-task').addEventListener('click',
function() {
    const taskInput = document.getElementById('new-task');
    const taskText = taskInput.value.trim();

    if (taskText !== '') {
        const taskList = document.getElementById('task-list');
```

```
        const listItem = document.createElement('li');
        listItem.className = 'task';
        listItem.innerHTML = `${taskText} <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
            if (event.target.tagName === 'LI') {
                listItem.classList.toggle('completed');
                saveTasks();
            }
        });
        taskList.appendChild(listItem);
        taskInput.value = '';
        saveTasks();
    }
});

function removeTask(button) {
    const listItem = button.parentElement;
    listItem.remove();
    saveTasks();
}

// Cargar tareas almacenadas al cargar la página
window.addEventListener('load', loadTasks);
```

PASO 5: EDITAR UNA TAREA

- **Elementos Añadidos**:
    - Dentro de cada <li>, se añade un botón <button onclick="editTask(this)">Editar</button> para editar la tarea.

```
function editTask(button) {
    const listItem = button.parentElement;
    const taskText = listItem.textContent.replace('Editar',
'').replace('Eliminar', '').trim();
    const taskInput = document.createElement('input');
    taskInput.type = 'text';
    taskInput.value = taskText;
    taskInput.addEventListener('blur', function() {
        listItem.textContent = `${taskInput.value} `;
```

```
        listItem.innerHTML += `<button
onclick="editTask(this)">Editar</button> <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
            if (event.target.tagName === 'LI') {
                listItem.classList.toggle('completed');
                saveTasks();
            }
        });
        saveTasks();
    });
    listItem.textContent = '';
    listItem.appendChild(taskInput);
    taskInput.focus();
}
```

PASO 6:ORDENAR TAREAS

Funcionalidad para ordenar las tareas por prioridad y fecha de vencimiento:

```
function sortTasks(criteria) {
    const taskList = document.getElementById('task-list');
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.sort((a, b) => {
        if (criteria === 'priority') {
            const priorities = { low: 1, medium: 2, high: 3 };
            return priorities[b.priority] - priorities[a.priority];
        } else if (criteria === 'dueDate') {
            return new Date(a.dueDate) - new Date(b.dueDate);
        }
    });
    taskList.innerHTML = '';
    tasks.forEach((task, index) => {
        const listItem = document.createElement('li');
        listItem.className = `task ${task.completed ? 'completed' :
''}`;
        listItem.setAttribute('data-index', index);
        listItem.innerHTML = `${task.text} (Prioridad:
${task.priority}, Vence: ${task.dueDate}) <button
onclick="editTask(this)">Editar</button> <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
```

```
            if (event.target.tagName === 'LI') {
                task.completed = !task.completed;
                listItem.classList.toggle('completed');
                saveTasks();
            }
        });
        taskList.appendChild(listItem);
    });
    saveTasks();
}

document.getElementById('sort-priority').addEventListener('click',
() => sortTasks('priority'));
document.getElementById('sort-due-date').addEventListener('click',
() => sortTasks('dueDate'));
```

Explicación:

- **Estructura de Datos**: Las tareas ahora tienen priority y dueDate.
- **Función** sortTasks: Ordena las tareas por prioridad o fecha de vencimiento.
- **Eventos de Ordenación**: Los botones de ordenación llaman a sortTasks con el criterio correspondiente.

PASO 7: ORDENAR LAS TAREAS POR PRIORIDAD Y FECHA DE VENCIMIENTO:

```
function sortTasks(criteria) {
    const taskList = document.getElementById('task-list');
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.sort((a, b) => {
        if (criteria === 'priority') {
            const priorities = { low: 1, medium: 2, high: 3 };
            return priorities[b.priority] - priorities[a.priority];
        } else if (criteria === 'dueDate') {
            return new Date(a.dueDate) - new Date(b.dueDate);
        }
    });
    taskList.innerHTML = '';
    tasks.forEach((task, index) => {
        const listItem = document.createElement('li');
        listItem.className = `task ${task.completed ? 'completed' :
''}`;
        listItem.setAttribute('data-index', index);
```

```
        listItem.innerHTML = `${task.text} (Prioridad:
${task.priority}, Vence: ${task.dueDate}) <button
onclick="editTask(this)">Editar</button> <button
onclick="removeTask(this)">Eliminar</button>`;
        listItem.addEventListener('click', function(event) {
            if (event.target.tagName === 'LI') {
                task.completed = !task.completed;
                listItem.classList.toggle('completed');
                saveTasks();
            }
        });
        taskList.appendChild(listItem);
    });
    saveTasks();
}

document.getElementById('sort-priority').addEventListener('click',
() => sortTasks('priority'));
document.getElementById('sort-due-date').addEventListener('click',
() => sortTasks('dueDate'));
```

EXPLICACIÓN DE LAS FUNCIONES Y ELEMENTOS:

- loadTasks: Carga las tareas desde localStorage y las muestra en la lista.
- saveTasks: Guarda las tareas en localStorage.
- add-task **Event Listener**: Añade una nueva tarea a la lista cuando se hace clic en el botón "Añadir".
- removeTask: Elimina una tarea de la lista.
- editTask: Permite editar el texto, la prioridad y la fecha de vencimiento de una tarea.
- filterTasks: Filtra las tareas para mostrar todas, solo las completadas o solo las pendientes.
- sortTasks: Ordena las tareas por prioridad o fecha de vencimiento.
- **Eventos de Filtro y Ordenación**: Los botones de filtro y ordenación llaman a filterTasks y sortTasks con los criterios correspondientes.