Informe del Segundo Laboratorio del Módulo de Introducción a los Lenguajes de Programación

Kevin Alexander Jaramillo Giraldo

Jaider Javier De La Rosa Berte

Politécnico PIO
Internacional de Occidente

Politécnico Pio

Escuela de ciencias

Técnico Laboral Asistente en Programación de Software

Introducción a los Lenguajes de Programación

27-09-2024

Introducción

Este proyecto fue desarrollado con el propósito de aplicar los conocimientos adquiridos hasta el momento en el módulo “Introducción a los lenguajes de programación” enmarcado en la malla curricular del programa “Técnico Laboral Asistente en Programación de Software” de la institución educativa Politécnico Pio Internacional de Occidente, adicionalmente, con la presentación de este escrito y los demás archivos enlazados a este escrito por medio de un link a un repositorio en GitHub, se pretende cumplir de manera responsable con la asignación realizada por el Ingeniero y Profesor Jaider Javier De La Rosa Berte. Las herramientas implementadas para llevar a cabo este trabajo son Visaul Studio Code para realizar todo lo referente a la programación de los archivos HTML, CSS y JavaScript que componen la parte del frontend de la página web realizada; adicionalmente se hace uso del programa Git para enlazar el repositorio local con el remoto, así como para subir los avances del repositorio local al repositorio de GitHub; se hace uso de GitHub para la creación y almacenamiento del proyecto remoto; se hace uso del programa PDF-Xchange Editor para la lectura de la asignación realizada por el docente encargado y de Microsoft Word para la redacción del presente informe de laboratorio. Implementar estos conceptos en un proyecto tan cercano a los requerimientos reales sobre interactividad en las páginas web considero que me permite, no solo comprender su funcionamiento teórico, sino también enfrentar desafíos prácticos relacionados con la lógica de programación en JavaScript, en CSS y en HTML, la organización de los archivos que conforman la página web y la interacción y enrutamiento entre sus diferentes componentes.

Objetivos de la actividad

- Aplicar conceptos fundamentales de HTML, CSS y JavaScript en el desarrollo de una aplicación web.
- Desarrollar una agenda interactiva que permita agregar, eliminar y visualizar eventos.
- Introducir el uso del DOM (Document Object Model) para la manipulación dinámica del contenido de la página.
- Implementar interactividad usando JavaScript y mejorar el diseño con CSS.

- Evidenciar los conocimientos adquiridos en el presente módulo hasta la fecha de entrega del presente documento.
- Cumplir con responsabilidad la asignación hecha por el docente Jaider Javier De La Rosa Berte.

Desarrollo de la actividad

Parte 1: Estructura HTML

En esta sección cree en un inicio la estructura básica del HTML utilizando HTML:5 y dando tablatura en el Visual Studio Code, después empecé a modificar la estructura inicial para agregar los elementos que se solicitaban en el documento del laboratorio, el orden que utilicé para la creación del esqueleto fue el especificado en el pdf de direccionamiento cargado en el campus, por tanto, primero cree el input para el título del evento a guardar y le dí su respectiva clase e identificador, procuré dar los nombres de las clases en español y los id en ingles para mantener el uso de ambos idiomas, a las clases de los elementos les di el uso exclusivo de ser los medios para invocar etiquetas del HTML desde el CSS, a los identificadores de los elementos del HTML les di el uso exclusivo de ser medios para invocar etiquetas HTML desde el archivo JavaScript. Para la parte de la estructura de los contenedores me guíe con el ejemplo que hicimos en clase sobre la asignación de notas para estudiantes del PIO.

Después del input de tipo texto para el titulo del evento, cree el input de tipo Date atendiendo a la instrucción que el profesor dio en la misma clase mencionada anteriormente, pese a que se construyó un registro de notas, el profesor verbalmente dijo que era lo que debía de hacerse para que la página tuviera un input de tipo Date, los otros dos elementos (botón para agregar y div vació para mostrar en una sección definida los elementos agregados con el botón) también se trabajaron en clase así que su creación e implementación fue un proceso relativamente familiar, me apoyé en el ejercicio realizado en clase para la codificación de estos elementos.

Adicional a todo lo anterior, cree una sección con un botón flotante porque quería agregarle a la agenda una canción relativamente relajante para hacer la experiencia del usuario más apacible. A continuación, se presenta el código del HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/AgendaInteractiva.css">
    <title>Agenda Interactiva</title>
</head>
<body>

    <div class="contenedor" id="container">

        <!-- Le pongo un título al aplicativo web-->
        <h1 class="fituloApp" id="appTittle">Agenda Interactiva</h1>

        <!-- A continuación creo toda la parte del formulario-->
        <form class="formulario" id="formulary"> <!-- Creo un formulario con
clase para css e id para el js -->
            <h2 class="subtitulo1" id="subTittle1">Esta agenda interactiva
permite al usuario crear y eliminar recordatorios para sus eventos, si
deseas usarla y crear recordatorios por favor llena el formulario.</h2>
            <label for="titulo del Evento" class="tituloEvento"
id="eventTittle">Título del Evento</label>
            <input type="text" name="EntradaTituloEvento"
id="InputTittleEvent"> <!-- Creo un input de tipo text que es una barra de
entrada con texto -->
            <label for="fecha" class="fecha" id="date">Fecha</label> <!-- La
etiqueta label añade una leyenda o etiqueta explicativa a elementos input --
>
            <input type="date" name="EntradaFecha" id="InputDate"> <!-- Creo
un input de tipo date que es una barra de entrada con fecha formato
dd/mm/aaaa que despliega calendario -->
            <label for="hora" class="hora" id="hour">Hora</label>
            <input type="time" name="EntradaHora" id="InputHour">
            <button type="submit" class="botonAgregarEvento" id="addEvent-
button">Agendar Evento</button>
        </form>

 <div class="contenedorFiltrar" id="filterContainer">
            <input type="date" class="filtrarFecha" id="filterDateInput">
            <button class="botonfiltrar" id="filterDateButton">Buscar por
Fecha</button>
        </div>
```

```
        <!-- A continuación creo el contenedor vacio para mostrar los
eventos registrados con su respectiva fecha-->
        <div class="contenedorEventos" id="containerEvents">
            <h2 class="subtitulo2" id="subTittle2">Eventos Registrados</h2>
            <ul class="listaEventos" id="eventList">
                <!-- Cada que se registre un evento aparecerá aquí-->
            </ul>
        </div>

    </div>

    <!-- Botón flotante para parchar con musiquita chill -->
    <button class="boton-flotante" id="floatbutton">Play Music</button>
    <audio src="utils/sounds/lofi_song.m4a" class="cancion" id="song"
></audio>


    <script src="js/AgendaInteractivaaa.js"></script>

</body>
</html>
```

Parte 2: Estilización con el CSS

En esta parte de la creación del trabajo no gasté mucho tiempo ya que el ejemplo que el profesor realizó, en la clase anteriormente mencionada, una estilización muy sobria y elegante para el ejemplo del registro de notas, en función a este estilo visual intenté estilizar mi página web, gráficamente no suelo tener un gusto que a las demás personas les figure con su concepción de belleza razón por la cual intenté seguir las enseñanzas del docente y fijarme en su escogencia de colores, el docente utilizó colores no muy intensos que catalogué como colores pastel, entonces pensé en un concepto de noche lluviosa para la agenda, razón por la cual implementé el atributo background-image para colocar una imagen que realicé con pront en una inteligencia artificial. Para la parte del botón flotante utilicé una documentación parecida al estilo de Bootstrap y posteriormente ajuste los parámetros de las rutas relativas y de los colores para que el diseño se correspondiera con las tonalidades manejadas en la página, como utilicé posición absoluta le puse una media querie al botón flotante para que cuando esté en páginas de pantalla pequeña no se vaya a desaparecer.

Profesor la verdad me faltó tiempo para estilizar a un diseño responsivo absolutamente cada parte de la página y ajustando los maxwidth y los minwidth no obstante igualmente fue con mucho esfuerzo. El código esta bastante comentado porque me equivocaba aveces entonces escribía la explicación del código para luego si necesito hacer lo mismo saber donde encuentro con certeza la solución y digo con certeza porque en ocasiones busco la solución en la documentación, pero no la sé implementar entonces no me funciona o la documentación es de versiones anteriores y no me funciona.

Leyendo sobre como estilizar un sitió web encontré el estilo de trabajo de css con grid entonces apliqué este tipo de display a ciertos elementos del html, adicional a lo anterior buscando dar animación a ciertos elementos del css para cumplir con la mejora opcional de hacer más atractiva la adición y la eliminación de eventos con su respectivo título, fecha y hora. Pese a que en pocas ocasiones recuerdo que se utilizara el símbolo de contenencia de elementos dentro de otros en el CSS opte por implementar el símbolo para asegurarme que otros elementos del mismo tipo en el html no se fueran a ver afectados al momento de crear una regla css pensando en que fuese aplicada específicamente solo en cierta sección de la página web. A continuación, se presenta el código de la sección Css:

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Times New Roman', Times, serif;

}
/* Estilo CSS para ocultar todo el contenido inicialmente */
body {
    visibility: hidden;
}

body {
    /* background-color: #486f99; */
    background-image:
URL('../utils/images/imagen_noche_lluviosa_ventana.webp');
    background-repeat: no-repeat;
    background-size: cover;
    font-family: 'Times New Roman', Times, serif;
    font-size: 16px;
    line-height: 1.5;
```

```
    color: #ffffff;
}

.contenedor {
    max-width: 800px;
    margin: 25px auto;
    padding: 50px; /* Espacio entre el borde y el contenido */
    border-radius: 30px;
    background-color: #304a67;
    box-shadow: 0 0 10px rgb(0, 0, 0); /* Agregar una sombra al contenedor*/

}

.contenedor:hover {
    background-color: #29425f; /* Color de fondo al pasar el cursor */
    box-shadow: 0px 15px 20px rgba(0, 0, 0, 0.3); /* Sombra al pasar el
cursor */
    transform: translateY(-7px); /* Movimiento hacia arriba, curiosamente al
ser negativo sube y al ser positivo baja */
}

h1 {
    font-size: 50px;
    margin-bottom: 20px;
    margin-top: -20px;
    text-shadow: 0px 0px 30px rgb(0, 0, 0),0px 0px 15px rgb(0, 0, 0);/*
Agregar una sombra al contenedor*/
    display: flex;
    justify-content: center; /* Centrar horizontalmente objetos con display
flex*/
}

h1:hover {
    color: #040a3d;
    text-shadow: 0 0 30px rgb(215, 214, 214),0 0 30px rgb(215, 214, 214); /*
Agregar una sombra al contenedor*/
}

.formulario {
    padding: 1px 20px 20px 20px;
    border-radius: 30px;
    background-color: #486f99;
    display: grid;

}
```

```css
.formulario>label {
    text-shadow: 0px 0px 5px rgb(0, 0, 0);/* Agregar una sombra al contenedor*/
}

.formulario>input{
    margin: 10px 0px; /* Margen superior e inferior y el margen izquierdo y derecho */
    padding: 10px; /* Espacio entre el borde y el contenido, incluso de los inputs, se ensanchan las cajas contenedoras input*/
    border: 1px solid #ccc;
    border-radius: 30px;
    width: 100%;
    box-sizing: border-box;
    text-align: center;
    box-shadow: 0px 0px 1.5px black ;
}

.formulario>button {
    margin: 20px 0px 0px 0px; /* Margen superior, lado derecho, margen inferior y margen izquierda */
    padding: 10px; /* Espacio entre el borde y el contenido, incluso de los inputs, se ensanchan las cajas contenedoras input*/
    border: 1px solid #ccc;
    border-radius: 30px;
    width: 100%;
    box-sizing: border-box;
    transition: all 300ms ease 0ms; /* Transicion de todos los valores */
    box-shadow: 0px 0px 1.5px black ;
}

.formulario>button:hover {
    background-color: #bdbdbd;
    transform: translateY(-5px); /* Movimiento hacia arriba, curiosamente al ser negativo sube y al ser positivo baja */
}

.contenedorFiltrar>.filtrarFecha {
    margin: 20px 0px 0px 0px; /* Margen superior, lado derecho, margen inferior y margen izquierda */
    padding: 10px; /* Espacio entre el borde y el contenido, incluso de los inputs, se ensanchan las cajas contenedoras input*/
    border: 1px solid #ccc;
    border-radius: 30px;
```

```
    width: 34%;
    box-sizing: border-box;
    transition: all 300ms ease 0ms; /* Transicion de todos los valores */
    box-shadow: 0px 0px 1.5px black ;
}

.contenedorFiltrar>.filtrarFecha:hover {
    background-color: #bdbdbd;
    transform: translateY(-5px); /* Movimiento hacia arriba, curiosamente al
ser negativo sube y al ser positivo baja */
}

.contenedorFiltrar>.botonfiltrar {
    margin: 20px 0px 0px 0px; /* Margen superior, lado derecho, margen
inferior y margen izquierda */
    padding: 10px; /* Espacio entre el borde y el contenido, incluso de los
inputs, se ensanchan las cajas contenedoras input*/
    border: 1px solid #ccc;
    border-radius: 30px;
    width: 33%;
    box-sizing: border-box;
    transition: all 300ms ease 0ms; /* Transicion de todos los valores */
    box-shadow: 0px 0px 1.5px black ;
}

.contenedorFiltrar>.filtrarFecha:hover {
    background-color: #bdbdbd;
    transform: translateY(-5px); /* Movimiento hacia arriba, curiosamente al
ser negativo sube y al ser positivo baja */
}


.formulario>h2 {
    font-size: 20px;
    margin: 20px;
    text-align: center;
    text-shadow: 0px 0px 3px rgb(0, 0, 0);/* Agregar una sombra al
contenedor*/
}

.contenedorEventos>h2 {
    font-size: 25px;
    margin: 20px;
    text-align: center;
```

```
    text-shadow: 0px 0px 10px rgb(0, 0, 0), 0px 0px 10px rgb(0, 0, 0);/*
Agregar una sombra al contenedor*/
}

.contenedorEventos>ul>li {
    list-style: none;
    display: block;
    background-color: #486f99;
    border-radius: 50px;
    margin: 10px;
    text-align: left;
    text-shadow: 0px 0px 3px rgb(0, 0, 0);/* Agregar una sombra al
contenedor*/
    text-transform: capitalize;
}

.contenedorEventos>ul>li>.boton-borrar {
    float: left; /* Flotar a la derecha */
    padding: 4px; /* Espacio entre el borde y el contenido */
    margin-right: 10px; /* Margen a la derecha */
    border: none; /* Quita el borde */
    border-radius: 10px; /* Redondear los bordes del botón */
    transition: all 300ms ease 0ms; /* Transicion de todos los valores */
    box-shadow: 0px 0px 3px black ;
}

.contenedorEventos>ul>li>.boton-borrar:hover {
    background-color: #f1f1f1ed;
    transform: translateY(-5px); /* Movimiento hacia arriba, curiosamente al
ser negativo sube y al ser positivo baja */
    box-shadow: 0px 0px 3px #ffffffde ;
}

.boton-flotante {
    font-size: 12px; /* Cambiar el tamaño de la tipografia */
    text-transform:capitalize; /* Texto en mayusculas */
    font-weight: bold; /* Fuente en negrita o bold */
    text-decoration: none; /* Quita el subrayado */
    color: #ffffff; /* Color del texto */
    border-radius: 250px; /* Borde del boton */
    letter-spacing: 2px; /* Espacio entre letras */
    background-color: #2f2ce1; /* Color de fondo */
    padding: 10px 10px; /* Relleno del boton, primer parametro a lo alto,
segundo a lo ancho*/
```

```
    position: fixed; /* Posición Fixed o fijada en la pantalla, esta opción habilita las propiedades siguientes de posición de pantalla*/
    bottom: 40px; /* Margen desde abajo */
    right: 40px;  /* Margen desde la derecha */
    transition: all 300ms ease 0ms; /* Transicion de todos los valores */
    box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.1);
    z-index: 1;
    border: none;
}
.boton-flotante:hover {
    background-color: #1e206c; /* Color de fondo al pasar el cursor */
    box-shadow: 0px 15px 20px rgba(0, 0, 0, 0.3); /* Sombra al pasar el cursor */
    transform: translateY(-7px); /* Movimiento hacia arriba, curiosamente al ser negativo sube y al ser positivo baja */
}

@media only screen and (max-width: 600px) { /* Query media para hacer responsive el bóton flotante*/
    .btn-flotante {
        font-size: 14px;
        padding: 12px 20px;
        bottom: 20px;
        right: 20px;
    }
}
```

Parte 3: Interactividad con JavaScript

Respecto al archivo de JavaScrip tuve muchos problemas por los que tuve que hacer varias veces el documento, me guíe inicialmente con el ejercicio que el profesor realizó en clase, adicional a lo anterior cuando ya no sabía que hacer leía la documentación, miraba videos, usaba IA, repasaba los ejercicios de la clase. Empecé el documento guardando en constantes los elementos traídos con el método getElementById, después creo la función para guardar los eventos en el localStorage del navegador, para ello creo una variable local de la función que me guardará el parseo al formato de sintaxis de objeto JSON de la lista de eventos, si no hay eventos me traerá una lista vacía, después de esto en la línea siguiente se hace un push o un empuje de los eventos a esta lista vacía que acabamos de guardar en la variable local o a la lista ya existente que parceamos para traerla, finalmente en el

localStorage del navegador se guarda con el método setItem la nueva lista actualizada o pusheada para posteriormente volver a traerla.

La línea del **events.forEach((event, index) => { addEventToDOM(event, index); });**: recorre el array **events** usando el ciclo **forEach** para cada evento con el índice index, se llama a la función addEventToDOM, pasando el evento y su índice en el array. Esto permite agregar cada evento a la lista en el DOM. Para el manejo de los eventos en el formulario utilicé la guía del ejercicio que realizó el profesor y para la funcionalidad del botón de música, buscar y demás utilicé documentación, ia y tutoriales de youtube, aveces los tutoriales no funcionaban y por eso tenía que reempezar el código porque no lograba descubrir que estaba mal. Adjunto envío el código.

```
const eventForm = document.getElementById('formulary');
const eventTittleEventInput = document.getElementById('InputTittleEvent');
const eventDateInput = document.getElementById('InputDate');
const eventHourInput = document.getElementById('InputHour');
const eventList = document.getElementById('eventList');
const filterDateButton = document.getElementById('filterDateButton');
const filterDateInput = document.getElementById('filterDateInput');

// Función para guardar en localStorage
function saveEventToLocalStorage(event) {
    let events = JSON.parse(localStorage.getItem('events')) || [];
    events.push(event);
    localStorage.setItem('events', JSON.stringify(events));
}

// Función para cargar los eventos desde localStorage y mostrarlos
function loadEvents() {
    let events = JSON.parse(localStorage.getItem('events')) || [];
    events.forEach((event, index) => {
        addEventToDOM(event, index);
    });
}

// Función para agregar un evento al DOM
function addEventToDOM(event, index) {
    const li = document.createElement('li');
    li.innerHTML = `
        <span>${event.titulo} - ${event.fecha} - ${event.hora}</span>
```

```
        <button class="boton-borrar" data-
index="${index}">Eliminar</button>`;
    eventList.appendChild(li);
}

// Función para eliminar un evento del localStorage
function deleteEventFromLocalStorage(index) {
    let events = JSON.parse(localStorage.getItem('events')) || [];
    events.splice(index, 1);
    localStorage.setItem('events', JSON.stringify(events));
    loadEvents(); // Recargar eventos
}

// Delegación de eventos para manejar los botones "Borrar"
eventList.addEventListener('click', function(e) {
    if (e.target.classList.contains('boton-borrar')) {
        const index = e.target.getAttribute('data-index');
        deleteEventFromLocalStorage(index);
        e.target.parentElement.remove(); // Eliminar del DOM
    }
});

// Manejo del envío de datos a través del formulario
eventForm.addEventListener('submit', function(e) {
    e.preventDefault();

    let title = eventTittleEventInput.value;
    let date = eventDateInput.value;
    let hour = eventHourInput.value;

    if (title === '' || date === '' || hour === '') {
        alert('Por favor completa todos los campos!!!');
        return;
    }

    let evento = {
        titulo: title,
        fecha: date,
        hora: hour
    };

    saveEventToLocalStorage(evento);
    addEventToDOM(evento, JSON.parse(localStorage.getItem('events')).length
- 1);
    eventForm.reset();
```

```
});

// Función para filtrar eventos por fecha
function filterEventsByDate() {
    const selectedDate = filterDateInput.value;
    const events = JSON.parse(localStorage.getItem('events')) || [];

    // Limpiar la lista actual de eventos
    eventList.innerHTML = '';

    // Filtrar y mostrar solo los eventos de la fecha seleccionada
    const filteredEvents = events.filter(event => event.fecha ===
selectedDate);

    if (filteredEvents.length > 0) {
        filteredEvents.forEach((event, index) => {
            addEventToDOM(event, index);
        });
    } else {
        eventList.innerHTML = '<li>No hay eventos para esta fecha</li>';
    }

}

// Asignar el evento click al botón para filtrar por fecha
filterDateButton.addEventListener('click', filterEventsByDate);

// Cargar los eventos guardados cuando se cargue la página
document.addEventListener('DOMContentLoaded', loadEvents);

// Obtener las referencias al botón y al audio
const button = document.getElementById('floatbutton');
const song = document.getElementById('song');

// Función para alternar entre reproducir y pausar
function PlayOPause() {
    if (song.paused) {
        song.play();
        button.textContent = 'Pause Music';
    } else {
        song.pause();
        button.textContent = 'Play Music';
    }
}
```

```
// Asignar el evento de clic al botón de música
button.addEventListener('click', PlayOPause);

// Creación de la función para esperar a que todo el contenido JavaScript
esté cargado porque al
// Función para esperar a que todo el contenido JavaScript esté cargado
window.addEventListener('load', function () {
    document.body.style.visibility = 'visible';
});
```

Conclusión

El informe de la segunda práctica de laboratorio del módulo “Introducción a los lenguajes de programación” presenta la aplicación de lo que he logrado aprender para crear un calendario web interactivo usando HTML, CSS y JavaScript. Adicionalmente considero que existen tres puntos importantes para recordar: aplicación práctica de conceptos fundamentales de HTML, CSS y JavaScript en un proyecto web, con énfasis en la creación y manipulación dinámica de DOM, interactividad y almacenamiento de eventos locales en el localStorage del navegador (Brave es el que uso usualmente); uso de herramientas y entornos de desarrollo, habiendo utilizado Visual Studio Code, Git, GitHub, y Editor de PDF-Xchange e IAs para desarrollar el proyecto, lo que indica familiaridad con un flujo de trabajo profesional; y desafíos de diseño e implementación porque, aunque una página funcional con interactividad y un diseño visual agradable fue creado, tuve dificultades como por ejemplo las mencionadas con los errores de Java Script. Todo lo anterior nos motiva a practicar aún más.