Informe de Laboratorio 03

Tema: NodeJS + Express

Nota

Estudiante	Escuela	Asignatura
David Alfredo Huamani Ollachica dhuamanio@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20230485

Laboratorio	Tema	Duración
03	NodeJS + Express	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 20 Mayo 2024	Al 24 Mayo 2024

1. Tarea

- Cree una aplicación NodeJS con Express, para administrar una agenda personal.
- **Rutas de la Aplicación:**
  - `Home (/")`: Página Principal.
- Trabaje todo en una misma interfaz.
- La aplicacion debe permitir
  - **Crear evento:** Fecha y hora. (Si ya existe el archivo no debería ingresar el evento). La primera línea es el título del evento, las demás líneas son la descripción del evento.
  - **Editar evento:** Se muestra el archivo donde está el detalle del evento.
  - **Eliminar evento.**
  - **Ver eventos:** Utilizar el formato árbol especificado anteriormente, donde debería incluirse sólo el título del evento.
- Utilice DockerFile para realizar operaciones automatizadas en Docker (incluido arrancar el servidor web nginx a traves de un puerto y copiar el proyecto web para acceder desde la máquina anfitrion.)
- Produción acceder a la aplicación NodeJS+Express a traves de un servidor web robusto (Nginx).
- Ejemplo: http://127.0.0.1:8084/lab04/

2. URL de Repositorio Github

- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/dev1d123/pw2_lab04`

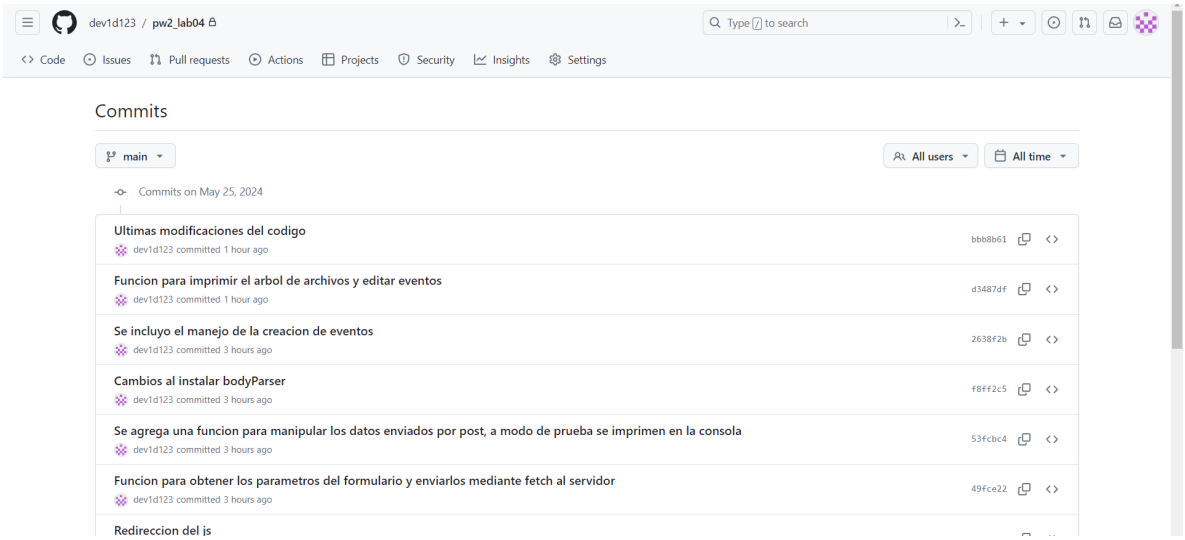

Figura 1: Imagen de los commits

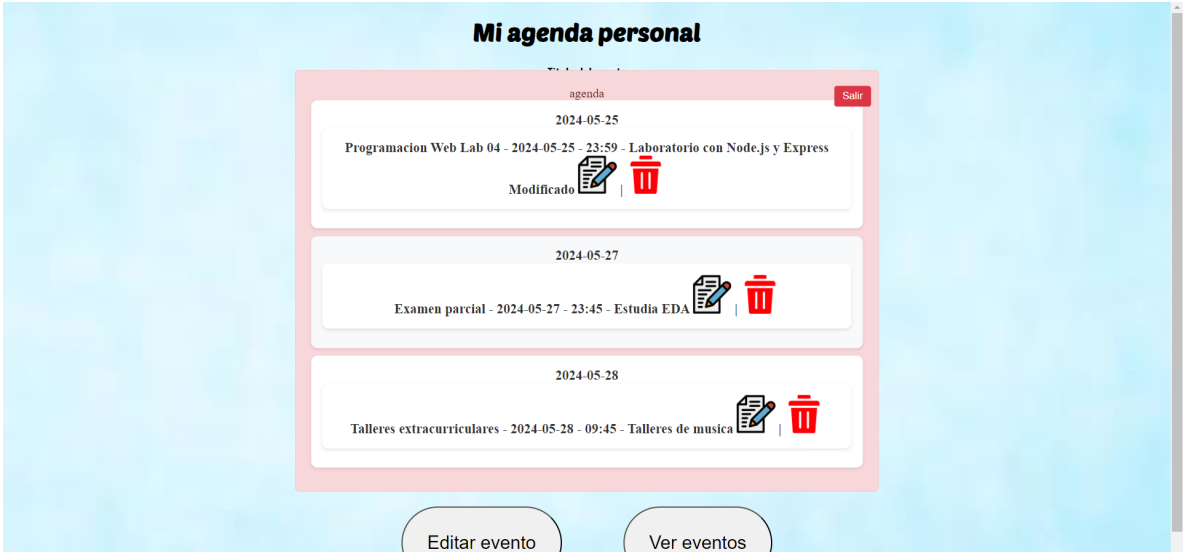

Figura 2: Ejecucion

3. Entregables

3.1. HTML

- Documento HTML, se implemento un formulario y unos botones para mostrar las opciones para ver y editar los eventos de la agenda.
- Se implemento todo en una sola interfaz es decir solo se utilizo un solo documento html.

Listing 1: public/html/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi agenda</title>
    <link rel="stylesheet" href="../css/styles.css">
</head>
<body>
    <header>
        <h1>Mi agenda personal</h1>
    </header>
    <main>
        <form id="myForm" class="myForm">
            <label for="title">Titulo del evento</label>
            <br>
            <input type="text" name="title" id = "title">
            <br>

            <label for="fecha">Fecha</label>
            <br>
            <input type="date" name="fecha" id = "fecha">
            <br>
            <label for="hora">Hora</label>
            <br>
            <input type="time" name="hora" id = "hora">
            <br>
            <label for="descripcion"></label>
            <br>
            <textarea name = "descripcion" rows = "20" cols ="20">Ingrese la descripcion de su
                evento!</textarea>
            <br>
            <input type="submit" name="enviar" value="Registrar evento">
            <br>
        </form>
        <div class="buttons">
            <button id="edite">Editar evento</button>
            <button id="see">Ver eventos</button>
        </div>
    </main>

    <div class="editarEvento" id = "editarEvento">
    </div>
    <div class="seeEventos">
        <ul id="agenda-tree"></ul>
        <button onclick="closeSee()">Salir</button>
```

```
    </div>
    <script src="../javascript/script.js"></script>
</body>
</html>
```

3.2. Estilos

- En el documento CSS se implementaron todos los estilos necesarios para el codigo, ademas se añadieron estilos para las listas y los divs ocultos. Por ultimo se agregaron fuentes de google.

Listing 2: public/css/styles.css

```
@import url('https://fonts.googleapis.com/css2?family=Poetsen+One&display=swap');
@import
    url('https://fonts.googleapis.com/css2?family=Oswald:wght@200..700&family=Poetsen+One&display=swap');
body{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    display: flex;
    justify-content: center;
    text-align: center;
    flex-direction: column;
    background-image: url("../img/bg.jpg");
    background-repeat: no-repeat;
    background-size: cover;
}

header{
    font-family: "Poetsen One", sans-serif;
    font-weight: 400;
    font-style: normal;
}
main{
    font-family: "Oswald", sans-serif;
    font-optical-sizing: auto;
    font-weight: 500;
    font-style: normal;
}
.buttons{
    display: flex;
    justify-content: center;
    gap: 5rem;
    margin-top: 4rem;
}

.buttons button{
    padding: 2rem;
    border-radius: 3rem;
    font-size: 1.5rem;
    transition: transform 0.3s, color 0.3s, background-color 0.3s;
    cursor: pointer;
}

button:hover{
```

```
    transform: scale(1.2);
    color: white;
    background-color: black;
}




.editarEvento, .seeEventos {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: #f8d7da;
    color: #721c24;
    padding: 20px;
    border: 1px solid #f5c6cb;
    border-radius: 5px;
}

.editarEvento button, .seeEventos button {
    margin-top: 10px;
    padding: 5px 10px;
    background-color: #dc3545;
    color: #fff;
    border: none;
    border-radius: 3px;
    cursor: pointer;
}
.editarEvento{
    display: none;
}

.seeEventos{
    display: none;
}
ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
}

li {
    padding: 15px;
    margin-bottom: 10px;
    background-color: #f8f9fa;
    border-radius: 8px;
    font-size: 1.1rem;
    font-weight: bold;
    color: #333;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```
li:nth-child(odd) {
    background-color: #ffffff;
}
```

3.3. Javascript lado del cliente

- En este documento se agrega la logica de la agenda por parte del servidor se hacen algunas instrucciones DOM, y se efectuan las llamadas al servidor mediante fetch.

Listing 3: Ejercicio3/index.html

```
var editar = document.getElementById("edite");
var ver = document.getElementById("see");

var f = document.querySelector(".editarEvento");
var s = document.querySelector(".seeEventos");

function closeEdit() {
    f.style.display = "none";
}

function closeSee() {
    s.style.display = "none";
}


ver.addEventListener('click', function(event){
    s.style.display = "block";
    fetch('/agenda-tree')
    .then(response => response.text())
    .then(tree => {
        document.getElementById('agenda-tree').innerHTML = tree;
    })
    .catch(error => console.error('Error fetching agenda tree:', error));

    console.log("nothing");
})

editar.addEventListener('click', function(event){

    f.style.display = "block";

    fetch('/agenda-editar')
    .then(response => response.text())
    .then(tree => {
        document.getElementById('editarEvento').innerHTML = tree;
    })
    .catch(error => console.error('Error fetching agenda tree:', error));

    console.log("nothing");
});

document.getElementById('myForm').addEventListener('submit', function(event) {
    event.preventDefault();
```

```
    const title = document.getElementById('title').value;
    const fecha = document.getElementById('fecha').value;
    const hora = document.getElementById('hora').value;
    const descripcion = document.querySelector('textarea[name="descripcion"]').value;

    const data = { title, fecha, hora, descripcion };

    fetch('/register', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: new URLSearchParams(data),
    })
    .then(response => response.text())
    .then(data => {
      alert(data);
    })
    .catch((error) => {
      console.error('Error:', error);
    });
});
```

3.4. Javascript lado del servidor, NodeJS + Express

- Aqui se implementa toda la logica usada en el servidor, usando NodeJS y express, se manejan las solicitudes del cliente de manera efectiva.

Listing 4: Ejercicio3/index.html

```
const path = require('path');
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('fs');
const app = express();
const port = 3000;
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.urlencoded({ extended: true }));
app.get('/', (req, res) => {
  res.sendFile(path.resolve(__dirname, 'public/html/index.html'));
});
app.post('/register', (req, res) => {
    var {title, fecha, hora, descripcion} = req.body;
    //se crea la ruta de direccion para los archivos
    const dirPath = path.join(__dirname, 'agenda', fecha);
    //verificacion adicional si no existe
    if (!fs.existsSync(dirPath)) {
        fs.mkdirSync(dirPath, { recursive: true });
    }
    var titulo = hora.replace(/:/g, '-');
    //Ruta del evento con su fecha como titulo
    const filePath = path.join(dirPath, `${titulo}.txt`);
    var fileContent = `Titulo del evento: ${title}\nFecha: ${fecha}\nHora:
        ${hora}\nDescripcion: ${descripcion}`;
```

```
    fs.writeFile(filePath, fileContent, (err) => {
        if (err) {
            console.error(err);
            res.status(500).send('Error al registrar el evento');
        } else {
            res.status(200).send('Evento registrado exitosamente');
        }
    });

    console.log("Los parametros enviados son: ", title, " ", fecha, " ", hora, " ",
        descripcion);
});

app.listen(port, () => {
  console.log(`Escuchando en: http://localhost:${port}`);
});

//Generar el arbol de directorios como una lista
function generateTree(dir) {
    const stats = fs.statSync(dir);
    if (stats.isDirectory()) {
        const files = fs.readdirSync(dir);
        let tree = `<ul>${path.basename(dir)}\n`;
        files.forEach(file => {
            const filePath = path.join(dir, file);
            const fileStats = fs.statSync(filePath);
            if (fileStats.isFile()) {
                const content = fs.readFileSync(filePath, 'utf-8');
                const matches = content.match(/Titulo del evento: (.+)\nFecha: (.+)\nHora:
                    (.+)\nDescripcion: (.+)/);
                if (matches) {
                    const title = matches[1];
                    const fecha = matches[2];
                    const hora = matches[3];
                    const descripcion = matches[4];
                    tree += `<li>${title} - ${fecha} - ${hora} - ${descripcion}</li>\n`;
                } else {
                    tree += `<li>${file}</li>\n`;
                }
            } else if (fileStats.isDirectory()) {
                tree += `<li>${generateTree(filePath)}</li>\n`;
            }
        });
        tree += `</ul>`;
        return tree;
    } else {
        return path.basename(dir);
    }
}

//Funcion para enviar el arbol!
app.get('/agenda-tree', (req, res) => {
    const agendaDir = path.join(__dirname, 'agenda');
    const tree = generateTree(agendaDir);
    res.send(tree);
```

```
});
function generateDelete(dir) {
    const stats = fs.statSync(dir);
    if (stats.isDirectory()) {
        const files = fs.readdirSync(dir);
        let tree = `<ul>${path.basename(dir)}\n`;
        files.forEach(file => {
            const filePath = path.join(dir, file);
            const fileStats = fs.statSync(filePath);
            if (fileStats.isFile()) {
                const content = fs.readFileSync(filePath, 'utf-8');
                const matches = content.match(/Titulo del evento: (.+)\nFecha: (.+)\nHora:
                    (.+)\nDescripcion: (.+)/);
                if (!matches) {
                    return res.status(404).send('No se encontraron datos validos en el
                        archivo.');
                }
                const title = matches[1];
                const fecha = matches[2];
                const hora = matches[3];
                const descripcion = matches[4];

                tree += `<li>
                            ${title} - ${fecha} - ${hora} - ${descripcion}
                            <a href="/edit?file=${filePath}"><img src="../img/edit.png"
                                width="50px"></a> |
                            <a href="/delete?file=${filePath}"><img src="../img/delete.png"
                                width="50px"></a>
                        </li>\n`;
            } else if (fileStats.isDirectory()) {
                tree += `<li>${generateDelete(filePath)}</li>\n`;
            }
        });
        tree += `</ul>`;
        tree += `<button onclick="closeEdit()" style="position: fixed;top: 10px;right: 10px;
            z-index: 9999;">Salir</button>`;
        return tree;
    } else {
        return path.basename(dir);
    }
}
app.get('/agenda-editar', (req, res) => {
    const agendaDir = path.join(__dirname, 'agenda');
    const tree = generateDelete(agendaDir);
    res.send(tree);
});

app.get('/edit', (req, res) => {
    console.log(req.query.file);
    const filePath = req.query.file;
    if (!filePath) {
        return res.status(400).send('Ruta del archivo no proporcionada.');
    }

    fs.readFile(filePath, 'utf-8', (err, content) => {
        if (err) {
```

```
            console.error(err);
            return res.status(500).send('Error al leer el archivo.');
        }

        const matches = content.match(/Titulo del evento: (.+)\nFecha: (.+)\nHora:
            (.+)\nDescripcion: (.+)/);
        if (!matches) {
            return res.status(404).send('No se encontraron datos validos en el archivo.');
        }
        const title = matches[1];
        const fecha = matches[2];
        const hora = matches[3];
        const descripcion = matches[4];
        res.send(`
            <div class="form-Edit">
                <form id="editForm" class="myForm" method="POST"
                    action="/update?file=${encodeURIComponent(filePath)}">
                    <label for="title">Titulo del evento</label>
                    <br>
                    <input type="text" name="title" id="title2" value="${title}" readonly>
                    <br>
                    <label for="fecha">Fecha</label>
                    <br>
                    <input type="date" name="fecha" id="fecha2" value="${fecha}" readonly>
                    <br>
                    <label for="hora">Hora</label>
                    <br>
                    <input type="time" name="hora" id="hora2" value="${hora}" readonly>
                    <br>
                    <label for="descripcion">Descripcion</label>
                    <br>
                    <textarea name="descripcion" rows="20" cols="20">${descripcion}</textarea>
                    <br>
                    <input type="submit" name="enviar" value="Actualizar evento">
                    <br>
                </form>
            </div>
        `);
    });
});

app.post('/update', (req, res) => {
    const { title, fecha, hora, descripcion } = req.body;
    const filePath = req.query.file;

    if (!filePath) {
        return res.status(400).send('Ruta del archivo no proporcionada.');
    }

    const newContent = `Titulo del evento: ${title}\nFecha: ${fecha}\nHora:
        ${hora}\nDescripcion: ${descripcion}`;

    fs.writeFile(filePath, newContent, (err) => {
        if (err) {
            console.error(err);
            return res.status(500).send('Error al actualizar el evento.');
```

```
        }
        res.redirect('/');
    });
});
app.get('/delete', (req, res) => {
    const filePath = req.query.file;
    if (!filePath) {
        return res.status(400).send('Ruta del archivo no proporcionada.');
    }
    fs.unlink(filePath, (err) => {
        if (err) {
            console.error(err);
            return res.status(500).send('Error al eliminar el archivo.');
        }
        res.redirect('/');
    });
});
```

4. Pregunta

4.1. Diferencias entre XmlHttpRequest, jQuery.ajax y Fetch

La diferencia principal entre las conexiones asíncronas utilizando el objeto `XmlHttpRequest`, `jQuery.ajax` y `Fetch` radica en su sintaxis y en la forma en que manejan las solicitudes y respuestas. Aquí hay una breve comparación con un ejemplo básico de cada uno:

1. **XmlHttpRequest:**
   - Este es un objeto proporcionado por el navegador para realizar solicitudes HTTP asíncronas desde JavaScript.
   - Requiere más código y es menos intuitivo en comparación con las opciones más modernas.

   Ejemplo:

```
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);
xhr.onreadystatechange = function() {
  if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
    console.log(xhr.responseText);
  }
};
xhr.send();
```

2. **jQuery.ajax:**
   - jQuery es una biblioteca de JavaScript que simplifica el manejo de eventos, animaciones y comunicaciones AJAX.
   - `$.ajax()` es un método de jQuery para realizar solicitudes HTTP asíncronas.
   - Es más fácil de usar y requiere menos código en comparación con `XmlHttpRequest`.

   Ejemplo:

```
$.ajax({
  url: 'https://api.example.com/data',
  method: 'GET',
  success: function(response) {
    console.log(response);
  }
});
```

3. **Fetch API:**

   - Fetch es una API moderna proporcionada por los navegadores para realizar solicitudes HTTP asíncronas.
   - Es más moderno y más fácil de usar que `XmlHttpRequest` y no requiere el uso de bibliotecas externas como jQuery.
   - Utiliza Promesas para manejar las respuestas.

   Ejemplo:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

En resumen, Fetch es la opción más moderna y nativa, jQuery.ajax es una opción más fácil de usar si ya está utilizando jQuery en su proyecto, y XmlHttpRequest es una opción más antigua que todavía se encuentra en algunos proyectos legacy.

5. Rúbricas

5.1. Sobre el informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	2	
Total		20		16	

6. Referencias

- https://www.w3schools.com/js/default.asp