.

Informe de Laboratorio 04

Tema: NodeJS + Express

Nota

Estudiante	Escuela	Asignatura
Rafael Diego Nina Calizaya rninacal@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Laboratorio	Tema	Duración
04	NodeJS + Express	06

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

1. URL del Repositorio:

https://github.com/DrN25/pw2_24a/tree/main/Lab04

2. Tarea:

Cree una aplicación NodeJS con express, para administrar una agenda personal.

Peticiones y Funciones destacadas.

Se crearon varias peticiones y funciones para el funcionamiento de la página usando NodeJS y Express, entre ellas las más destacadas son:

- **app.post('/submit' ...):** Esta petición se encarga de crear los eventos. Para ello recibe 3 atributos(fecha, hora, contenido), con los cuales crea una carpeta (si es que no existe) y un documento .txt con su contenido. Una vez creado el evento se almacena en la carpeta .ªgenda"de manera funcional.

```
16  app.post('/submit', (req, res) => {
17          const {fecha, hora, contenido} = req.body;
18          const folderPath = path.join(__dirname, 'agenda' , fecha);
19          const filePath = path.join(folderPath, `${hora}.txt`);
20          if(!fs.existsSync(folderPath)) {
21                  fs.mkdirSync(folderPath, { recursive: true });
22          }
23          fs.writeFileSync(filePath, contenido);
24          res.send('Archivo guardado correctamente.');
25  });
```

- **app.get('/buscar' ...):** Esta petición es la primera parte de la sección Editar. Esta solo recibe 2 atributos(fecha, hora) y con estos verifica que el documento elegido exista, si es así se coloca el contenido del documento en el textarea continuo al botón para que el usuario pueda editar el documento ya presente, caso contrario se notifica que no existe un documento con los datos enviados.

```
27  app.get('/buscar', (req, res) => {
28          const {fecha, hora} = req.query;
29          const filePath = path.join(__dirname, 'agenda', fecha, `${hora}.txt`);
30          if(fs.existsSync(filePath)) {
31                  const contenido = fs.readFileSync(filePath, 'utf-8');
32                  res.json({contenido});
33          } else {
34                  res.status(404).send('Archivo no encontrado');
35          }
36  });
```

- **app.post('/editar' ...):** Esta petición es la segunda parte de la sección Editar. Tras la verificación de la petición anterior y con el texto ya modificado, al presionar el botón de enviar se mandaran los 3 atributos(fecha, hora, contenido) como en la petición /submit, y tras ubicar al archivo se reescribirá el mismo pero con el nuevo contenido entregado por el usuario.

```
38  app.post('/editar', (req, res) => {
39          const {fecha, hora, contenido} = req.body;
40          const folderPath = path.join(__dirname, 'agenda', fecha);
41          const filePath = path.join(folderPath, `${hora}.txt`);
42          if(fs.existsSync(filePath)) {
43                  fs.writeFileSync(filePath, contenido);
44          res.send('Archivo editado exitosamente');
45          } else {
46                  res.status(404).send('Archivo no encontrado');
47          }
48  });
```

- **app.post('/eliminar' ...):** Esta petición se encarga de eliminar los eventos. Para ello se reciben 2 atributos(fecha, hora), con los cuales se ubicará al documento para luego proceder a eliminarlo de su carpeta designada.

```
50  app.post('/eliminar', (req, res) => {
51          const { fecha, hora } = req.body;
52          const folderPath = path.join(__dirname, 'agenda', fecha);
53          const filePath = path.join(folderPath, `${hora}.txt`);
54          if (fs.existsSync(filePath)) {
55                 fs.unlinkSync(filePath);
56          res.send('Archivo eliminado exitosamente');
57          } else {
58                  res.status(404).send('Archivo no encontrado');
59          }
60  });
```

- **app.get('/arbol' ...):** Esta petición se encarga de realizar el proceso para retornar el árbol que contiene a los directorios y documentos. Este llama a la función crearArbol() mandando como atributo la dirección del directorio, la cual retorna un árbol de su contenido con el formato pedido en la práctica.

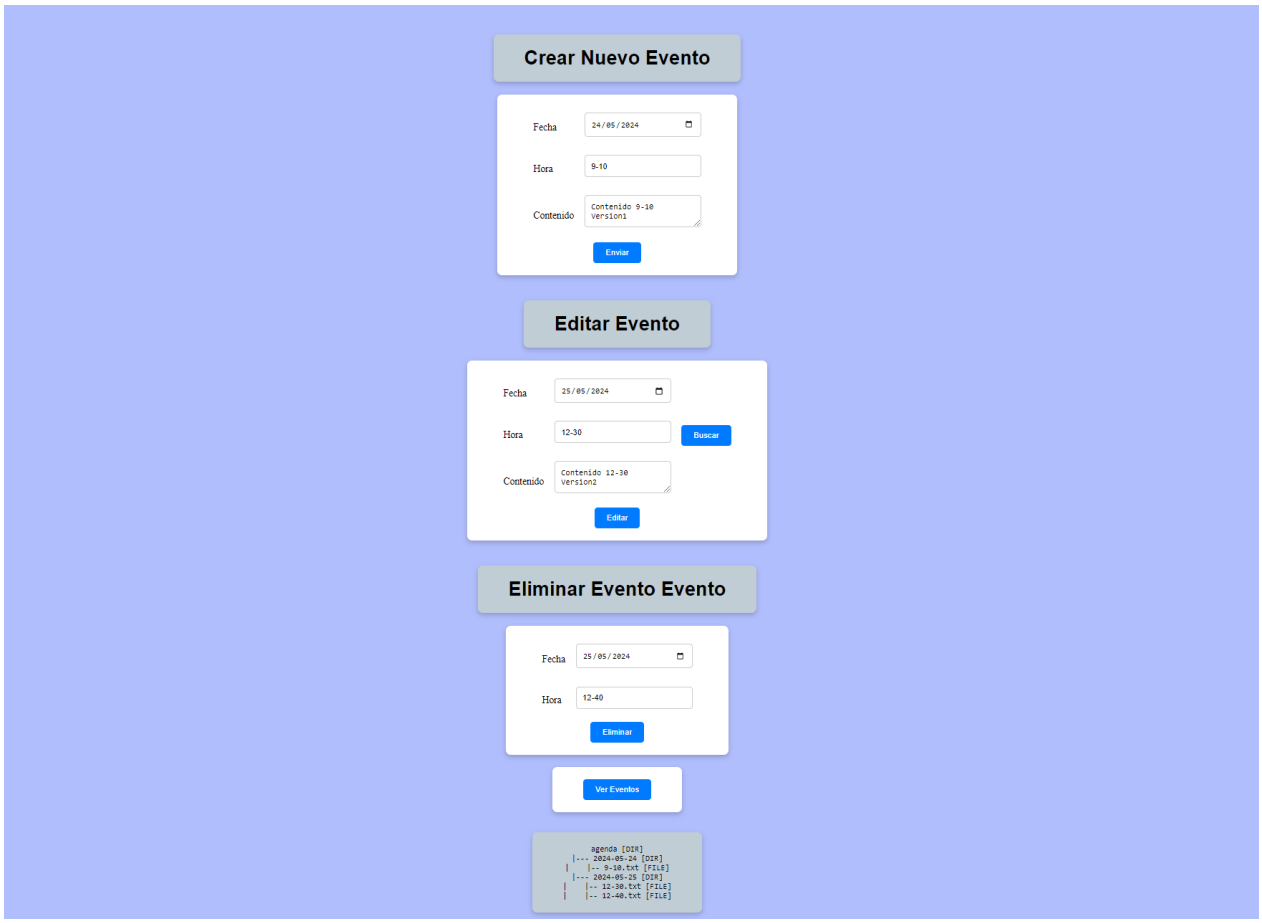
```
app.get('/arbol', (req, res) => {
        const arbol = crearArbol(path.join(__dirname, 'agenda'));
        res.send(`${arbol.replace(/\n/g, '<br>')}`);
});
```

- **crearArbol():** Esta función lee el contenido del directorio actual, creando una constante la cual contendrá el árbol a devolver. Se itera sobre cada elemento, donde dependiendo si es carpeta o documento se colocará un prefijo -guiones u otros, y se utilizan — para indicar la jerarquía de carpetas y documentos.

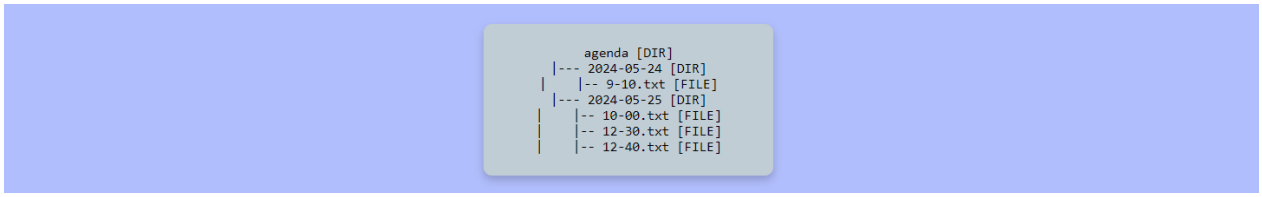
```
const crearArbol = (dirActual) => {
        let resultado = `${path.basename(dirActual)} [DIR]\n`;
        const items = fs.readdirSync(dirActual);
        items.forEach((item) => {
                const rutaItem = path.join(dirActual, item);
                const esDirectorio = fs.statSync(rutaItem).isDirectory();
                const prefijo = esDirectorio ? '[DIR]' : '[FILE]';
                const guion = esDirectorio ? '---' : '--';
                resultado += `|${guion} ${item} ${prefijo}\n`;
                if (esDirectorio) {
            const subItems = fs.readdirSync(rutaItem);
                                subItems.forEach((subItem) => {
                                resultado += `|    |-- ${subItem} [FILE]\n`;
                        });
                }
        });
        return resultado;
};
```

Funcionamiento de la Página Web.

- **Página Web:**

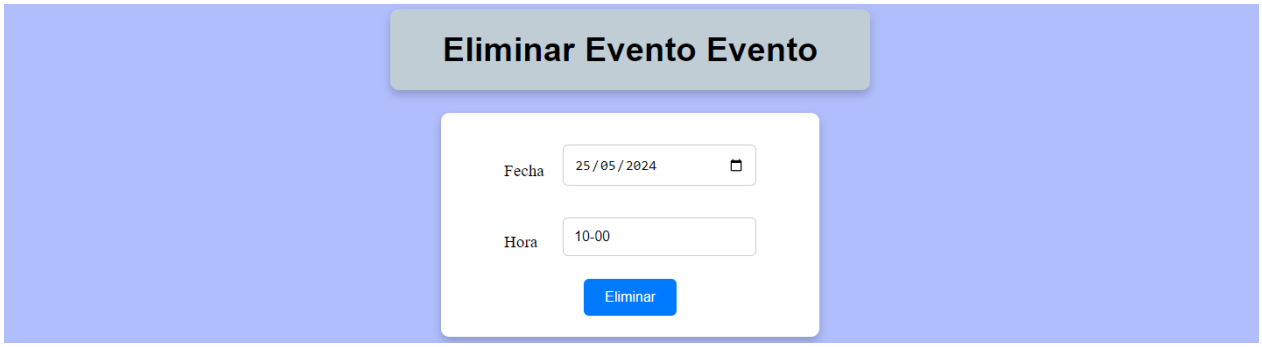


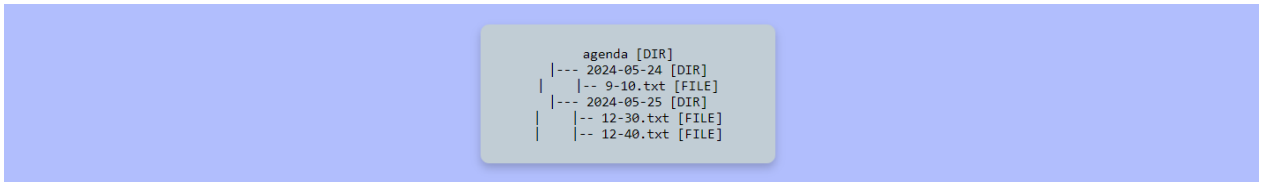
- **Probando Creación de Eventos:**



- **Probando Edición de Eventos:**

- **Probando Eliminación de Eventos:**





Pregunta.

Mencione la diferencia entre conexiones asíncronas usando el objeto XmlHttpRequest, JQuery.ajax y Fetch. Justifique su respuesta con un ejemplo muy básico. Eje: Hola Mundo, IMC, etc.

- **XmlHttpRequest:** Es la API original de JavaScript, requiere manejar de forma manual múltiples eventos.

```javascript
const xml = new XMLHttpRequest();
xml.open('GET', 'https://url');
xml.onload = function() {
  if (xml.status === 200) {
    console.log(xml.responseText); // Se imprime "Hola Mundo"
  } else {
    console.error('Error:', xml.statusText);
  }
};
xmlh.send();
```

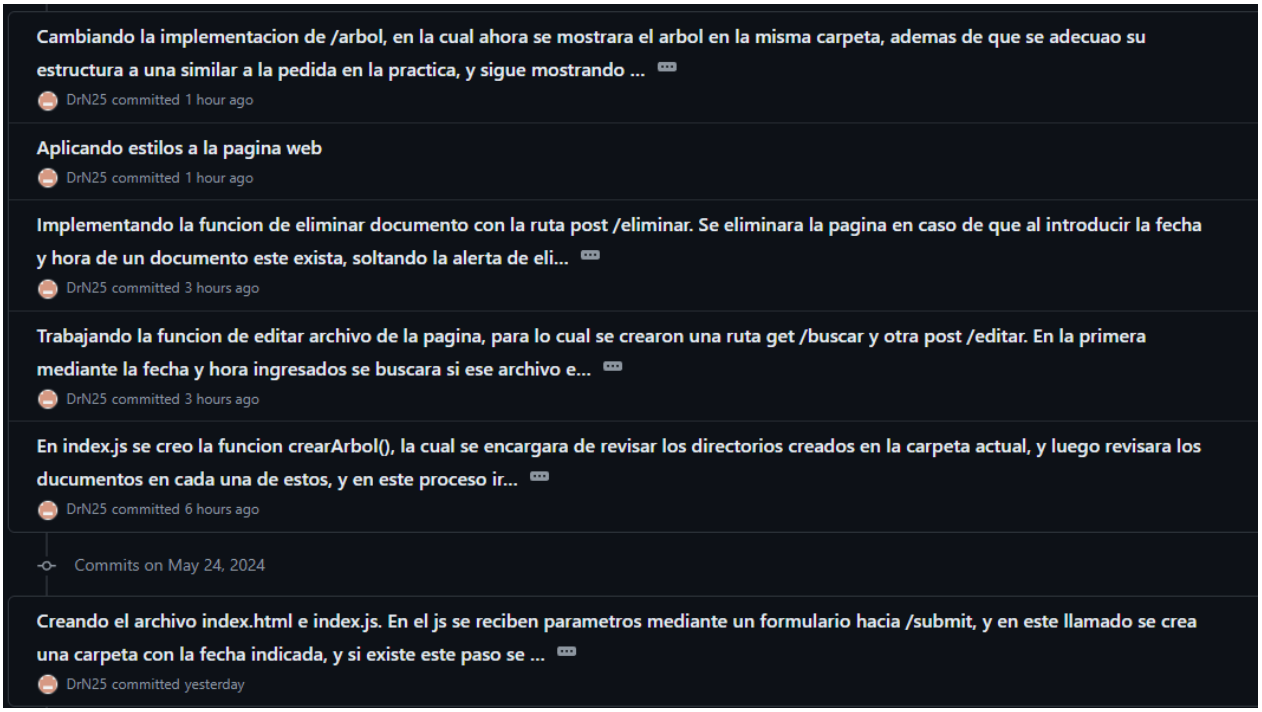
- **JQuery.ajax:** Es una biblioteca de JavaScript, la cual usa la función $.ajax al configurar objetos.

```javascript
$.ajax({
  url: 'https://url',
  success: function(data) {
    console.log(data); // Se imprime "Hola Mundo"
  },
  error: function(error) {
    console.error('Error:', error);
  }
});
```

- **Fetch:** Es la API más moderna. Su funcionamiento viene acompañado de "promesas", manejando respuestas como .then y .catch.

```javascript
fetch('https://url')
  .then(response => response.text())
  .then(text => console.log(text)) // Se imprime "Hola Mundo"
  .catch(error => console.error('Error:', error));
```

3. Commits realizados:



4. Rúbrica:

	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

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o onservación)	4	X	4	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
3. Ejecución	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	X	3	
4. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	1.5	
7. Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	x	3	
TOTAL		20		16.5	