

## Informe de Laboratorio 04

### Tema: NodeJS + Express

Nota

Estudiante	Escuela	Asignatura
Fernandez Huarca, Rodrigo Alexander rfernandezh@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20230465

Laboratorio	Tema	Duración
04	Node JS + Express	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 22 Mayo 2024	Al 25 Mayo 2024

### 1. Tarea

- Cree una aplicación NodeJS con Express, para administrar una agenda personal.
- Home ("/") : Pagina Principal
- Trabaje todo en una misma interfaz.
- Ejemplo de estructura de la agenda cuando se explora "Eventos".
- La aplicación 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 muestran el archivo donde esta el detalle del evento)
  - Eliminar evento.

## 2. Acceso al Repositorio en GitHub

- URL para el laboratorio 04 para clonar o recuperar.
  - <https://github.com/RodrigoFH/FernandezHuarcaRodrigoAlexander.git>
- URL para el laboratorio 04 en el Repositorio GitHub.
  - <https://github.com/RodrigoFH/FernandezHuarcaRodrigoAlexander/tree/main/Laboratorio04/personal-agenda>

## 3. Actividad:

### 3.1. Archivos HTML usados:

- Para esta sección se creó 3 HTML para una mejor división de los segmentos de la aplicación:  
**index.html:** Este archivo representa la página principal de la agenda personal. Aquí es donde se muestran todos los eventos almacenados. Los usuarios pueden ver la lista de eventos y tienen enlaces para editar o eliminar eventos, así como un enlace para agregar un nuevo evento.

Listing 1: index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <link rel="stylesheet" href="css/styles.css">
7     <title>Personal Agenda</title>
8 </head>
9 <body>
10     <h1>Personal Agenda</h1>
11     <a href="add.html">Add Event</a>
12     <ul id="eventsList"></ul>
13
14     <script src="js/script.js"></script>
15 </body>
16 </html>
```

**add.html:** Este archivo contiene el formulario para agregar un nuevo evento. Se accede a esta página cuando el usuario hace clic en el enlace "Add Event" en la página principal. Separar esta funcionalidad en un archivo independiente permite que el código HTML sea más limpio y específico para la tarea de agregar eventos.

Listing 2:add.html

---

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/styles.css">
7   <title>Add Event</title>
8 </head>
9 <body>
10   <h1>Add Event</h1>
11   <form id="addEventForm">
12     <input type="date" name="date" required>
13     <input type="time" name="time" required>
14     <input type="text" name="title" placeholder="Event Title" required>
15     <textarea name="description" placeholder="Event Description"
16 required></textarea>
17     <button type="submit">Add Event</button>
18   </form>
19   <a href="/">Back to Agenda</a>
20
21   <script src="js/script.js"></script>
22 </body>
</html>
```

---

**edit.html:** Este archivo contiene el formulario para editar un evento existente. Se accede a esta página cuando el usuario hace clic en el enlace "Edit" junto a un evento en la página principal.

Listing 3:edit.html

---

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/styles.css">
7   <title>Edit Event</title>
8 </head>
9 <body>
10   <h1>Edit Event</h1>
11   <form id="editEventForm">
```

---

```
12     <input type="hidden" name="id">
13     <input type="date" name="date" required>
14     <input type="time" name="time" required>
15     <input type="text" name="title" required>
16     <textarea name="description" required></textarea>
17     <button type="submit">Save Changes</button>
18 </form>
19 <a href="/">Back to Agenda</a>
20
21 <script src="js/script.js"></script>
22 </body>
23 </html>
```

### 3.2. Hoja de estilos CSS:

- Para esta sección se uso solo una hoja de estilos para que todas las paginas guarden un estilo similar.

Listing 4:styles.css

```
1 body {
2     font-family: Arial, sans-serif;
3     margin: 0;
4     padding: 0;
5     padding: 20px;
6     background-color: #f0f0f0;
7 }
8
9 h1 {
10     color: #1e95ac;
11 }
12
13 form {
14     margin-bottom: 20px;
15 }
16
17 input, textarea {
18     display: block;
19     margin-bottom: 10px;
20     padding: 10px;
21     width: 100%;
22     box-sizing: border-box;
```

```
23 }
24
25 button {
26     padding: 10px 20px;
27     background-color: #289ca7;
28     color: white;
29     border: none;
30     cursor: pointer;
31 }
32
33 button:hover {
34     background-color: #289ca7;
35 }
36
37 ul {
38     list-style: none;
39     padding: 0;
40 }
41
42 li {
43     background-color: white;
44     padding: 20px;
45     margin-bottom: 10px;
46     border: 1px solid #ddd;
47 }
48
49 form[style="display:inline;"] {
50     display: inline;
51 }
```

### 3.3. Scripts Usados:

- **app.js:** Este archivo es el servidor principal de la aplicación. Utiliza Node.js y Express para manejar las solicitudes del cliente y gestionar los datos de los eventos. Aquí están sus responsabilidades principales:

#### Listing 5:app.js

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const fs = require('fs');
4 const path = require('path');
```

```
5 const app = express();
6 const port = 3000;
7
8 app.use(bodyParser.json());
9 app.use(bodyParser.urlencoded({ extended: true }));
10 app.use(express.static('public'));
11
12 let events = require('./events.json');
13
14 app.get('/', (req, res) => {
15     res.sendFile(path.join(__dirname, 'public', 'index.html'));
16 });
17
18 app.get('/add', (req, res) => {
19     res.sendFile(path.join(__dirname, 'public', 'add.html'));
20 });
21
22 app.get('/edit/:id', (req, res) => {
23     res.sendFile(path.join(__dirname, 'public', 'edit.html'));
24 });
25
26 app.get('/api/events', (req, res) => {
27     res.json(events);
28 });
29
30 app.post('/api/events', (req, res) => {
31     const newEvent = {
32         id: Date.now(),
33         date: req.body.date,
34         time: req.body.time,
35         title: req.body.title,
36         description: req.body.description
37     };
38     events.push(newEvent);
39     fs.writeFileSync('events.json', JSON.stringify(events, null, 2));
40     res.status(201).json(newEvent);
41 });
42
43 app.put('/api/events/:id', (req, res) => {
44     const id = parseInt(req.params.id);
```

```
45     const index = events.findIndex(e => e.id === id);
46     if (index !== -1) {
47         events[index] = {
48             id: id,
49             date: req.body.date,
50             time: req.body.time,
51             title: req.body.title,
52             description: req.body.description
53         };
54         fs.writeFileSync('events.json', JSON.stringify(events, null, 2));
55         res.json(events[index]);
56     } else {
57         res.status(404).json({ error: 'Event not found' });
58     }
59 });
60
61 app.delete('/api/events/:id', (req, res) => {
62     const id = parseInt(req.params.id);
63     events = events.filter(e => e.id !== id);
64     fs.writeFileSync('events.json', JSON.stringify(events, null, 2));
65     res.status(204).send();
66 });
67
68 app.listen(port, () => {
69     console.log(App running at http://localhost:${port}`);
70 });
```

- **script.js:** Es el código frontend. Se encarga de la interacción con el usuario, manipula el DOM y realiza solicitudes a las rutas API del servidor.

#### Listing 6:script.js

```
1 document.addEventListener('DOMContentLoaded', () => {
2     const eventsList = document.getElementById('eventsList');
3
4     if (eventsList) {
5         fetch('/api/events')
6             .then(response => response.json())
7             .then(events => {
8                 events.forEach(event => {
9                     const li = document.createElement('li');
```

```
10         li.innerHTML = `
11             <strong>${event.date} ${event.time} -
12             ${event.title}</strong><br>
13             ${event.description}
14             <form action="#" method="POST" class="deleteForm" data-
15             id="${event.id}">
16                 <button type="submit">Delete</button>
17             </form>
18             <a href="edit.html?id=${event.id}">Edit</a>
19         `;
20         eventsList.appendChild(li);
21     });
22
23     document.querySelectorAll('.deleteForm').forEach(form => {
24         form.addEventListener('submit', function (e) {
25             e.preventDefault();
26             const id = this.getAttribute('data-id');
27             fetch(`/api/events/${id}`, {
28                 method: 'DELETE'
29             }).then(() => location.reload());
30         });
31     });
32
33     const addEventForm = document.getElementById('addEventForm');
34     if (addEventForm) {
35         addEventForm.addEventListener('submit', function (e) {
36             e.preventDefault();
37             const formData = new FormData(this);
38             const data = Object.fromEntries(formData.entries());
39             fetch('/api/events', {
40                 method: 'POST',
41                 headers: { 'Content-Type': 'application/json' },
42                 body: JSON.stringify(data)
43             }).then(() => location.href = '/');
44         });
45     }
46
47     const editEventForm = document.getElementById('editEventForm');
```



```
48   if (editEventForm) {
49       const urlParams = new URLSearchParams(window.location.search);
50       const id = urlParams.get('id');
51       fetch(`/api/events/${id}`)
52           .then(response => response.json())
53           .then(events => {
54               const event = events.find(e => e.id == id);
55               editEventForm.elements['id'].value = event.id;
56               editEventForm.elements['date'].value = event.date;
57               editEventForm.elements['time'].value = event.time;
58               editEventForm.elements['title'].value = event.title;
59               editEventForm.elements['description'].value =
60 event.description;
61               });
62       editEventForm.addEventListener('submit', function (e) {
63           e.preventDefault();
64           const formData = new FormData(this);
65           const data = Object.fromEntries(formData.entries());
66           fetch(`/api/events/${data.id}`, {
67               method: 'PUT',
68               headers: { 'Content-Type': 'application/json' },
69               body: JSON.stringify(data)
70           }).then(() => location.href = '/');
71       });
72   }
73 });
```

## Add Event

[Add Event](#)[Back to Agenda](#)

## Personal Agenda

[Add Event](#)**2024-05-25 20:35 - Web Programming lab paper**

Make the paper for the lab 4 in Web Programming

[Delete](#)[Edit](#)**2024-05-25 20:38 - Sent a Message to my friend**

Sent a important message idk

[Delete](#)[Edit](#)

## Edit Event

05/25/2024



08:35 PM



Web Programming lab paper

Make the paper for the lab 4 in Web Programming


Save Changes

[Back to Agenda](#)

Commits on May 25, 2024


[Se movieron add y edit junto a index para que puedan ser aceptados por la solicitud del script](#)

ff31294   

 RdrigoFH committed 7 hours ago

[Se revisaron los recursos de la api para solucionar el error](#)

349ace2   

 RdrigoFH committed 7 hours ago


[La app no funciona adecuadamente, se realizaron algunos cambios](#)

8709959   

 RdrigoFH committed 7 hours ago


[Se agrego el listening para correr el servidor](#)

10fe4b9   

 RdrigoFH committed 7 hours ago

[Se implemento la logica del delete](#)

12b89f6   

 RdrigoFH committed 7 hours ago

[Se agregaron las solicitudes para agregar nuevos eventos con su titulo hora fecha y descripcion](#)

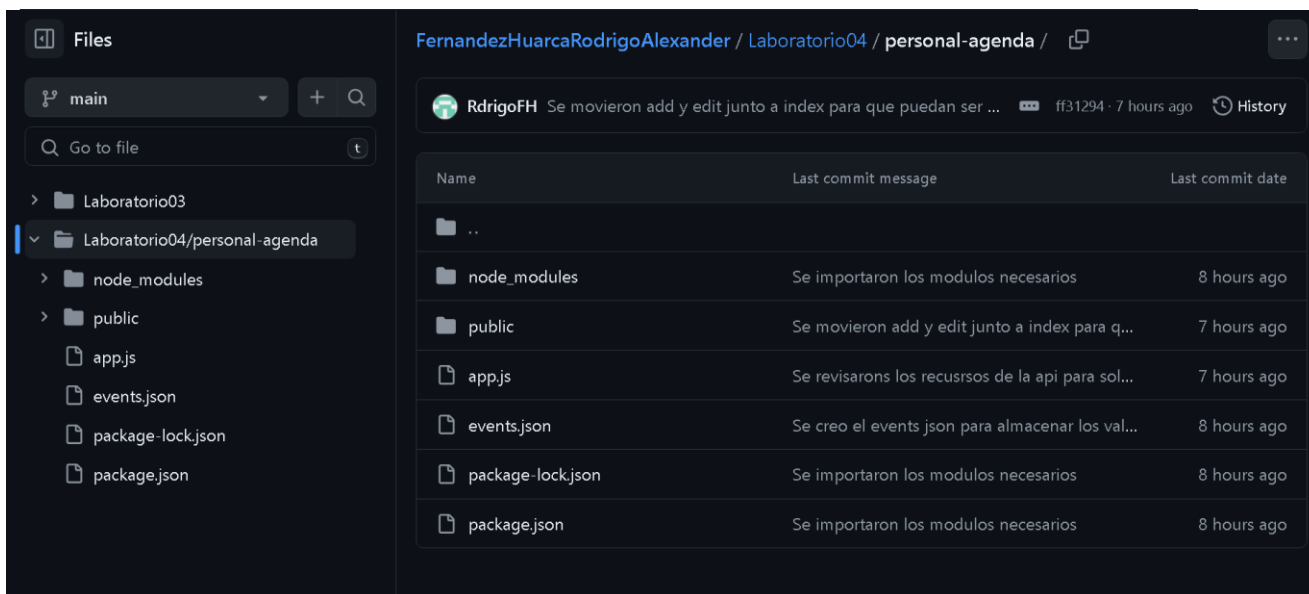
2e8fe9a   

 RdrigoFH committed 7 hours ago

[se creo el post para los nuevos eventos](#)

1405a4d   

 RdrigoFH committed 7 hours ago



Name	Last commit message	Last commit date
..		
node_modules	Se importaron los modulos necesarios	8 hours ago
public	Se movieron add y edit junto a index para que puedan ser ...	7 hours ago
app.js	Se revisaron los recursos de la api para sol...	7 hours ago
events.json	Se creo el events json para almacenar los val...	8 hours ago
package-lock.json	Se importaron los modulos necesarios	8 hours ago
package.json	Se importaron los modulos necesarios	8 hours ago

#### 4. Cuestionario:

- 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.

El objeto **XMLHttpRequest** es la forma tradicional de hacer solicitudes HTTP en JavaScript. Es un objeto nativo del navegador que permite enviar y recibir datos desde y hacia un servidor de manera asíncrona.

**jQuery.ajax** es un método proporcionado por la biblioteca jQuery que simplifica las solicitudes asíncronas. Abstrae muchas de las complejidades de XMLHttpRequest y proporciona una interfaz más amigable y concisa.

Listing 7:Ejemplo usando XMLHttpRequest

```
1 var xhr = new XMLHttpRequest();
2 xhr.open("GET", "https://api.example.com/hello-world", true);
3 xhr.onreadystatechange = function () {
4     if (xhr.readyState === 4 && xhr.status === 200) {
5         console.log(xhr.responseText); // Hola Mundo
6     }
7 };
8 xhr.send();
```

---

**Listing 8:Ejemplo usando jQuery.ajax**

---

```
1 $.ajax({
2   url: "https://api.example.com/hello-world",
3   method: "GET",
4   success: function(response) {
5       console.log(response); // Hola Mundo
6   },
7   error: function(error) {
8       console.error(error);
9   }
10 });
```

---

## **5. Rubrica**

### **5.1. Sobre el informe**

Tabla 1: Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	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 cumplió 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 los ítems.
- 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 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
<b>2. Commits</b>	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
<b>3. Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
<b>4. Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
<b>5. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	1.5	
<b>6. Fechas</b>	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos.	2	X	2	
<b>8. Madurez</b>	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	
<b>Total</b>		20		15.5	

## 6. Referencias

- <https://www.w3schools.com/js/>
- <https://learnjavascript.online/>