Informe de Laboratorio 04

Tema: NodeJS + Express

Nota

Estudiante	Escuela	Asignatura
Kevin Andree Llacma Quispe kllacma@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programacion web 2 Semestre: I Código: 20200585

Laboratorio	Tema	Duración
04	NodeJS + Express	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	-	-

Programación Web
Laboratorio 04
Tema: NodeJS + Express

29 de mayo de 2024

1. Marco teórico

1.1. NodeJS

- Node.js es un entorno de servidor de código abierto
- Node.js es gratis
- Node.js se ejecuta en varias plataformas (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js usa JavaScript en el servidor
- Sitio web : https://nodejs.org/es

1.2. Express

- Infraestructura web rápida, minimalista y flexible para Node.js
- Aplicaciones web: Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.
- API: Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla.
- Rendimiento: Express proporciona una delgada capa de características de aplicación web básicas, que no ocultan las características de Node.js que tanto ama y conoce.

2. Desarollo del lab

2.1. Descripcion

Cree un teclado random para banca por internet.
Para realizar la app agenda seguimos los siguientes pasos
.Inicializamos el proyecto y agregamos dependencias
.npm init -y
.npm install express body-parser ejs

```
PS F:\Pweb2_D\Pweb2_D\Lab_04\agenda-app> npm init -y
Wrote to F:\Pweb2_D\Pweb2_D\Lab_04\agenda-app\package.json:

{
  "name": "agenda-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



    Directory: F:\Pweb2_D\Pweb2_D\Lab_04\agenda-app


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/28/2024   4:53 PM            224 package.json
```

```
PS F:\Pweb2_D\Pweb2_D\Lab_04\agenda-app> npm install express body-parser ejs

added 80 packages, and audited 81 packages in 7s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS F:\Pweb2_D\Pweb2_D\Lab_04\agenda-app>
```

Importamos Express, body-parser para manejar los datos del formulario, y los módulos fs y path para trabajar con el sistema de archivos y las rutas.

```
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('fs');
const path = require('path');
```

Configuramos Express, habilitamos body-parser para analizar datos del formulario y configuramos EJS para plantillas.

```
const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.set('view engine', 'ejs');
```

Definimos el directorio donde se almacenarán los eventos y nos aseguramos de que existe.

```
const AGENDA_DIR = path.join(__dirname, 'agenda');


if (!fs.existsSync(AGENDA_DIR)) {
    fs.mkdirSync(AGENDA_DIR);
}
```

getAgendaStructure():
Esta función lee el directorio de la agenda y devuelve un objeto estructurado con las fechas y eventos.

```
function getAgendaStructure() {
    const agenda = {};
    fs.readdirSync(AGENDA_DIR).forEach(dateDir => {
        const datePath = path.join(AGENDA_DIR, dateDir);
        if (fs.statSync(datePath).isDirectory()) {
            agenda[dateDir] = [];
            fs.readdirSync(datePath).forEach(eventFile => {
                const eventPath = path.join(datePath, eventFile);
                const eventContent = fs.readFileSync(eventPath, 'utf-8');
                const eventTitle = eventContent.split('\n')[0];
                agenda[dateDir].push({ time: eventFile.replace('.txt', ''), title: eventTitle });
            });
        }
    });
    return agenda;
}
```

Utilizamos fs.readdirSync para leer el contenido del directorio AGENDADIR. Este método devuelve una lista de nombres de archivos y directorios dentro de AGENDADIR. Iteramos sobre cada elemento utilizando forEach.

```
fs.readdirSync(AGENDA_DIR).forEach(dateDir => {
```

Para cada elemento en AGENDADIR, construimos la ruta completa utilizando path.join. Esto nos da la ruta completa del subdirectorio que corresponde a una fecha específica.

```
const datePath = path.join(AGENDA_DIR, dateDir);
```

Utilizamos fs.statSync para obtener información sobre datePath. Si datePath es un directorio (isDirectory()), continuamos con el procesamiento.

```
if (fs.statSync(datePath).isDirectory()) {
```

Luego

- Se verifica si datePath es un directorio.
- Se inicializa un arreglo vacío para almacenar los eventos de la fecha.
- Se leen todos los archivos dentro del directorio de la fecha.
- Para cada archivo de evento, se construye su ruta completa.
- Se lee el contenido del archivo de evento.
- Se extrae el título del evento (la primera línea del contenido del archivo).
- Se agrega un objeto que contiene la hora y el título del evento al arreglo correspondiente a la fecha.

```
if (fs.statSync(datePath).isDirectory()) {
    agenda[dateDir] = [];
    fs.readdirSync(datePath).forEach(eventFile => {
        const eventPath = path.join(datePath, eventFile);
        const eventContent = fs.readFileSync(eventPath, 'utf-8');
        const eventTitle = eventContent.split('\n')[0];
        agenda[dateDir].push({ time: eventFile.replace('.txt', ''), title: eventTitle });
    });
}
```

Obtiene la estructura de la agenda y la pasa a la vista index.ejs.

```
app.get('/', (req, res) => {
    const agenda = getAgendaStructure();
    res.render('index', { agenda });
});
```

Recibe datos del formulario, crea un nuevo evento y redirige a la página principal si el evento no existe.

```
app.post('/create', (req, res) => {
    const { date, time, title, description } = req.body;
    const dateDir = path.join(AGENDA_DIR, date);
    const eventFile = path.join(dateDir, `${time}.txt`);

    if (!fs.existsSync(dateDir)) {
        fs.mkdirSync(dateDir);
    }

    if (!fs.existsSync(eventFile)) {
        fs.writeFileSync(eventFile, `${title}\n${description}`);
        res.redirect('/');
    } else {
        res.send('Evento ya existe');
    }
});
```

Obtiene los detalles del evento para mostrarlos en un formulario de edición y luego actualiza el archivo con los cambios.

```
app.get('/edit/:date/:time', (req, res) => {
    const { date, time } = req.params;
    const eventFile = path.join(AGENDA_DIR, date, `${time}.txt`);
    if (fs.existsSync(eventFile)) {
        const eventContent = fs.readFileSync(eventFile, 'utf-8').split('\n');
        const title = eventContent[0];
        const description = eventContent.slice(1).join('\n');
        res.render('edit', { date, time, title, description });
    } else {
        res.send('Evento no encontrado');
    }
});

app.post('/edit/:date/:time', (req, res) => {
    const { date, time } = req.params;
    const { title, description } = req.body;
    const eventFile = path.join(AGENDA_DIR, date, `${time}.txt`);

    if (fs.existsSync(eventFile)) {
        fs.writeFileSync(eventFile, `${title}\n${description}`);
        res.redirect('/');
    } else {
        res.send('Evento no encontrado');
    }
});
```

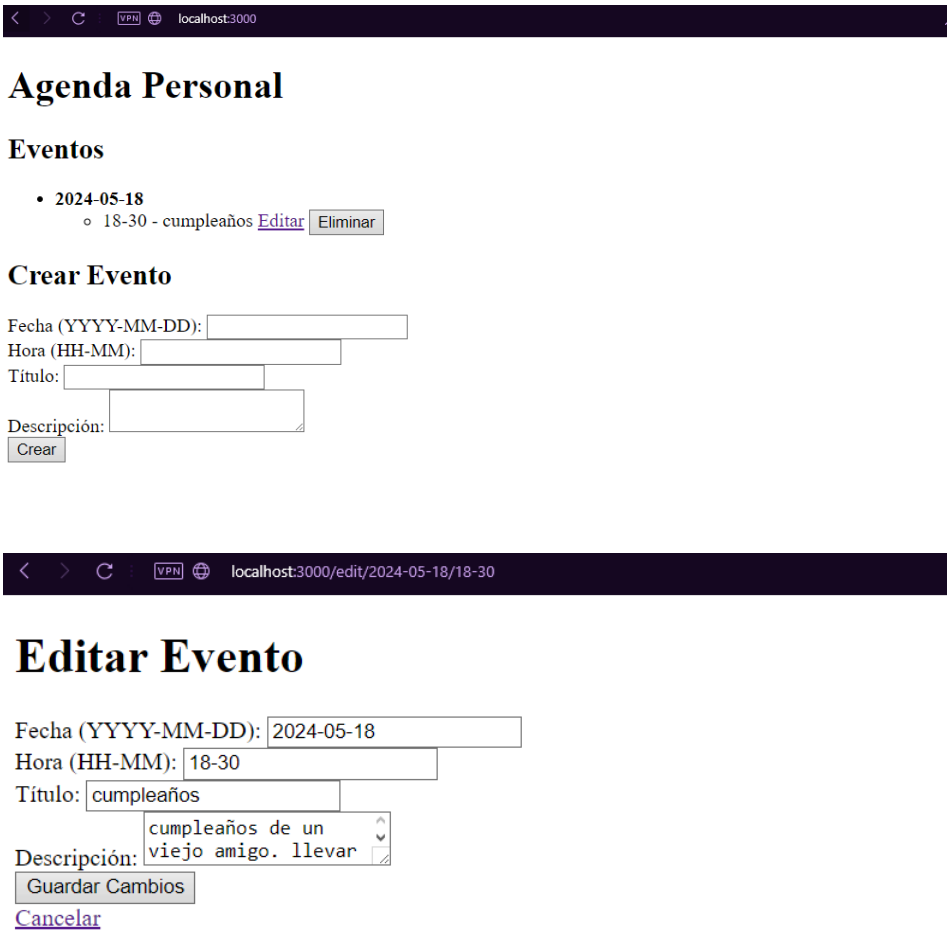
Elimina el archivo del evento y redirige a la página principal.

```
app.post('/delete/:date/:time', (req, res) => {
    const { date, time } = req.params;
    const eventFile = path.join(AGENDA_DIR, date, `${time}.txt`);
    if (fs.existsSync(eventFile)) {
        fs.unlinkSync(eventFile);
        res.redirect('/');
    } else {
        res.send('Evento no encontrado');
    }
});

```

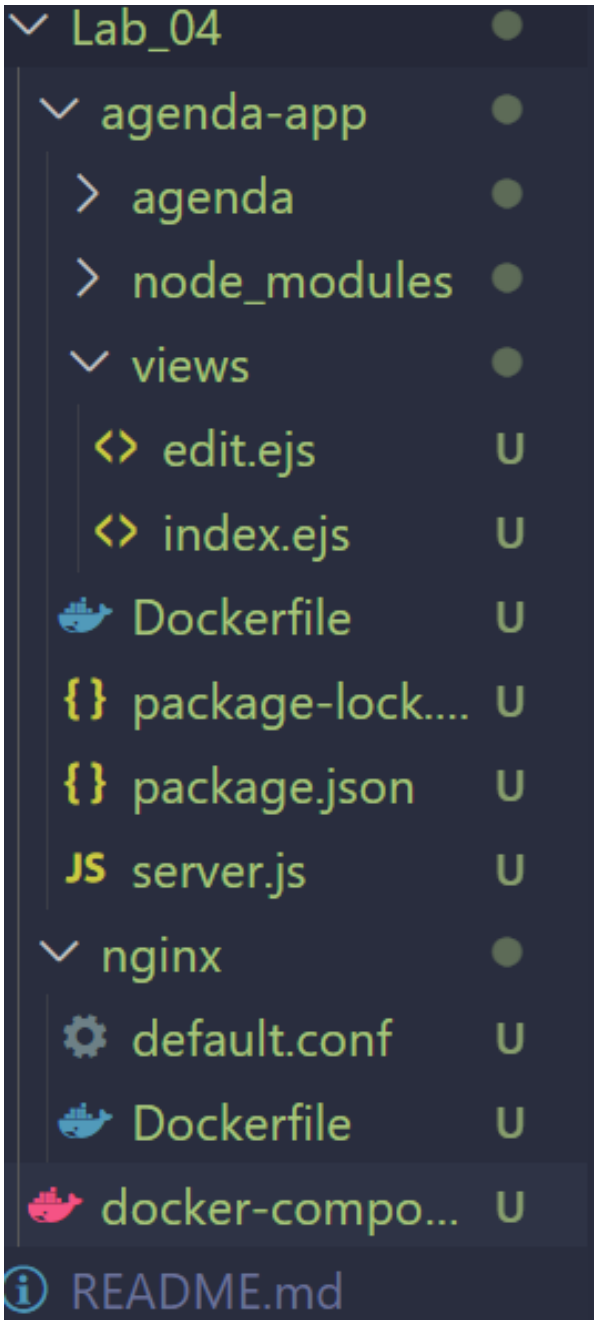
Inicia el servidor en el puerto 3000 y muestra un mensaje en la consola.

```
app.listen(3000, () => {
    console.log('Servidor escuchando en http://localhost:3000');
});

```

Prueba de la agenda:



Se uso docker file para correr la aplicacion:



- Define la imagen base node:14.
- Establece el directorio de trabajo en /usr/src/app.
- Copia los archivos package.json y package-lock.json y ejecuta npm install para instalar las dependencias.
- Copia el resto del código de la aplicación.

- Expone el puerto 3000.
- Define el comando para iniciar la aplicación.

```
FROM node:14

WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]

```

- Define que Nginx escuche en el puerto 80.
- Configura la ubicación raíz / para pasar las solicitudes a la aplicación NodeJS en http://node:3000.
- Configura la ubicación /static para servir archivos estáticos directamente desde el directorio public de la aplicación.

```
version: '3'
services:
  node:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - .:/usr/src/app
    networks:
      - app-network

  nginx:
    build: ./nginx
    ports:
      - "80:80"
    depends_on:
      - node
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```