

DESARROLLO DE APLICACIONES WEB AVANZADO

LABORATORIO N° 02

CREAR UN SERVIDOR WEB




Alumnos		Nota
Sosa Hualpa, Anyelina Lucía		
Grupo	B	
Fecha de Entrega	22/03/2024	
Docente	Renato Usnayo Cáceres	

OBJETIVOS:

- Identificar las principales características del módulo HTTP de node.js
- Manipulación de variables y funciones comunes del core de node.js
- Exportar sus propios módulos.

SEGURIDAD:

	Advertencia: En este laboratorio está prohibida la manipulación del hardware, conexiones eléctricas o de red; así como la ingestión de alimentos o bebidas.
---	---

FUNDAMENTO TEÓRICO:

- Revisar el texto guía que está en el campus Virtual.

NORMAS EMPLEADAS:

- No aplica

RECURSOS:

- En este laboratorio cada alumno trabajará con un equipo con Windows 10.

METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:

- El desarrollo del laboratorio es individual

PROCEDIMIENTO:

1. Configuración del proyecto

Crea una carpeta para tu proyecto y navega a ella en tu terminal.

Inicializa un proyecto de Node.js ejecutando el siguiente comando y siguiendo las instrucciones:

<code>npm init</code>

```
D:\5toSemestre\daw\lab2>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (lab2)
version: (1.0.0)
description: daw - lab02
entry point: (index.js)
test command:
git repository:
keywords:
author: Anyelina Sosa
license: (ISC)
About to write to D:\5toSemestre\daw\lab2\package.json:
{
  "name": "lab2",
  "version": "1.0.0",
  "description": "daw - lab02",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Anyelina Sosa",
  "license": "ISC"
}

Is this OK? (yes) yes
```

Ingresamos datos como la descripción y el nombre del autor, que sería nuestro nombre.

2. Instalación de dependencias

Vamos a necesitar el módulo http de Node.js para crear el servidor y el módulo fs para trabajar con el sistema de archivos. Además, utilizaremos el módulo path para manejar rutas de archivos de manera segura. Instala estas dependencias ejecutando:

```
npm install http fs path
```

```
D:\5toSemestre\daw\lab2>npm install http fs path
added 6 packages, and audited 7 packages in 6s
found 0 vulnerabilities
```

Vemos que se han instalado las dependencias sin ningún problema.

3. Creación del servidor

Crea un archivo llamado `server.js` en la carpeta del proyecto.

```
const http = require('http');
const fs = require('fs');
const path = require('path');

const server = http.createServer((req, res) => {
  // Evitar solicitud de favicon.ico
  if (req.url === '/favicon.ico') {
    res.writeHead(204); // Respuesta exitosa sin contenido
    res.end();
    return;
  }

  // Obtén la ruta absoluta del archivo solicitado
  const filePath = path.join(__dirname, 'public', req.url);

  console.log('Ruta del archivo solicitado:', filePath); // Mensaje de depuración

  // Verifica si filePath es un archivo o un directorio
  if (fs.statSync(filePath).isDirectory()) {
    console.error('Se intentó leer un directorio:', filePath);
    res.writeHead(500, { 'Content-Type': 'text/html' });
    res.end('Error del servidor');
    return;
  }

  // Lee el archivo solicitado y responde con su contenido
  fs.readFile(filePath, (err, content) => {
    if (err) {
      console.error('Error al leer el archivo:', err);
      if (err.code === 'ENOENT') {
        console.error('Archivo no encontrado:', filePath);
        res.writeHead(404, { 'Content-Type': 'text/html' });
        res.end('Archivo no encontrado');
      } else {
        res.writeHead(500, { 'Content-Type': 'text/html' });
        res.end('Error del servidor');
      }
    } else {
      const ext = path.extname(filePath);
      let contentType = 'text/html';
      switch (ext) {
        case '.js':
          contentType = 'text/javascript';
          break;
        case '.css':
```

```
        contentType = 'text/css';
        break;
      case '.json':
        contentType = 'application/json';
        break;
      case '.png':
        contentType = 'image/png';
        break;
      case '.jpg':
        contentType = 'image/jpeg';
        break;
    }

    console.log('Sirviendo archivo:', filePath);

    // Archivo encontrado, envía el contenido como respuesta
    res.writeHead(200, { 'Content-Type': contentType });
    res.end(content);
  }
});
});

const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Servidor en funcionamiento en http://localhost:${PORT}`);
});
```

Explique el funcionamiento del código

Primero se importan los módulos 'http', 'fs' (sistema de archivos), y 'path' (manipulación de rutas de archivos Node.js). Se crea el servidor HTTP utilizando la función 'http.createServer()'. Dentro de esta se proporciona una función de devolución de llamada que manejará las solicitudes entrantes y generará las respuestas correspondientes, evitamos la solicitud del ícono de favoritos, pues esto es innecesario.

Luego, obtenemos la ruta absoluta del archivo solicitado combinando el directorio actual ('__dirname') con la carpeta "public" y la ruta solicitada en la URL y se muestra por medio de consola.

Se verifica que la ruta solicitada es un directorio o un archivo usando 'fs.statSync()'. Si es un directorio, devuelve un error 500 y finaliza la respuesta.

En caso la ruta solicitada sea un archivo, se lee su contenido usando 'fs.readFile()'. Y si hay un error al leer el archivo se responde con "Error al leer el archivo" por medio de consola. Si el error es 'EOENT' se responderá con un estado 404, de lo contrario con un estado de 500.

En caso no existieran errores, se determina el tipo de contenido según su extensión y se establece el encabezado 'Content-Type' correspondiente.

Se muestra por consola el archivo que se está sirviendo. Luego, se responde con un estado de 200 y el contenido del archivo solicitado.

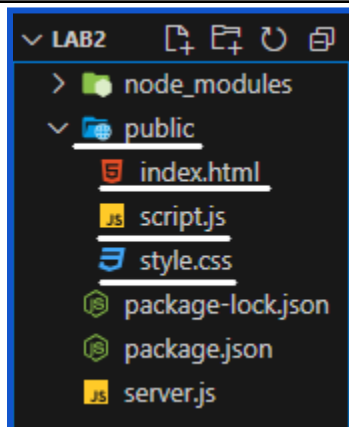
El servidor se inicia escuchando el puerto 3000, y una vez el servidor está funcionando, se muestra un mensaje en la consola indicando la URL donde se puede acceder al servidor.

4. Creación de la carpeta de archivos estáticos

Crea una carpeta llamada public en la raíz del proyecto. Coloca los archivos HTML, CSS, JavaScript y otros archivos estáticos que desees servir en esta carpeta.

Asegúrate de tener una estructura de carpetas similar a esta:

- (Nombre de tu proyecto)
- public
 - index.html
 - style.css
 - script.js
 - (otros archivos)
- server.js



Se crea la carpeta y archivos tal como pide la guía.

Archivo index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Mi Página</title>
</head>
```

```
<body>
  <h1>Hola, esta es mi página</h1>
  <p>Bienvenido al mundo de Node.js y archivos estáticos.</p>
  <button id="boton-saludo">Saludar</button>
  <p id="saludo"></p>
  <script src="script.js"></script>
</body>
</html>
```

Explique el funcionamiento del código

Es una hoja HTML, en el head se enlaza una hoja de estilos "style.css", también se le asigna un título a la página.

En el body se asigna un encabezado principal, que sería el título de la página, un párrafo que da la bienvenida al usuario y nos proporciona una breve descripción de la página. También un botón de id "boton-saludo", debajo de este se crea un párrafo vacío de nombre "saludo" y por último se enlaza un script de nombre "script.js"

Archivo style.css

```
/* Estilos para el contenido */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f0f0f0;
}

h1 {
  color: #333;
}

p {
  color: #666;
}
```

Archivo script.js

```
// Función para saludar al hacer clic en un botón
function saludar() {
  const mensaje = '¡Hola desde el script!';

  // Mostrar el mensaje en un elemento HTML con el id 'saludo'
  const saludoElement = document.getElementById('saludo');
  saludoElement.textContent = mensaje;
}
```

```
}  
  
// Asignar la función 'saludar' al evento 'click' del botón  
const botonSaludo = document.getElementById('boton-saludo');  
botonSaludo.addEventListener('click', saludar);
```

Explique que hace el código

Se crea la función “saludar”, en esta está la constante ‘mensaje’ que almacena un saludo, también otra de nombre ‘saludoElement’ que obtiene el elemento HTML con el id “saludo” usando ‘document.getElementById(‘saludo’)', que a su vez actualiza el contenido de texto del elemento ‘saludoElement’ con el mensaje de saludo usando ‘textContent’.

La constante ‘botonSaludo’ selecciona el elemento del botón de saludo en el HTML mediante su id y lo asigna a la variable ‘botonSaludo’.

Y en la última línea, se agrega un “event listener” al botón de saludo que escucha el evento de clic ‘click’ y llama a la función “saludar” cuando se hace clic al botón.

5. Ejecución del servidor

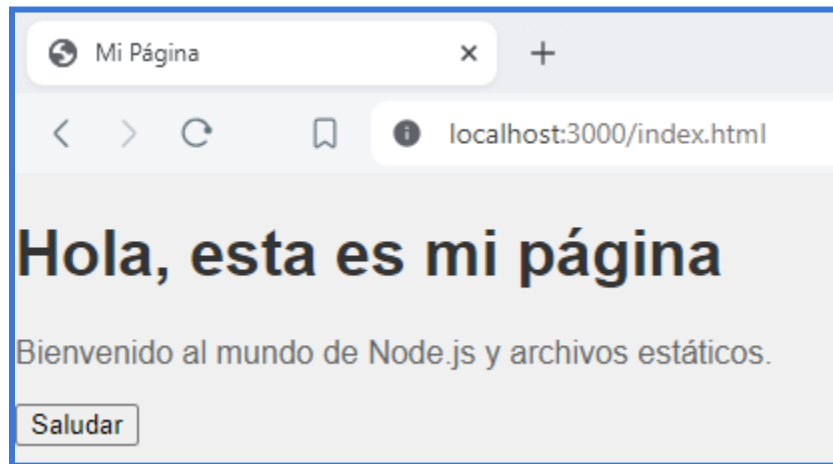
En la terminal, ejecuta el siguiente comando para iniciar el servidor:

```
node server.js
```

Tu servidor de archivos web ahora está en funcionamiento. Abre tu navegador y accede a <http://localhost:3000/index.html> para ver los archivos servidos estáticamente desde la carpeta public.

```
D:\5toSemestre\daw\lab2>node server.js  
Servidor en funcionamiento en http://localhost:3000  
Ruta del archivo solicitado: D:\5toSemestre\daw\lab2\public\index.html  
Ruta del archivo solicitado: D:\5toSemestre\daw\lab2\public\  
Se intentó leer un directorio: D:\5toSemestre\daw\lab2\public\  
Ruta del archivo solicitado: D:\5toSemestre\daw\lab2\public\index.html  
Sirviendo archivo: D:\5toSemestre\daw\lab2\public\index.html  
Sirviendo archivo: D:\5toSemestre\daw\lab2\public\index.html  
Ruta del archivo solicitado: D:\5toSemestre\daw\lab2\public\style.css  
Ruta del archivo solicitado: D:\5toSemestre\daw\lab2\public\script.js  
Sirviendo archivo: D:\5toSemestre\daw\lab2\public\style.css  
Sirviendo archivo: D:\5toSemestre\daw\lab2\public\script.js
```

Vemos que han aparecido las impresiones por medio de la consola.



Vemos que ha iniciado el servidor correctamente, además de que al abrir el navegador se pueden ver los archivos servidos estáticamente desde la carpeta “public”.

TAREA

¿Cuál es el propósito de un servidor de archivos web y por qué es útil en el desarrollo de aplicaciones web?

Un servidor de archivos web es un tipo de servidor que se utiliza para almacenar y distribuir archivos **a través de protocolos web, como HTTP o HTTPS**. Su propósito principal es **proporcionar acceso a archivos estáticos**, como archivos HTML, CSS, JavaScript, imágenes y otros recursos multimedia a los clientes que los solicitan **a través del navegador web**.

Es bastante útil porque:

- Puede almacenar y entregar archivos estáticos como HTML, CSS, JavaScript, imágenes y otros recursos multimedia a los usuarios finales. Esto permite que los navegadores web carguen y rendericen las páginas web correctamente.
- Muchas aplicaciones web modernas requieren una combinación de archivos estáticos y contenido generado dinámicamente, y este tipo de servidor puede integrarse con tecnologías de servidor como PHP, Python, Ruby on Rails, Node.js, etc., para procesar solicitudes dinámicas y generar respuestas personalizadas según las necesidades de la aplicación.
- Están diseñados para manejar grandes cantidades de tráfico web y distribuir archivos de manera eficiente a través de la red, lo que es fundamental para garantizar un rendimiento óptimo y una experiencia de usuario fluida, especialmente en sitios web con alto volumen de visitantes.
- Los servidores de archivos web como Apache HTTP Server, Nginx, Microsoft IIS, entre otros, suelen ser fáciles de configurar y mantener, pues ofrecen una amplia gama de opciones de configuración para adaptarse a las necesidades específicas de la aplicación web y pueden escalarse según sea necesario para satisfacer la demanda del tráfico.
- Suelen proporcionar mecanismos de seguridad integrados, como la capacidad de habilitar el cifrado SSL/TLS para garantizar la seguridad de las comunicaciones entre el servidor y los clientes. También pueden implementar reglas de acceso y autenticación para proteger los recursos sensibles y prevenir ataques maliciosos.

¿Cuál es la diferencia entre un servidor web y un servidor de archivos web?

Un servidor web y un servidor de archivos web son dos tipos diferentes de servidores que cumplen funciones distintas en el contexto de la infraestructura de Internet y las aplicaciones web.

Servidor web:

Ejemplos de servidores web populares: Apache HTTP Server, Nginx, Microsoft IIS y LiteSpeed.

- Es un software diseñado para recibir solicitudes HTTP (y a menudo HTTPS) de clientes, como navegadores web, y responder a esas solicitudes sirviendo contenido web.
- Su función principal: Manejar solicitudes HTTP, procesarlas y devolver las respuestas correspondientes. Esto puede incluir la entrega de archivos HTML, CSS, JavaScript, imágenes, archivos multimedia y otros tipos de contenido web.
- Son responsables de interpretar y ejecutar el código de servidor, como PHP, Python, Ruby, etc., si está presente en las páginas web solicitadas. Esto significa que pueden ejecutar scripts y aplicaciones dinámicas para generar contenido en tiempo real.

Servidor de archivos web:

Ejemplos de servidores de archivos web : Apache HTTP Server en su configuración básica, servidores de almacenamiento en la nube como Amazon S3 o servicios especializados en la entrega de contenido estático como Cloudflare.

- Es un tipo específico de servidor web que se utiliza principalmente para almacenar y distribuir archivos estáticos a través de protocolos web, como HTTP o HTTPS.
- Su función principal: Proporcionar acceso a archivos estáticos, como archivos HTML, CSS, JavaScript, imágenes y otros recursos multimedia, a los clientes que los solicitan a través del navegador web.
- Aunque algunos servidores de archivos web pueden admitir cierto grado de contenido dinámico, su funcionalidad en este sentido es limitada en comparación con los servidores web convencionales.
- Los servidores de archivos web son adecuados para sitios web y aplicaciones que principalmente sirven contenido estático y que no requieren un procesamiento dinámico intensivo.

(Escoja una opción)

Opción 1: Al finalizar sus estudios, usted decide independizarse y crear su empresa. El primer paso para entrar al mundo del comercio global es tener un sitio web, sin él no existimos hoy en día. Cree una página web (El rubro de la empresa es de su elección), con sus páginas básicas: Inicio, Nosotros (nuestro equipo, misión, visión), Nuestros Servicios (debe ser una galería de imágenes con su descripción), Catálogo de Clientes y Contáctenos (El formulario de contacto nos redirigirá a una página de confirmación de atención a nuestra solicitud ("muy pronto nos pondremos en contacto")). Toda la web debe utilizar node.js

Creación del proyecto en la carpeta "l2-tarea":

> npm init

```
D:\5toSemestre\daw>cd l2-tarea

D:\5toSemestre\daw\L2-tarea>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (l2-tarea)
version: (1.0.0)
description: tarea lab 2
entry point: (index.js)
test command:
git repository:
keywords:
author: Anyelina Sosa
license: (ISC)
About to write to D:\5toSemestre\daw\L2-tarea\package.json:

{
  "name": "l2-tarea",
  "version": "1.0.0",
  "description": "tarea lab 2",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Anyelina Sosa",
  "license": "ISC"
}

Is this OK? (yes) y
```

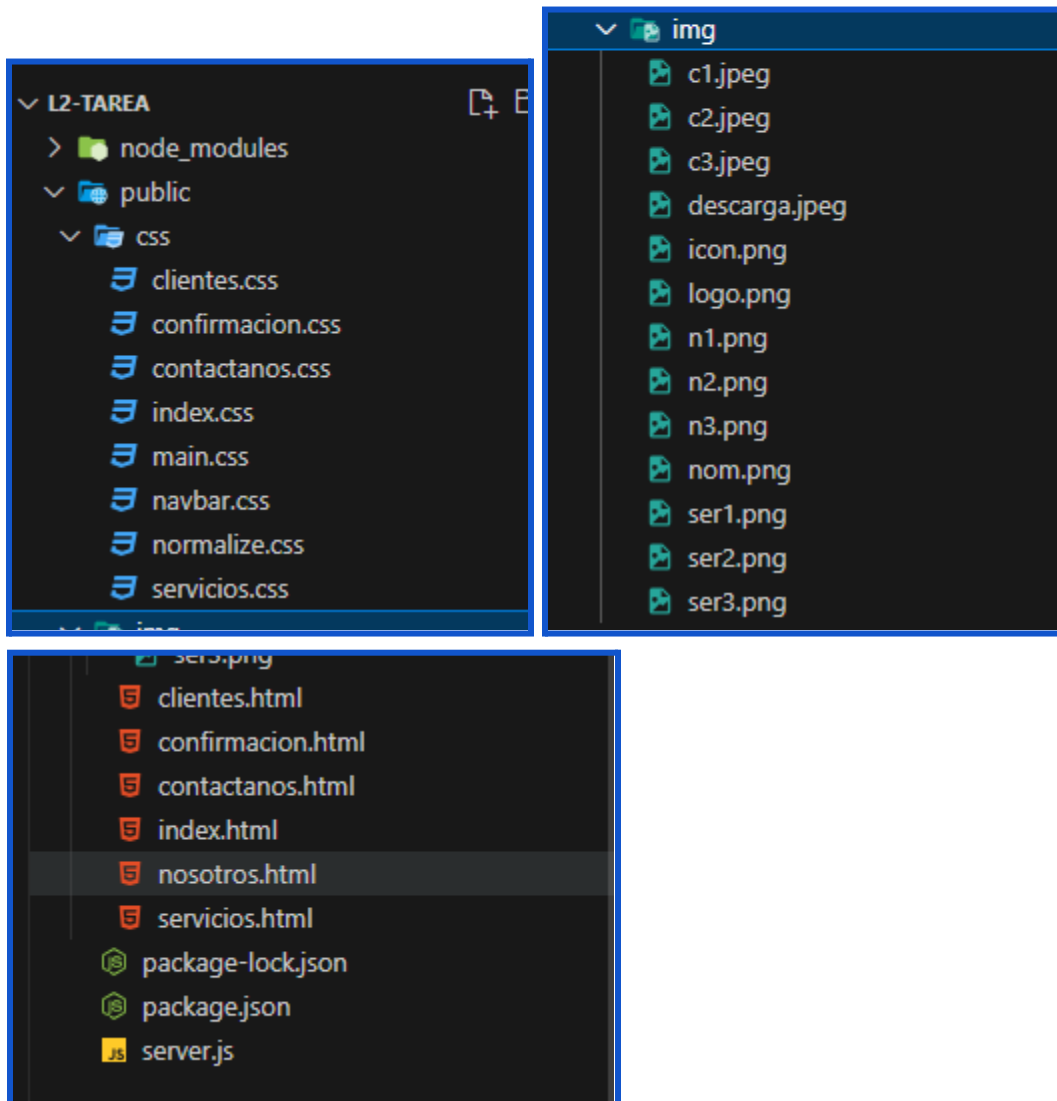
Instalación de dependencias

> npm install http fs path

```
D:\5toSemestre\daw\L2-tarea>npm install http fs path

added 6 packages, and audited 7 packages in 3s

found 0 vulnerabilities
```



server.js:

Se ha usado Bootstrap y Bootstrap Icons

```
package.json > ...
1  {
2    "name": "12-tarea",
3    "version": "1.0.0",
4    "description": "tarea lab 2",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Anyelina Sosa",
10   "license": "ISC",
11   "dependencies": {
12     "bootstrap": "^5.3.3",
13     "bootstrap-icons": "^1.11.3",
14     "fs": "^0.0.1-security",
15     "http": "^0.0.1-security",
16     "path": "^0.12.7"
17   }
18 }
```

Código:

link: <https://github.com/anyelinash/dawlab2>

Ejecución:

index.html:



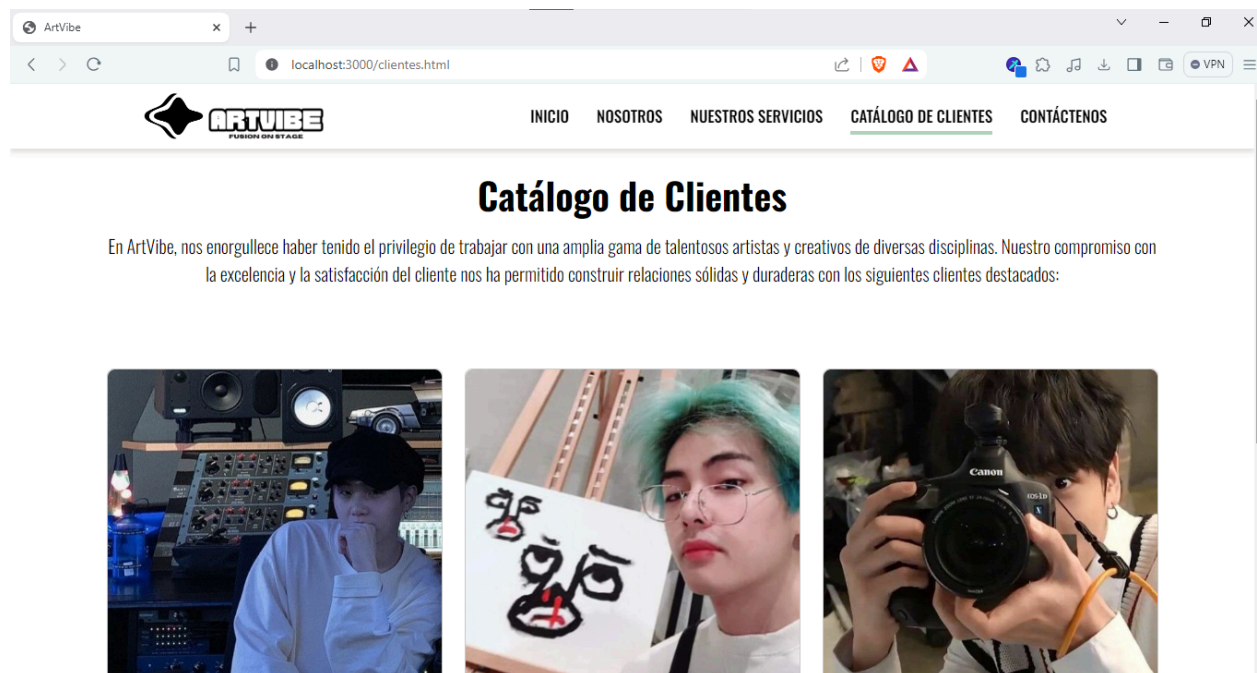
[nosotros.html:](#)



[servicios.html:](#)



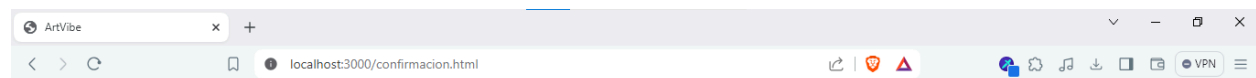
[catalogo.html:](#)



[contactanos.html:](#)



confirmacion.html:



Consola:


```
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\img\c3.jpeg
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\main.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\clientes.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\navbar.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\c1.jpeg
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\c2.jpeg
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\c3.jpeg
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\img\icon.png
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\icon.png
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\img\nom.png
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\nom.png
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\contactanos.html
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\contactanos.html
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\css\main.css
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\css\navbar.css
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\css\contactanos.css
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\img\icon.png
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\img\nom.png
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\main.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\navbar.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\contactanos.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\nom.png
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\img\icon.png
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\confirmacion.html
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\confirmacion.html
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\css\confirmacion.css
Ruta del archivo solicitado: D:\5toSemestre\daw\L2-tarea\public\css\main.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\confirmacion.css
Sirviendo archivo: D:\5toSemestre\daw\L2-tarea\public\css\main.css
```

Opción 2: Desarrolle una página web sobre tu portafolio personal con sus páginas básicas: Inicio, sobre mí, experiencia laboral, educación, habilidades, proyectos y contáctame (El formulario de contacto nos redirigirá a una página de confirmación de atención a nuestra solicitud (“muy pronto nos pondremos en contacto”)). Toda la web debe utilizar node.js

- Utiliza una biblioteca de estilos para mejorar el diseño

Colocar capturas del código y capturas de ejecución

Grabar explicando el código y ejecución de este en máximo 4 minutos

link: https://drive.google.com/drive/u/0/folders/1T1qW8uAIGVTd9VcyxAkBRZGRec_Vr82d

OBSERVACIONES: *(Las observaciones son las notas aclaratorias, objeciones y problemas que se pudo presentar en el desarrollo del laboratorio)*

- Con el comando “npm install http fs path” se instalan los paquetes fs y path en el proyecto.
- Para usar una biblioteca de estilos como Bootstrap tenemos que instalar la dependencia y a la vez poner un “link” en la hoja HTML.
- Se observó que “fs” en Node.js, abreviatura de "File System" (Sistema de Archivos), que es un módulo integrado que proporciona una interfaz para interactuar con el sistema de archivos del sistema operativo en el que se ejecuta Node.js.
- Vimos que por consola se sirvieron los archivos html.
- Con “addEventListener” se pudieron adjuntar un evento a un elemento del DOM (Documento Object Model) y ejecutar una función "listener".

CONCLUSIONES: *(Las conclusiones son una opinión sobre tu trabajo, explicar cómo resolviste las dudas o problemas presentados en el laboratorio. Además de aportar una opinión crítica de lo realizado)*

- El comando “npm install http fs path” es utilizado para instalar paquetes en un proyecto Node.js. En este laboratorio se vio que se instalan los paquetes fs y path. fs proporciona funcionalidades para interactuar con el sistema de archivos, mientras que path proporciona utilidades para trabajar con rutas de archivos y directorios. Esta instalación permite acceder a estas funcionalidades en el proyecto.
- Para utilizar una biblioteca de estilos como Bootstrap en un proyecto web, es necesario instalar la dependencia de Bootstrap a través de herramientas como npm o incluirlo directamente en el proyecto. Además, se debe enlazar el archivo CSS de Bootstrap en el archivo HTML correspondiente utilizando la etiqueta <link>.
- El módulo fs en Node.js proporciona una interfaz para interactuar con el sistema de archivos del sistema operativo en el que se ejecuta Node.js. Permite realizar una variedad de operaciones relacionadas con archivos y directorios, como leer y escribir archivos, crear y eliminar directorios, entre otras. Esta capacidad es fundamental para muchas aplicaciones Node.js que necesitan interactuar con archivos.
- Observar que los archivos HTML se sirvieron desde la consola indica que se utilizó Node.js para crear un servidor web capaz de entregar archivos estáticos, como archivos HTML. Esto sugiere el uso de Node.js en el lado del servidor para manejar solicitudes HTTP y servir contenido web.
- addEventListener es un método en JavaScript que permite adjuntar eventos a elementos del DOM y ejecutar una función cuando ocurre ese evento. Esto proporciona una forma flexible de interactuar con la interacción del usuario en una página web, permitiendo la ejecución de código en respuesta a acciones como clics de mouse, pulsaciones de teclas, entre otros.