Guía de ejercicios - Introducción a Node (I)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Glosario - Introducción a Node

- **API REST:** Interfaz de programación de aplicaciones, que se apoya en la arquitectura REST para el desarrollo de aplicaciones en red.

- **Backend**: Cuando nos referimos al Backend, hablamos de la lógica de negocios que no vemos, la que se conecta a la fuente de datos por ejemplo, el conjunto de acciones que sucede entre el servidor y la aplicación.

- **Bucle:** En programación, es una secuencia que se ejecuta repetidas veces.

- **createServer:** En el módulo http de Node, es el método que crea un servidor y que recibe como parámetro una función callback con el objeto request y response

- **CRUD:** Por sus siglas en inglés Create Read Update Delete (Crear, leer, actualizar, eliminar), es la representación de un sistema administrativo que maneja datos, archivos o cualquier entidad que pueda ser aplicada a sus siglas.

- **Frontend:** Es la parte que ven los usuarios, por eso cuando nos referimos a esto decimos que está del lado del cliente. Es toda la interacción que sucede entre el usuario y la aplicación.

- **FTP:** Por sus siglas en inglés File Transfer Protocol (Protocolo de transferencia de archivos) es el protocolo que se encarga de transferir archivos. Permite a los usuarios cargar o descargar archivos en su hosting o servidor contratado, protegidos por un sistema de credenciales únicas referentes al servicio, usuario y clave de acceso.

- **Listen:** En el módulo http de Node, es el método que define el puerto por el cual se dispondrá el servidor. Normalmente se escribe concatenando el método createServer

- **Middleware:** Se puede definir como el proceso o función que se ejecuta en la capa de comunicación en un sistema desarrollado bajo la arquitectura cliente-servidor.

- **Node:** Entorno de ejecución multiplataforma y de código abierto, utilizado para desarrollar aplicaciones escritas en código JavaScript.

- **Performance:** Desempeño con respecto al rendimiento de un computador, un sistema operativo, un programa o una conexión a red.

- **Payload:** Se entiende por payload como el “paquete” o conjunto de datos transmitidos dentro de una comunicación, aplicable para funciones (argumentos), para los verbos HTTP que conllevan a un envío formateado de información (JSON o XML), entre otros.

- **Servidor Web:** Un servidor Web o Servidor HTTP, es un programa que procesa una aplicación en un servidor o máquina, genera conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo respuestas.

Reflexiona:

- ¿Node debería ser una alternativa a considerar si necesitara crear una aplicación que se ejecute en diferentes plataformas?
- ¿Node debería ser una alternativa a considerar si necesitara disponibilizar los datos de una base de datos como una API REST?
- ¿Ocuparía Node para hacer cambios estéticos en mis sitios web?
- ¿Podremos con Express hacer todo lo que hemos hecho con Node js en los módulos anteriores?
- Con lo poco que has visto hasta ahora, ¿Te parece que el desarrollo de un servidor con Express tiene un código más limpio y cómodo de escribir que con Node puro?

Actividad guiada: Ruta genérica

A continuación, se presenta un ejercicio para crear una ruta genérica en un servidor con Express. En este caso, se creará una ruta genérica que devuelva un mensaje personalizado para cualquier ruta indefinida.

- **Paso 1**: Crea un nuevo archivo llamado app.js y configura un servidor básico con Express:

```
// app.js
const express = require("express");
const app = express();
const port = 3000;

app.listen(port, () => {
```

```
  console.log(`El servidor está inicializado en el puerto ${port}`);
});
```

- **Paso 2:** Agrega una ruta GET para la página de inicio, que devuelva un mensaje simple:

```
// app.js

// ... (código del Paso 1)

app.get("/", (req, res) => {
  res.send("¡Bienvenido a mi sitio web!");
});
```

- **Paso 3:** Agrega una ruta GET para una página "Acerca de", que devuelva información sobre ti:

```
// app.js

// ... (código de los Pasos 1 y 2)

app.get("/about", (req, res) => {
  res.send("Soy un entusiasta de la programación y me encanta aprender nuevas tecnologías.");
});
```

- **Paso 4:** Agrega una ruta genérica para manejar cualquier otra ruta no definida:

```
// app.js

// ... (código de los Pasos 1, 2 y 3)

app.get("*", (req, res) => {
  res.send("<h1>404 - Página no encontrada</h1><p>Lo siento, la página que buscas no existe.</p>");
});
```

- **Paso 5:** Ejecuta el servidor y prueba las rutas:

Para ejecutar el servidor, abre una terminal, ve al directorio donde tienes el archivo app.js y ejecuta el siguiente comando:

```
node app.js
```

Luego, abre tu navegador y prueba las siguientes rutas:

1. Página de inicio: http://localhost:3000/. Deberías ver el mensaje "¡Bienvenido a mi sitio web!".

2. Página "Acerca de": http://localhost:3000/about. Deberías ver el mensaje "Soy un entusiasta de la programación y me encanta aprender nuevas tecnologías."

3. Cualquier otra página, por ejemplo: http://localhost:3000/contacto. Deberías ver el mensaje "404 - Página no encontrada. Lo siento, la página que buscas no existe."

El paso 4 es donde hemos agregado la ruta genérica para manejar cualquier ruta no definida previamente. Al acceder a cualquier ruta diferente a las especificadas en los pasos anteriores, se mostrará el mensaje personalizado del servidor. Esto ayuda a mejorar la experiencia del usuario al proporcionar una respuesta amigable para rutas inexistentes.

¡Manos a la obra! - GET/usuario/:nombre

Desarrollar una ruta **GET /usuario/:nombre** que almacene el parámetro nombre en una constante y devuelva un mensaje de éxito en caso de que el nombre empiece con una vocal.

¡Manos a la obra! - Spotify

Desarrollar una ruta GET /musica que redireccione al cliente al sitio web de Spotify.

¡Manos a la obra! - GET/colores

Desarrollar un middleware para la ruta **GET /colores** que evalúe si el parámetro “color” es igual a “Azul”, de ser correcto devolver un mensaje de éxito, de lo contrario un mensaje de fracaso.

¡Manos a la obra! - Servidor con Express

Desarrollar un servidor con Express que sirva un sitio web estático y utilice un fichero .CSS alojado en un directorio local.

¡Continúa aprendiendo y practicando!

Soluciones

1. Desarrollar una ruta **GET /usuario/:nombre** que almacene el parámetro nombre en una constante y devuelva un mensaje de éxito en caso de que el nombre empiece con una vocal.

```
app.get("/usuario/:nombre", (req, res) => {
  const nombre = req.params.nombre;
  const test = nombre.match(/^[aeiouAEIOU]/)
  test
    ? res.send("Si, tu nombre empieza con una vocal")
    : res.send("No, tu nombre no empieza con una vocal");
});
```

2. Desarrollar una ruta **GET /musica** que redireccione al cliente al sitio web de Spotify.

```
app.get("/musica",  (req, res) => {
  res.redirect("https://www.spotify.com/cl/");
});
```

3. Desarrollar un middleware para la ruta **GET /colores** que evalúe si el parámetro "color" es igual a "Azul", de ser correcto devolver un mensaje de éxito, de lo contrario un mensaje de fracaso.

```
app.use("/colores/:color", (req, res, next) => {
  const { color } = req.params;
  color == "Azul" ? next() : res.send("No es azúl");
});

app.get("/colores/:color", (req, res, next) => {
```

```
  res.send("Si, es azúl")
});
```

4. Desarrollar un servidor con Express que sirva un sitio web estático y utilice un fichero con código CSS alojado en un directorio local.

```
const express = require("express");
const app = express();

app.listen(3000, () => {
  console.log("El servidor está inicializado en el puerto 3000");
});
app.use(express.static("assets"));

app.get("/", (req, res) => {
  res.sendFile(__dirname + '/index.html')
})
```

```
body{
    background: Green;
    color: white;
}
```

```
<link rel="stylesheet" href="estilos.css">
<h1>¡Color esperanza!</h1>
```