Introducción a Node

Middlewares y devolución de sitios web estáticos

{desafío} latam_

Describir las características fundamentales del entorno Node.js y su utilidad para el desarrollo de aplicaciones web.

- Unidad 1:
  Introducción a Node
- Unidad 2:
  Node y el gestor de paquetes
- Unidad 3:
  Persistencia

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Reconocer el uso de los middlewares en el desarrollo de servidores con el framework Express.*

- *Construir un servidor que utilice middlewares para el procesamiento de una lógica antes de la ejecución de la ruta.*

{desafío} latam_

¿Cómo se define una ruta genérica en Express y qué propósito tiene?

{desafío} latam_

¿Cuáles son algunas propiedades del objeto request en Node.js y Express que nos permiten acceder a la información de una consulta HTTP?

{desafío} latam_

/* Middlewares */

Middlewares

Entendiendo que las rutas creadas en nuestro servidor contienen una función que se ejecuta en el momento que son consultadas, por su parte, los middlewares son funciones que se activan antes que nuestras rutas y nos sirven como filtro o estación previa, donde podemos definir diferentes validaciones. Su sintaxis es muy parecida a la creación de una ruta, pero el método que ocupamos es diferente, como puedes ver en el siguiente código de ejemplo:

```
// Middleware
app.use(<PATH>, (req, res, next) => {
  // Validaciones
  next()
});
```

Middlewares

```
// Middleware
app.use(<PATH>, (req, res, next) => {
  // Validaciones
  next()
});
```

- El método para la instancia “app” se llama “use”, a diferencia de las rutas que se definen según el método HTTP que deseemos disponer en nuestro servidor.
- Ocupamos un tercer parámetro llamado “next”, que será una función que al ejecutarse permitirá la continuación de la consulta, bien sea a otro middleware o a la ruta que se está consultando.

Demostración
"El tiempo secreto"

Ejercicio guiado

El tiempo secreto

Validar la propiedad Authorization dentro de las cabeceras de una consulta realizada a la ruta **GET /Tiempo**. El objetivo será simular la verificación de tokens que se realizan en servicios web cuyos recursos están restringidos con verificación de usuarios basada en tokens. ¿Y cómo vamos a evaluar si existe o no la propiedad Authorization? Primero la vamos a extraer con el objeto request y a través de un operador ternario condicionamos la respuesta del servidor.

Nuestro condicional evaluará el valor de la propiedad Authorization y en caso de devolver “true”, se procederá a ejecutar la función “next”, de esa forma, el cliente podrá pasar a la lógica que tenemos en la propia ruta **GET /Tiempo**, donde se estará devolviendo un objeto con el tiempo del servidor.

Sigue los pasos para realizar el siguiente ejercicio:

Ejercicio guiado

El tiempo secreto

- ***Paso 1:*** Agregar un middleware para la ruta **/Tiempo.**
- ***Paso 2:*** Utilizar el parámetro “request” y su método “header” para almacenar en una constante el valor de la propiedad “Authorization”.
- ***Paso 3:*** A través de un operador ternario, evalúa la constante creada en el paso anterior y en caso de ser “true”, ejecutar el parámetro “next()”, de lo contrario responder con un mensaje “¿Quién es?”.
- ***Paso 4:*** Crear una ruta **GET /Tiempo.**
- ***Paso 5:*** Crear y devolver un objeto que incluya la fecha actual. Para esto, ocupa el Date.now()

Ejercicio guiado

El tiempo secreto

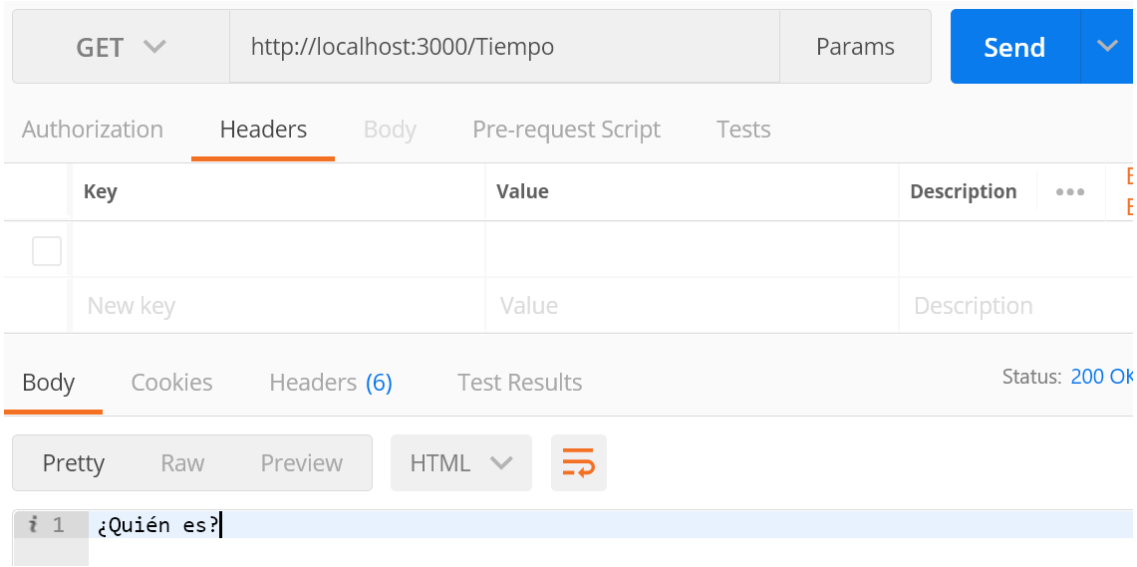
Tómate unos minutos para leer el código e identifica las diferencias entre un middleware y una ruta.

```
// Paso 1
app.use("/Tiempo", (req, res, next) => {
  // Paso 2
  const Auth = req.header("Authorization");
  // Paso 3
  Auth ? next() : res.send("¿Quién es?");
});

// Paso 4
app.get("/Tiempo", (req, res) => {
  // Paso 5
  const tiempo = { time: Date.now() };
  res.send(tiempo);
});
```

Ejercicio guiado

El tiempo secreto

¿Qué pasaría si se consultara entonces a la ruta /Tiempo? Utiliza POSTMAN para realizar esta consulta sin especificar nada en las cabeceras. Deberás obtener lo que te mostramos en la siguiente imagen:

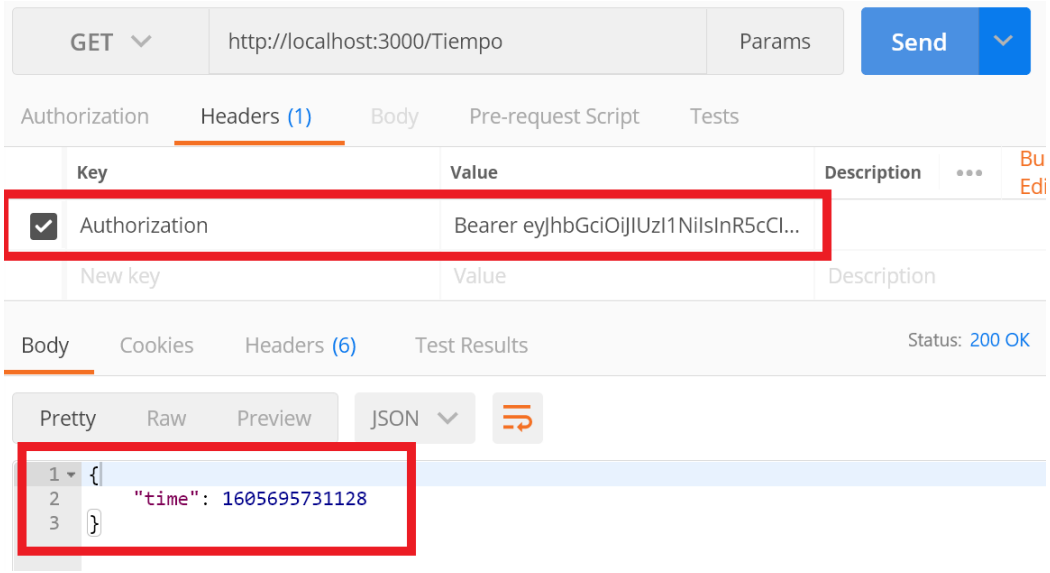


{desafío} latam_

Ejercicio guiado

El tiempo secreto

Ahora, desde el mismo POSTMAN incluye la propiedad Authorization en las cabeceras enviando un token cualquiera y deberás obtener lo que te mostramos en la siguiente imagen:



{desafío} latam_

/* Devolución de sitios web estáticos */

{desafío}
latam_

Publicando directorio

Cuando desarrollamos servidores con Node en módulos pasados, tuvimos la necesidad de crear una ruta por cada archivo estático que deseábamos importar en nuestro documento HTML, o simplemente disponer un fichero o archivo estático al cliente. Con Express, podemos a través de un middleware especificar un directorio y hacer público o “liberar” su contenido, ¿Cómo? **Utilizando el método “static” de la instancia de Express.**

La instrucción que debemos ocupar es la siguiente:

```
app.use(express.static("<Nombre o dirección de la carpeta o archivo>"));
```

Publicando directorio

```
app.use(express.static("<Nombre o dirección de la carpeta o archivo>"));
```

Si desglosamos esta instrucción tendremos lo siguiente:

- Estamos creando un middleware para incluir un comportamiento en nuestro servidor.
- Se está usando la instancia directa del paquete Express.
- En la instancia de Express se está ocupando un método llamado “static”, el cual permite definir a través de un String, la dirección de un directorio de archivos estáticos. Sin embargo, también puede ser usado para devolver un único archivo, por ejemplo, un index.html.

/* Sirviendo un sitio web estático */

Sirviendo un sitio web estático

Una de las principales funcionalidades de los servidores web, es servir aplicaciones clientes, cuyo contenido esté alojado en el servidor. De esta manera podemos mezclar ambos mundos (frontend y backend) en un mismo entorno de desarrollo. ¿Qué ventaja consigo con esto? El poder que tenemos a nivel operativo en el backend no lo podemos conseguir desde el frontend y las interacciones con los usuarios que logramos en las aplicaciones clientes no lo podemos programar desde el backend, por lo que la unión de estos 2 mundos nos aumentan inmensamente las posibilidades y amplían nuestro rango de funcionalidades

Demostración
“Devolviendo un sitio web”

Ejercicio guiado

Devolviendo un sitio web

Servir un sitio web estático desde un servidor desarrollado con Express, donde se solicita imprimir una cabecera y una imagen importada desde un directorio público definido en el servidor.

Consideraciones previas:

1. Crear un documento “index.html” con el siguiente código en tu árbol de archivos a la misma altura que el script donde tienes escrito tu servidor:

```
<h1>¡Soy un sitio web bebé =)!</h1>
<img src="Paisaje.jpeg" />
```

Ejercicio guiado

Devolviendo un sitio web

2. Crear una carpeta “assets” en tu servidor.

2. Dentro de la carpeta assets agrega alguna imagen que tengas guardada en tu sistema operativo. En mi caso usaré una imagen llamada “Paisaje.jpeg”.

2. Express contiene un método llamado “sendFile” en el objeto response de una ruta. Este método permite devolver un archivo especificando su ruta como argumento. Sin embargo, esta ruta se debe definir por medio de una concatenación con una variable de entorno llamada “__dirname”, la cual contiene la referencia a la dirección dentro del árbol de archivos en donde está ubicado el archivo en el que se ocupa.

Ejercicio guiado

Devolviendo un sitio web

Con las consideraciones previas realizadas, sigue los pasos para el desarrollo de este ejercicio:

- **Paso 1:** Crear un servidor con Express que escuche el puerto 3000.
- **Paso 2:** Ocupar un middleware y el método “static” de Express para declarar la carpeta “assets” como directorio público del servidor.
- **Paso 3:** Crear una ruta GET raíz que devuelva el documento index.html.

Ejercicio guiado

Devolviendo un sitio web

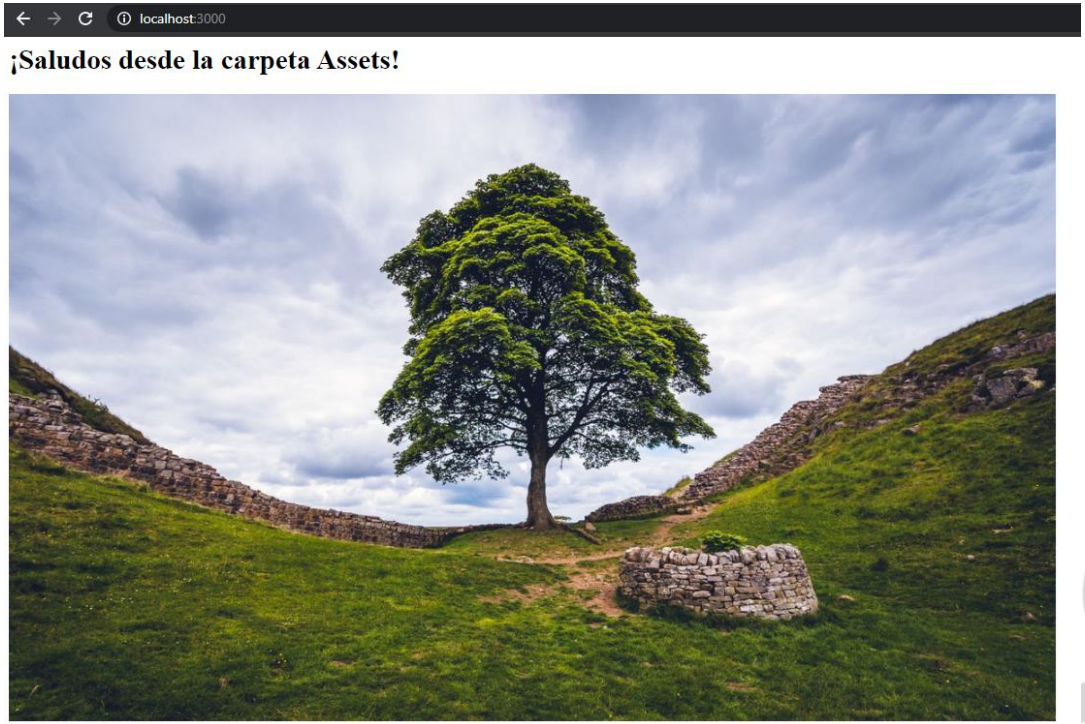
```
// Paso 1
const express = require("express");
const app = express();

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

// Paso 2
app.use(express.static("assets"));

// Paso 3
app.get("/", (req, res) => {
  res.sendFile(__dirname + '/index.html')
})
```

{desafío} latam_

Ejercicio guiado

Devolviendo un sitio web

Ahora, abre tu navegador y consulta la dirección http://localhost:3000/, deberás recibir lo que te mostramos en la siguiente imagen:



{desafío} latam_

Resumen

- Los middlewares son funciones que se ejecutan antes de emitir una respuesta en una aplicación cliente-servidor. Se utilizan para realizar validaciones y procesar datos de la consulta, y tienen acceso al objeto "request" y al objeto "response" para extraer parámetros, payload, cabeceras y métodos utilizados en la consulta.

- Los middlewares se definen en el servidor utilizando el método "use" en lugar de los métodos HTTP utilizados para definir rutas. Estas funciones actúan como filtros previos a las rutas y se activan antes de ellas. Se utiliza un tercer parámetro llamado "next" que permite la continuación de la consulta hacia otro middleware o hacia la ruta consultada

Resumen

- Servir un sitio web estático desde un servidor desarrollado con Express ofrece la ventaja de combinar el desarrollo frontend y backend en un mismo entorno. Esto amplía las posibilidades y funcionalidades al aprovechar las capacidades operativas del backend y las interacciones con los usuarios logradas en las aplicaciones clientes. Al devolver un sitio web estático, se pueden imprimir elementos como cabeceras e imágenes importadas desde un directorio público definido en el servidor, lo que permite una integración más completa y potente entre ambos mundos.

¿Cuál es la función principal de los middlewares en el desarrollo de servidores?

¿Cuál es la ventaja de utilizar Express para servir un sitio web estático en comparación con el desarrollo puro con Node?

{desafío} latam_

Próxima sesión...

- *Describir las características fundamentales del entorno Node para el desarrollo de aplicaciones web.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales