Node y el gestor de paquetes

Paquete Jimp y Nodemailer

{desafío} latam_

Competencia general de la unidad

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

{desafío}
latam_

¿Qué aprenderás en esta sesión?

- *Construir un servidor que al ser consultado devuelva una imagen procesada con el paquete Jimp*
- *Ejecutar procedimiento para bajar una aplicación Node utilizando la línea de comandos*
- *Construir una aplicación Node que envíe correos electrónicos con el paquete Nodemailer*

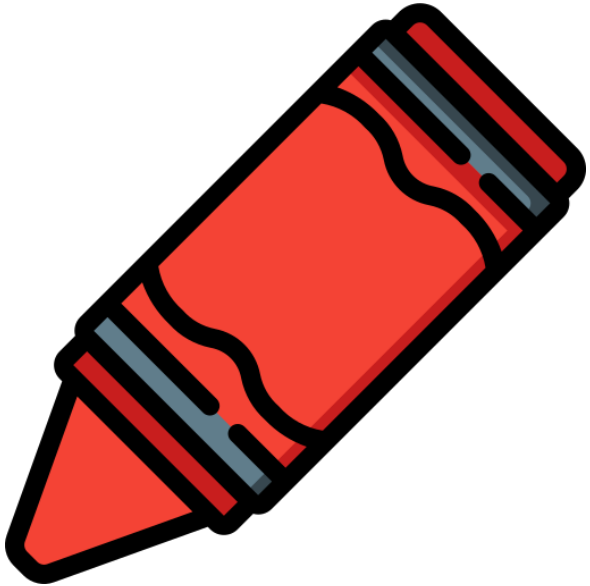
¿Qué herramientas se pueden utilizar en Node.js para manipular y procesar imágenes, enviar correos electrónicos, y cómo se puede detener adecuadamente la ejecución de una aplicación en proceso?

{desafío} latam_

/* El paquete Jimp */

El paquete Jimp

El paquete Jimp pertenece a los paquetes que trabajan con archivos multimedia, en este caso con imágenes específicamente, ofreciéndonos diferentes funcionalidades como por ejemplo, el redimensionamiento, recorte, filtros, orientación, brillo, contraste, entre otros, que puedes conseguir en su repositorio de NPM.



El paquete Jimp

Características

Manipulación de imágenes

Filtros y efectos

Dibujo y texto

Manipulación de píxeles

Soporte para formatos de imagen

Capacidad de carga y guardado

Manipulación de canales

Compatibilidad con promesas

Demostración
"Aplicando el paquete Jimp"

{desafío}
latam_

Ejercicio guiado

Aplicando el paquete Jimp

Crear un servidor que al consultarse devuelva una imagen redimensionada a 250 pixeles de ancho y un alto automático, además de aplicarle el filtro sepia. ¿Y cómo se hace eso? A continuación te mostramos la sintaxis correspondiente a la instancia de Jimp que usarás para lograrlo:

```
Jimp.read(<dirección de la imagen>, <callback(err,imagen)>)
```

Ejercicio guiado

Aplicando el paquete Jimp

Dentro del callback recibiremos como primer parámetro el error en caso de existir y como segundo parámetro la instancia de la imagen leída. Lo siguiente será usar la sintaxis con la instancia de la imagen:

```
imagen
.resize(<ancho>, <alto>)
.sepia()
.writeAsync(<nombre_nueva_imagen>)
.then()
```

Ejercicio guiado

Aplicando el paquete Jimp

La dirección de la imagen puede apuntar a una imagen local o remota, en este caso se usará la URL de una imagen de Google, siéntete libre de usar la que quieras para el experimento.

Ahora que sabemos cómo usar el paquete Jimp, prosigue con los siguientes pasos para la solución de este ejercicio:

- **Paso 1:** Crea una instancia del paquete Jimp

```
const Jimp = require('jimp');
```

Ejercicio guiado

Aplicando el paquete Jimp

- **Paso 2:** Crea una ruta raíz GET /

```
app.get("/", async (req, res) => {
```

- **Paso 3:** Define con el objeto response que la cabecera de la respuesta tendrá un contenido de tipo image/png

```
    res.setHeader('Content-Type', 'image/png')
```

Ejercicio guiado

Aplicando el paquete Jimp

- **Paso 4:** Utiliza el método read que se encuentra dentro de la instancia del paquete Jimp para leer una URL y posteriormente usar los métodos resize, sepia y writeAsync para cambiar el tamaño, asignar el filtro sepia y posteriormente almacenar la imagen procesada con nombre img.png.

```
   const imagen = await Jimp.read(
'https://miviaje.com/wp-
content/uploads/2016/05/shutterstock_337174700.jpg'
     )
    await imagen
        .resize(250, Jimp.AUTO)
        .sepia()
        .writeAsync('img.png')
```

Ejercicio guiado

Aplicando el paquete Jimp

- **Paso 5:** Concluye la consulta con la data del archivo de imagen creado. Utiliza el File System para esto.

```
    const imagenData = fs.readFileSync('img.png')
    res.send(imagenData)
})
```

Como dato extra, esto es 100% escalable y si lo aplicamos en un plano más real, la dirección de la imagen debería ser asignada por un usuario en un formulario desde el cliente.

¿Cómo crees que la capacidad de manipular y procesar imágenes programáticamente utilizando una biblioteca como Jimp podría influir en la forma en que las aplicaciones web y móviles brindan experiencias visuales personalizadas y atractivas a los usuarios?

{desafío} latam_

/* Cancelando una aplicación en proceso */

Cancelando una aplicación en proceso

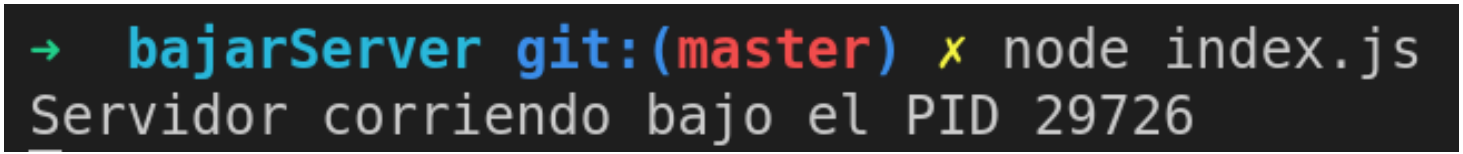
Cancelar un proceso en ejecución, como un servidor, es esencial para liberar recursos. Cada proceso en un sistema operativo tiene un PID único que lo identifica. Al conocer este PID, es posible terminarlo voluntariamente, lo que libera los recursos y el puerto que estaba usando.

El PID se obtiene del objeto "process" y varía con cada inicio. Aunque un servidor puede no tener procesos internos, su ocupación de puerto importa. El código proporcionado ayuda a entender el proceso.

```
const express = require('express');
const app = express();
app.listen(8080, console.log('Servidor corriendo bajo el PID',
process.pid))
```

Cancelando una aplicación en proceso

Si procedes a levantar el servidor deberás recibir un mensaje como el que te muestro en la siguiente imagen.



Como puedes notar recibí el número “29726”, el cual representa el identificador del proceso que mi sistema operativo ejecutó para levantar el servidor. Para cancelar este proceso de forma manual bastará con presionar Crtl + C. Sin embargo, lo queremos hacer por medio de la línea de comando. Esto se consigue diferente según el sistema operativo

Cancelando una aplicación en proceso

- En MAC y LINUX disponemos del siguiente comando:

```
kill -9 <pid>
```

- Pero en windows se usará el siguiente comando:

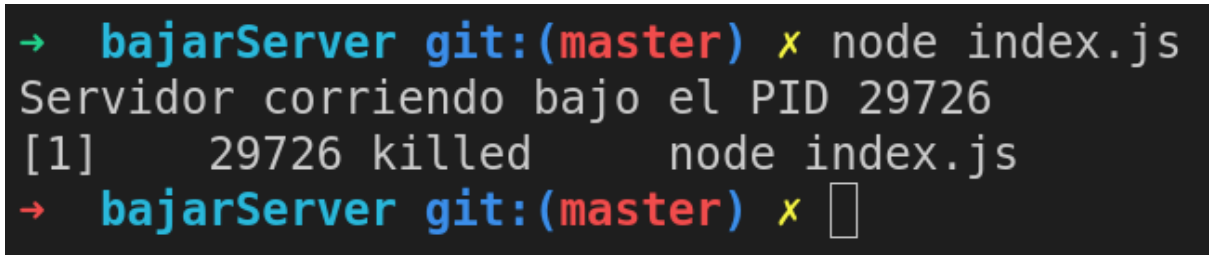
```
taskkill /F /PID <pid>
```

En donde deberás cambiar <pid> por el número identificador del proceso.

Ahora intentemos cancelar el servidor levantado, para esto deberás abrir otra terminal y proceder con la instrucción, en este caso se ocupará el siguiente comando:

```
kill -9 29726
```

Obteniendo como resultado lo que se muestra en la siguiente imagen:



/* Nodemailer */

Nodemailer

Nodemailer es otro paquete muy conocido de NPM. Nos sirve como herramienta para hacer envíos de correos electrónicos con Node y cuenta con un sitio oficial, en el que encontrarás toda la información que necesites en caso de que quieras profundizar en su API y sus herramientas.

Nodemailer

Propiedades de nodemailer

Nodemailer al ser importado en una variable nos ofrece diferentes propiedades y métodos. El que utilizaremos en esta clase será el método “createTransport”, el cual recibe como argumento un objeto de configuración que especifica el servicio(host) y las credenciales de autenticación, pero ¿Qué sería un Transport? Sucede que SMTP es el medio de transporte principal en Nodemailer para entregar mensajes. SMTP es también el protocolo utilizado entre diferentes hosts de correo electrónico, por lo que es verdaderamente universal.

Casi todos los proveedores de entrega de correo electrónico admiten el envío basado en SMTP, incluso si impulsan principalmente su envío basado en API. Las API pueden tener más funciones, pero el uso de estas también significa el bloqueo del proveedor, mientras que en el caso de SMTP solo se necesita cambiar las opciones de configuración para reemplazar un proveedor por otro y listo.

Nodemailer

Propiedades de nodemailer

Veamos como se ve lo anterior con el siguiente código en donde te muestro una sintaxis básica de este método:

```
let transporter = nodemailer.createTransport({
 service: <proveedor>,
 auth: {
   user: <correo electrónico host>,
   pass: <contraseña del electrónico host>,
 },
})
```

Nodemailer

Propiedades de nodemailer

Estas propiedades se describen de la siguiente manera:

- **service:** Proveedor de correos electrónicos. Este debe ser un servicio dedicado al envío de correos, en esta lectura ocuparemos a Gmail para esto.

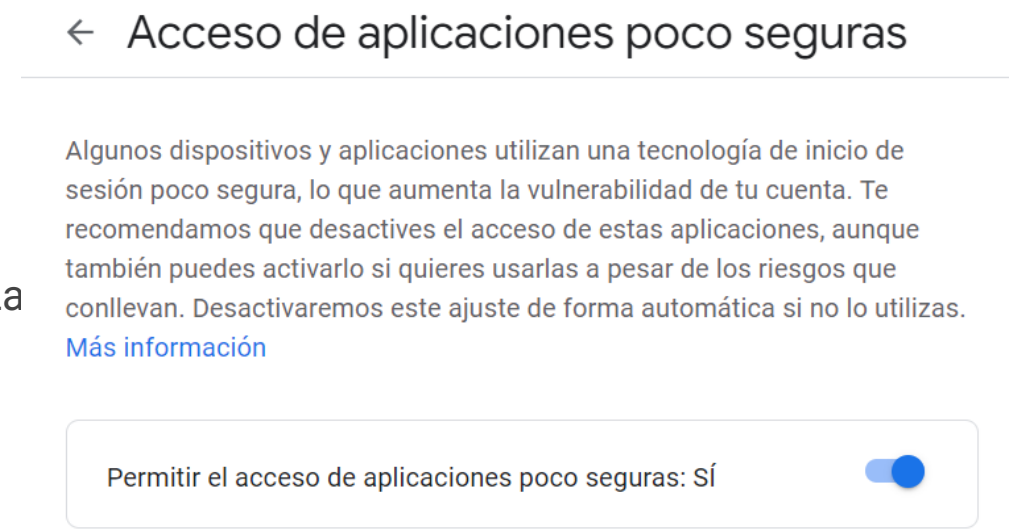
- **auth:** En formato de objeto, representa las credenciales del usuario que está ejecutando el envío del correo, y recibe las siguientes propiedades:

    - **user:** La dirección del correo que servirá como host para el envío del correo electrónico
    - **pass:** La contraseña del correo que servirá como host para el envío del correo electrónico

Nodemailer

Propiedades de nodemailer

Para poder utilizar tu propio correo electrónico gmail, debes activar la opción en el siguiente link. Gmail permite utilizar tu correo como proveedor, pero necesita que lo especifiques con tu cuenta personal. La opción debe quedar como se muestra en la siguiente imagen:



Nodemailer

Propiedades de nodemailer

Por motivos de esta lectura se creó un correo electrónico Gmail, de manera que si no quieres usar tu correo personal y no tienes uno alterno, puedas utilizarlo sin problema.

Las credenciales de este correo electrónico son las siguientes:

- **Correo:** nodemailerADL@gmail.com
- **Contraseña:** vamoscontodo

Importante: NO MODIFICAR la contraseña del gmail otorgado. Si la contraseña no se encuentra vigente, solicitar al docente que se contacte con Soporte Académico.

{desafío} latam_

Nodemailer

Propiedades de nodemailer

El proveedor de correos será el mismo Gmail, por lo que el código expuesto anteriormente quedará de la siguiente manera:

```
let transporter =
nodemailer.createTransport({
 service: 'gmail',
 auth: {
   user: 'nodemailerADL@gmail.com',
   pass: "vamoscontodo",
 },
})
```

Nodemailer

Propiedades de nodemailer

En síntesis, este usuario corresponde a la cuenta que tu proveedor de correos te asigne y ya que estamos usando Gmail, corresponderá a las credenciales del correo directamente.

Lo siguiente, es ocupar un método de nuestra variable “transporter” llamado “sendMail”, el cual tiene la siguiente sintaxis:

```
transporter.sendMail(<opciones_correo>,<callback>)
```

Nodemailer

Propiedades de nodemailer

Como puedes ver, necesitamos pasarle las opciones del correo electrónico que queremos enviar, este será un objeto que podemos crear en otra variable y tiene las siguientes propiedades básicas:

```
let mailOptions = {
     from: <correo electrónico que envía el mensaje>,
     to: <el o los correos a los que se quiere mandar el
mensaje>,
     subject: <asunto del correo>,
     text: <contenido del correo>,
   }
```

Nodemailer

Propiedades de nodemailer

Tenemos como opciones o propiedades principales las siguientes:

- from: Correo de origen, este dato debe ser tipo String.
- to: Destinatario o destinatarios, en caso de ser solo 1 correo, se puede especificar como String, pero si quisiéramos enviar el correo a varias personas para hacer un envío de correos electrónicos masivos, podemos pasar como vale un arreglo de Strings especificando todos los correos de destino.
- subject: Asunto del correo, este dato debe ser tipo String
- text: Mensaje en formato de texto plano.

Nodemailer

Propiedades de nodemailer

Para probar por primera vez Nodemailer, llenaremos los datos de la siguiente manera:

```
let mailOptions = {
     from: 'nodemailerADL@gmail.com',
     to: 'nodemailerADL@gmail.com',
     subject: 'Nodemailer Test',
     text: 'Probando… 1,2,3...',
   }
```

¿De qué manera la combinación de la manipulación programática de imágenes con Jimp y el envío de correos electrónicos automatizados a través de Nodemailer podría dar lugar a la creación de aplicaciones innovadoras y efectivas que ofrezcan experiencias únicas tanto a nivel visual como en la comunicación con los usuarios?

{desafío} latam_

Próxima sesión...

- *Reconocer el uso del protocolo SMTP para el envío de correos electrónicos.*
- *Construir un servidor que procese el envío de un correo electrónico a partir de una consulta HTTP*

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales