Guía de ejercicios - Node y el gestor de paquetes

 ¡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!

Actividad guiada: Mi primer envío de correo electrónico en Node

Desarrollar una aplicación básica que envíe un correo electrónico con el paquete nodemailer, pero antes deberás instalar el paquete, así que abre la consola y escribe el siguiente comando y luego sigue los siguientes pasas para llegar a la solución:

```
npm i nodemailer
```

Con esto tenemos todo listo para realizar nuestra primera prueba, sigue los pasos para hacer el envío de tu primer correo electrónico con nodemailer.

- ***Paso 1:*** Importar el paquete Nodemailer a una constante.
- ***Paso 2:*** Guardar en una variable "transporter" el llamado al método createTransport pasándole como configuración las siguientes propiedades:
    - service: gmail
    - auth:
        - user: nodemailerADL@gmail.com
        - pass: vamoscontodo
- ***Paso 3:*** Crear una variable "mailOptions" para definir las opciones del correo electrónico de prueba, las cuales serán las siguientes:
    - from: nodemailerADL@gmail.com
    - to: nodemailerADL@gmail.com
    - subject: Nodemailer Test
    - text: Probando... 1,2,3...
- ***Paso 4:*** Invoca el método sendMail de la instancia transporter pasándole como argumento las opciones del correo y en su segundo parámetro recibirás un callback, con el error y la data correspondiente al envío.
- ***Paso 5:*** Usa condicionales para imprimir por consola el error y la data correspondiente al envío en caso de existir.

```
// Paso 1
const nodemailer = require('nodemailer')
```

```
// Paso 2
let transporter = nodemailer.createTransport({
 service: 'gmail',
 auth: {
   user: 'nodemailerADL@gmail.com',
   pass: 'vamoscontodo',
 },
})

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

// Paso 4
transporter.sendMail(mailOptions, (err, data) => {
 // Paso 5
 if (err) console.log(err)
 if (data) console.log(data)
})
```

Ahora si abres la terminal y levantas la aplicación deberás obtener lo que se muestra en la siguiente imagen:

Imagen 1. Mensaje por consola indicando que el correo fue enviado con éxito.
Fuente: Desafío Latam

Con este mensaje tenemos la certeza de que el correo fue enviado sin problemas.

¿Pero es así realmente? Para verificarlo abre la cuenta de Gmail y revisa la bandeja de "recibidos" y deberás tener lo que se muestra en la siguiente imagen:

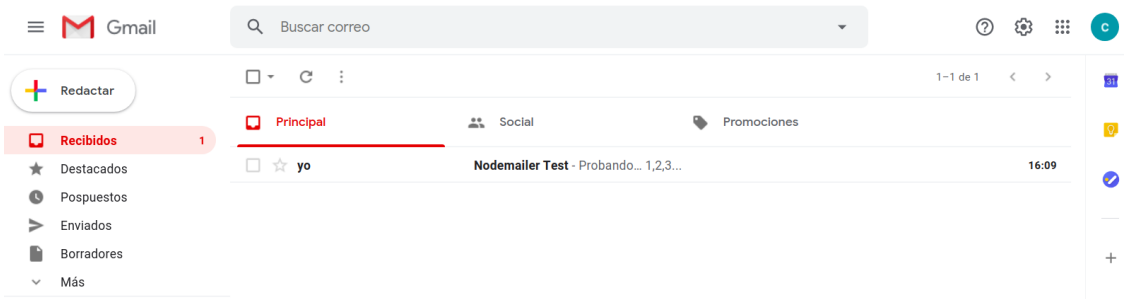

Imagen 2. Bandeja de entrada indicando que se recibió el correo electrónico.
Fuente: Desafío Latam

¡Maravilloso! Hemos logrado enviar un correo electrónico en simples pasos con el paquete nodemailer.

¡Manos a la obra! - Aplicación con colores

Crear una aplicación que devuelva por consola un mensaje de fondo amarillo y color de texto verde.

¡Manos a la obra! - Utiliza un UUID

Crear una aplicación que devuelva por consola los últimos 6 caracteres de un UUID.

¡Manos a la obra! - Aplicación de fecha

Crear una aplicación que devuelve la fecha actual sumando 10 días y en el siguiente formato "dddd"

¡Manos a la obra! - Particionando un arreglo

Basado en el ejercicio “Pares o nones”, crea una aplicación que particione el siguiente arreglo

```
const numeros = [true, 0, null, undefined, '', 22, false]
```

El objetivo será dividir el arreglo en dos grupos en donde el primero contenga los elementos evaluados lógicamente como true y el segundo los elementos considerados lógicamente como false.

¡Manos a la obra! - Indicadores económicos

Basado en el ejercicio de Rick and Morty, desarrolla una aplicación en Node que utilice Axios para consultar el valor del dólar en Chile utilizando la Siguiente API de indicadores económicos.

¡Manos a la obra! - Imagen procesada con resize, grayscale y quality

Crear un servidor que al ser consultado devuelva una imagen procesada con los métodos resize, grayscale y quality.

El método grayscale al igual que el sepia, no necesitan recibir un parámetro, no obstante el quality recibe un número del 0 al 100 indicando el porcentaje de calidad. Por ejemplo, si quisieras bajar a un 60% la calidad de una imagen ocuparían el método quality como te muestro a continuación

```
quality(60)
```

Soluciones

1. Crear una aplicación que devuelva por consola un mensaje de fondo amarillo y color de texto verde

```
console.log(chalk.green.bgYellow('Hola Mundo!'))
```

2. Crear una aplicación que devuelva por consola los últimos 6 caracteres de un uuid

```
console.log(uuidv4().slice(30))
```

3. Crear una aplicación que devuelva la fecha actual sumando 10 días y en el siguiente formato “dddd”

```
console.log(moment().add(10, 'days').format('dddd'))
```

4. Basado en el ejercicio “Pares o nones”, crea una aplicación que particione el siguiente arreglo

```
const numeros = [true, 0, null, undefined, '', 22, false]
console.log(_.partition(numeros, (n) => n))
```

5. Basado en el ejercicio de Rick and Morty, desarrolla una aplicación en Node que utilice Axios para consultar el valor del dólar en Chile utilizando la api de https://mindicador.cl/api.

```
const axios = require("axios");

axios
  .get("https://mindicador.cl/api")
  .then((data) => {
    const dolar = data.data.dolar.valor;
    console.log("El valor del dolar en Chile es: " + dolar);
  })
  .catch((e) => {
    console.log(e);
  });
```

6. Crear un servidor que al ser consultado devuelva una imagen procesada con los métodos resize, grayscale y quality.

   El método grayscale al igual que el sepia no necesitan recibir un parámetro, no obstante el quality recibe un número del 0 al 100 indicando el porcentaje de calidad.

```
app.get("/", async (req, res) => {
    res.setHeader('Content-Type', 'image/png')
    const imagen = await Jimp.read(
'https://miviaje.com/wp-content/uploads/2016/05/shutterstock_337174700.jpg'
     )
    await imagen
        .resize(550, Jimp.AUTO)
       .greyscale()
       .quality(20)
       .writeAsync('img.png')
    const imagenData = fs.readFileSync('img.png')
    res.send(imagenData)
})
```