



Curso complementario Desarrollo de Back-end con Node.js - MongoDB

ING - DIEGO CASALLAS

22810019-1



www.sena.edu.co

Node.js - Email



Enviar Correo electrónico

- Configuración de servidor Gmail SMTP
 - Guía de configuración [Link](#)
- Usar un servicio de mensajería como u otras
 - <https://app.brevo.com/>

SMTP

Es un Protocolo Simple de Transferencia de Correo, es un protocolo estándar utilizado para enviar correos electrónicos a través de Internet.

Funciona como un sistema de mensajería que permite que los servidores de correo electrónico se comuniquen y transfieran mensajes entre sí, desde el remitente hasta el destinatario final.

¿Cómo funciona SMTP?

1. **Envío del correo:** Cuando envías un correo electrónico, tu cliente de correo (como Outlook, Gmail, etc.) se conecta a un servidor SMTP y le entrega el mensaje.
2. **Transferencia entre servidores:** El servidor SMTP del remitente se encarga de encontrar el servidor SMTP del destinatario, utilizando registros DNS y otros protocolos de red.
3. **Entrega al destinatario:** Una vez que el correo llega al servidor SMTP del destinatario, este lo almacena en el buzón del destinatario.
4. **Recepción del correo:** El destinatario utiliza un protocolo de recepción de correo (como POP3 o IMAP) para acceder y leer sus mensajes.

¿Qué son los puertos SMTP?

- **Los puertos SMTP son números de puerto específicos en un servidor que permiten la comunicación a través de SMTP.**
- **Los puertos más comunes son:**
 - **25:** El puerto SMTP original, pero ya no se recomienda para enviar debido a problemas de seguridad y bloqueo por parte de los ISP.
 - **465:** Un puerto SSL/TLS antiguo, pero no recomendado por seguridad.
 - **587:** El puerto SMTP recomendado para el envío de correo electrónico, que utiliza STARTTLS para cifrar la conexión.
 - **2525:** Una alternativa al puerto 587, a menudo utilizado cuando el **587** está bloqueado.

Estructura y Funcionamiento de SMTP:

1. Conexión:

- a. El cliente se conecta al servidor SMTP a través de un puerto específico (generalmente el 25, 587 o 465).

2. Comandos:

- a. El cliente envía comandos al servidor para iniciar la sesión, identificar remitente, destinatario y transferir el contenido del mensaje.

3. Respuestas:

- a. El servidor responde a los comandos con códigos numéricos y mensajes opcionales, confirmando la recepción o indicando errores.

4. Transacciones SMTP:

a. **Una transacción SMTP consta de tres secuencias de comandos/respuesta:**

- i. **MAIL FROM:** Establece la dirección de retorno del mensaje (remitente).
- ii. **RCPT TO:** Establece los destinatarios del mensaje.
- iii. **DATA:** Inicia la transferencia del contenido del mensaje (cuerpo y encabezado), que finaliza con un punto en una línea separada.

5. Cierre de la sesión:

a. El cliente puede cerrar la sesión con el comando **QUIT**.

i. Componentes clave:

1. **Cliente SMTP (Agente de Envío de Correo):** La aplicación que envía el correo electrónico (ej. Outlook, Gmail).
2. **Servidor SMTP (Agente de Transferencia de Correo):** El servidor que recibe y reenvía el correo electrónico.
3. **Comandos SMTP:** Instrucciones de texto para controlar la comunicación (ej. HELO, MAIL FROM, RCPT TO, DATA, QUIT).
4. **Códigos de respuesta SMTP:** Códigos numéricos que indican el estado de la transacción (ej. 250 para éxito, 550 para error).

6. Seguridad:

- a. SMTP por sí solo no cifra la comunicación. Se recomienda utilizar puertos seguros como 587 (con STARTTLS) o 465 (con SSL/TLS implícito) para cifrar la conexión y proteger el contenido del correo electrónico.

Node.js - Email



Enviar Correo electrónico

- **Instalar dependencias**

- Verificar las dependencias en el archivo package.js
- De no tenerlas realizar la instalación

- **npm install nodemailer**

- **npm install ejs**

```
"keywords": [],
"author": "",
"license": "ISC",
"description": "",
"dependencies": {
  "bcrypt": "^6.0.0",
  "body-parse": "^0.1.0",
  "cookie-parse": "^0.4.0",
  "dotenv": "^17.2.0",
  "ejs": "^3.1.10",
  "express": "^5.1.0",
  "jsonwebtoken": "^9.0.2",
  "mongoose": "^8.16.3",
  "nodemailer": "^7.0.5"
},
"devDependencies": {
  "@types/express": "^5.0.3",
  "nodemon": "^3.1.10"
}
```

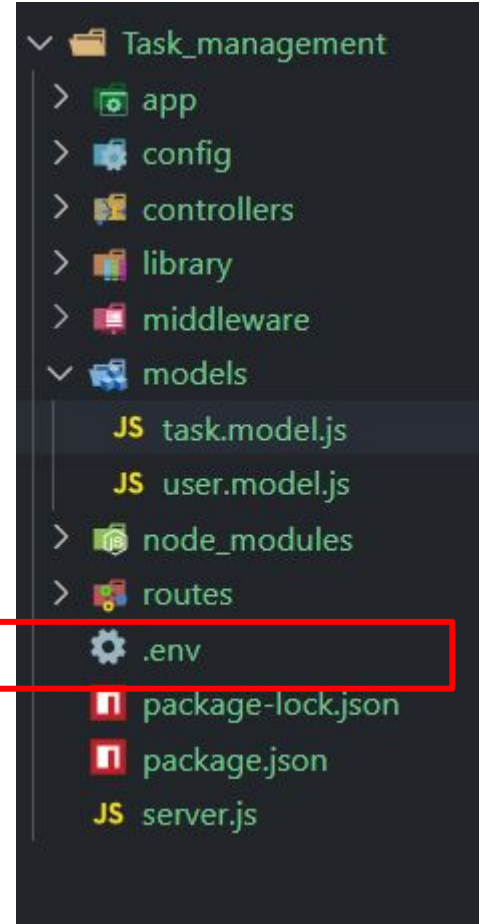
Node.js - Email



.env

```
Task_management > .env
  Import to Postman
1  SERVER_PORT="3000"
2  JWT_SECRET="jwt_secret_key"
3  MONGODB_URI=mongodb://localhost:27017/task_app
4
5  EMAIL_HOST="smtp.hostinger.com"
6  EMAIL_PORT=465
7  EMAIL_SECURE=false
8  EMAIL_USER="*****"
9  EMAIL_PASSWORD="*****"
```

ACTUALITZAR



Node.js - Email

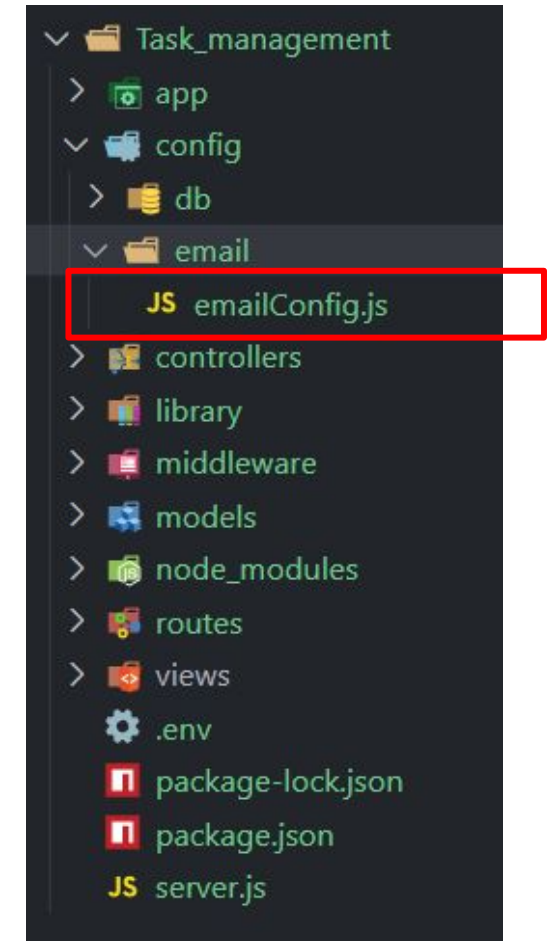


Crear la configuración de correo

config/email->emailConfig.js

```
Task_management > config > email > JS emailConfig.js > ...
1 import nodemailer from 'nodemailer';
2 import dotenv from 'dotenv';
3
4 dotenv.config();
5
6 const transporter = nodemailer.createTransport({
7   host: process.env.EMAIL_HOST || 'smtp.hostinger.com',
8   port: process.env.EMAIL_PORT || 465,
9   secure: process.env.EMAIL_SECURE || true, // true para 465, false para 587
10  auth: {
11    user: process.env.EMAIL_USER,
12    pass: process.env.EMAIL_PASSWORD,
13  },
14 });
15
16 export default transporter;
```

AJUSTAR



Node.js - Email



- Crear la plantilla para enviar el correo electrónico
 - Crear la carpeta views
 - Crear carpeta emails
 - Crear plantilla welcome-email.ejs

Task_management > views > emails > <% welcome-email.ejs > html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <style>
5      body { font-family: Arial, sans-serif; }
6      .header { background-color: #3498db; padding: 20px; color: white; }
7    </style>
8  </head>
9  <body>
10   <div class="header">
11     <h1>!Welcome, <%= name %>!</h1>
12   </div>
13   <p>Thank you for registering on our platform</p>
14 </body>
15 </html>
```

Task_management

- > app
- > config
- > controllers
- > library
- > middleware
- > models
- > node_modules
- > routes
- > views
 - emails
 - <% welcome-email.ejs
- .env
- package-lock.json
- package.json
- server.js

Node.js - Email

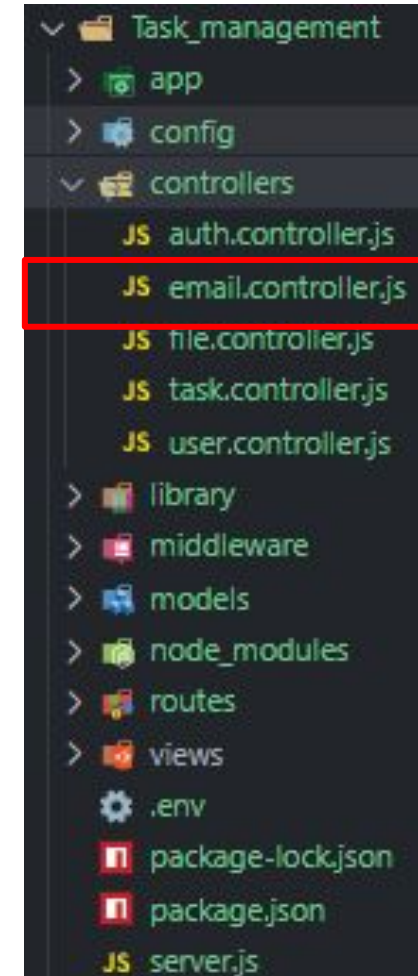


Crear el controlador para el envío de correo

controllers->email.controller.js

```
Task_management > controllers > JS email.controller.js > ...
1  import transporter from '../config/email/emailConfig.js';
2  import { fileURLToPath } from 'url';
3  import path from 'path';
4  import ejs from 'ejs';
5
6  // Configuring routes for ES modules
7  const __filename = fileURLToPath(import.meta.url);
8  const __dirname = path.dirname(__filename);
9
10 class EmailController {
11   /**
12    * Send an email using the provided options.
13    * @param {Object} options - { to, subject, template, data, attachments }
14    */
15   static async sendEmail(options) {
16     try {
17       const { to, subject, template, data, attachments } = options;
18
19       // Render template if provided
20       const html = template
21         ? await ejs.renderFile(
22             path.join(__dirname, `../views/emails/${template}.ejs`),
23             data
24           )
25       : options.html;
```

PLANTILLA



The image shows a file explorer window with a dark theme. The project structure is as follows:

- Task_management
 - app
 - config
 - controllers
 - auth.controller.js
 - email.controller.js (highlighted with a red box)
 - file.controller.js
 - task.controller.js
 - user.controller.js
 - library
 - middleware
 - models
 - node_modules
 - routes
 - views
 - .env
 - package-lock.json
 - package.json
 - server.js

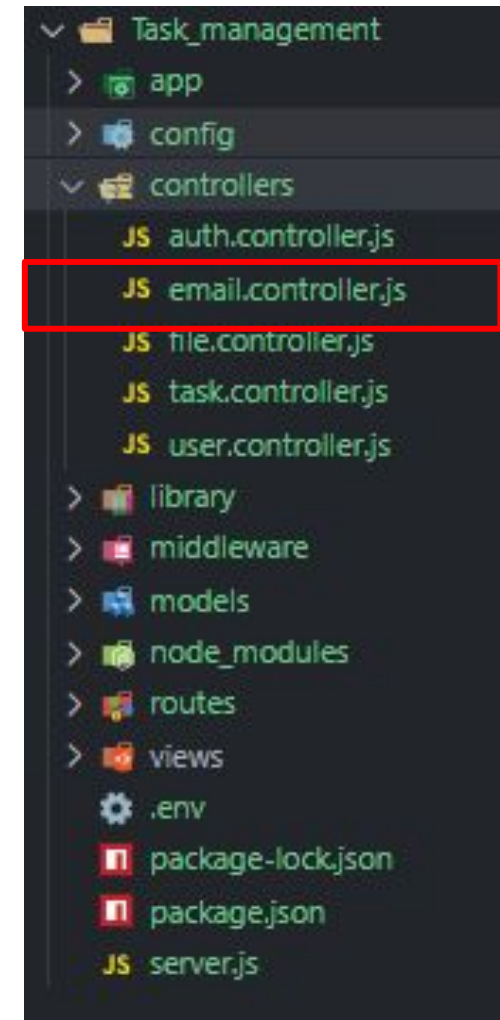
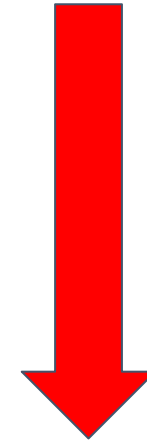
Node.js - Email



Crear el controlador para el envío de correo

controllers->email.controller.js

```
26
27     const mailOptions = {
28       from: `"My App" <${process.env.EMAIL_USER}>`,
29       to,
30       subject,
31       html: html || '<p>Email sended successfully!</p>',
32       attachments,
33     };
34
35     const info = await transporter.sendMail(mailOptions);
36     console.log('Sent succedly:', info.messageId);
37     return info;
38   } catch (error) {
39     console.error('Error in sendEmail:', error);
40     throw new Error('Failed to send email');
41   }
42 }
43
44 // Methods for specific email types
45 static async sendWelcomeEmail(user) {
46   return this.sendEmail({
47     to: user.email,
48     subject: 'Welcome to My App',
49     template: 'welcome-email',
50     data: { name: user.name },
51   });
52 }
```



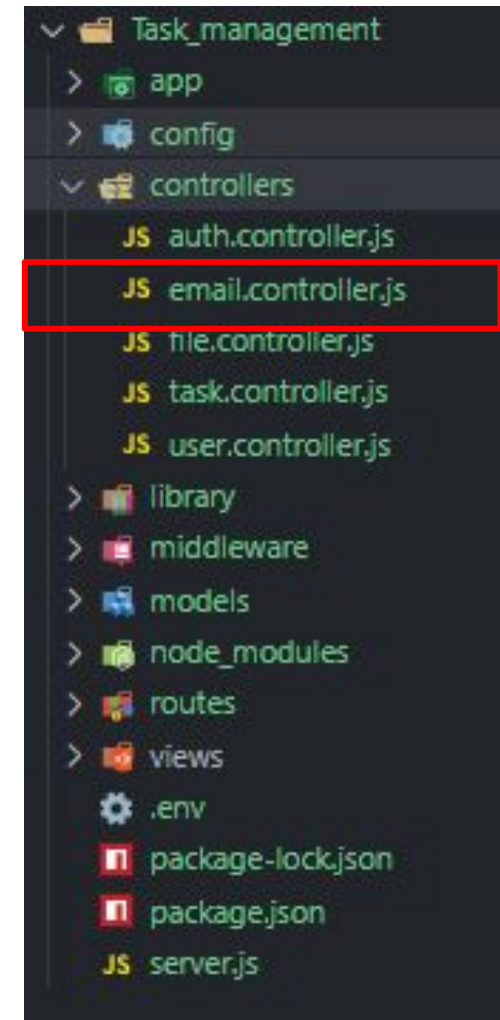
Node.js - Email



Crear el controlador para el envío de correo

controllers->email.controller.js

```
53 // Method to send password reset email
54 static async sendPasswordReset(user, resetLink) {
55   return this.sendEmail({
56     to: user.email,
57     subject: 'Change Password',
58     template: 'reset-password',
59     data: { name: user.name, resetLink },
60   });
61 }
62 }
63
64 export default EmailController;
```



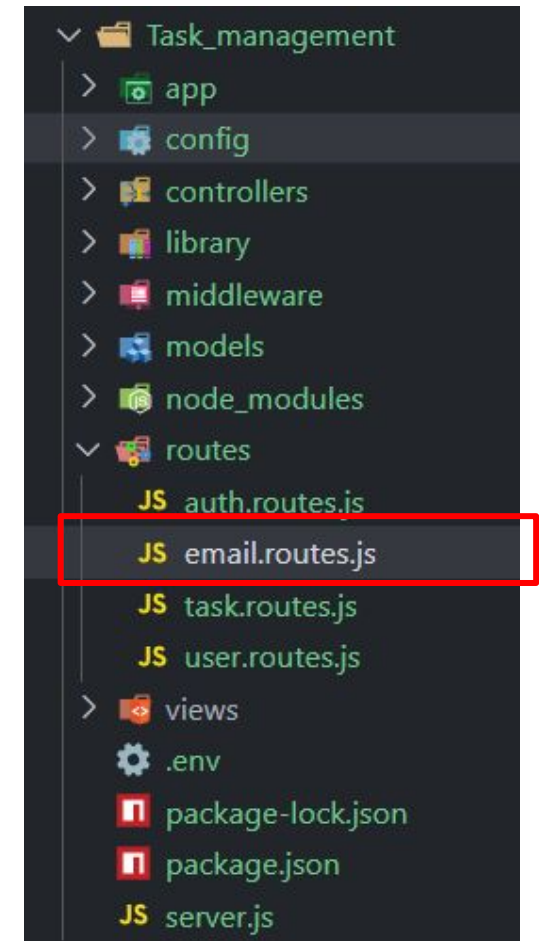
Node.js - Email



Crear la ruta

routes->email.routes.js

```
Task_management > routes > JS email.routes.js > router.post() callback
1 import { Router } from "express";
2 import EmailController from '../controllers/email.controller.js';
3 // Importing the verifyToken middleware to protect routes
4
5 import {verifyToken} from '../middleware/authMiddleware.js';
6 const router = Router();
7 const name='/send-welcome';
8
9 router.post(name, verifyToken, async (req, res) => {
10   try {
11     await EmailController.sendWelcomeEmail(req.body);
12     res.json({ success: true });
13   } catch (error) {
14     res.status(500).json({ error: error.message });
15   }
16 });
17
18 export default router;
```



Node.js - Email

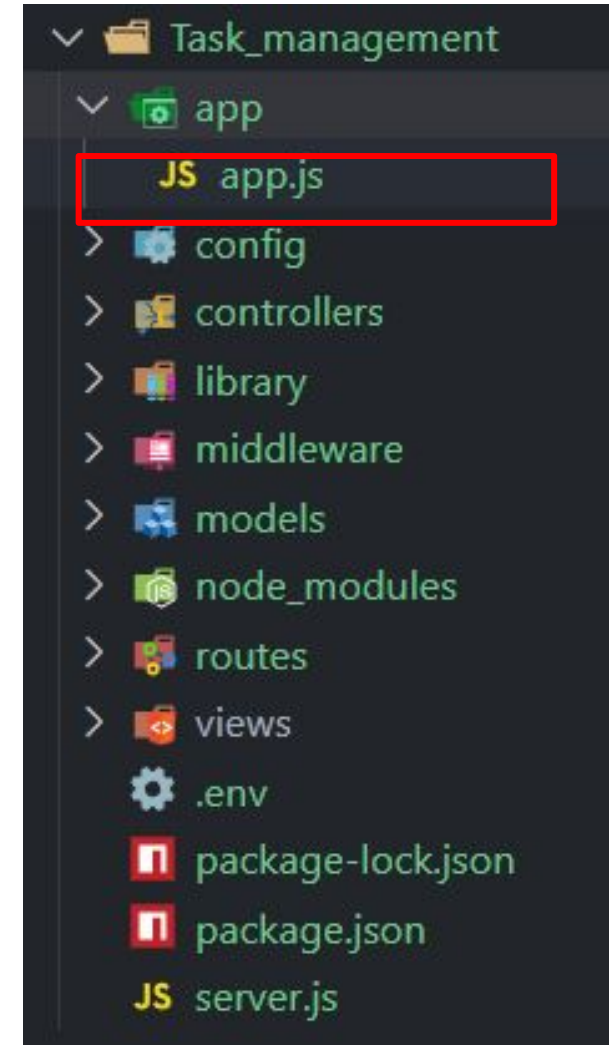


Cargar la ruta a la aplicación

app->app.js

```
Task_management > app > JS app.js > ...
1  import express from 'express';
2  import { connectDB } from '../config/db/connection.js'; // Import the connection
3  // Import routes
4  import authRouter from '../routes/auth.routes.js';
5  import userRouter from '../routes/user.routes.js';
6  import taskRouter from '../routes/task.routes.js';
7  import emailRouter from '../routes/email.routes.js';
8
9  const app = express();
10 app.use(express.json());
11 app.use(express.urlencoded({ extended: true }));
12
13 // Connect to MongoDB
14 connectDB();// Call the connection function
15
16 // Use routes
17 app.use('/api', authRouter);
18 app.use('/api', userRouter);
19 app.use('/api', taskRouter);
20 app.use('/api', emailRouter);
21
22 app.use((req, res, next) => {
23   res.status(404).json({
24     message: 'Endpoint losses'
25   });
26 });
27
28 export default app;
```

RUTAS



Node.js - Email



Realizar pruebas

TOKEN

POST

http://localhost:3000/api/send-welcome

Params

Authorization

Headers (11)

Body

Scripts

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1 {

2 |

3 | "email": "diehercasvan@gmail.com",

4 | "name": "Diego Casallas"

5 }

Body

Cookies (1)

Headers (7)

Test Results

200 OK

5.91 s

251 B

{}

JSON

Preview

Visualize

1 {

2 |

3 | "success": true

4 }

RUTA

DATOS

RESPUESTA

TASK-MONGO-NODE

>

auth

>

user

>

task

>

Email

POST

sendEmail

>

TEST

MÉTODO	RUTA	CONTROLADOR	MODELO
POST	http://localhost:3000/api/send-welcome	Email	N/A

Node.js - Email



Welcome to My App

Recibidos x



My App <noreply@dendrite.com.co>
para mi ▼

14:06 (hace 16 minutos)



!Welcome, Diego Casallas!

Thank you for registering on our platform

← Responder

→ Reenviar





G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682



www.sena.edu.co