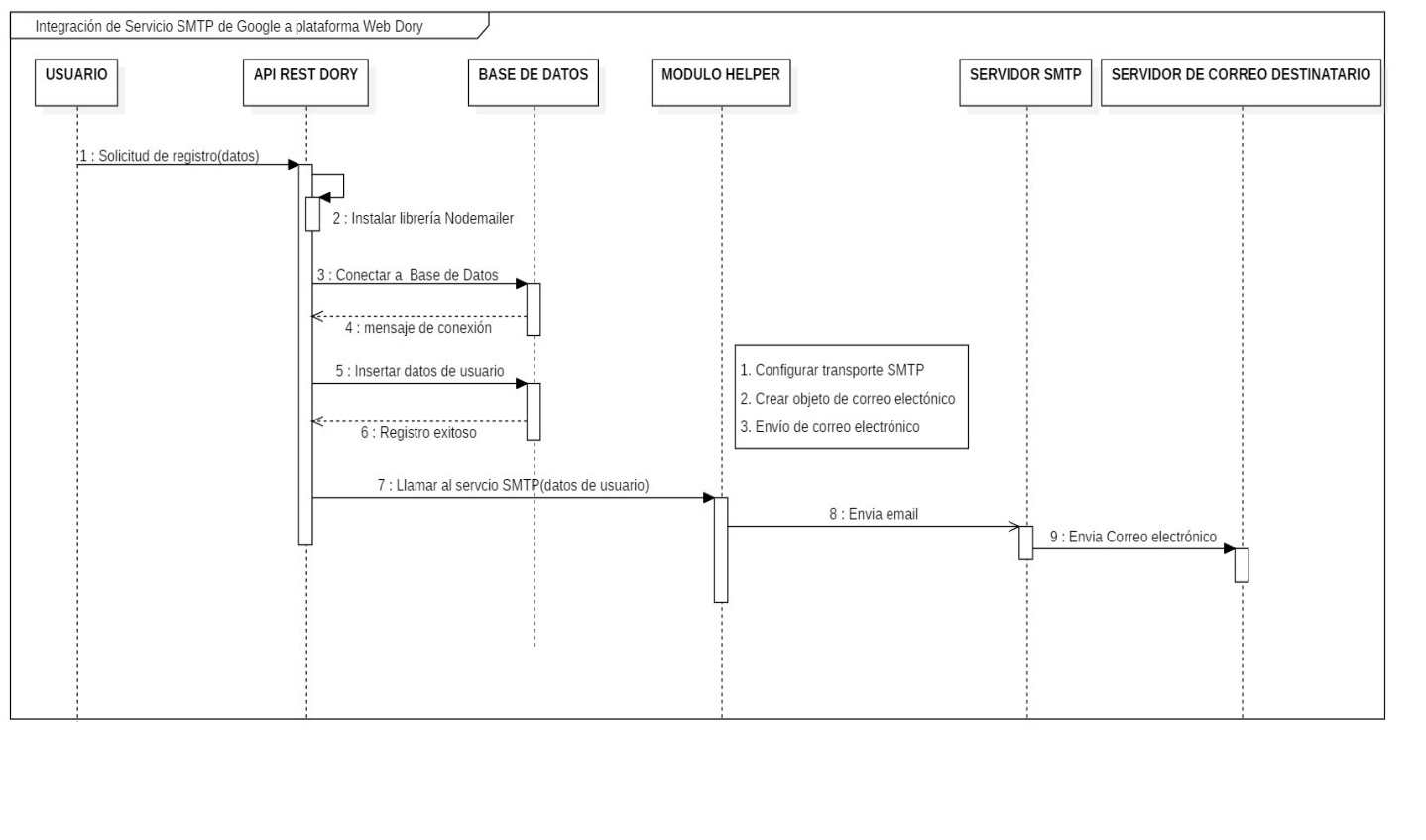
**Como integrar el servicio SMTP de Google en el api rest de la plataforma web Dory**

**Objetivo:** Integrar el servicio SMTP de Google en la plataforma web Dory.

**Procedimiento:**

**Recursos necesarios:**

* Node.js
* Visual Studio Code
* Express
* Cuenta de servicio de Google
* Api rest para la comunicación con la plataforma web y la base de datos
* Cliente de correo electrónico para recibir el mensaje de bienvenida

Piezas de software necesarias:

* Nodemailer

Ver Anexos (tecnologías y piezas de software)

**Pasos:**

1. Configurar el entorno de desarrollo. Ver (Como preparar el entorno de desarrollo).
2. Instalar piezas de software necesarias como: nodemailer (ver imagen 1 ).

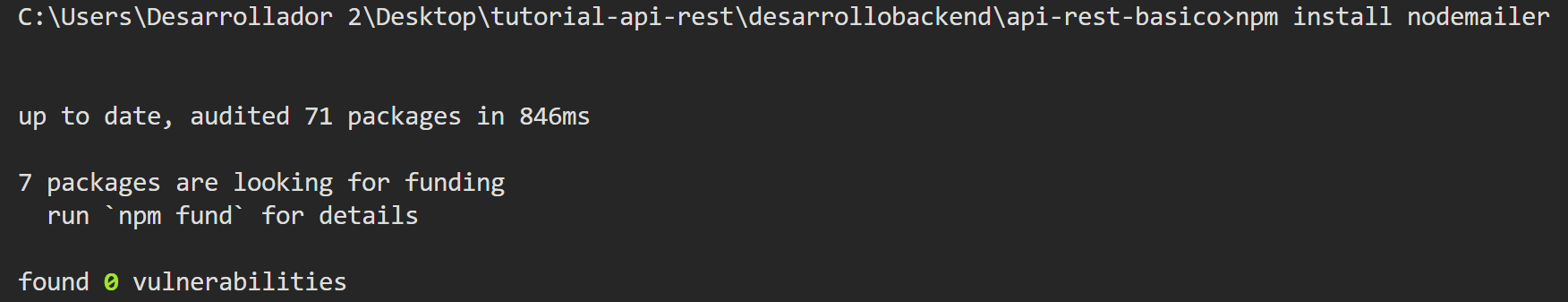


Imagen 1. *Instalación de modulo nodemailer*

1. Crear directorio “helper.js” donde se importará el módulo “nodemaider”, se implementará el servicio de envió del formulario SMTP y se exportará para poder usarlos en otros archivos.

* Importar librería “nodemailer” que permite interactuar con el servicio SMTP (ver imagen 2).



Imagen 2. *Importe de librería nodemailer*

* La function **sendEmail** se declara con tres parámetros “email” (dirección de correo electrónico), “tema” (Asunto del correo electrónico) y “contentHtml” (contenido Html del correo electrónico (ver imagen 3).

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 3. *Function sendEmail*

* Se crea un objeto “**transporter**” que específica los detalles de la conexión SMTP con el servidor de Gmail.

Texto

Descripción generada automáticamente

Imagen 4. *Objeto transporter*

* Se utiliza el objeto “transporter” para enviar el correo electrónico ( ver imagen 5)

Texto

Descripción generada automáticamente

Imagen 5. U*tilización del objeto transporter*

* *El campo "from" especifica la dirección de correo electrónico y el nombre del remitente que aparecerá en el correo electrónico.*
* *El campo "to" contiene la dirección de correo electrónico del destinatario.*
* *El campo "subject" especifica el asunto del correo electrónico.*
* *El campo "html" contiene el cuerpo del correo electrónico en formato HTML.*
* Se guarda la información del envío del correo electrónico en la variable "**info**" para cualquier procesamiento adicional o manejo de errores.
* Finalmente, se retorna un objeto que contiene un mensaje indicando que el enlace con el asunto especificado se ha enviado con éxito al correo electrónico proporcionado y se exporta el archivo para ser usado en otros.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 6. *Objeto de mensaje exitoso y exportación de la function sendEmail*

* Archivo “helper.js” donde se implementó el servicio SMTP (ver imagen 7)

Texto

Descripción generada automáticamente

Imagen 7. *Función de envió de formulario en el Archivo helper.js*

1. En nuestro archivo principal “index.js” se crea la ruta al endpoint para comunicarse con el servicio de registro del usuario donde se utilizará el servicio SMTP de Google (ver imagen 8).

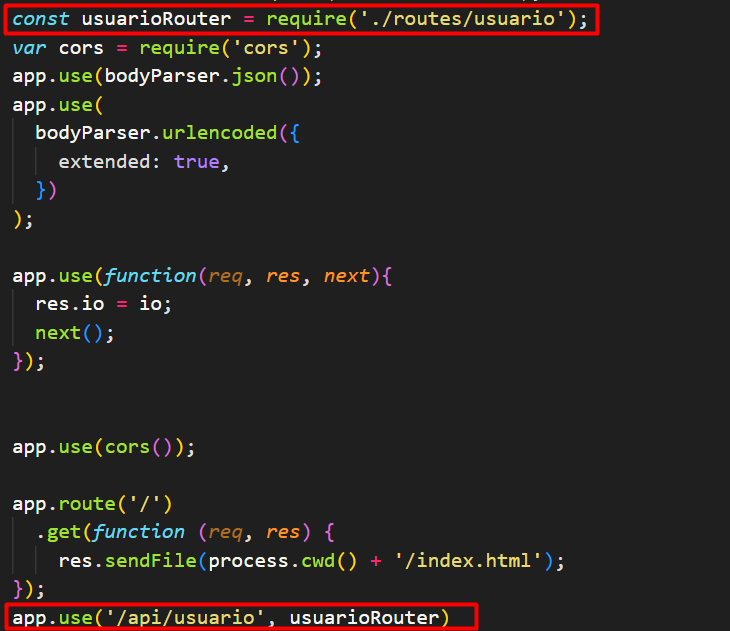


Imagen 8. *Creación de ruta al endpoint de usuario*

1. En nuestro archivo de rutas del usuario requerimos los servicios del usuario y en el endpoint de registro nos comunicamos con el servicio de registro (ver imagen 9).

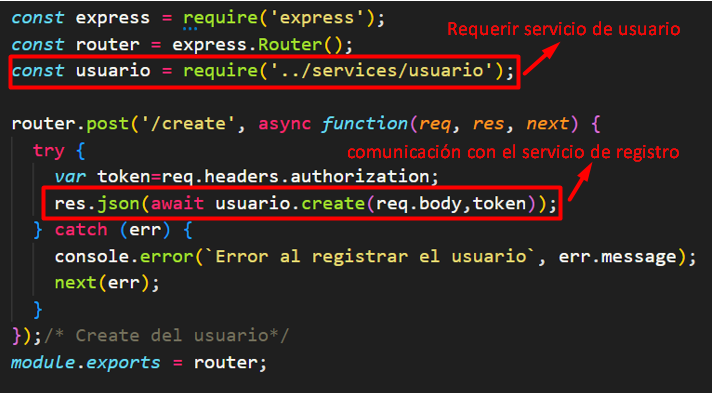


Imagen 9. *Archivo de rutas de usuario.js*

1. En nuestro archivo de servicios “usuario.js” requerimos el archivo “helper.js” donde configuramos las opciones SMTP (ver imagen 10) y enviamos nuestro mensaje a el correo electrónico con la función **sendEmail** (ver imagen 11).



Imagen 10. *Requerimiento de directorio “helper.js”*

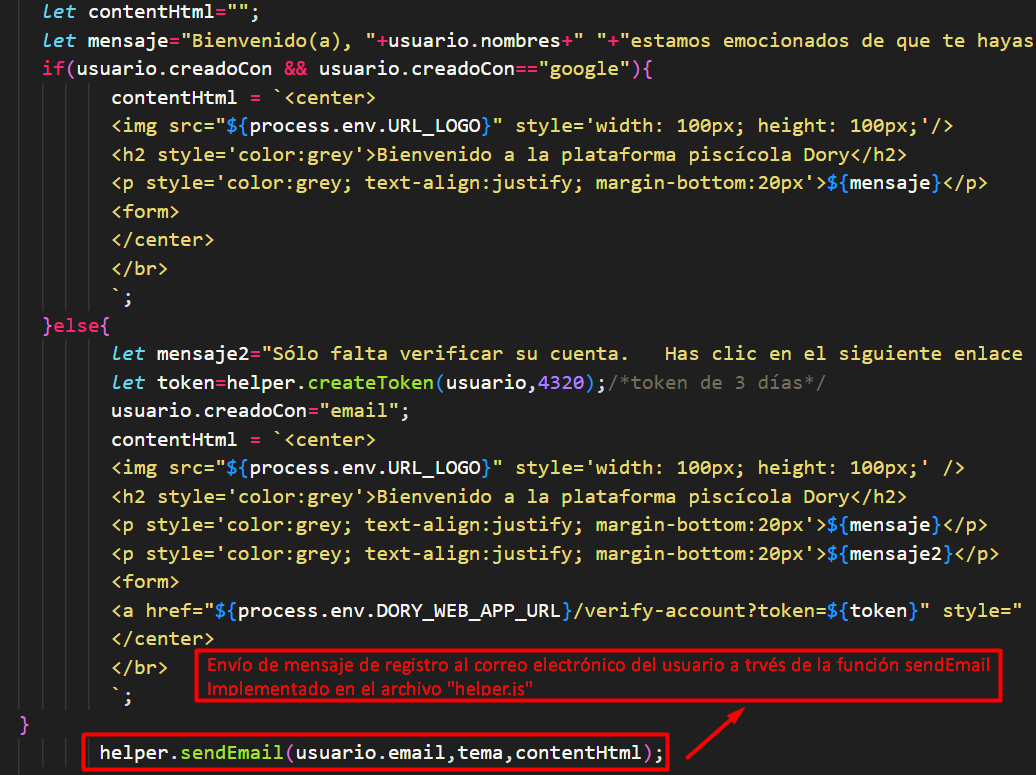


Imagen 11. *Envió de mensaje al correo electrónico*

1. Reiniciamos el servidor y probamos el servicio de registro en Postman (ver imagen 12).

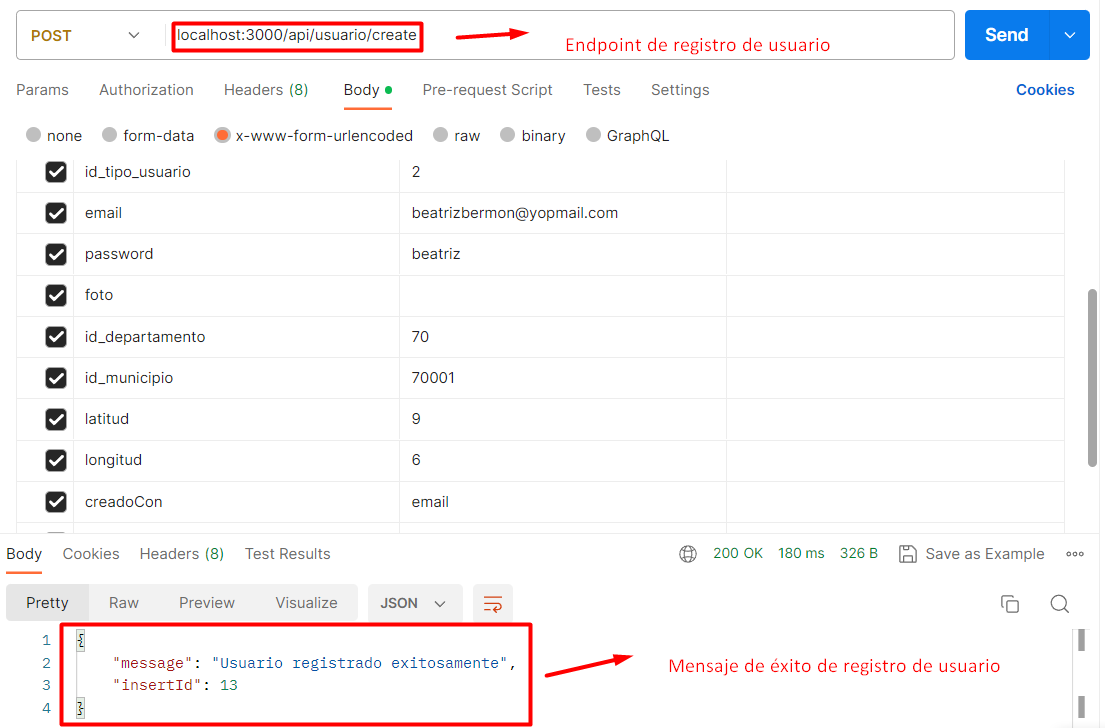


Imagen 12. *Pruebas del envío de mensajes*

1. Verificamos el envío del formulario inspeccionando el email del usuario que realizó el registro (ver imagen 13).



Imagen 13. *Inspección del email*