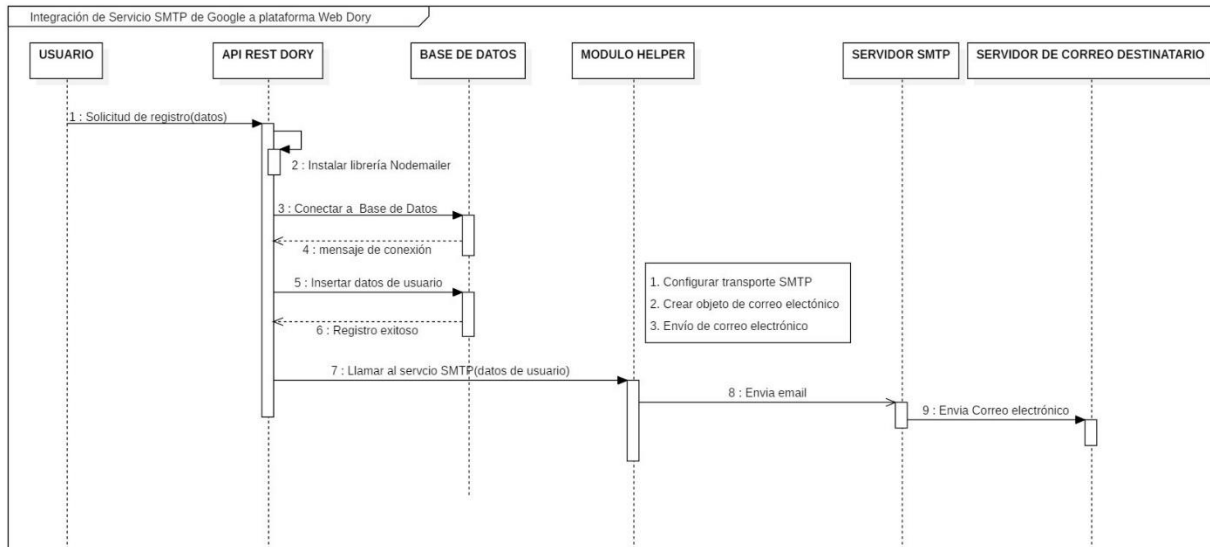


# Integración del servicio SMTP de Google en el api rest de la plataforma web Dory

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



## Recursos necesarios:

- Node.js versión 16.14.1 + npm versión 8.5.0
- Visual Studio Code versión 1.18.1
- Express versión 4.17.1
- 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 versión 6.7.2

Ver Anexos (tecnologías y piezas de software)

## Pasos:

1. Retomando la receta "Abrir Api Rest Dory", en el módulo "helper.js" se importa la librería "nodemailer", se implementa el servicio de envío del formulario SMTP y se exporta para poder usarlo en otros archivos.

- Importar librería "nodemailer" que permite interactuar con el servicio SMTP (ver imagen 1).

```
const nodemailer = require('nodemailer');
```

Imagen 1. 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 2).

```
function sendEmail(email,tema,contentHtml) {  
    //código para enviar el correo electrónico  
};
```

Imagen 2. Function sendEmail

- Se crea un objeto "**transporter**" que especifica los detalles de la conexión SMTP con el servidor de Gmail (ver imagen 3).

```
let transporter = nodemailer.createTransport({  
  host: "smtp.gmail.com",  
  port: 587,  
  secure: false, // true for 465, false for other ports --color gris #343A40  
  auth: {  
    user: "plataforma.piscicola@gmail.com", // generated ethereal user  
    pass: [REDACTED], // generated ethereal password  
  },  
});
```

Imagen 3. Objeto transporter

- Se utiliza el objeto "transporter" para enviar el correo electrónico (ver imagen 4)

```
// send mail with defined transport object
let info = transporter.sendMail({
  from: tema+' <plataforma.piscicola@gmail.com>', // user
  to: email, //email
  subject: tema, // Subject line
  html: contentHtml, // html body
});
```

Imagen 4. Utilización del objeto transporter

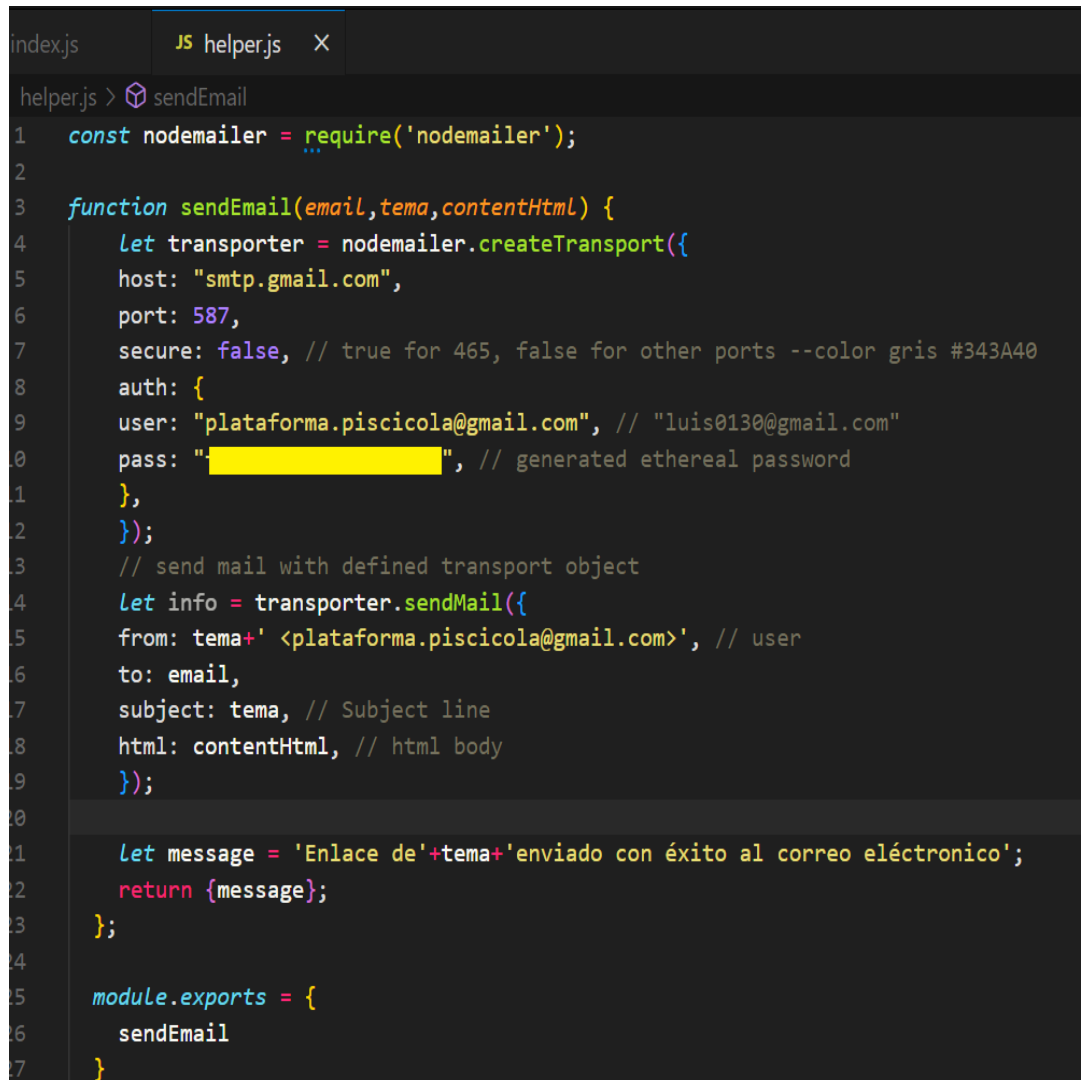
- El campo "from" especifica la dirección de correo electrónico y el nombre del remitente que aparecerá en el correo electrónico.
  - El campo "to" contiene la dirección de correo electrónico del destinatario.
  - El campo "subject" especifica el asunto del correo electrónico.
  - El campo "html" contiene 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 (ver imagen 5).

```
let message = 'Enlace de'+tema+'enviado con éxito al correo electrónico';
return {message};
};

module.exports = {
  sendEmail
}
```

Imagen 5. Objeto de mensaje exitoso y exportación de la función sendEmail

- Módulo "helper.js" donde se implementó el servicio SMTP (ver imagen 6).



```
index.js JS helper.js X
helper.js > sendEmail
1  const nodemailer = require('nodemailer');
2
3  function sendEmail(email,tema,contentHtml) {
4      let transporter = nodemailer.createTransport({
5          host: "smtp.gmail.com",
6          port: 587,
7          secure: false, // true for 465, false for other ports --color gris #343A40
8          auth: {
9              user: "plataforma.piscicola@gmail.com", // "luis0130@gmail.com"
10             pass: "██████████", // generated ethereal password
11         },
12     });
13     // send mail with defined transport object
14     let info = transporter.sendMail({
15         from: tema+' <plataforma.piscicola@gmail.com>', // user
16         to: email,
17         subject: tema, // Subject line
18         html: contentHtml, // html body
19     });
20
21     let message = 'Enlace de'+tema+'enviado con éxito al correo electrónico';
22     return {message};
23 };
24
25 module.exports = {
26     sendEmail
27 }
```

Imagen 6. Función de envío de formulario en el módulo helper.js

2. 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 7).

```
const usuarioRouter = require('./routes/usuario');
var cors = require('cors');
app.use(bodyParser.json());
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);

app.use(function(req, res, next){
  res.io = io;
  next();
});

app.use(cors());

app.route('/')
  .get(function (req, res) {
    res.sendFile(process.cwd() + '/index.html');
  });
app.use('/api/usuario', usuarioRouter)
```

Imagen 7. Creación de ruta al endpoint de usuario

3. 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 8).

```
const express = require('express');
const router = express.Router();
const usuario = require('./services/usuario');

router.post('/create', async function(req, res, next) {
  try {
    var token=req.headers.authorization;
    res.json(await usuario.create(req.body,token));
  } catch (err) {
    console.error(`Error al registrar el usuario`, err.message);
    next(err);
  }
}); /* Create del usuario*/
module.exports = router;
```

Imagen 8. Archivo de rutas de usuario.js

4. En nuestro archivo de servicios "usuario.js" requerimos el módulo "helper.js" donde configuramos las opciones SMTP (ver imagen 9) y enviamos nuestro mensaje a el correo electrónico con la función **sendEmail** (ver imagen 10).

```
const helper = require('./helper');
```

Imagen 9. Requerimiento del módulo "helper.js"

```

let contentHtml="";
let mensaje="Bienvenido(a), "+usuario.nombres+" "+"estamos emocionados de que te hayas
if(usuario.creadoCon == "google"){
  contentHtml = `<center>
    
    <h2 style='color:grey;'>Bienvenido a la plataforma piscícola Dory</h2>
    <p style='color:grey; text-align:justify; margin-bottom:20px;'>${mensaje}</p>
    <form>
  </center>
</br>
`;
}else{
  let mensaje2="Sólo falta verificar su cuenta. Has clic en el siguiente enlace
  let token=helper.createToken(usuario,4320);/*token de 3 días*/
  usuario.creadoCon="email";
  contentHtml = `<center>
    
    <h2 style='color:grey;'>Bienvenido a la plataforma piscícola Dory</h2>
    <p style='color:grey; text-align:justify; margin-bottom:20px;'>${mensaje}</p>
    <p style='color:grey; text-align:justify; margin-bottom:20px;'>${mensaje2}</p>
    <form>
    <a href="${process.env.DORY_WEB_APP_URL}/verify-account?token=${token}" style="
  </center>
  </br>
  `;
  Envío de mensaje de registro al correo electrónico del usuario a través de la función sendEmail
  Implementado en el archivo "helper.js"
  helper.sendEmail(usuario.email,tema,contentHtml);
}

```

Imagen 10. *Envío de mensaje al correo electrónico*

- Reiniciamos el servidor y probamos el servicio de registro en Postman (ver imagen 11).

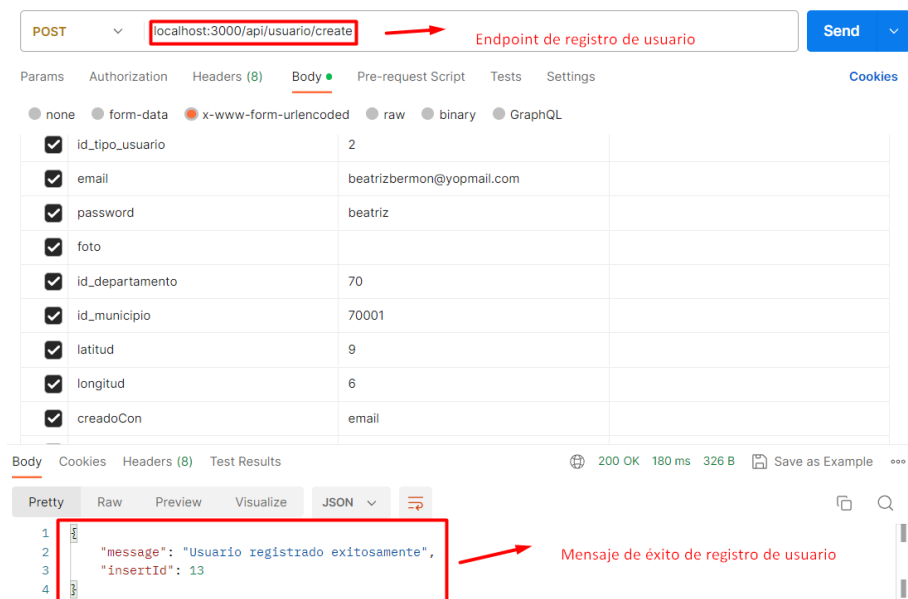


Imagen 11. *Pruebas del envío de mensajes*

- Verificamos el envío del formulario inspeccionando el email del usuario que realizó el registro (ver imagen 12).



## Bienvenido a la plataforma piscícola Dory

Bienvenido(a), Samuel estamos emocionados de que te hayas registrado con nosotros, somos un equipo conformado por emprendedores y profesionales que trabajan día a día para promover la productividad y competitividad de la cadena piscícola del Departamento de Sucre, en alianza con los grupos de investigación, Gestión de la Producción y la Calidad y GINTEING, de la Universidad de Sucre y la Corporación Universitaria Antonio José de Sucre.

Sólo falta verificar su cuenta. Has clic en el siguiente enlace para confirmar su correo electrónico

[Verificar cuenta de usuario](#)

Imagen 12. *Inspección del email*