

Universidad Catolica De
Honduras

Nuestra Señora Reina De La Paz

Campus Global



Investigación paquetes nodejs

Catedrático:

Ing. Carlos Flores

Alumno:

Roberto Carlos Castillo Castellanos

Sección:

1301

Fecha:

2/03/2022

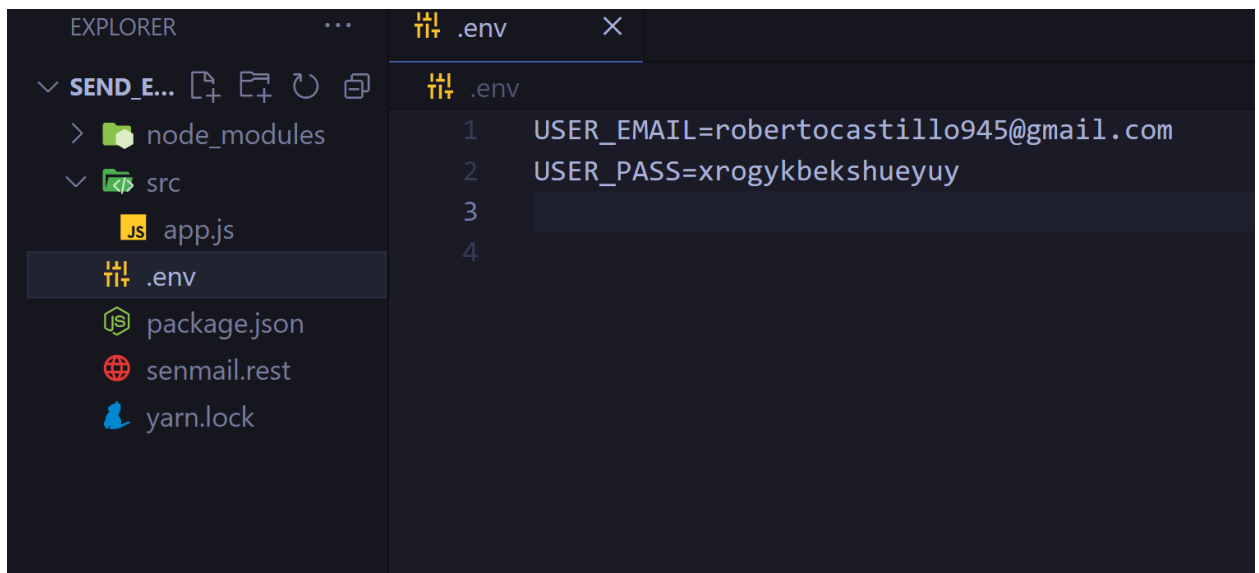
Índice

dotenv	2
jsonwebtoken	3
passport	4
passport-jwt	6
nodemailer	8

dotenv

Este es un paquete que nos ayuda a establecer variables de entorno, las variables de entorno nos ayudan a mantener información delicada privada, esta información es requerida en el código pero el lugar de escribirla directamente en el código la escribimos en un archivo llamado `.env` y desde el código hacemos referencia a la información almacenada en ese archivo como ser la dirección de una api y la clave para el acceso a esta también contraseñas de servicios como ser la contraseña a nuestra cuenta de gmail y el nombre de usuario.

Ejemplo de archivo `.env`

A screenshot of the Visual Studio Code editor interface. On the left, the 'EXPLORER' sidebar shows a file tree with folders 'node_modules' and 'src', and files 'app.js', '.env', 'package.json', 'senmail.rest', and 'yarn.lock'. The '.env' file is selected. The main editor area shows the content of the '.env' file with line numbers 1 through 4. The content is: '1 USER_EMAIL=robertocastillo945@gmail.com', '2 USER_PASS=xrogykbekshueyuy', '3', and '4'.

```
1 USER_EMAIL=robertocastillo945@gmail.com
2 USER_PASS=xrogykbekshueyuy
3
4
```

Ejemplo de acceso a las variables en el archivo `.env`

```
10 |  
11 | let transporter = nodemailer.createTransport({  
12 |   service: "gmail",  
13 |   auth: {  
14 |     user: process.env.USER_EMAIL,  
15 |     pass: process.env.USER_PASS,  
16 |   },  
17 | });
```

jsonwebtoken

json web token es un paquete que nos ayuda con la autorización de usuarios o manejo de sesiones, este paquete genera una clave única para darle autorización a un usuario esta clave se envía al usuario y es almacenada en la aplicación cliente no en el servidor generalmente las sesiones se almacenan en el servidor pero con json web token se podría decir que el dominio de la sesión la tiene la aplicación cliente dado que siempre y cuando el usuario no borre o pierda su token o clave seguirá teniendo acceso a los recursos que el servidor proporciona a menos que el token tenga una fecha de expiración si el token expira el usuario pierde el acceso. Los web token se pueden configurar con:

- Un tiempo de expiración
- Una carga útil o payload el cual contiene información del usuario como ser id de usuario, email, etc.

Ejemplo de uso:

```
1  const { sign } = require("jsonwebtoken");
2
3  function genAccessToken(user) {
4    return sign(user, process.env.ACCESS_TOKEN_SECRET, {
5      expiresIn: "10m",
6    });
7  }
8
9  function genRefreshToken(user) {
10   return sign(user, process.env.REFRESH_TOKEN_SECRET);
11 }
```

passport

Passport es un paquete que nos ayuda a manejar sesiones de usuarios en nuestra aplicación

passport tiene muchas estrategias o modos de manejo de sesiones tales como:

- local: las contraseñas y usuarios se guardan y manejan de forma local en nuestra aplicación esto quiere decir que nos encargamos de gran parte del proceso como encriptar contraseñas y guardar los usuarios a la base de datos.
- externa: facebook, google u otra empresa se encarga de manejar las contraseñas
- jwt: usa json web tokens para autorizar usuarios

Ejemplo de uso

```
40 passport.use(  
41   new LocalStrategy(  
42     async (username, password, done) => {  
43       try {  
44         const user = await User.findOne({ username })  
45  
46         if (!user) {  
47           return done(null, false);  
48         }  
49         if (!user.verifyPassword(password)) {  
50           return done(null, false);  
51         }  
52         return done(null, user);  
53       } catch (error) {  
54         return done(error);  
55       }  
56     })  
57   )  
58 )
```

```
59   app.post(  
60     "/login",  
61     passport.authenticate(  
62       "local",  
63       { failureRedirect: "/login" }  
64     ),  
65     (req, res)=> {  
66       res.redirect("/");  
67     }  
  )
```

passport-jwt

Ejemplo de uso

```
42  const {
43    Strategy,
44    ExtractJwt: { fromAuthHeaderAsBearerToken },
45  } = require("passport-jwt");
46
47  let opts = {
48    jwtFromRequest: fromAuthHeaderAsBearerToken(),
49    secretOrKey: "secret",
50    issuer: "accounts.examplesoft.com",
51    audience: "yoursite.net",
52  };
```

```
54  passport.use(
55    new Strategy(
56      opts,
57      async (jwt_payload, done) => {
58        let {sub} = jwt_payload
59        try {
60          const user = await User.findOne({ id:sub })
61          return done(null, user || false);
62        } catch (error) {
63          return done(error, false);
64        }
65      })
66  );
```

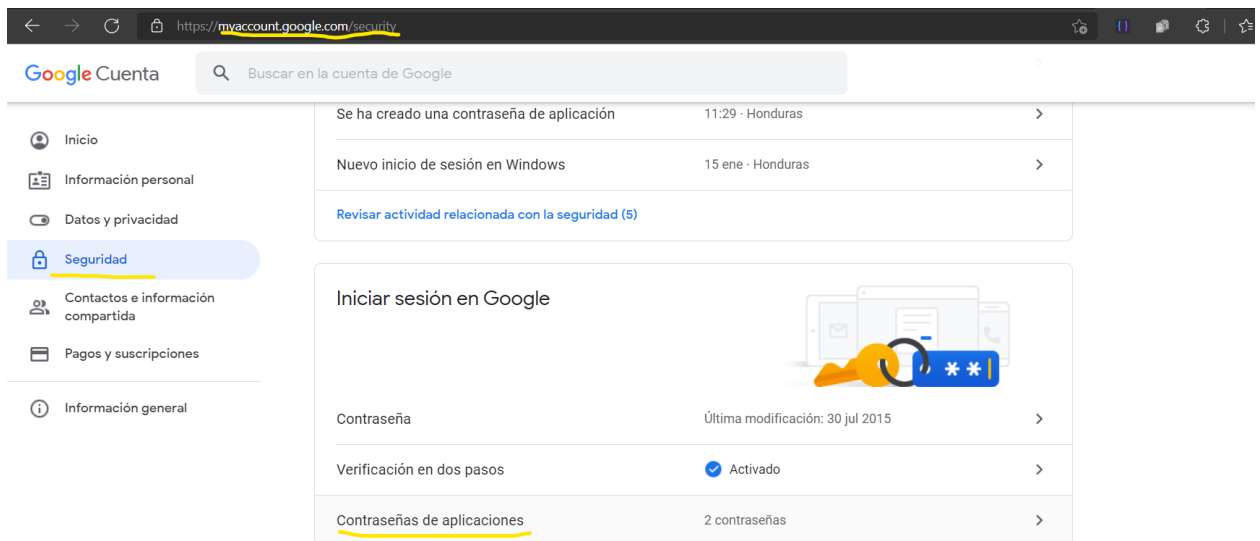


```
8   app.post(  
9     "/profile",  
0     passport.authenticate("jwt", { session: false }),  
1     (req, res) => {  
2       res.send(req.user.profile);  
3     }  
4   );
```


nodemailer

Este es un paquete que nos ayuda con el envío de correos a los usuarios podemos conectarlo con nuestro servidor de correo o con gmail. para conectarte con gmail debemos crear credenciales de aplicación

Crear credenciales de aplicación.



Tus contraseñas de aplicación

Nombre	Fecha de creación	Último uso	
nodejsrobertcastillo	11:31	11:39	

Selecciona la aplicación y el dispositivo para los que quieres generar la contraseña de aplicación.

Seleccionar aplicación ▼

Seleccionar dispositivo

iPhone

iPad

BlackBerry

Mac

Windows Phone

Ordenador con Windows

Otra (*nombre personalizado*)

GENERAR

← Contraseñas de aplicaciones

Las contraseñas de aplicación te permiten acceder a tu sesión en tu cuenta de Google desde aplicaciones

instaladas en dispositivos que no admiten la verificación en dos pasos. No tendrás que recordarlas porque solo tienes que introducirlas una vez. [Más información](#)

Tus contraseñas de aplicación

Nombre

Fecha de creación

Último uso

nodejsrobertcastillo

11:31

11:39

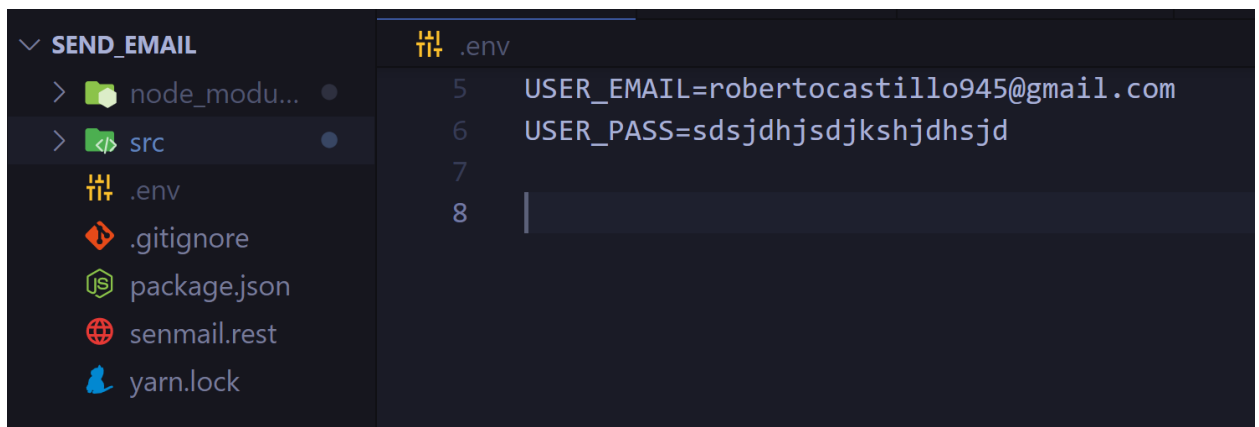


Selecciona la aplicación y el dispositivo para los que quieres generar la contraseña de aplicación.

nodemailerrobertcastillo X

GENERAR

A continuación nos dará una clave la cual debemos usar en nuestra aplicación.



Ejemplo de uso.

```
9   const nodemailer = require("nodemailer");
10
11   let transporter = nodemailer.createTransport({
12     service: "gmail",
13     auth: {
14       user: process.env.USER_EMAIL,
15       pass: process.env.USER_PASS,
16     },
17   });
```

```
app.post("/", async (req, res) => {
  const { sender, from, to, subject, text, html } = req.body;

  try {
    let info = await transporter.sendMail({
      from: `"${sender}" <${from}>`,
      to,
      subject,
      text,
      html,
    });
    res.json({ info, msg: "Mail sent" });
  } catch (error) {
    throw new Error(error);
  }
})
```

```
Send Request
4 POST http://localhost:3000/
5 Content-Type: application/json
6
7 {
8   "sender": "Robert Castillo 🐼",
9   "from": "robertocastillo945@gmail.com",
10  "to": "robertocastill63@gmail.com",
11  "subject": "Hello ✓",
12  "text": "Hello world?",
13  "html": "<b>Hello world?</b>"
14 }
```

Hello ✓

Recibidos x



Robert Castillo 🐼 <robertocastillo945@gmail.com>

para mí ▼



inglés ▼



español ▼

[Traducir mensaje](#)

Hello world?



Responder



Reenviar

