



**Clase 35.** Programación Backend

# ***Envío de mensajes y Seguridad***



## ***OBJETIVOS DE LA CLASE***

- Conocer el módulo Nodemailer para el envío de mails desde una App de Node.
- Enviar mails con un servidor sin autenticación y con Gmail.
- Conocer el módulo Twilio para el envío de SMS desde una App de Node.

# ***CRONOGRAMA DEL CURSO***

Clase 34



**Productos Cloud:AWS**

Clase 35



**Envío de mensajes y  
Seguridad**

Clase 36



**Twilio & OWASP**

***NODEMAILER***

# *¿De qué se trata?*



- **Nodemailer** es un módulo para aplicaciones Node que permite el envío de correos electrónicos de forma sencilla.
- El proyecto comenzó en 2010 cuando no había una opción sensata para enviar mensajes de correo electrónico.
- Hoy es la solución a la que la mayoría de los usuarios de Node recurren por defecto.



# ***Algunas características***



- Implementación en un solo módulo con cero dependencias.
- Gran foco en la seguridad.
- Proxies para conexiones SMTP.
- Compatibilidad con Windows: se instala con NPM como cualquier módulo.
- Compatibilidad con contenido HTML y la alternativa de texto sin formato.
- Posibilidad de agregar archivos adjuntos a los mensajes.
- Entrega segura de correo electrónico utilizando TLS / STARTTLS.
- Diferentes métodos de transporte además del soporte SMTP incorporado.
- Autenticación Sane OAuth2.

# ***ENVIAR MAILS CON SERVIDOR ETHEREAL***

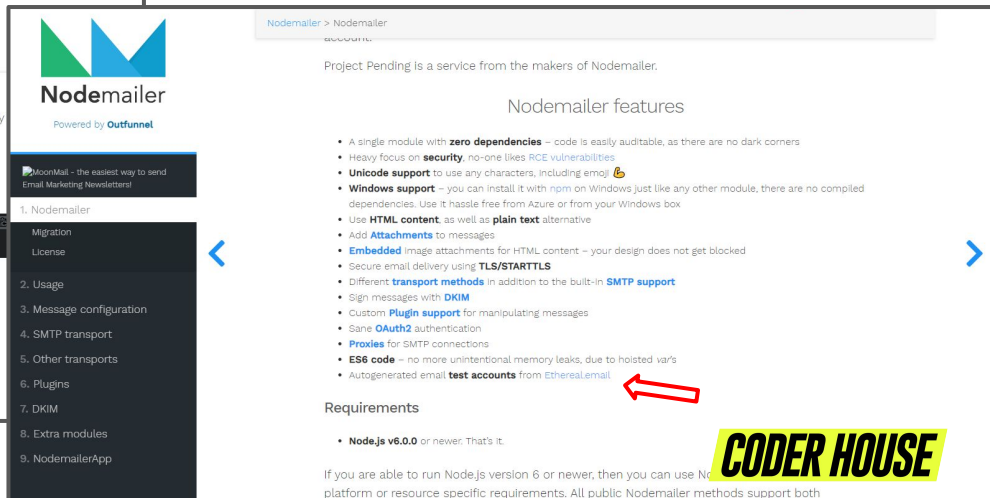
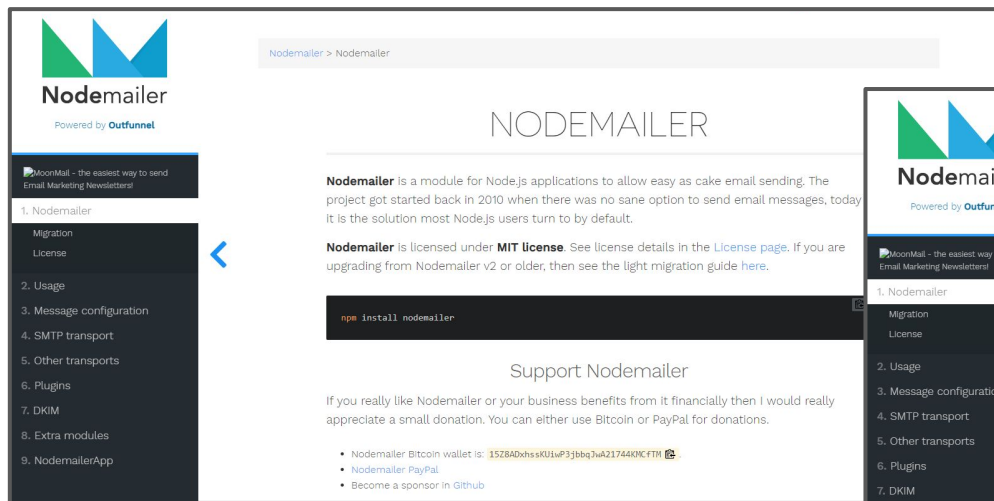


# Comenzando con Nodemailer

Ejemplo  
en vivo



1. Ingresamos a la página de la documentación de Nodemailer desde <https://nodemailer.com>.
2. Nos dirigimos a [Ethereal.email](#) (servidor SMTP sin autenticación), donde crearemos una cuenta de mail de testing para luego probar el módulo Nodemailer en nuestro proyecto.



Bajemos en la página



**CODER HOUSE**





# Comenzando con Nodemailer

Ejemplo  
en vivo



- Una vez allí, clickeamos en el botón “Create a Ethereum Account” para crear una cuenta..
- Se genera la cuenta con la que podremos abrirlo desde el botón “Open mailbox”.

Ethereal Home FAQ Help Messages Logout

## Ethereal

[+ Create Ethereum Account](#) [Subscribe to newsletter](#)

Ethereal is a fake SMTP service, mostly aimed at Nodemailer users (but not limited to). It's a completely free anti-transactional email service where messages never get delivered.

Instead, you can generate a vanity email account right from Nodemailer, send an email using that account just as you would with any other SMTP provider and finally preview the sent message here as no emails are actually delivered. So far Ethereal has caught **73 730 015** emails sent using Ethereal testing accounts.

### IMAP/POP3

Ethereal addresses act as real IMAP/POP3 accounts. You can access sent and received messages using your favorite email client, whatever that happens to be.

### Inbound

The same account acts as a real inbound email address. You can send emails to that address from anywhere and these messages also end up on the account mailbox.

### Automatic

All account registration can be handled by the Nodemailer API. If you have lost authentication details for a previous account, then just create a new one.

Messages caught in the last 15 days

Feedback



Ethereal Home FAQ Help Messages Logout

## Account created

Here you can find all the details needed to access your new Ethereum test account. Remember that if sending messages through SMTP then no message is actually delivered, all messages are caught and you can see these in the [Messages](#) page or by using your favorite IMAP/POP3 client.

### Account credentials

Account details generated using [faker.js](#)

NB! these credentials are shown only once. If you do not write these down then you have to create a new account.

Name	Ford Blanda
Username	ford.blanda@ethereal.email (also works as a real inbound email address)
Password	ERsp2fauH37t24hsg

[Download as CSV](#) [Open Mailbox](#)

### Nodemailer configuration

```
const transporter = nodemailer.createTransport({
  host: 'smtp.ethereal.email',
  port: 587,
  auth: {
    user: 'ford.blanda@ethereal.email',
    pass: 'ERsp2fauH37t24hsg'
  }
});
```

DHDMailer configuration

**CODER HOUSE**

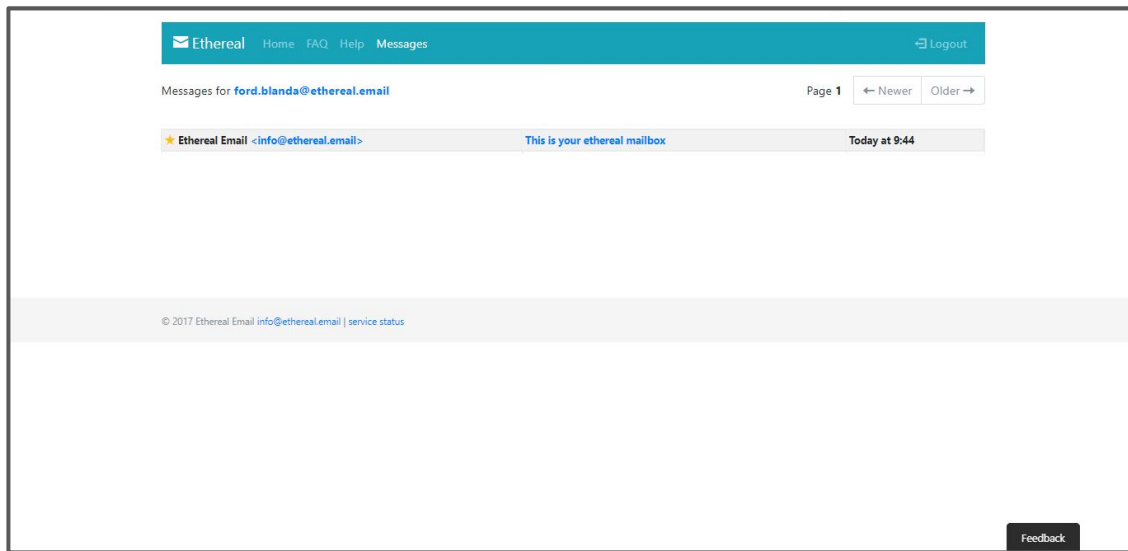


# Comenzando con Nodemailer

Ejemplo  
en vivo



5. Podremos ver en el mailbox entonces los mails que enviemos desde nuestra aplicación una vez que comencemos a usarlo.





# ***Configurar Nodemailer***

Ejemplo  
en vivo



1. Instalar el módulo de Nodemailer como dependencia de nuestro proyecto:

```
$ npm install nodemailer
```

2. Luego, configuramos los mails en el archivo index.js de nuestro proyecto con el código que vemos a continuación.
3. Debemos configurar un transporte donde se indica el puerto, el host (Ethereal en este caso) y el objeto Auth.



# ***Configurar Nodemailer***

Ejemplo  
en vivo



Los datos de user y pass del objeto auth del transporte, los obtenemos de la cuenta de Ethereum que nos creamos.

```
import { createTransport } from 'nodemailer';

const TEST_MAIL = 'xxxxxxx@ethereum.email'

const transporter = createTransport({
  host: 'smtp.ethereum.email',
  port: 587,
  auth: {
    user: TEST_MAIL,
    pass: 'xxxxxxx'
  }
});
```



# ***Configurar Nodemailer***

Ejemplo  
en vivo



- Configuramos las opciones de mail donde indicamos quién lo envía, a qué direcciones, el asunto del mail y el cuerpo del mismo (puede ser texto plano o HTML).

```
const mailOptions = {  
  from: 'Servidor Node.js',  
  to: TEST_MAIL,  
  subject: 'Mail de prueba desde Node.js',  
  html: '<h1 style="color: blue;">Contenido de prueba desde <span  
style="color: green;">Node.js con Nodemailer</span></h1>'  
}
```



# ***Configurar Nodemailer***

Ejemplo  
en vivo



- Para enviar finalmente el mail, utilizamos el método *sendMail* del transporter con las opciones como argumento. Este método devuelve una promesa.

```
try {  
  const info = await transporter.sendMail(mailOptions)  
  console.log(info)  
} catch (error) {  
  console.log(err)  
}
```

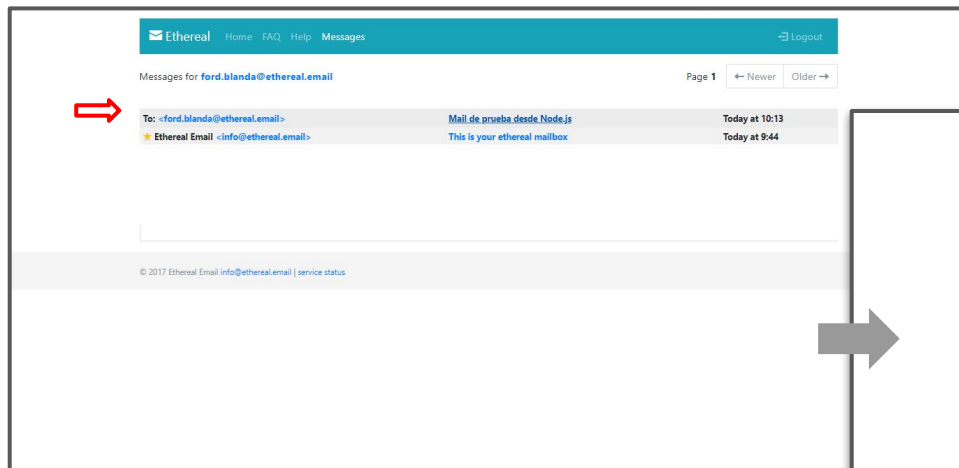


# Configurar Nodemailer

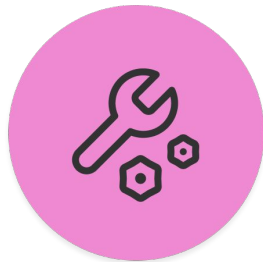
Ejemplo  
en vivo



- Si ejecutamos el código anterior, y volvemos al mailbox de Ethereum, podemos ver el mail de ejemplo enviado.



**CODER HOUSE**



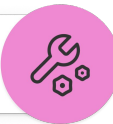
# ***ENVIAR MAIL DESDE NODE CON ETHEREAL***

*Tiempo: 10 minutos*



# ***Enviar mail desde Node con Ethereum***

Desafío  
generico



*Tiempo: 10 minutos*

Realizar un módulo Node.js llamado '/enviarmail', que cada vez que se lo ejecute envíe un correo electrónico hacia una cuenta Ethereum (<https://ethereum.email/>) creada para tal fin.

- El asunto del mail y el mensaje a enviar se recibirán como body de la request.
- El mensaje podrá tener formato HTML.
- Se representará por consola el resultado de la operación.
- Verificar en la cuenta de Ethereum que se haya recibido el mail con el asunto y contenido correspondiente.

***ENVIAR MAILS USANDO GMAIL***



# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



- En la documentación de Nodemailer, podemos ir al [apartado de Gmail](#) donde vamos a tener información de cómo utilizarlo.

The screenshot shows the Nodemailer documentation interface. On the left is a dark sidebar with the Nodemailer logo and a list of navigation items: 'MoonMail - the easiest way to send Email Marketing Newsletters!', '1. Nodemailer', '2. Usage' (with sub-items 'SMTP? Say what?' and 'Using Gmail'), 'Delivering bulk mail', '3. Message configuration', '4. SMTP transport', '5. Other transports', '6. Plugins', '7. DKIM', '8. Extra modules', and '9. NodemailerApp'. The main content area is titled 'USING GMAIL' and contains text explaining Gmail's security measures and how to configure Nodemailer for OAuth2 authentication. It mentions that Gmail blocks 'less secure' apps by default and provides instructions on how to enable OAuth2 or use an application-specific password. The text also notes the daily sending limits of Gmail (500 recipients per day).

**CODER HOUSE**



# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



Google Account



← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

## Your app passwords

Name	Created	Last used	
coderhouse	7:06 PM	7:11 PM	

Select the app and device you want to generate the app password for.

Select app



Select device



GENERATE

- Para que nos pueda funcionar Nodemailer con Gmail, una solución propuesta es crear una contraseña alternativa para usar solo en nuestra aplicación. Para generarla, accedemos a esta página: <https://security.google.com/settings/security/apppasswords>

**CODER HOUSE**



# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



- En el transporter, en lugar del campo *host* configuramos un campo *service* como “gmail” y luego el objeto *auth* con el *user* y *pass* generados para nuestra cuenta de Gmail.
- Las opciones de envío son iguales a las del ejemplo anterior

```
const transporter = createTransport({  
  service: 'gmail',  
  port: 587,  
  auth: {  
    user: 'mimaildegmail@gmail.com',  
    pass: 'mipassdegmail'  
  }  
});
```

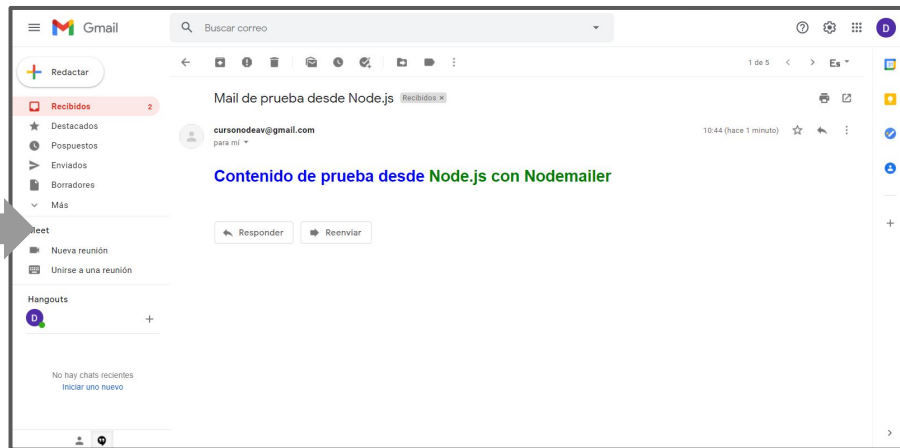
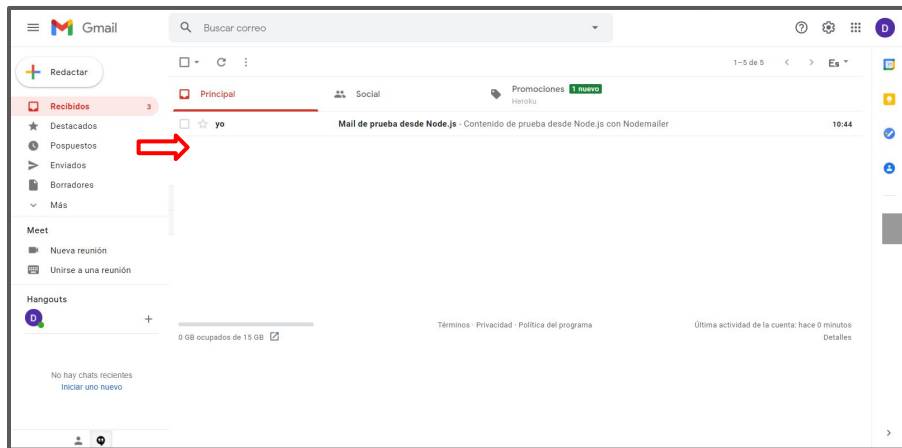


# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



- Una vez que ejecutamos el código podemos ir a la cuenta de Gmail a la que le enviamos el mail y chequear que haya llegado correctamente.





# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



- Podemos agregarle un archivo adjunto al mail que enviamos desde Node.
- Para eso, agregamos en el objeto *mailOptions* la clave de *attachments* y dentro la ruta del archivo a adjuntar.

```
index.js > mailOptions > html
3 const transporter = nodemailer.createTransport({
4   service: 'gmail',
5   auth: {
6     user: 'cursonodeav@gmail.com',
7     pass: '...',
8   }
9 })
10
11 const mailOptions = {
12   from: 'Servidor Node.js',
13   to: 'cursonodeav@gmail.com',
14   subject: 'Mail de prueba desde Node.js',
15   html: '<h1 style="color: blue;">Contenido de prueba con archivos adjuntos desde <span style="color: green;"',
16   attachments: [
17     { // filename and content type is derived from path
18       path: 'nodemailer.png'
19     }
20 ]
21 }
22
23 hi
```

```
PS C:\Cursos\Coderhouse\CursoBackend\Clase35\nodeMailerGmail> node .\index.js
{
  accepted: [ 'cursonodeav@gmail.com' ],
  rejected: [],
  envelopeTime: 603,
  messageTime: 721,
  messageSize: 6473,
  response: '250 2.0.0 OK 1621000300 44sm4602888qtb.45 - smtp',
  envelope: { from: '', to: [ 'cursonodeav@gmail.com' ] },
  messageId: '<3435815d-3b49-bc5e-4af2-4feb1f5ae27@Instructor>'
}
```

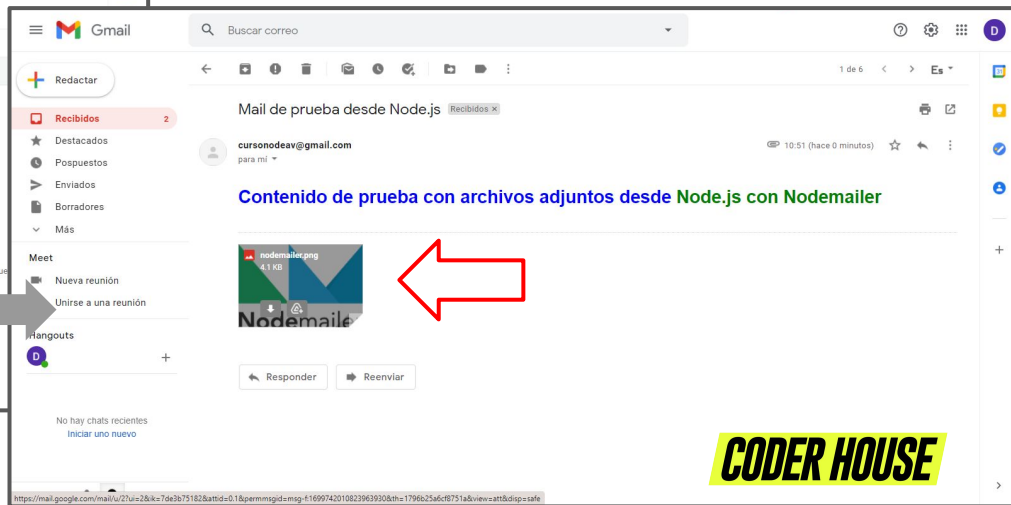
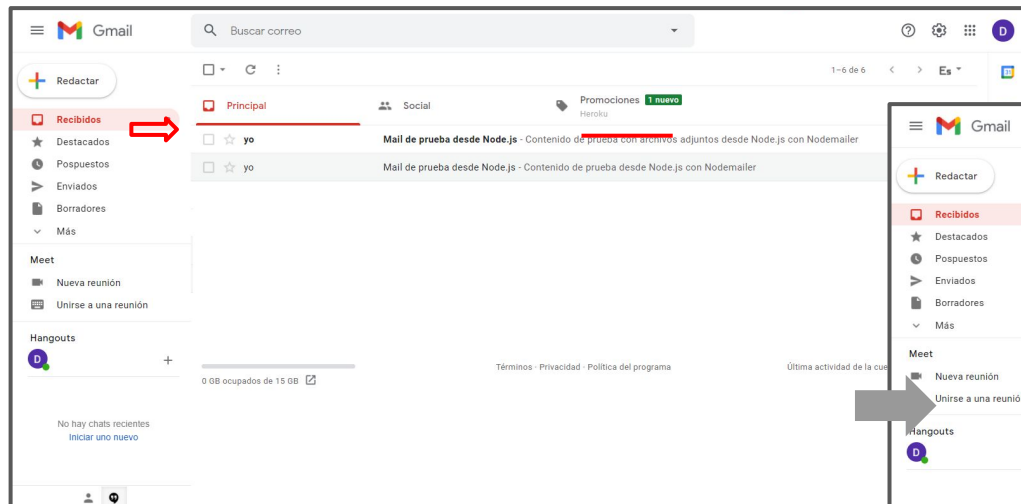


# Configurar Nodemailer con Gmail

Ejemplo  
en vivo



- Ejecutamos nuevamente el código con este cambio y volvemos a la cuenta de Gmail y vemos que llegó este segundo mail con el archivo que adjuntamos.



**CODER HOUSE**



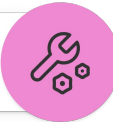


# ***ENVIAR MAIL DESDE NODE CON GMAIL***

*Tiempo: 10 minutos*

# Enviar mail desde Node con Gmail

Desafío  
generico



*Tiempo: 10 minutos*

Al ejercicio anterior agregarle un tercer parámetro que indique la cuenta email destino, y un cuarto parámetro opcional, que permita en caso de pasarlo, adjuntar un archivo local (mediante path) al email enviado.

- Se utilizará Gmail como servidor de correo por parte de nodemailer.
- Comprobar en ambos casos que los correos lleguen a la cuenta indicada y la información corresponda.

**NOTA:** crear una cuenta de gmail a tal fin y habilitar el acceso a aplicaciones poco seguras.



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

# ***TWILIO SMS***



# ¿De qué se trata?



- **Twilio** es un servicio de comunicación en la nube que permite un sin fin de procesos como lo son: enviar y recibir sms, enviar y recibir llamadas de voz, enviar y recibir llamadas de video y muchos más.
- Actúa como intermediario, ofreciendo un WebHook HTTPS para que se envíe una solicitud POST a la URL que queramos cada vez que se reciba un mensaje y responder en esa misma solicitud.
- El módulo de Node *twilio* ayuda a escribir el código de los *request* HTTP a la API de Twilio.
- Con el registro en la página, Twilio nos regala 15 USD para comenzar a probar sus servicios.

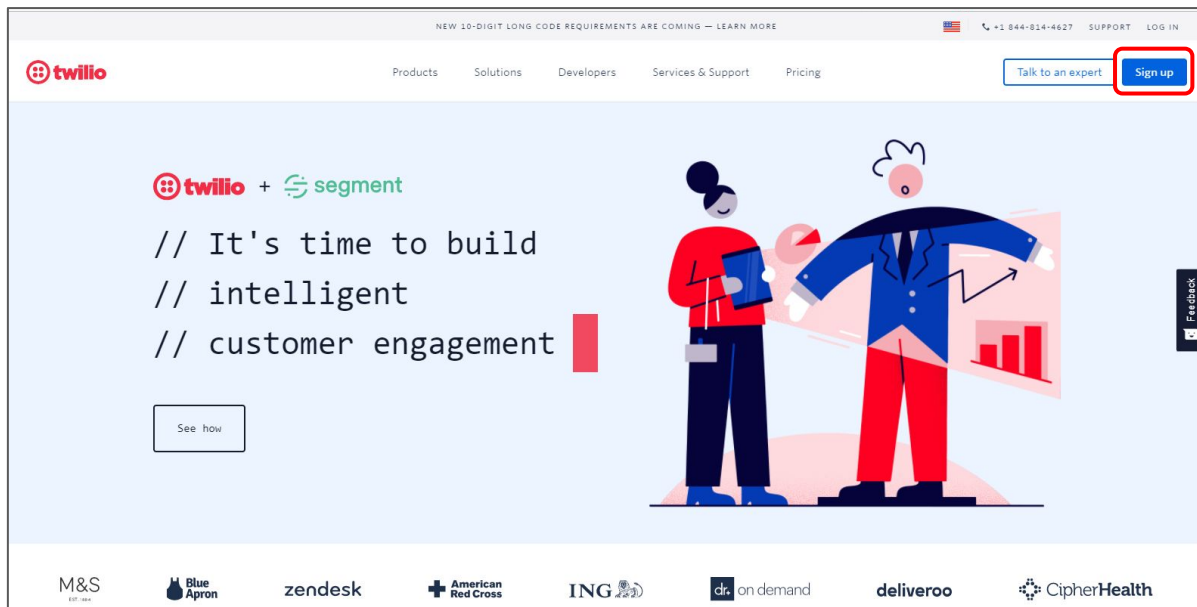


# Comenzando con Twilio

Ejemplo  
en vivo



Ingresamos en [Twilio](#) para comenzar a crear una cuenta desde “Sign up”.



**CODER HOUSE**



# Comenzando con Twilio

Ejemplo  
en vivo



Completamos los datos e iniciamos nuestro período de prueba gratis. Luego, vamos al [login](#) e iniciamos sesión con el mail con que nos registramos.

The image shows two overlapping screenshots of the Twilio website. The background screenshot is the 'Get started with a free Twilio account' page, which includes a list of services (SMS marketing, Omnichannel contact center, Call tracking, Web chat, Push notifications, Alerts and notifications, Phone verification) and a registration form with fields for First Name, Last Name, Email, and Password. A red box highlights the 'Start your free trial' button. The foreground screenshot is the 'Log in' page, which has a red box around the email input field and the 'Next' button. A grey arrow points from the 'Start your free trial' button to the 'Log in' page. The bottom right corner features the 'CODER HOUSE' logo.

Get started with a free Twilio account.  
No credit card required.

WITH TWILIO YOU CAN BUILD:

- ✓ SMS marketing
- ✓ Omnichannel contact center
- ✓ Call tracking
- ✓ Web chat
- ✓ Push notifications
- ✓ Alerts and notifications
- ✓ Phone verification

First Name \*

Last Name \*

Email \*

Password (14+ Characters) \*

☐ I accept the [Twilio Terms of Service](#) and have read the [Twilio Privacy Statement](#). If I am a micro- or small enterprise or a not-for-profit organisation in the EEA or UK, I agree to the [European Electronic Communications Code Rights Waiver](#).

Start your free trial

Log in

Email

Next

Don't have an account yet? [Sign up for free.](#)

Send Beautiful Transactional Emails at Scale

Whether you're sending 100 emails or 100 billion—Twilio SendGrid delivers.

Twilio SendGrid's flexible Email API delivers your transactional email in seconds, without interruption. Plus, the intuitive console and native support for Handlebars syntax make personalization easy.

Try it for free

© 2021 Twilio Inc. All rights reserved. [Privacy Policy](#) | [Terms of Service](#)

**CODER HOUSE**



# Comenzando con Twilio

Ejemplo  
en vivo




Vemos que como prueba, Twilio nos da una cantidad de USD para usar para SMS y un número que es el número de salida. Este es el número que el usuario verá cuando le llegue el mensaje, el número desde el que se envía.

DevelopMonitor

No pinned products yet!  
[Explore Products](#)

Docs and Support

mariano.aquino@gmail.com's Account Dashboard



Hi there! Want to get an app running with no code?  
Check out our most popular use cases

See app samples ↗

Project Info

TRIAL BALANCE

\$21.62

TRIAL NUMBER

+14156884237

Need more numbers?

REFERRAL PROGRAM

Refer your network to Twilio — give \$10, get \$10.  
[Referral Dashboard](#)

ACCOUNT SID

AC429651f99e462fbfa8dcd49223746d53

AUTH TOKEN

Show

PROJECT NAME

Here's how your Twilio Trial account works:

- You can send messages and make calls to [verified numbers](#).
- Messages and calls include a note about this coming from a "Twilio trial account."

Learn more about [your trial](#) or [upgrade](#) to remove restrictions.





# Comenzando con Twilio

Ejemplo  
en vivo




Luego, vamos a *verified numbers* y vemos los números que están verificados para poder enviarles mensajes desde nuestra cuenta de prueba.

DevelopMonitor

No pinned products yet!  
[Explore Products](#)

Docs and Support

## mariano.aquino@gmail.com's Account Dashboard



Hi there! Want to get an app running with no code?  
Check out our most popular use cases

See app samples ↗

Project Info

TRIAL BALANCE

\$21.62

TRIAL NUMBER

+14156884237

Need more numbers?

REFERRAL PROGRAM

Refer your network to Twilio — give \$10, get \$10.  
[Referral Dashboard](#)

ACCOUNT SID

AC429651f99e462fbfa8dcd49223746d53

AUTH TOKEN

Show

PROJECT NAME

Here's how your Twilio Trial account works:

- You can send messages and make calls to **verified numbers**.
- Messages and calls include a note about this coming from a "Twilio trial account."

Learn more about [your trial](#) or [upgrade](#) to remove restrictions.

**CODER HOUSE**



# Comenzando con Twilio

Ejemplo  
en vivo



Acá podemos ingresar un número de teléfono e invitarlo a colaborar con nuestra cuenta. Recibirá un código, que debemos ingresar aquí para confirmar la verificación. A continuación ya podemos enviarle SMS desde Twilio.

Develop

Monitor

No pinned products yet!  
[Explore Products](#)

## Verified Caller IDs

Add a new Caller ID

Verify a number that you own to use it as a caller ID or as the 'To' number for outbound calls/messages from the Sandbox Number. The phone numbers you buy from Twilio, or port to Twilio, can always be used as caller IDs. [Learn more about Verified Caller IDs](#)

Number

Enter the exact number

Friendly Name

Enter the exact friendly name

Filter

Clear filter

Number	Friendly Name	Actions
5111 8738 8881	mi celu	<div>Remove</div>

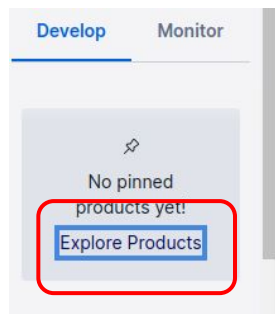


# Comenzando con Twilio

Ejemplo  
en vivo



Para poder enviar mensajes a Argentina, hay que configurarlo.



## All Products

Programmable communications

Super Network

Internet of Things

Solutions

Account security

## Explore Products

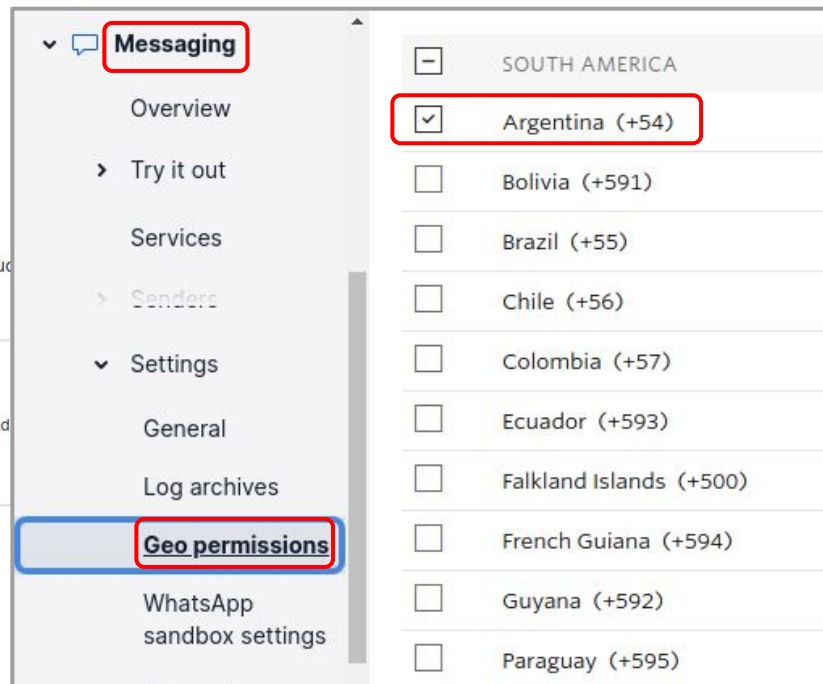
Use this menu to explore, add and remove products

### Programmable communications

[Messaging](#)

Send and receive messages via SMS, MMS, and

[Docs](#)



**CODER HOUSE**



# *Configurar Twilio en nuestra app*

Ejemplo  
en vivo



Vamos ahora a nuestro proyecto e instalamos Twilio como dependencia.

```
$ npm install twilio
```

Luego configuramos el envío de mensajes como muestra el siguiente código.

Los datos de accountSid y authToken lo sacamos de la cuenta de Twilio que creamos.

```
import twilio from 'twilio'

const accountSid = 'xxxxxxxxxxxxxxxxxxxx'
const authToken = 'xxxxxxxxxxxxxxxxxxxx'

const client = twilio(accountSid, authToken)

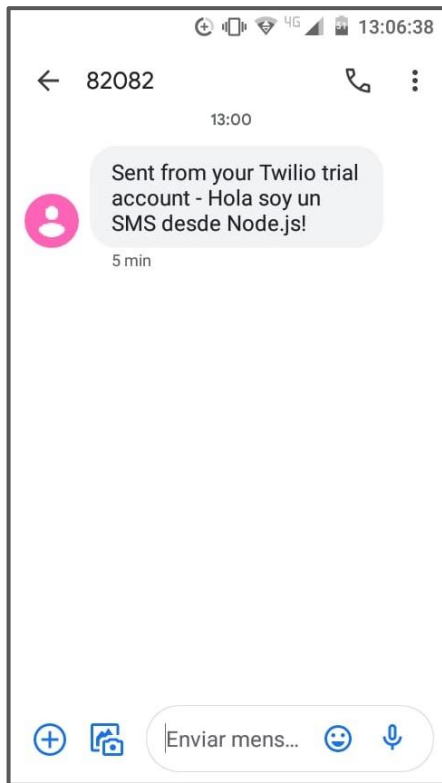
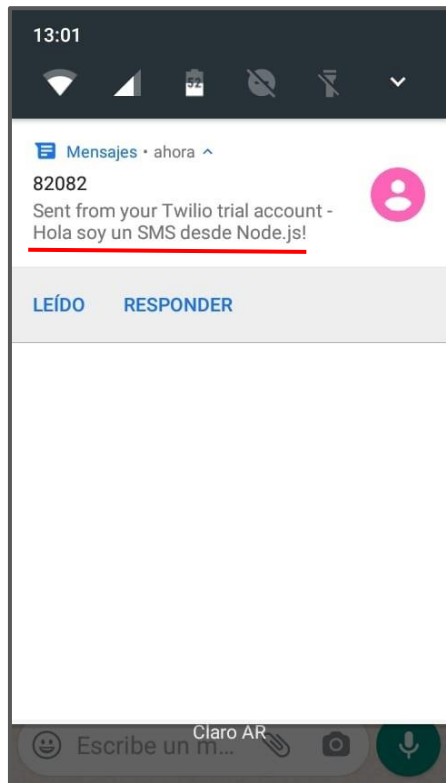
try {
  const message = await client.messages.create({
    body: 'Hola soy un SMS desde Node.js!',
    from: '+14156884237',
    to: '+541199998888'
  })
  console.log(message)
} catch (error) {
  console.log(error)
}
```

**CODER HOUSE**



# ***Configurar Twilio en nuestra app***

Ejemplo  
en vivo



Finalmente, si vamos al celular del número al que le enviamos el mensaje, chequeamos que haya llegado.



# ***ENVIAR SMS DESDE NODE***

*Tiempo: 10 minutos*

# Enviar SMS desde Node

Tiempo: 10 minutos

Desafío  
generico



Realizar en Node.js un programa llamado 'enviarSMS' que permita enviar mensajes de texto.

- Crear a tal fin, una cuenta en Twilio (<https://www.twilio.com/>) habilitando un número telefónico gratis para la operación.
- El número telefónico destino y el mensaje a enviar se le pasarán a la aplicación por línea de línea de comandos.
- Verificar que el mensaje SMS llegue al número indicado y que la operación se refleje en la consola de Twilio.

**NOTA:** Twilio nos dá un número gratis para el envío de SMS que viene con un monto en USD. Cada vez que enviemos un mensaje, se descontará el costo de la operación de dicho monto.

**CODER HOUSE**

***¿PREGUNTAS?***







# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Nodemailer y cómo enviar mails desde una App de Node
  - Twilio y cómo enviar SMS desde una App de Node.
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***