

Esta clase va a ser

- grabada

Clase 30. PROGRAMACIÓN BACKEND

Mailing y mensajería

Objetivos de la clase

- Conocer y utilizar el módulo Nodemailer para el desarrollo de mensajería.
- Conocer el modelo de mensajería de Twilio
- Desarrollar un modelo práctico de mailing

Sobre el protocolo SMTP

SMTP – Protocolo para mailing

Por sus siglas, Simple Mail Transfer Protocol, o protocolo de transferencia de mail simple, es el protocolo que los aplicativos utilizan siempre que tienen que hacer llegar un correo electrónico.

Éste utiliza los puertos 25/TCP o 587/TCP, dependiendo del cliente, o bien 465/TCP para SMTPS



Nodemailer

Para poder realizar envío de mensajería a partir de nuestros aplicativos, la librería por defecto que se utiliza es **nodemailer**, éste nos permitirá conectar con múltiples hosts y servicios de mailing.



Nodemailer

Para trabajar, necesitamos un servicio

Nodemailer conecta con hosts y servicios, por lo que para poder hacer envíos de gmail, hotmail, etc, ocupamos hacer uso de los servicios de gmail, outlook, sendinblue, sendgrid, entre otros, según sea el caso.

Es importante que elijamos primero qué servicio ocuparemos para conectar con nodemailer.



Envío de mensajes con gmail


Lo primero: habilitar contraseña de aplicaciones


Google necesita que tu aplicación realice un proceso de autenticación como cualquier otro, sin embargo, utilizar tu contraseña de gmail de manera explícita (Sí, aún estando en variables de entorno), no es poca cosa para éste.


Por ello, Google tiene una configuración especial llamada “contraseña para aplicaciones” el cual permite que tu aplicativo realice una autenticación con tu correo electrónico sin ocupar tu contraseña personal.


Habilitar autenticación por dos pasos

Para poder utilizar las contraseñas para aplicaciones, necesitamos activar la autenticación en dos pasos desde nuestra configuración de nuestra cuenta de Google.

 Seguridad

 Contactos e información compartida

 Pagos y suscripciones

 Información general



Contraseña

Última modificación: 14 ago 2018



Verificación en dos pasos

 Activada



Contraseñas de aplicaciones

1 contraseña



Creación de una contraseña de aplicación.

← Contraseñas de aplicaciones

La Contraseña de la aplicación te permite acceder a tu cuenta de Google desde apps en dispositivos que no son compatibles con la verificación en 2 pasos. Solo debes ingresarla una vez para que no tengas que recordarla. [Más información](#)

No tienes contraseñas de la aplicación.

Selecciona la app y el dispositivo para los que quieras generar la Contraseña de la aplicación.

CoderNode

GENERAR

Una vez habilitada la autenticación en dos pasos, podemos crear una contraseña de aplicación, basta colocarle el nombre y nos dará una contraseña única.

Contraseña de aplicación generada

Tu contraseña de aplicación para el dispositivo

rouv mczp qswj cgrq

Esta contraseña generada es única y no se puede revisar nuevamente, si deseamos utilizarla a largo plazo, ¡mejor guardarla en algún lado!

Utilizando nuestra cuenta de gmail

Ahora levantaremos un servidor express como es habitual, además de instalar el módulo de nodemailer a partir de

```
npm install nodemailer
```

Posteriormente, se importa al mismo nivel de app.js (al menos dentro de esta prueba).

Se creará un endpoint “/mail” para poder probar el envío de nodemailer.

Estructura base

JS app.js



{ } package.json

```
src > JS app.js >  app.get('/mail') callback
1  import express from 'express';
2  import nodemailer from 'nodemailer';
3
4  const app = express();
5
6  app.get('/mail', async(req, res)=>{
7    |
8  })
```

Conectando nuestra cuenta de gmail

Una vez hecho nuestro endpoint, solo falta inicializar un transporte de Gmail para poder comenzar a enviar correos.

El transporte variará según sea el servicio a utilizar, por lo que es importante revisar la documentación que implica cada servicio.

Podemos tener múltiples transportes configurados, aunque se recomienda tener un único sistema base para éste. (Debido al reconocimiento de origen del cliente y los correos spam)

```
const transport = nodemailer.createTransport({  
  service: 'gmail',  
  port: 587,  
  auth: {  
    user: 'tu correo de gmail',  
    pass: 'La contraseña de aplicación de tu correo'  
  }  
})
```

Enviando nuestro correo

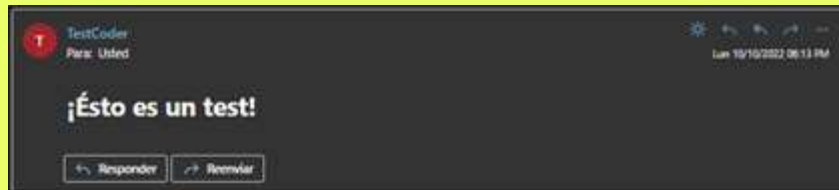
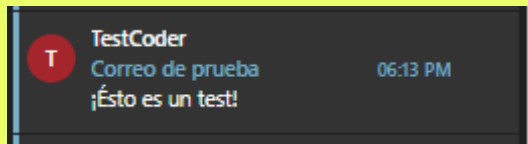
Un envío de correo es esencialmente:

- from: ¿Quién envía el correo?
- to : ¿A quién envió el correo?
- subject: ¿Qué asunto tiene el correo?
- html: Plantilla completa con toda la información a enviar, podemos enviar textos planos, o plantillas suficientemente complejas.
- attachments, es un conjunto de archivos o imágenes que deseamos que aparezcan en el correo.

```
✓ app.get('/mail', async(req, res)=>{  
  ✓   let result = await transport.sendMail({  
    from: 'Coder Tests <tucorreodegmail>',  
    to: 'correo adonde queremos enviar',  
    subject: 'Correo de prueba',  
    html: `  
      ✓   <div>  
        |   <h1>¡Ésto es un test!</h1>  
      </div>  
    `,  
    attachments: []  
  })  
})
```

¡Recibimos un correo!

Finalmente, podemos revisar nuestra bandeja de entrada:



¡Hemos recibido nuestro primer correo!

Notamos que, al colocar el formato de "from" como lo hicimos, logramos ocultar de manera explícita el correo del cual estamos enviando.

**Envío con attachments
e imágenes.**

Enviando imágenes y documentos.

Muy difícilmente un correo electrónico cuenta sólo con texto. De manera que el uso de imágenes y anexar documentos se vuelve imprescindible.

Para ello, haremos uso del campo "attachments" el cual es una de las propiedades al momento del envío del correo.

Utilizarlo en la plantilla de html es ligeramente más complejo, ya que tenemos que ligar el attachment a un "cid", con el fin de que podamos utilizarlo dentro del cuerpo del html enviado.

Si no relacionamos un cid, al momento de enviar el correo las imágenes saldrán rotas o arrojará un error al momento de cargar.

Agregando un attachment

perrito2.jpg src\i...
package.json
CLASE30MAILINGMENSAJ...
node_modules
src
images
perrito1.jpg
perrito2.jpg
app.js
utils.js
package-lock.json
package.json

```
15 app.get('/mail', async(req, res)=>{  
16     let result = await transport.sendMail({  
17         from: 'TestCoder <mauricioespinosaflares25@gmail.com>',  
18         to: 'ing_mauricioespinosa@hotmail.com',  
19         subject: 'Correo de prueba',  
20         html: `  
21             <div>  
22                 <h1>¡Esto es un test, pero con imágenes, mira!</h1>  
23                   
24             </div>  
25             `,  
26         attachments: [{  
27             filename: 'perrito1.jpg',  
28             path: __dirname + '/images/perrito1.jpg',  
29             cid: 'perrito1'  
30         }]  
31     })  
32     res.send({status: "success", result: "Email Sent"})  
33 }  
34
```

Agregando un attachment





Envío de correos

Duración: 15min



ACTIVIDAD EN CLASE

Envío de correos

A partir del módulo de nodemailer

Realizar el envío de un correo electrónico a partir de tu correo de Gmail, enviando documentos anexos, así también como imágenes en el formato de html.

El concepto de envío será libre.



Break

¡10 minutos y volvemos!

Envío de mensajería con Twilio

Twilio

Twilio es una plataforma de comunicaciones que nos permitirá establecer un sistema de comunicación más directo con el cliente.

En muchas ocasiones, nuestros correos podrían dar a parar a la bandeja de spam del cliente (Según cómo lo tenga configurado), de modo que ocuparemos realizar mensajería más directa con éste, según sea la intención de nuestro aplicativo

Twilio nos permitirá establecer SMS, whatsapp, chatbots, mensajes de voz pregrabados, etc.

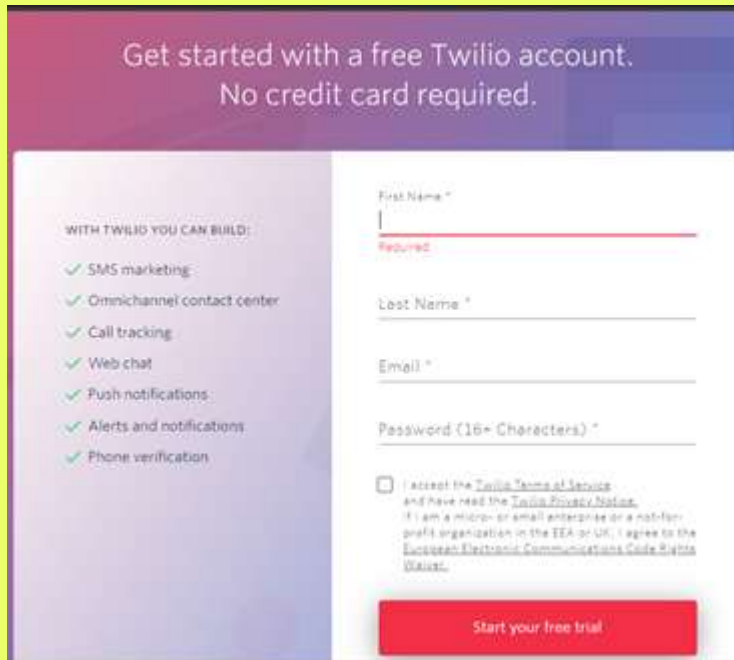


Cuenta de Twilio

Es importante notar que registrarse en Twilio habla directamente de un free trial, lo cual sí implica que habrá dinero de por medio

Sin embargo, nuestro registro es totalmente gratuito, además, para nuestras pruebas nos regala 15 USD para el envío de mensajes que ocupemos durante las pruebas.

No se requiere tarjeta de para registrarse, por lo que no afectará a futuro



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 *

Required

Last Name *

Email *

Password (16+ Characters) *

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

Start your free trial



Toma 3 minutos para registrarte y
verificar tu email

Entendiendo el panel de Twilio

Panel de Twilio



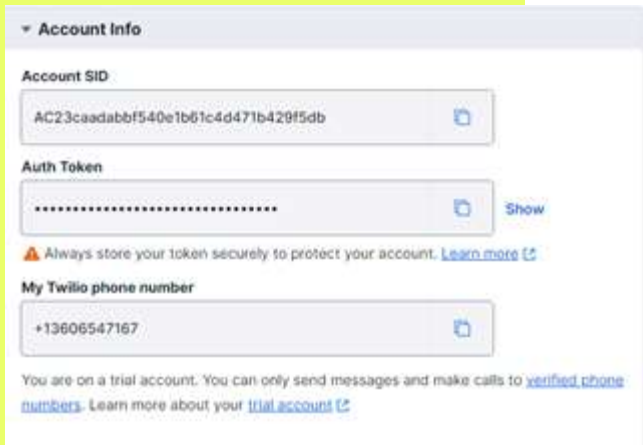
Una vez registrado y logueado, notaremos un panel de bienvenida, además de un campo "Trial" donde se nos indica que tenemos 15.50 USD de prueba para nuestras implementaciones.

Para comenzar a utilizarlo, daremos click al botón "Get a trial phone number" para contar con un teléfono de prueba para envío de mensajes.

Panel de Twilio

En la parte inferior de nuestro panel, encontraremos tres elementos importantes:

- Account SID, importante para poder identificar nuestro aplicativo de Twilio en el código
- Auth Token, clave secreta de autenticación para poder utilizar esta cuenta.
- My Twilio phone number, que es el número de prueba recién generado.



Cuida los números verificados

My Twilio phone number

+13606547167



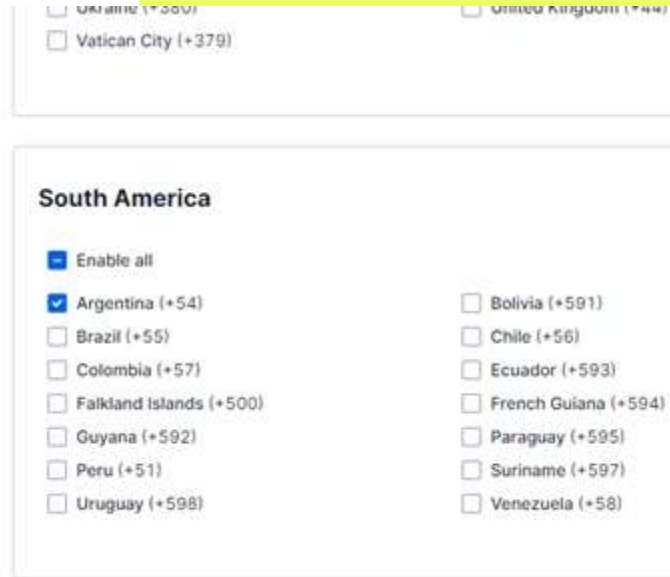
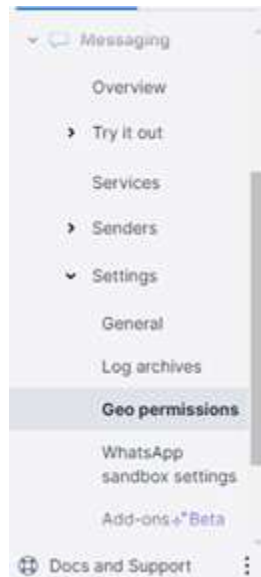
You are on a trial account. You can only send messages and make calls to verified phone numbers. Learn more about your [trial account](#) 

Al ser una cuenta para pruebas, no podemos spammear a cualquier amigo o familiar con nuestros mensajes. La cuenta sólo podrá enviar mensajería a los números verificados.

Nuestro número verificado por defecto siempre será el número de verificación que colocamos al momento de nuestro registro, por lo que no es necesario agregar otro más.

¡Marca los permisos por localización!

Es muy importante que, para que Twilio reconozca los prefijos* correspondientes, es importante que marquemos las casillas de los lugares a utilizar.



*también llamados ladas o código, refiere al signo “+” acompañado de un número que representa la característica telefónica de cada país.

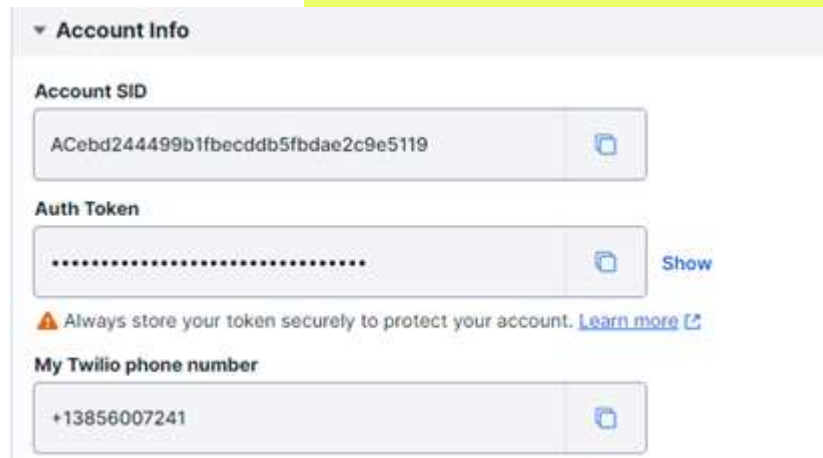
Enviando mensajes con Twilio

Tomando las variables necesarias

Para poder resolver un envío de mensaje de twilio ocuparemos tres campos:

- Account SID
- Auth Token
- Twilio Number

Estos tres datos los encontramos en nuestra consola de Twilio.



Instalar twilio + setear variables

```
JS app.js X
src > JS app.js > app.get('/sms') callback
1 import express from 'express';
2 import nodemailer from 'nodemailer';
3 import __dirname from './utils.js';
4 import twilio from 'twilio';
5
6 const app = express();
7
8 const TWILIO_ACCOUNT_SID= "ACebd244499b1fbecddb5fbdae2c9e5119";
9 const TWILIO_AUTH_TOKEN = "REDACTED";
10 const TWILIO_SMS_NUMBER="+13856007241";
11
```

Utilizamos npm para poder hacer uso de Twilio con el comando

```
npm install twilio
```

Y colocamos nuestras variables necesarias a nivel app.js (Aunque vendría bien tenerlas en variables de entorno).

Inicializando al cliente

Una vez que tenemos las variables necesarias, podemos inicializar nuestro cliente para envío de mensajes.

```
const client = twilio(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN);
```

Ahora podemos proceder a enviar nuestro primer mensaje SMS

Sms endpoint

```
✓ app.get('/sms', async (req, res) => {  
  ✓ let result = await client.messages.create({  
    body: 'Esto es un mensaje SMS',  
    from: TWILIO_SMS_NUMBER,  
    to: 'Tu número de prueba'  
  })  
  res.send({status:"success",result:"Message sent"})  
})
```

Voilà!, ¡recibimos nuestro primer mensaje!



¡Importante!

Recuerda que cada mensaje enviado y cada implementación genera un coste para Twilio. ¡Te recomendamos aprovecharlo al máximo para decidir si es la opción que necesitas!



Envío de mensajería

Duración: 15min



ACTIVIDAD EN CLASE

Envío de mensajería

A partir del módulo de Twilio

Realizar el envío de un sms conectado con tu teléfono personal, el cual enviará el mensaje:

`Gracias, \${nombre}, tu solicitud del producto \${producto} ha sido aprobada`

Donde **nombre** y **producto** deberán recibirse por query params.



Tercera entrega de tu Proyecto final

Se profundizará sobre los roles de los usuarios, las autorizaciones y sobre la lógica de compra.



Mejorando la arquitectura del servidor

Objetivos generales

- ✓ Profesionalizar el servidor

Objetivos específicos

- ✓ Aplicar una arquitectura profesional para nuestro servidor
- ✓ Aplicar prácticas como patrones de diseño, mailing, variables de entorno. etc.

Se debe entregar

- ✓ Modificar nuestra capa de persistencia para aplicar los conceptos de Factory (opcional), DAO y DTO.

Se debe entregar

- ✓ El DAO seleccionado (por un parámetro en línea de comandos como lo hicimos anteriormente) será devuelto por una Factory para que la capa de negocio opere con él. (Factory puede ser opcional)
- ✓ Implementar el patrón Repository para trabajar con el DAO en la lógica de negocio.
- ✓ Modificar la ruta /current Para evitar enviar información sensible, enviar un DTO del usuario sólo con la información necesaria.



Mejorando la arquitectura del servidor

Se debe entregar

- ✓ Realizar un middleware que pueda trabajar en conjunto con la estrategia "current" para hacer un sistema de autorización y delimitar el acceso a dichos endpoints:
- Sólo el administrador puede crear, actualizar y eliminar productos.
- Sólo el usuario puede enviar mensajes al chat.
- Sólo el usuario puede agregar productos a su carrito.

Se debe entregar

- ✓ Crear un modelo Ticket el cual contará con todas las formalizaciones de la compra. Éste contará con los campos
 - `Id` (autogenerado por mongo)
 - `code`: String debe autogenerarse y ser único
 - `purchase_datetime`: Deberá guardar la fecha y hora exacta en la cual se formalizó la compra (básicamente es un `created_at`)
 - `amount`: Number, total de la compra.
 - `purchaser`: String, contendrá el correo del usuario asociado al carrito.



Mejorando la arquitectura del servidor

Se debe entregar

- ✓ Implementar, en el router de carts, la ruta `/:cid/purchase`, la cual permitirá finalizar el proceso de compra de dicho carrito.
- La compra debe corroborar el stock del producto al momento de finalizarse
 - Si el producto tiene suficiente stock para la cantidad indicada en el producto del carrito, entonces restarlo del stock del producto y continuar.
 - Si el producto no tiene suficiente stock para la cantidad indicada en el producto del carrito, entonces no agregar el producto al proceso de compra.

Se debe entregar

- Al final, utilizar el servicio de Tickets para poder generar un ticket con los datos de la compra.
- En caso de existir una compra no completada, devolver el arreglo con los ids de los productos que no pudieron procesarse.

Una vez finalizada la compra, el carrito asociado al usuario que compró deberá contener sólo los productos que no pudieron comprarse. Es decir, se filtran los que sí se compraron y se quedan aquellos que no tenían disponibilidad.



Mejorando la arquitectura del servidor

Formato

- ✓ Link al repositorio de Github con el proyecto (sin node_modules)
- ✓ Además, archivo .env para poder correr el proyecto.

Sugerencias

- ✓ Te recomendamos ver el vídeo explicativo disponible en la carpeta de clase

¿Preguntas?

Muchas gracias.

Resumen de la clase hoy

- ✓ Nodemailer
- ✓ Nodemailer con gmail
- ✓ Twilio
- ✓ SMS con Twilio

Opina y valora
esta clase

#DemocratizandoLaEducación