

Esta clase va a ser

- grabada
a

Clase 16. PROGRAMACIÓN BACKEND

Mailing y mensajería & Mocks

Temario

15 – Parte II

Desarrollo de un servidor web basado en capas

- ✓ Práctica de desarrollo de aplicación backend completa.
- ✓ Conexión front-back

16 – Parte I

Mailing y mensajería

- ✓ Nodemailer
- ✓ Twilio
- ✓ Desarrollo práctico de correo y mensajería

16 – Parte II

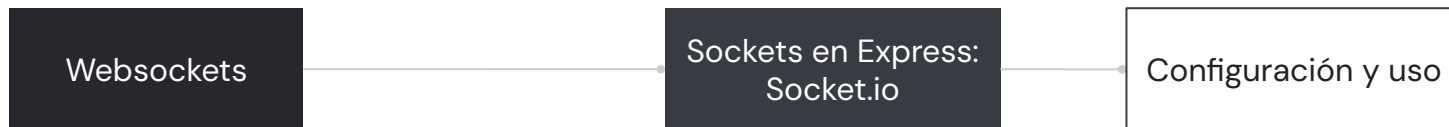
Mocks

- ✓ TDD
- ✓ Desarrollo de prácticas preventivas a partir de mocks

Objetivos de la clase – Parte I

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

MAPA DE CONCEPTOS



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.

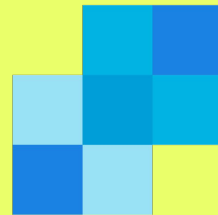
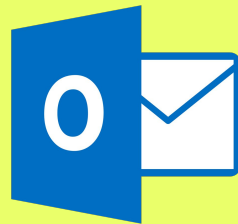


Nodemailer

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

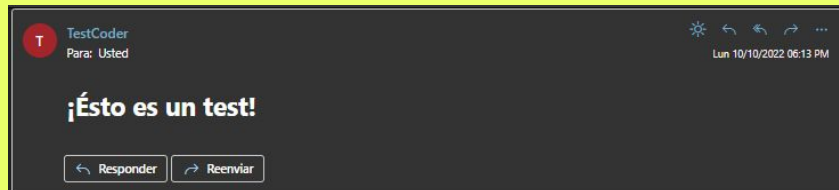
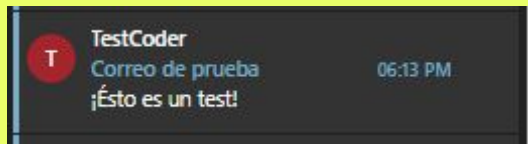
Un envío de correo es esencialmente:

- from: ¿Quién envía el correo?
- to : ¿A quién envío 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\...

package.json

CLASE30MAILINGMENSAJ...

> node_modules

src

images

perrito1.jpg

perrito2.jpg

JS app.js

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>¡Ésto 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 Waiver](#).

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.

The screenshot displays the Twilio console interface. At the top, a dark blue header bar contains the Twilio logo, the text 'Console', the account name 'My first Twilio account', a search bar with the placeholder 'Jump to...', and links for 'Account' and 'Billing'. Below the header, the main content area is divided into a left sidebar and a central panel. The sidebar includes a 'Develop' tab, a 'Monitor' tab, and a list of navigation items: 'Phone Numbers', 'Messaging', 'Explore Products', and 'Docs and Support'. The central panel features a large welcome message 'Ahoy [redacted], welcome to Twilio!' followed by a section titled 'Connect to 3rd-party applications'. This section lists three requirements: 'Account SID and Auth token', 'Twilio phone number', and 'Upgraded Twilio account'. A button labeled 'Get a trial phone number' is prominently displayed, along with a link to 'Read 3rd-party integration FAQ'. To the right of this section is a 'Invite teammates' box with a button 'Invite teammates ->'. Below that is a 'Talk to Sales' box with a button 'Talk to Sales'. At the bottom, there are two expandable sections: 'Account Info' showing 'Account SID' and 'Helpful links' with a link 'How does Twilio work?'. An illustration of two people stacking blocks is positioned between the 'Connect to 3rd-party applications' and 'Invite teammates' sections.

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.

▼ Account Info

Account SID

AC23caadabbf540e1b61c4d471b429f5db

Auth Token

.....

Show

⚠ Always store your token securely to protect your account. [Learn more](#)

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](#)

▼ Helpful links

[How does Twilio work?](#)

Understand how to use Twilio in a 2-minute video.

[API documentation](#)

Learn the basics of Twilio APIs.

[Support help center](#)


Troubleshoot common issues.

Cuida los números verificados

My Twilio phone number

+13606547167



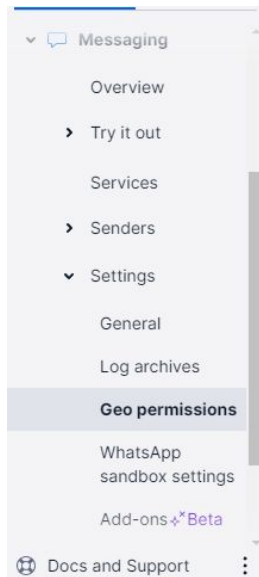
You are on a trial account. You can only send messages and make calls to verified phone numbers. Learn more about your [trial account](#) 

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.



☐ Ukraine (+380) ☐ United Kingdom (+44)
☐ Vatican City (+379)

South America

☒ Enable all

☒ Argentina (+54)

☐ Brazil (+55)

☐ Colombia (+57)

☐ Falkland Islands (+500)

☐ Guyana (+592)

☐ Peru (+51)

☐ Uruguay (+598)

☐ Bolivia (+591)

☐ Chile (+56)

☐ Ecuador (+593)

☐ French Guiana (+594)

☐ Paraguay (+595)

☐ Suriname (+597)

☐ Venezuela (+58)

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

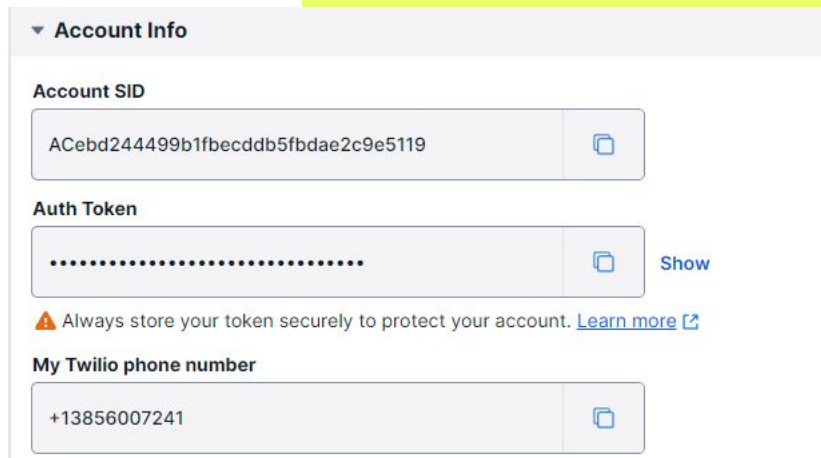
Enviando mensajes con Twilio

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.






▼ Account Info

Account SID


ACebd244499b1fbecddb5fbdae2c9e5119 

Auth Token

.....  [Show](#)

 Always store your token securely to protect your account. [Learn more](#) 

My Twilio phone number

+13856007241 

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 = " ";
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 aquí:
- ✓ Aquí iría el link del vídeo explicativo***

¿Preguntas?

Resumen de la parte I

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



Break

¡30 minutos y volvemos!

Temario

16 – Parte II

Mocks

- ✓ [TDD](#)
- ✓ [Desarrollo de prácticas preventivas a partir de mocks](#)

17 – Parte I

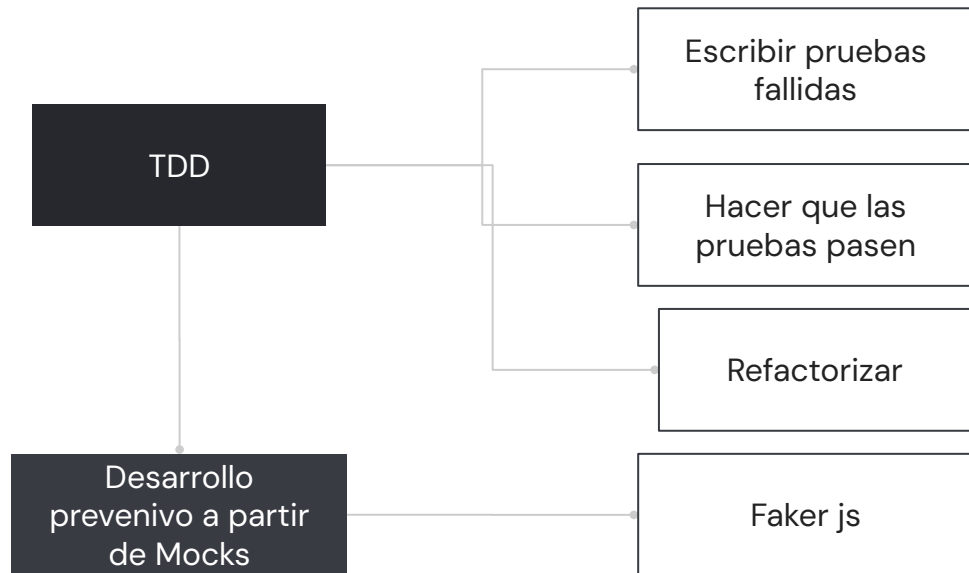
Optimización

- ✓ Rendimiento en producción
- ✓ Comprensión con Brotli
- ✓ Middleware para manejo de errores

Objetivos de la clase – Parte II

- Entender y aplicar el concepto de TDD
- Comprender el concepto de Mocks
- Realizar un desarrollo práctico de Mocking.

MAPA DE CONCEPTOS



TDD

Sobre el desarrollo y los errores

¿Cuántos errores has cometido en tu código desde que comenzaste a desarrollar? ¿Qué tan graves suelen ser esos errores?

Seguramente tus respuestas serán **“muchos”** y **“no muy graves”**. El nivel de gravedad de un error es bastante relativo, sin embargo, muchas veces lo podemos medir por qué tanto impacto tiene directamente en la experiencia del cliente.

Si aún no te has desenvuelto en un ambiente productivo, sabrás que todos los errores que se cometen no salen de tu computadora, es decir, todo se queda en un “entorno de desarrollo”.

Errores en entorno productivo

¿Qué pasa cuando ocurre un error estando en un entorno productivo? Recordemos que este entorno es el nivel en el que el cliente puede visualizar e interactuar con todo, de manera que un error en este punto puede ser bastante grave.

Seamos desarrolladores frontend o backend, recordamos que la experiencia del usuario se puede ver afectada de manera que puede generar molestias al utilizar nuestra página, o incluso llevarlo a no querer volver a utilizarlo.



Tres principales errores:

- ✓ **De compilación:** "Mi código no compila", ocurre cuando un código no puede iniciarse debido a que el compilador encontró algún error antes de ejecutarlo.
- ✓ **De ejecución:** "Mi código explotó", ocurre cuando el código sí logra compilarse, pero a lo largo de su trabajo ocurre un error, usualmente no controlado.
- ✓ **Lógicos:** "Funciona, pero no funciona", ocurre cuando el código compila y se ejecuta correctamente, pero el resultado no es el esperado.

Consecuencias

- ✓ Las consecuencias podrán variar, algunas pueden ser muy simples...

"He estado experimentando que mi cámara tiembla descontroladamente cada vez que abro Snapchat o uso la cámara para Instagram", escribió uno. "Sin embargo, no tengo ningún problema cuando uso la aplicación de cámara normal".

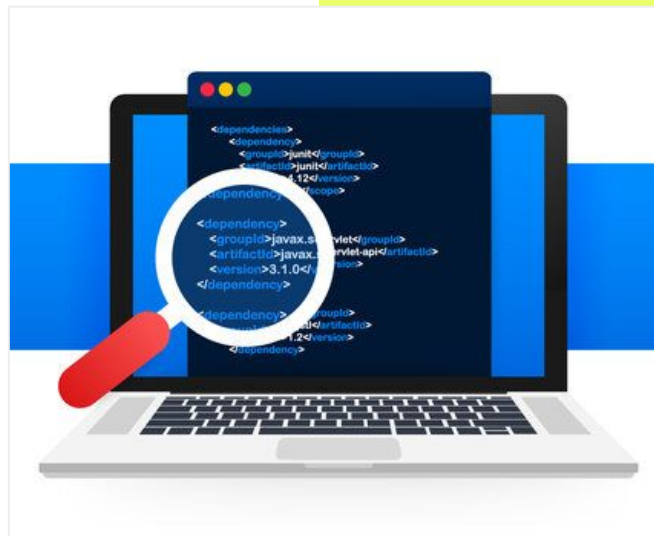
- ✓ Y otros que ya no causan tanta gracia

Con base en los ingresos por US\$29.000 millones del último trimestre, *Fortune* señala que **la compañía perdió US\$99,75 millones en ingresos** por el número de horas inactivas de publicidad.

¡Piensa en tus pruebas!

Cuando nuestro código comienza a representar una empresa, organización, o cliente, cuya funcionalidad o errores definen las ganancias de la misma, tenemos que pensar que nuestro código, **no puede fallar**.

Pensar en un aplicativo con margen de error de 0% es utópico. Sin embargo, hacer pruebas de nuestro código reducirá en gran medida el margen de error de dicho módulo.

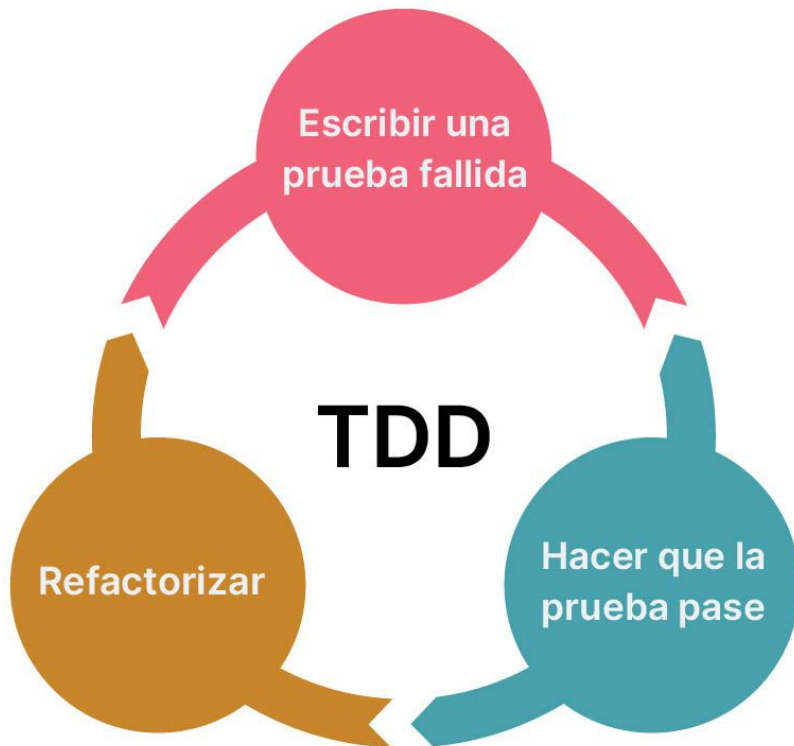


Test Driven Development

Una de las mejores formas de abordar un problema, por muy grande o pequeño que sea, es a partir de una práctica de programación llamada **Test Driven Development (TDD, o desarrollo orientado a pruebas)**.

Éste consiste en escribir las pruebas del módulo antes incluso de desarrollar dicho módulo, con el fin de dirigir nuestro desarrollo hacia el cumplimiento de estas pruebas. De esta manera, podemos visualizar cómo nuestro código irá cumpliendo punto a punto las cosas que debe abarcar en su uso.

El TDD es algo subjetivo y hay quienes agregan pasos extras y conceptos adicionales, pero en términos generales consiste en lo siguiente:

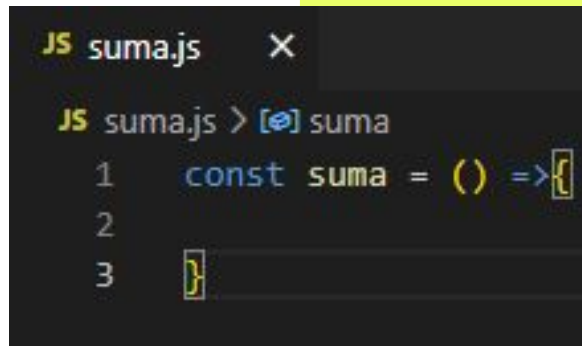


Etapas 1: Escribir pruebas fallidas

Escribir una prueba fallida

Con alguna herramienta de testing, podemos colocar las pruebas que necesitemos con el fin de corroborar una funcionalidad.

Se dice que la primera prueba debe fallar, pues ya que nuestro código aún no ha sido implementado, no hay forma de que se pase. Por ejemplo, pensemos en una función:



```
JS suma.js X
JS suma.js > [e] suma
1  const suma = () => {
2
3  }
```


¿Qué pruebas hacer con una suma?

En este punto, podríamos comenzar a pensar en múltiples escenarios para poner a prueba.

Estos escenarios pueden hacerse desde un módulo de testing, sin embargo, por fines prácticos, lo haremos sin módulo y ejecutando directamente la función (No te preocupes, más adelante vamos a implementar módulos de testing reales).

Algunos escenarios a plantear podrían ser:

- ✓ La función debe devolver null si algún parámetro no es numérico.
- ✓ La función debe devolver 0 si no se pasó ningún parámetro
- ✓ La función debe poder realizar la suma correctamente.
- ✓ La función debe poder hacer la suma con cualquier cantidad de números.

Implementando el primer escenario

El primer punto a testear se podría hacer de la siguiente manera:

```
//1. La función debe devolver null si algún parámetro no es numérico
console.log("Test 1: La función debe devolver null si algún parámetro no es numérico")
let resultTest1 = suma("2",2);
if(resultTest1===null) {
  console.log("Test 1 pasado");
  testsPasados++;
}
```

Y, si corremos el código, podremos notar que no pasamos la primera prueba

```
Test 1 no pasado, se recibió undefined, pero se esperaba null
```

Implementando el resto de escenarios (1)

```
JS suma.js  X
JS suma.js > [0] resultTest1
1  const suma = (num1,num2) =>{
2    if(!num1||!num2) return 0;
3    if(typeof num1!="number"||typeof num2!="number") return null;
4  }
5
6  //Escenarios
7  let testsPasados = 0 ;
8  let testsTotales = 4;
9  //1. La función debe devolver null si algún parámetro no es numérico
10 console.log("Test 1: La función debe devolver null si algún parámetro no es numérico")
11 let resultTest1 = suma("2",2);
12 if(resultTest1===null) {
13   console.log("Test 1 pasado");
14   testsPasados++;
15 }
16 else console.log(`Test 1 no pasado, se recibió ${typeof resultTest1}, pero se esperaba null`);
17 //2. La función debe devolver 0 si no se pasó ningún parámetro
18 console.log("Test 2: La función debe devolver 0 si no se pasó ningún parámetro")
19 let resultTest2 = suma();
20 if(resultTest2===0) {
21   console.log("Test 2 pasado");
22   testsPasados++;
23 }
24 else console.log(`Test 2 no pasado, se recibió ${resultTest2}, pero se esperaba 0`);
```

Implementando el resto de escenarios (2)

```
//3. La función debe poder realizar la suma correctamente
console.log("Test 3: La función debe resolver la suma correctamente.");
let resultTest3 = suma(2,3);
✓ if(resultTest3===5) {
  console.log("Test 3 pasado");
  testsPasados++;
}
else console.log(`Test 3 no pasado, se recibió ${resultTest3}, pero se esperaba 6`);
//4. La función debe poder hacer la suma con cualquier cantidad de números
let resultTest4 = suma(1,2,3,4,5);
✓ if(resultTest4===15) {
  console.log("Test 4 pasado");
  testsPasados++;
}
else console.log(`Test 4 no pasado, se recibió ${resultTest4}, pero se esperaba 15`);
if(testsPasados===testsTotales) console.log("Todos los tests se han pasado con éxito");
else console.log(`Se pasaron ${testsPasados} tests de un total de ${testsTotales}`);
```

Ejecutando escenarios

```
Test 1: La función debe devolver null si algún parámetro no es numérico
Test 1 no pasado, se recibió undefined, pero se esperaba null
Test 2: La función debe devolver 0 si no se pasó ningún parámetro
Test 2 no pasado, se recibió undefined, pero se esperaba 0
Test 3: La función debe resolver la suma correctamente.
Test 3 no pasado, se recibió undefined, pero se esperaba 6
Test 4 no pasado, se recibió undefined, pero se esperaba 15
Se pasaron 0 tests de un total de 4
```

Etapas 2: Hacer que las pruebas pasen

Test 1: parámetro no numérico

Resolvamos por pasos, suponiéndose que se reciben dos valores numéricos, entonces podemos hacer:

```
const suma = (num1,num2) =>{  
  if(typeof num1!="number" || typeof num2!="number") return null;  
}
```

Hasta ahora entonces tenemos:

```
Test 1: La función debe devolver null si algún parámetro no es numérico  
Test 1 pasado  
Test 2: La función debe devolver 0 si no se pasó ningún parámetro  
Test 2 no pasado, se recibió null, pero se esperaba 0  
Test 3: La función debe resolver la suma correctamente.  
Test 3 no pasado, se recibió undefined, pero se esperaba 6  
Test 4 no pasado, se recibió undefined, pero se esperaba 15  
Se pasaron 1 tests de un total de 4
```


Test 2: ningún parámetro

Podemos agregar una nueva validación para parámetros vacíos:

```
const suma = (num1,num2) =>{  
  if(!num1||!num2) return 0;  
  if(typeof num1!="number"||typeof num2!="number") return null;  
}
```

Hasta ahora entonces tenemos:

```
Test 1: La función debe devolver null si algún parámetro no es numérico  
Test 1 pasado  
Test 2: La función debe devolver 0 si no se pasó ningún parámetro  
Test 2 pasado
```


Test 3: Suma

Ejecutamos el código necesario para hacer la suma (no importa la optimización)

```
const suma = (num1,num2) =>{  
  if(!num1||!num2) return 0;  
  if(typeof num1!="number"||typeof num2!="number") return null;  
  let result = num1+num2;  
  return result;  
}
```

Ya casi finalizamos:

```
Test 1: La función debe devolver null si algún parámetro no es numérico  
Test 1 pasado  
Test 2: La función debe devolver 0 si no se pasó ningún parámetro  
Test 2 pasado  
Test 3: La función debe resolver la suma correctamente.  
Test 3 pasado  
Test 4 no pasado, se recibió 3, pero se esperaba 15  
Se pasaron 3 tests de un total de 4
```

Test 4: ¿ *n* parámetros?

Notamos que en este punto será necesario hacer cambios más grandes, ya que para *n* número de parámetros, no servirán las validaciones que tenemos.

Entonces toca reformular el problema.
¿Cómo trabajar con cualquier número de inputs? Procedemos a los cambios:

```
1  const suma = (...nums) =>{  
2    if(nums.length===0) return 0;  
3    let validInput = true;  
4    for(let i = 0; i < nums.length&&validInput; i++){  
5      if(typeof nums[i] !== "number"){  
6        validInput=false;  
7      }  
8    }  
9    if(!validInput) return null;  
10   let result = 0;  
11   for(let i=0;i<nums.length;i++){  
12     result+=nums[i];  
13   }  
14   return result;  
15 }
```

Etapas 3: Refactorizar

¡Listo!

¡Logramos pasar nuestros tests! Sin embargo, creo que notamos algo bastante serio:

El código funciona, de manera que pasamos las pruebas necesarias para considerarse “funcional”, sin embargo, ¿es óptimo?

Tenemos que buscar ahora una forma de que nuestro código sea más limpio y sintetizar el código.

```
Test 1: La función debe devolver null si algún parámetro no es numérico
Test 1 pasado
Test 2: La función debe devolver 0 si no se pasó ningún parámetro
Test 2 pasado
Test 3: La función debe resolver la suma correctamente.
Test 3 pasado
Test 4: La función debe poder funcionar con cualquier cantidad de números
Test 4 pasado
Todos los tests se han pasado con éxito
```

Sobre el arte de la refactorización

Al refactorizar, no pensamos en nuevas funcionalidades ni nada extra al código que tenemos, sino que **buscamos la manera de poder sintetizar tareas que podrían implementarse de otra forma.**

Al refactorizar, conseguimos mantener una funcionalidad idéntica para no afectar los tests, pero realizando tareas mucho más limpias.

Es la etapa final del TDD, una vez realizado ésto, podemos dar nuestro código por finalizado. 🎉

Refactorización

Función suma antes

```
1  const suma = (...nums) =>{
2    if(nums.length===0) return 0;
3    let validInput = true;
4    for(let i = 0; i < nums.length&&validInput; i++){
5      if(typeof nums[i] !== "number"){
6        validInput=false;
7      }
8    }
9    if(!validInput) return null;
10   let result = 0;
11   for(let i=0;i<nums.length;i++){
12     result+=nums[i];
13   }
14   return result;
15 }
```

Función suma ahora

```
const suma = (...nums) =>{
  if(nums.length===0) return 0;
  if(!nums.every(num=>typeof num==="number")) return null;
  return nums.reduce((prev,current)=>prev+current)
}
```

No hay afección en los tests

Todos los tests se han pasado con éxito

Listo, ¡Un módulo desarrollado a partir de un estilo de trabajo TDD!

El desarrollo basado en tests es bastante solicitado en nivel empresarial, ya que es una práctica que reduce en gran medida la posibilidad de errores “no contemplados”.

Evidentemente no es posible escribir TODOS los tests para TODOS los escenarios, pero es de gran apoyo.

Existen múltiples formas de testear estas funcionalidades, las cuales abordaremos más adelante en el módulo, para darte la seguridad de testear tu código y entregar un trabajo de calidad en tu trabajo.

¡Importante!

El TDD no es una herramienta, no es un módulo externo, no es algo de internet que puede implementarse. Es una forma de programar y, por lo tanto, necesitarás profundizar, practicarlo e implementarlo en entornos reales por tu cuenta.



Aplicando TDD

Duración: 20–25 min



ACTIVIDAD EN CLASE

Aplicando TDD

Aplicar bajo el modelo de trabajo de TDD:

Una función de login (con usuarios hardcodeados user = coderUser , password = 123)

- ✓ Si se pasa un password vacío, la función debe consologear ("No se ha proporcionado un password")
- ✓ Si se pasa un usuario vacío, la función debe consologear ("No se ha proporcionado un usuario")
- ✓ Si se pasa un password incorrecto, consologear ("Contraseña incorrecta")
- ✓ Si se pasa un usuario incorrecto, consologear ("Credenciales incorrectas")
- ✓ Si el usuario y contraseña coinciden, consologear ("logueado")



Break

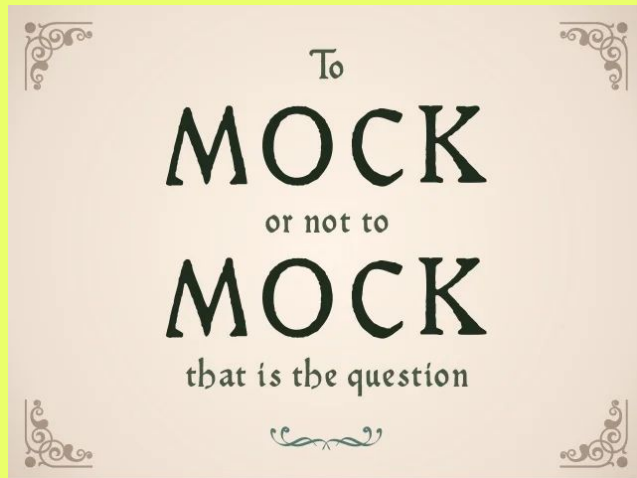
¡10 minutos y volvemos!

Desarrollo preventivo a partir de mocks

¿Qué es un mock?

Dentro de sus múltiples traducciones, podemos tomar **mock** como una “imitación” de un dato real. Nos es altamente útil para poder crear datos “supuestos” con el fin de probar la funcionalidad de alguna función

Un dato mock no debe comprometer jamás una estructura productiva, por lo que solo se usa en entornos de desarrollo.



Usos comunes de mocks

El escenario bajo el cual se suelen usar mocks son:

- ✓ Para generar usuarios falsos y probar módulos de registros o logins y sesiones.
- ✓ Para generar productos de prueba en un ecommerce.
- ✓ NPCs o personajes prueba para algún videojuego.
- ✓ Conjuntos de estadísticas para módulos de análisis
- ✓ Conjuntos de coordenadas para sistemas de geolocalización

Faker-js

Faker-js

Faker-js es un módulo externo de node js derivado del original **fakerjs**, el cual es un módulo pensado para poder realizar modelos de prueba de diferentes formas, en diferentes idiomas y diferentes escenarios.

¡Faker es extremadamente potente!

Podemos crear mocks de cosas como:

- ✓ nombres, apellidos, edades
- ✓ colores, fechas, números
- ✓ animales, países, géneros musicales
- ✓ vehículos, ipv4s, coordenadas geográficas
- ✓ ¡Es un mundo de datos para probar!



Algunas de las cosas que podemos simular con faker

Address

buildingNumber
cardinalDirection
city
cityName
cityPrefix
citySuffix
country
countryCode
county
direction
latitude
longitude
nearbyGPSCoordinate
ordinalDirection
secondaryAddress
state
stateAbbr
street
streetAddress
streetName
streetPrefix
streetSuffix
timeZone
zipCode
zipCodeByState

Date

between
betweens
birthdate
future
month
past
recent
soon
weekday

Lorem

lines
paragraph
paragraphs
sentence
sentences
slug
text
word
words

Fake

fake

Mersenne

rand
seed
seed_array

Finance

account
accountName
amount
bic
bitcoinAddress
creditCardCVV
creditCardIssuer
creditCardNumber

Music

genre
songName

Name

Referencia completa de la página [aquí](#)

Caso práctico

En la startup que trabajamos, el equipo de data science y analytics necesita realizar algunos algoritmos de análisis a partir de los usuarios que tenemos en nuestra base de datos.

Sin embargo, ya que la empresa es nueva, no contamos con una base lo suficientemente amplia para poder alimentar las analíticas del algoritmo que están desarrollando.

Necesitaremos “simular” múltiples usuarios con el fin de que la respuesta tenga una cantidad de usuarios lo suficientemente abundante para que ellos testeen sus propios algoritmos





Hands on lab

En esta instancia de la clase **repasaremos** algunos de los conceptos vistos en clase con una aplicación

¿De qué manera?

El profesor demostrará cómo hacerlo y tú lo puedes ir replicando en tu computadora. Si surgen dudas las puedes compartir para resolverlas en conjunto de la mano de los tutores.

Tiempo estimado: **30 minutos**

Mocking API

¿Cómo lo hacemos? **Desarrollaremos un servidor que pueda devolver los datos de prueba que necesitamos para poder presentarlos al equipo correspondiente, para ésto, tendremos que considerar los siguientes elementos:**

- ✓ Necesitaremos crear un modelo de productos de prueba para alimentar los usuarios de prueba.
- ✓ Además, necesitaremos crear usuarios de prueba, cuyo carrito corresponda a un arreglo alimentado por los productos mock creados previamente.
- ✓ Contaremos con un endpoint `/api/users`, el cual se encargará de devolver a los usuarios de prueba.
- ✓ Además, tendremos una función `"generateUsers"` y una función `"generateProducts"`

Snapshot: app.js

JS app.js X

src > JS app.js > ...

```
1 import express from 'express';
2 import usersRouter from './routes/users.js'
3 const app = express();
4 const PORT = process.env.PORT || 8080;
5
6 app.use('/api/users', usersRouter);
7 app.listen(PORT, () => console.log(`Listening on ${PORT}`))
```

Snapshot: users.js

```
JS app.js JS users.js X
src > routes > JS users.js > [e] default
1  import { Router } from 'express';
2  import { generateUser } from '../utils.js';
3
4  const router = Router();
5
6  router.get('/', async(req,res)=>{
7    let users = []
8    for(let i=0;i<100;i++){
9      users.push(generateUser())
10   }
11   res.send({status:"success",payload:users})
12 })
13
14 export default router;
```

Snapshot: utils.js (generateUser)

Recuerda que aquí comienza la magia del mocking, así que hay que hacer npm install del módulo de faker-js, el cual se hace:

```
npm install @faker-js/faker
```

```
JS app.js JS utils.js X
src > JS utils.js > [0] generateUser
1 import {faker} from '@faker-js/faker';
2
3 faker.locale = 'es'; //Seteamos el idioma de los datos.
4
5 export const generateUser = () =>{
6   let numOfProducts = parseInt(faker.random.numeric(1,{bannedDigits:['0']}))
7   let products= [] ;
8   for(let i=0;i<numOfProducts;i++){
9     products.push(generateProduct());
10  }
11  return {
12    name: faker.name.firstName(),
13    last_name: faker.name.lastName(),
14    sex:faker.name.sex(),
15    birthDate:faker.date.birthdate(),
16    phone:faker.phone.number(),
17    products,
18    image:faker.internet.avatar(),
19    id:faker.database.mongodbObjectId(),
20    email:faker.internet.email()
21  }
22 }
```

Snapshot: utils.js (generateProduct)

```
--  
24  export const generateProduct = () =>{  
25    return {  
26      title: faker.commerce.productName(),  
27      price: faker.commerce.price(),  
28      department : faker.commerce.department(),  
29      stock: faker.random.numeric(1),  
30      id: faker.database.mongodbObjectId(),  
31      image:faker.image.image()  
32    }  
33  }
```


Snapshot: respuesta del endpoint (¿Cuántos datos!)

```
{
  "status": "success",
  "payload": [
    {
      "name": "Elvira",
      "last_name": "Orellana",
      "sex": "female",
      "birthDate": "1966-09-04T06:09:21.660Z",
      "phone": "925.886.006",
      "products": [
        {
          "title": "Fantástico Pescado",
          "price": "600.00",
          "department": "Bebes",
          "stock": "7",
          "id": "5ff32b52fef095fcfa9eeaa0",
          "image": "https://loremflickr.com/640/480/cats"
        },
        {
          "title": "Pequeño Algodón Teclado",
          "price": "407.00",
          "department": "Hogar",
          "stock": "6",
          "id": "daaaaade5ae73cf3c329ccci",
          "image": "https://loremflickr.com/640/480/cats"
        },
        {
          "title": "Sabroso Metal Gorro",
          "price": "959.00",
          "department": "Papelería",
          "stock": "1",
          "id": "af2da97eaeddd0082bb451fa",
          "image": "https://loremflickr.com/640/480/nightlife"
        },
        {
          "title": "Refinado Algodón Teclado",
          "price": "884.00",
          "department": "Decoración",
          "stock": "6",
          "id": "e0ccaddbfcd1da3eb6b0d70d5",
          "image": "https://loremflickr.com/640/480/nature"
        },
        {
          "title": "Refinado Acero Atún",
          "price": "597.00",
          "department": "Salud",
          "stock": "5",
          "id": "bcd2cf5cc1194afa15726f4",
          "image": "https://loremflickr.com/640/480/people"
        },
        {
          "title": "Sorprendente Plástico Bicicleta",
          "price": "697.00",
          "department": "Decoración",
          "stock": "5",
          "id": "0cd5aa3f7ff5df72a19defe",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Ergonómico Algodón Pizza",
          "price": "668.00",
          "department": "Deportes",
          "stock": "5",
          "id": "a2a05f46ca4e0fe92dc1bc7d",
          "image": "https://loremflickr.com/640/480/technics"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/451.jpg",
      "email": "Lucas_Guardado63@hotmail.com"
    },
    {
      "name": "Raúl",
      "last_name": "Rojas",
      "sex": "male",
      "birthDate": "1944-11-22T14:51:40.045Z",
      "phone": "970.484.950",
      "products": [
        {
          "title": "Ergonómico Metal Ensalada",
          "price": "465.00",
          "department": "Joyería",
          "stock": "7",
          "id": "5edc3b6d55cbbfbf61568e1",
          "image": "https://loremflickr.com/640/480/nightlife"
        },
        {
          "title": "Guapa Granito Teclado",
          "price": "452.00",
          "department": "Hogar",
          "stock": "2",
          "id": "0403bb73cd2233bca5c9cafc",
          "image": "https://loremflickr.com/640/480/animals"
        },
        {
          "title": "Sabroso Madera Sopa",
          "price": "353.00",
          "department": "Bricolaje",
          "stock": "7",
          "id": "389644a8fd042e4ad4bf36",
          "image": "https://loremflickr.com/640/480/cats"
        },
        {
          "title": "Sabroso Hormigon Silla",
          "price": "221.00",
          "department": "Parafarmacia",
          "stock": "7",
          "id": "eb146da593ca01eab48b69",
          "image": "https://loremflickr.com/640/480/nature"
        },
        {
          "title": "Genérico Plástico Salchichas",
          "price": "461.00",
          "department": "Bebes",
          "stock": "9",
          "id": "d9f0bdca7ed31597afd1a73e",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Sorprendente Metal Gorro",
          "price": "586.00",
          "department": "Hogar",
          "stock": "6",
          "id": "1fbfb38b4be30898d32b3bf",
          "image": "https://loremflickr.com/640/480/city"
        },
        {
          "title": "Rústico Metal Guantes",
          "price": "622.00",
          "department": "Moda",
          "stock": "6",
          "id": "f0aab7204e8b5d394fcc3eb4",
          "image": "https://loremflickr.com/640/480/nature"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/41.jpg",
      "email": "cf8ec15f6ac1d881bcecdcf",
      "email": "Maraluisa_Rivas@hotmail.com"
    },
    {
      "name": "Joaquín",
      "last_name": "Araña",
      "sex": "male",
      "birthDate": "1980-10-28T15:19:14.358Z",
      "phone": "970902537",
      "products": [
        {
          "title": "Práctico Acero Bacon",
          "price": "796.00",
          "department": "Cine",
          "stock": "9",
          "id": "190fd24ae3790d92c43b6fcf",
          "image": "https://loremflickr.com/640/480/food"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/114.jpg",
      "id": "a68f75fc9cf2fd1de8de825c",
      "email": "Manuel.Prado54@hotmail.com"
    },
    {
      "name": "Adela",
      "last_name": "Paredes",
      "sex": "male",
      "birthDate": "1968-04-20T22:00:17.283Z",
      "phone": "946-738-328",
      "products": [
        {
          "title": "Increible Metal Teclado",
          "price": "61.00",
          "department": "Salud",
          "stock": "8",
          "id": "1a80e3d7cc6501ab5ac9ecbe",
          "image": "https://loremflickr.com/640/480/animals"
        },
        {
          "title": "Pequeño Acero Teclado",
          "price": "806.00",
          "department": "Cine",
          "stock": "8",
          "id": "8595e9fbb348ba414df495c",
          "image": "https://loremflickr.com/640/480/abstract"
        },
        {
          "title": "Hecho a mano Plástico Atún",
          "price": "5.00",
          "department": "Electrónica",
          "stock": "3",
          "id": "d56ace04e84b14c8def3a0f",
          "image": "https://loremflickr.com/640/480/food"
        },
        {
          "title": "Ergonómico Dadrillo Raton",
          "price": "100.00",
          "department": "Videojuegos",
          "stock": "7",
          "id": "ac3d171ec5f99638322bcd",
          "image": "https://loremflickr.com/640/480/food"
        },
        {
          "title": "Refinado Madera Ensalada",
          "price": "612.00",
          "department": "Hogar",
          "stock": "3",
          "id": "8239b23da9df242bae9120e",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Sorprendente Metal Salchichas",
          "price": "56.00",
          "department": "Música",
          "stock": "8",
          "id": "cab9acbe2aff6e3be7c4b8a",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Refinado Madera Salchichas",
          "price": "295.00",
          "department": "Papelería",
          "stock": "5",
          "id": "bebde7b1eeddb7b347aeac4",
          "image": "https://loremflickr.com/640/480/sports"
        },
        {
          "title": "Pequeño Metal Raton",
          "price": "85.00",
          "department": "Electrónica",
          "stock": "1",
          "id": "880dc2cd0a6c3ac3e79e7d",
          "image": "https://loremflickr.com/640/480/transport"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/204.jpg",
      "id": "d0d73af89f3adeafab133f8",
      "email": "Dborca_Garcia@hotmail.com"
    },
    {
      "name": "Gonzalo",
      "last_name": "Ledesma",
      "sex": "female",
      "birthDate": "1956-04-22T21:03:43.347Z",
      "phone": "984.005.961",
      "products": [
        {
          "title": "Fantástico Granito Gorro",
          "price": "984.00",
          "department": "Mascotas",
          "stock": "2",
          "id": "cfbb3f8dfcaefc32bad4cc51",
          "image": "https://loremflickr.com/640/480/people"
        },
        {
          "title": "Rústico Acero Salchichas",
          "price": "150.00",
          "department": "Hogar",
          "stock": "1",
          "id": "7ce3cd8da423d797eadbeac6",
          "image": "https://loremflickr.com/640/480/fashion"
        },
        {
          "title": "Sorprendente Dadrillo Pescado",
          "price": "672.00",
          "department": "Informática",
          "stock": "5",
          "id": "d607cc5f14debfbcb0187e9",
          "image": "https://loremflickr.com/640/480/city"
        },
        {
          "title": "Sorprendente Madera Teclado",
          "price": "343.00",
          "department": "Joyería",
          "stock": "2",
          "id": "c4d5a99eb7cac9cd6195dec",
          "image": "https://loremflickr.com/640/480/sports"
        },
        {
          "title": "Genérico Madera Guantes",
          "price": "379.00",
          "department": "Electrónica",
          "stock": "7",
          "id": "c5b695bc315b843d0ed7d899",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Pequeño Hormigon Raton",
          "price": "225.00",
          "department": "Decoración",
          "stock": "1",
          "id": "cbf70ebcbebf83f7abbc0858",
          "image": "https://loremflickr.com/640/480/nightlife"
        },
        {
          "title": "Sorprendente Metal Camiseta",
          "price": "302.00",
          "department": "Electrónica",
          "stock": "2",
          "id": "40d333de6482aa9f70decaf1",
          "image": "https://loremflickr.com/640/480/people"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/1185.jpg",
      "id": "b1127cbcb3358fde865df7ff",
      "email": "Alejandra87@yahoo.com"
    },
    {
      "name": "Federico",
      "last_name": "Almaráz",
      "sex": "female",
      "birthDate": "1984-05-17T07:18:45.698Z",
      "phone": "901-173-379",
      "products": [
        {
          "title": "Artesanal Algodón Teclado",
          "price": "605.00",
          "department": "Moda",
          "stock": "4",
          "id": "140f61ec23bad286ac8ba8bd",
          "image": "https://loremflickr.com/640/480/cats"
        },
        {
          "title": "Sorprendente Granito Bicicleta",
          "price": "190.00",
          "department": "Moda",
          "stock": "8",
          "id": "1d5f8e7cd6ae9b79b7fae9",
          "image": "https://loremflickr.com/640/480/business"
        },
        {
          "title": "Genérico Algodón Pizza",
          "price": "906.00",
          "department": "Decoración",
          "stock": "8",
          "id": "69a86b23bccaad81f4f450f",
          "image": "https://loremflickr.com/640/480/technics"
        },
        {
          "title": "Rústico Metal Queso",
          "price": "3.00",
          "department": "Cine",
          "stock": "6",
          "id": "8f8f7ad646cc8b21c86d4168",
          "image": "https://loremflickr.com/640/480/sports"
        }
      ],
      "image": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGvixi6V76LhCkZu6pnFt5AJBiYvHy/avatar/904.png",
      "id": "f4078f47f3c12a4f8cf0716",
      "email": "Gilberto_Calderon@yahoo.com"
    }
  ]
}
```



Más peticiones

Duración: 20–25 min



ACTIVIDAD EN CLASE

Más peticiones

A partir del servidor recién desarrollado

Se nos han solicitado algunas modificaciones para con el desarrollo del mocking:

- ✓ Ahora la generación de usuarios necesita que se separen por roles, los posibles roles son:
 - cliente
 - vendedor
- ✓ Un booleano que indique si el usuario es premium (no importando el rol)
- ✓ El producto debe tener un campo "code" que sea alfanumérico
- ✓ El producto debe contar con una breve descripción, ya sea por lorem o por producto.
- ✓ El usuario debe mostrar su actual ocupación laboral.

¿Cómo sé dónde están estos campos? Vamos a investigar <https://fakerjs.dev/api/>

¡Caso resuelto!

¡Un día de trabajo más! ahora podemos generar 50-100-1000-10000 datos diferentes y todo lo necesario para que otros sectores puedan seguir trabajando.

En este caso particular estamos haciéndolo en un proyecto aislado, ¡más adelante tocará aplicarlo para funcionalidades más complejas, como pruebas de carga, de funcionalidades, etc.



¿Preguntas?

Muchas gracias.

Resumen de la parte II

- ✓ Qué es el TDD
- ✓ Qué son los mocks
- ✓ faker.js
- ✓ Aplicar los conceptos en una mocking API sencilla.

Opina y valora
esta clase

#DemocratizandoLaEducación