



Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/09/25 Grupo:TIID211



Materia:Desarrollo Apps Moviles
Profesor: Ivan Isay Guerra Lopez
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/9/25



Introducción

Se presentan prácticos de JavaScript diseñados para mejorar la competencia en el uso de funciones, módulos, promesas y manejo de operaciones asíncronas con ``Async/Await``. Estos problemas son clave en el desarrollo web de hoy, porque nos ayudan a organizar mejor el código, reutilizar las funciones y manejar tareas que necesitan esperar, como ver una base de datos o una API.

En el primer ejercicio (a) se creó un módulo (`utils.js`) con una función llamada `restar`, que se importó de un archivo principal (`main.js`). Esto permitió ejecutar pruebas con varios valores, mostrar cómo la exportación/importación en JavaScript ayuda a mantener el código organizado y reutilizable.

The screenshot shows the Visual Studio Code interface with a project named 'PAM-211'. The Explorer sidebar on the left lists the following files and folders:

- ▼ PAM-211
 - ▼ Actividad 2
 - > Practica 2
 - > Practica1
 - > Practica3
 - API.html
 - JS API.js
 - main.html
 - JS main.js
 - JS mainusuarios.js
 - usuarios.html
 - JS usuarios.js
 - utils.js
 - > Prueba
 - .gitattributes
 - MenuPAM.java

The main editor window displays the content of 'JS main.js' with the following code:

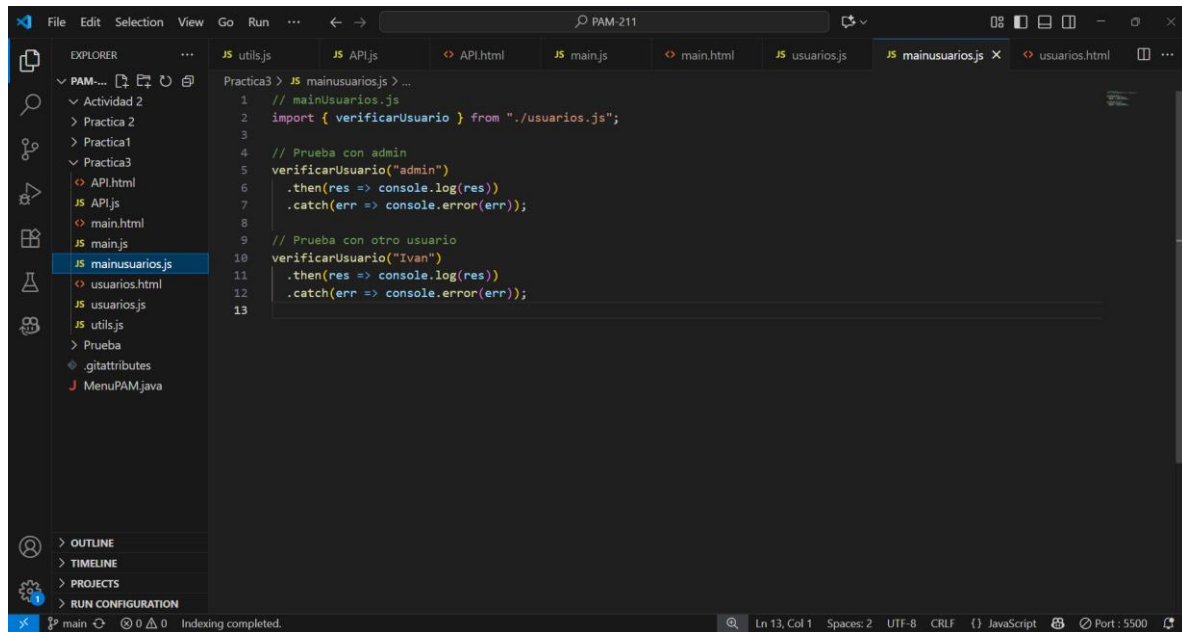
```
1 import { restar } from './utils.js';
2
3 console.log(restar(10, 5)); // 5
4 console.log(restar(20, 7)); // 13
5 console.log(restar(3, 8)); // -5
6 console.log(restar(100, 25)); // 75
7
```

The status bar at the bottom indicates the current position is 'Ln 7, Col 1', the file is encoded in 'UTF-8', and the editor is configured for 'JavaScript'.

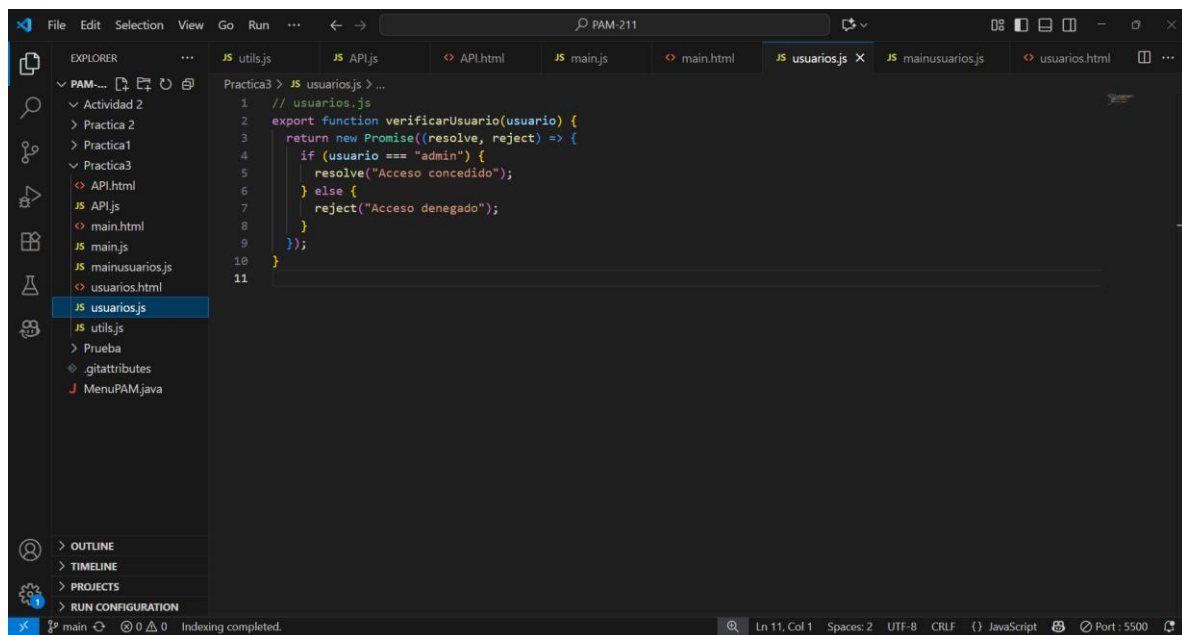


Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/09/25 Grupo:TIID211

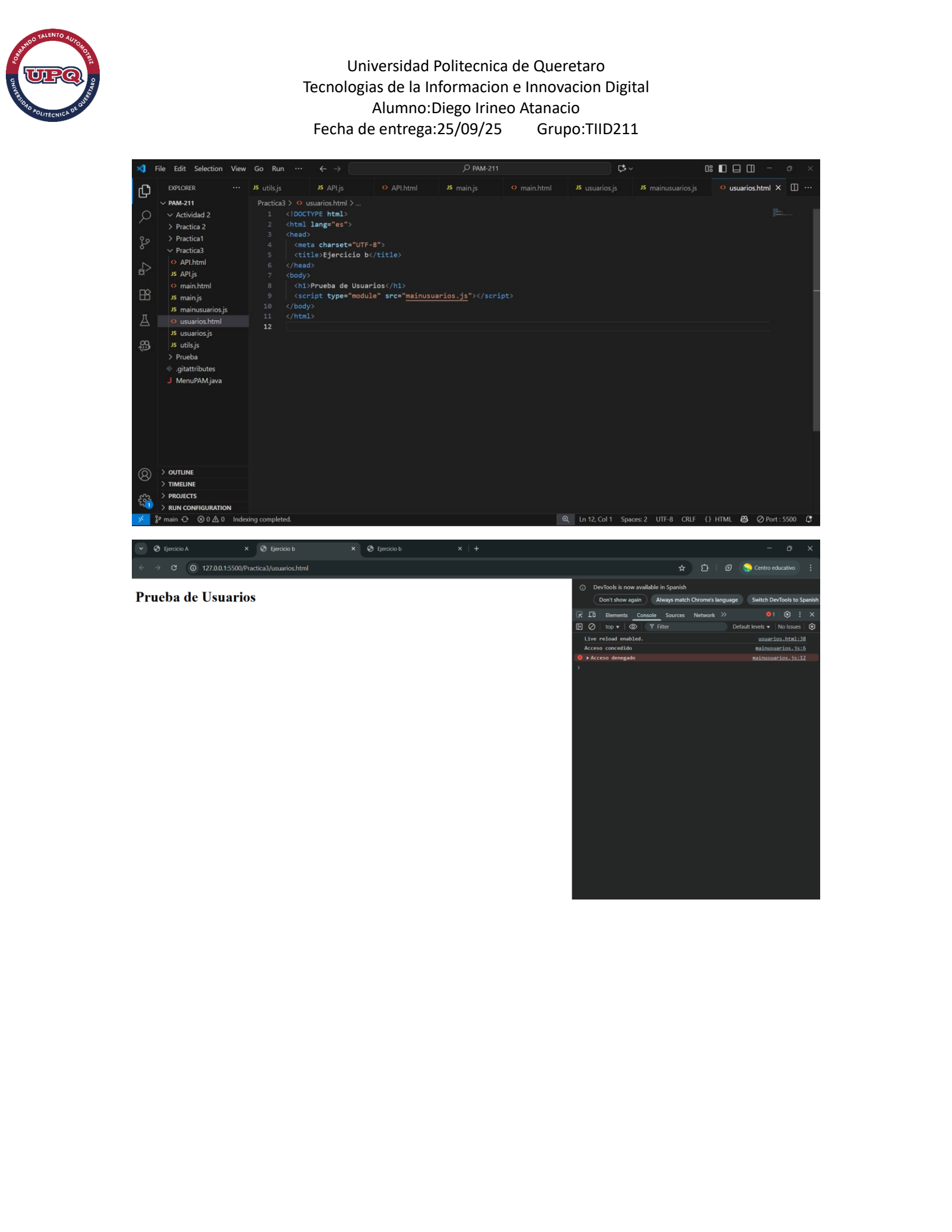
En el segundo ejercicio (b) se implementó una función llamada verificarUsuario, la cual retorna una promesa. Esta promesa se resuelve con “Acceso concedido” cuando el usuario es “admin” y se rechaza con “Acceso denegado” en cualquier otro caso. Este ejercicio permitió comprender cómo se utilizan las promesas y cómo manejar los resultados mediante los métodos `.then()` y `.catch()`.



```
1 // mainusuarios.js
2 import { verificarUsuario } from "../usuarios.js";
3
4 // Prueba con admin
5 verificarUsuario("admin")
6   .then(res => console.log(res))
7   .catch(err => console.error(err));
8
9 // Prueba con otro usuario
10 verificarUsuario("Ivan")
11   .then(res => console.log(res))
12   .catch(err => console.error(err));
13
```



```
1 // usuarios.js
2 export function verificarUsuario(usuario) {
3   return new Promise((resolve, reject) => {
4     if (usuario === "admin") {
5       resolve("Acceso concedido");
6     } else {
7       reject("Acceso denegado");
8     }
9   });
10 }
11
```



```
File Edit Selection View Go Run ... PAM-211
EXPLORER
  PAM-211
    Actividad 2
    Practica 2
    Practica1
    Practica3
      API.html
      JS API.js
      main.html
      main.js
      mainusuarios.js
      usuarios.html
      usuarios.js
      utils.js
    Prueba
      .gitattributes
      MenuPAM.java
  OUTLINE
  TIMELINE
  PROJECTS
  RUN CONFIGURATION

Practica3 > usuarios.html > ...
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Ejercicio b</title>
6 </head>
7 <body>
8   <h1>Prueba de Usuarios</h1>
9   <script type="module" src="mainusuarios.js"></script>
10 </body>
11 </html>
12
```

Ejercicio A Ejercicio b Ejercicio b +

127.0.0.1:5500/Practica3/usuarios.html

DevTools is now available in Spanish

Don't show again Always match Chrome's language Switch DevTools to Spanish

Elements Console Sources Network >>

top Filter Default levels No Issues

Live reload enabled. usuarios.html:18

Acceso concedido mainusuarios.js:6

Acceso denegado mainusuarios.js:12



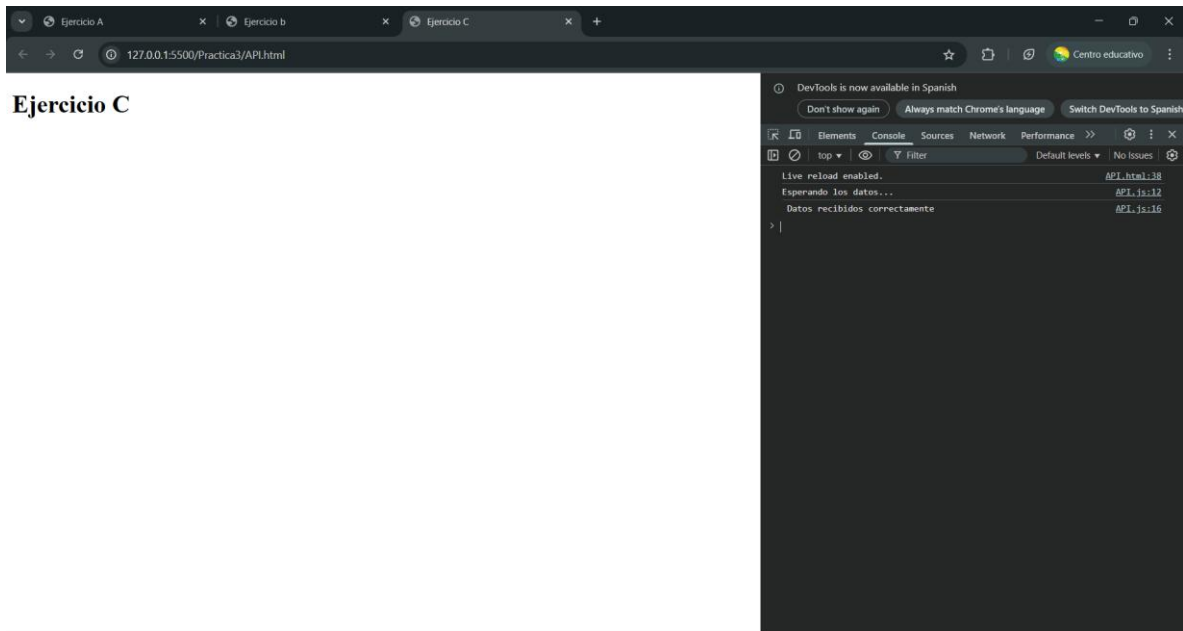
Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/09/25 Grupo:TIID211

Finalmente, en el tercer ejercicio (c) se desarrolló la función obtenerDatos, que simula una llamada a una API utilizando `setTimeout`. Para esperar la respuesta se aplicó la estructura `async/await`, mostrando primero un mensaje de “Esperando los datos...” y posteriormente “Datos recibidos correctamente”. Con ello se ilustró cómo se maneja de forma clara y legible la programación asincrónica en JavaScript.

```
1 // Función que simula una petición a una API con retraso
2 function simularPeticionAPI() {
3   return new Promise((resolve) => {
4     setTimeout(() => {
5       resolve("Datos recibidos correctamente");
6     }, 3000);
7   });
8 }
9
10 // Función async que usa await para esperar el resultado
11 async function obtenerDatos() {
12   console.log("Esperando los datos...");
13
14   try {
15     const resultado = await simularPeticionAPI();
16     console.log(resultado);
17   } catch (error) {
18     console.error("Error al obtener datos:", error);
19   }
20 }
21
22 // Llamada a la función
23 obtenerDatos();
24
```

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Ejercicio C</title>
6 </head>
7 <body>
8   <h1>Ejercicio C </h1>
9   <script type="module" src="API.js"></script>
10 </body>
11 </html>
12
```

Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/09/25 Grupo:TIID211





Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Alumno:Diego Irineo Atanacio
Fecha de entrega:25/09/25 Grupo:TIID211

Conclusion

Hacer estos ejercicios nos ayudó a obtener un mejor control de algunas cosas clave de JavaScript, como romper el código en trozos más pequeños, usar promesas para manejar cosas de asíncrono y administrar las operaciones de async con `Async/Await`. Estas herramientas son clave para que las aplicaciones de hoy funcionen sin problemas, porque le permiten administrar mejor los pasos del programa, mantener las cosas ordenadas y hacer que todo el proceso se sienta más natural cuando se trata de tareas que requieren mucho tiempo como solicitudes de servidor o base de datos.