Notas sobre el uso y creación de Promesas

1- Ejemplo de uso de promesas usando timers/promises con async/await

“Esta simetría con el código síncrono culmina con la mejora sintáctica async/await en ECMASCript 2017:”

```
import {setTimeout} from "timers/promises";

console.log('Comienzo');

async function esperar(miliSegundos) {
   await setTimeout(miliSegundos);
   console.log('Tiempo ' + miliSegundos);
}

await esperar(5000);

console.log('Terminado');
```

Fuente:
https://nodejs.org/dist/latest-v18.x/docs/api/timers.html#timerspromisessettimeoutdelay-value-options

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/async_function

2- Creando una promesa

Fuente:

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using_promises

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Promise

Crear y usar una promesa alrededor de una vieja API de callbacks

```
console.log("Comienzo");

const fun = (miliSegundos) => {
    return new Promise( (reject,resolve) => {
        if (miliSegundos == undefined) throw "Error de parametro"
        if (miliSegundos > 5000) reject("Demasiado tiempo");
        setTimeout(() => {
            resolve("Proceso");
        }, miliSegundos);
    })
}

const error = (texto) => { console.log("Error: " + texto);}
const proceso = (texto) => { console.log("Hizo esto: " + texto); }
const unTrow = (texto) => { console.log("Throw: " + texto); }
const final = () => console.log("Final");


fun()
    .then(error,proceso)
    .catch(unTrow)
    .finally(final);
```