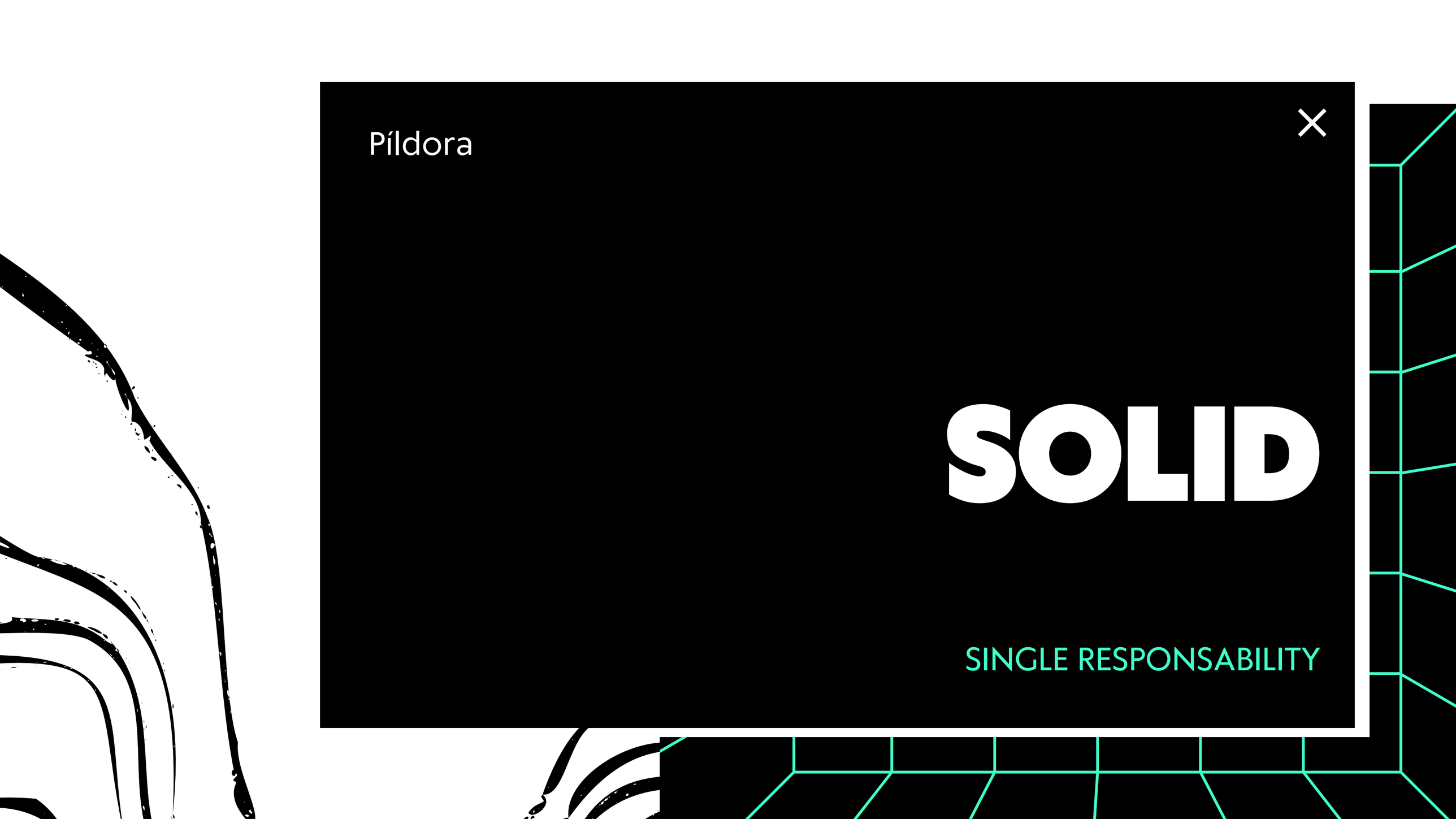
Píldora

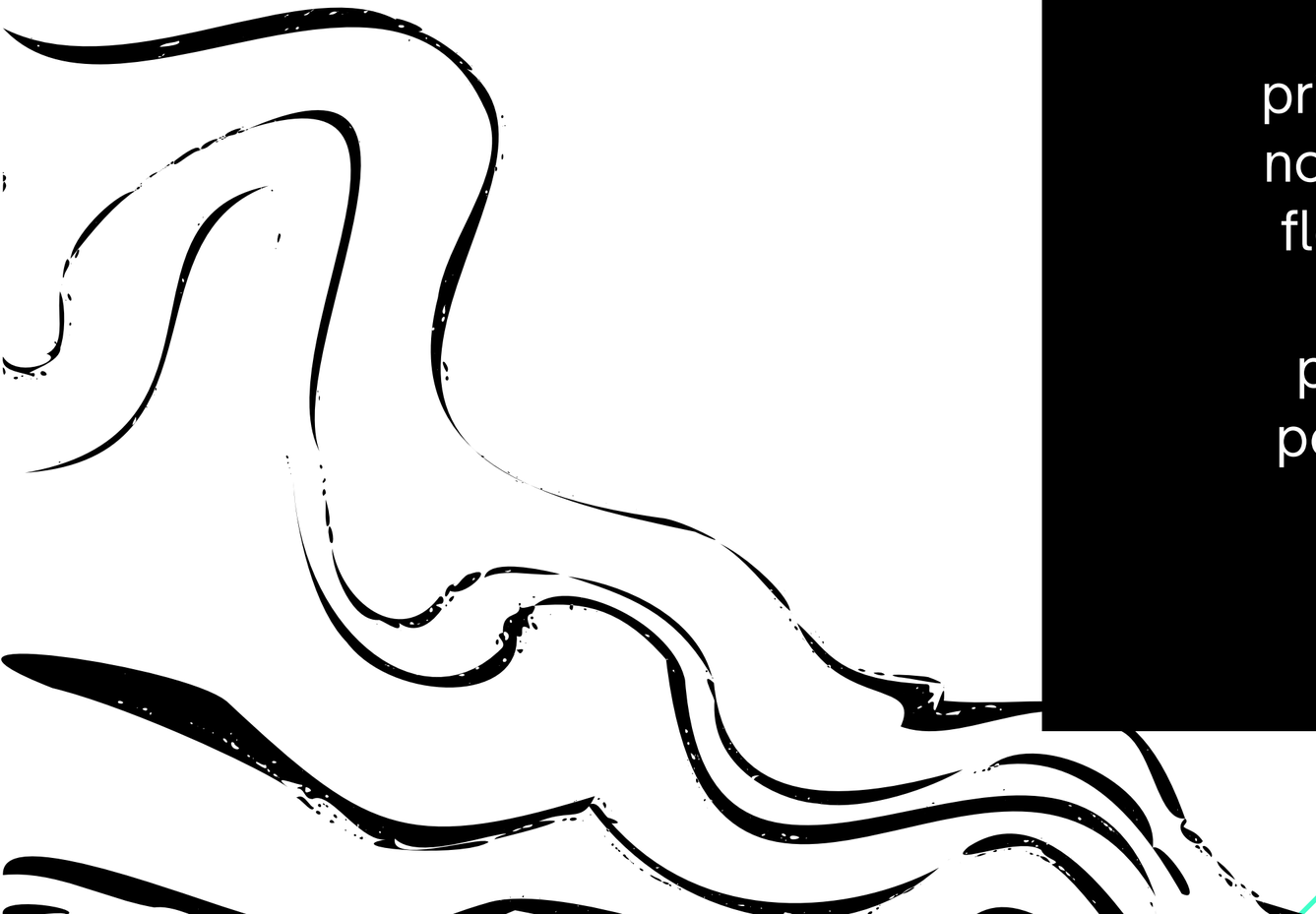
SOLID

SINGLE RESPONSABILITY



SOLID

¿Qué es?



SOLID

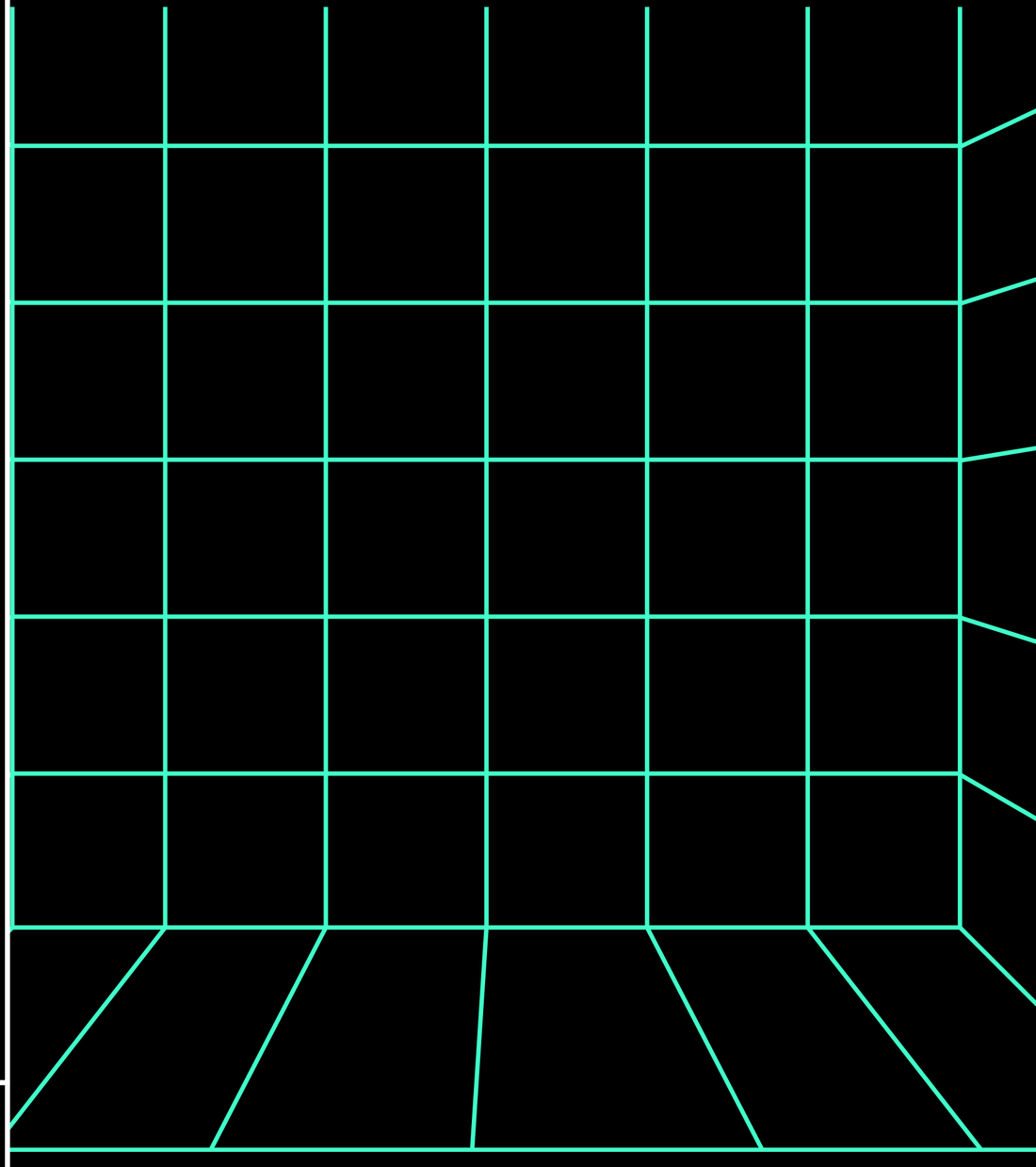
SOLID es un conjunto de cinco principios de diseño de software que nos ayudan a crear aplicaciones más flexibles, mantenibles y escalables. Estos principios son clave en la programación orientada a objetos, pero se pueden aplicar en cualquier lenguaje, incluido JavaScript.

Importancia de SOLID

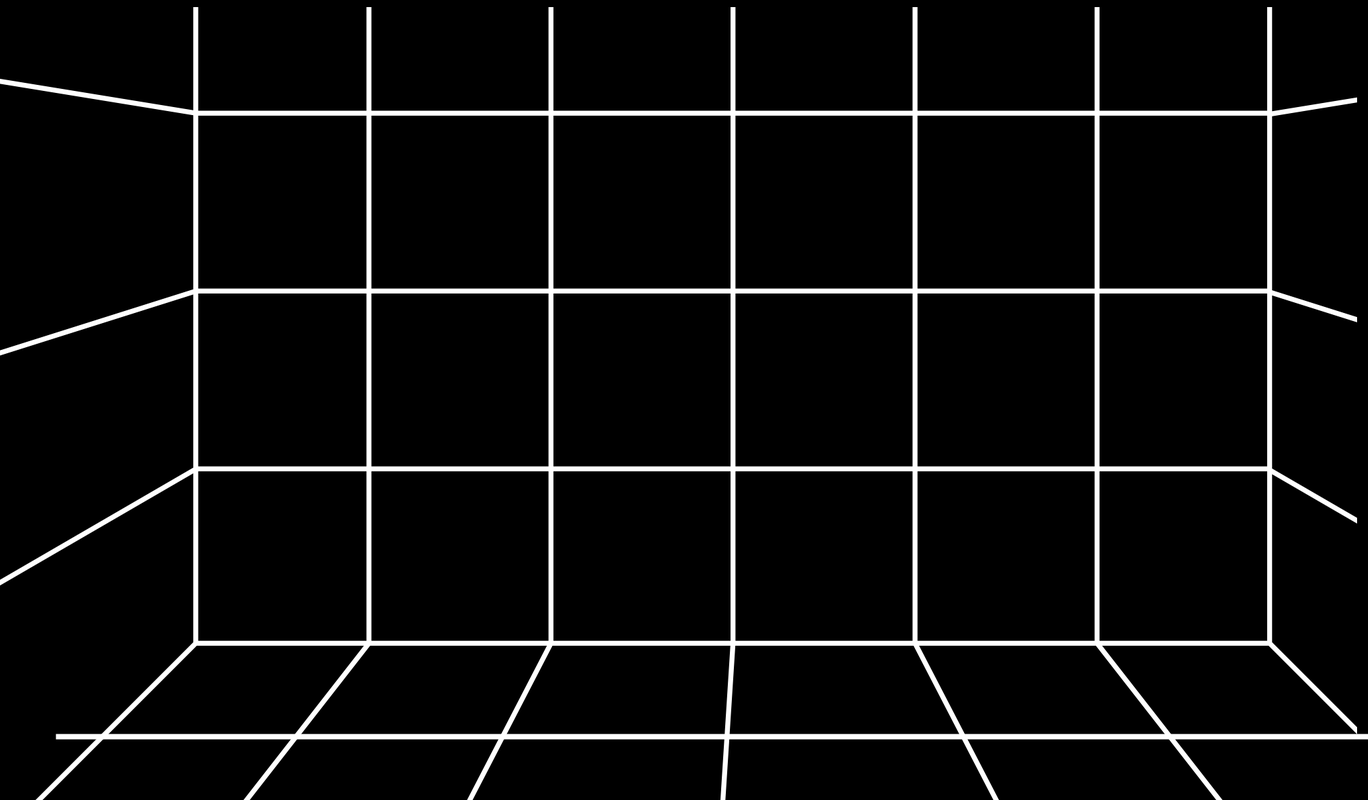
SOLID se enfoca en mejorar la calidad del software y su capacidad para manejar cambios futuros.

Los cinco principios SOLID

1. S: Principio de Responsabilidad Única (Single Responsibility Principle - SRP)
2. O: Principio de Abierto/Cerrado (Open/Closed Principle)
3. L: Principio de Sustitución de Liskov (Liskov Substitution Principle)
4. I: Principio de Segregación de Interfaces (Interface Segregation Principle)
5. D: Principio de Inversión de Dependencia (Dependency Inversion Principle)



Primer Principio: Principio de Responsabilidad Única (SRP)



¿Qué es?

El Principio de Responsabilidad Única (SRP) establece que una clase, función o módulo debe tener una única razón para cambiar, es decir, debe encargarse de una única tarea o responsabilidad.
Esto significa que cada unidad de código (como una clase o función) debe estar dedicada a una tarea específica y no mezclar responsabilidades.

EJEMPLO

Supongamos que tenemos una clase que se encarga de manejar usuarios en una aplicación y también envía emails.

```
class User {
  constructor(name, email) {
    this.name = name;
    this.email = email;
  }

  saveToDatabase() {
    console.log(`Saving ${this.name} to the database`);
    // Lógica para guardar al usuario en la base de datos
  }

  sendEmail() {
    console.log(`Sending email to ${this.email}`);
    // Lógica para enviar un email
  }
}
```

Este sería un ejemplo de violación del principio SRP

LO HAREMOS EN EL VSC

“¿Por qué es importante?”

- Facilidad de mantenimiento: Si algo va mal o necesitas cambiar una parte de tu sistema, es más fácil localizar y corregir el problema sin afectar otras partes.
- Mejora la legibilidad: Tener unidades de código con una única responsabilidad las hace más comprensibles.
- Evita el acoplamiento: Cada clase o función maneja su propia tarea sin depender excesivamente de otros componentes.
- Favorece la escalabilidad: Cuando el código está dividido en partes bien definidas, es más fácil añadir nuevas funcionalidades sin romper el sistema.

Conclusión

El Principio de Responsabilidad Única (SRP) es esencial para evitar clases y funciones demasiado grandes y difíciles de mantener. Al separar las responsabilidades, logramos un código más modular, fácil de mantener y con menos riesgos de errores cuando se realizan cambios.

Gracias por su atención.