TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

ANALISIS Y DISEÑO DE SOFTWARE

PRINCIPIO DE RESPONSABILIDAD ÚNICA (SRP)

PRINCIPIO DE RESPONSABILIDAD ÚNICA (SRP)

El Principio de Responsabilidad Única (SRP - *Single Responsibility Principle*), establece que una clase debe tener una única razón para cambiar, es decir, debe encargarse de una única responsabilidad o funcionalidad dentro del sistema. Este principio fue introducido como parte fundamental de los principios SOLID y busca mejorar la cohesión interna de las clases, reduciendo la complejidad y facilitando su mantenimiento (Leiva, 2021).

Aplicar SRP permite evitar que una clase se convierta en un "objeto todoterreno", con múltiples roles que dificultan su comprensión y reutilización. Una clase bien diseñada bajo este principio, se convierte en una unidad autónoma y específica, cuya lógica se enfoca exclusivamente en una tarea del sistema.

Problemas derivados de la violación del SRP

Cuando una clase asume más de una responsabilidad, se producen diversos inconvenientes:

- Aumento del acoplamiento interno, dificultando los cambios en una funcionalidad, sin afectar otras.
- Pruebas unitarias más complejas, dado que la clase depende de varios comportamientos diferentes.
- Riesgo de errores al modificar una parte de la lógica que está entrelazada con otras funciones.
- Difícil mantenimiento, especialmente en proyectos a largo plazo o desarrollados por múltiples personas.

Ejemplo de clase que viola SRP

Considérese el siguiente ejemplo de Java:

En este caso, la clase ReporteVentas está asumiendo tres responsabilidades distintas:

1. Lógica de negocio (calcularTotales).
2. Presentación o formato del reporte (imprimirReporte).
3. Persistencia de datos (guardarEnArchivo).

Cualquier cambio en el formato del reporte o en la forma de guardarlo en disco, obligaría a modificar esta misma clase, incluso si la lógica del cálculo de ventas permanece igual. Esto viola el SRP.

Aplicación correcta del SRP

Al aplicar SRP, la lógica anterior se puede dividir en clases separadas, cada una con una responsabilidad clara:

Con esta separación:

- CalculadoraVentas se centra solo en las operaciones matemáticas.
- FormateadorReporte se encarga exclusivamente de cómo se ve el reporte.
- GuardadorArchivo controla el almacenamiento del archivo.

Esto reduce el acoplamiento, mejora la claridad del código y permite realizar cambios aislados, sin afectar las demás clases.

Ventajas de seguir el SRP

- **Mayor mantenibilidad:** los cambios se aplican de manera focalizada.
- **Pruebas más simples:** se puede probar cada clase por separado.
- **Reutilización de código:** las clases pueden usarse en otros contextos sin duplicar lógica.
- **Facilidad para trabajar en equipo:** múltiples desarrolladores pueden trabajar en diferentes clases sin conflictos.

El Principio de Responsabilidad Única, no solo mejora la calidad del diseño orientado a objetos, sino que también promueve la claridad estructural, la modularidad y la escalabilidad del sistema (Leiva, 2021). Al limitar cada clase a una única tarea, se favorece la evolución del software sin comprometer la estabilidad del sistema, una práctica esencial en el desarrollo profesional de soluciones informáticas robustas.