DAM 1 DAW 1 / Entornos de desarrollo

# Optimización y documentación

## 4.2 Principios SOLID - Practicamos


Autor: Lisa ERIKSEN
Fecha: 2025 / 2026

# 1. Introducción

- Comprender los **principios SOLID**.
- Identificar **violaciones de SOLID** en código.
- Proponer **mejoras de diseño orientado a objetos**.
- Justificar decisiones técnicas de forma razonada.

# 2. Enunciado General

Se proporciona una serie de fragmentos de código (en pseudocódigo / Java simplificado) relacionados con la gestión de usuarios. Analiza cada caso e indica **qué principio SOLID se incumple** y **cómo podría mejorarse el diseño**.

## 2.1 Single Responsibility Principle (SRP)

Observa la siguiente clase:

```java
class UserManager {
    public void validateUser(User user) {
        // validación de datos
    }

    public void saveUser(User user) {
        // guardar en base de datos
    }

    public void sendEmail(User user) {
        // enviar email
    }
}
```

**Preguntas**:

1. Define el principio SRP en unas líneas.
2. ¿Se cumple el principio SRP? Justifica tu respuesta.
3. Propón una posible reorganización de las clases.

## 2.2 Open / Closed Principle (OCP)

Observa la siguiente clase:

```java
class DiscountCalculator {
    public double calculateDiscount(String userType) {
        if (userType.equals(anObject: "NORMAL")) {
            return 0.05;
        } else if (userType.equals(anObject: "PREMIUM")) {
            return 0.10;
        }
        return 0;
    }
}
```

**Preguntas**:

1. ¿Qué problema presenta este diseño respecto al OCP?
2. Explica cómo se podría mejorar usando herencia o interfaces.

## 2.3 Liskov Substitution Principle (LSP)

Observa la siguiente clase:

```java
class Rectangle {
    public void setWidth(int w) { }
    public void setHeight(int h) { }
}


class Square extends Rectangle {
    public void setWidth(int w) {
        super.setWidth(w);
        super.setHeight(w);
    }
}
```

**Preguntas**:

1. ¿Por qué este diseño puede violar el principio LSP?
2. Explica el problema con tus propias palabras.

## 2.4 Interface Segregation Principle (ISP)

Observa la siguiente interface:

```
interface UserActions {
    void login();
    void register();
    void generateReport();
}
```

**Preguntas**:

1. ¿Qué principio SOLID se incumple?
2. Propón una mejora del diseño.

## 2.5 Dependency Inversion Principle (DIP)

Observa la siguiente clase:

```
class UserController {

    private MySQLUserRepository repository = new MySQLUserRepository();

}
```

**Preguntas**:

1. ¿Por qué este diseño no cumple el DIP?
2. Explica cómo se podría corregir.