A close-up of a logo

Description automatically generated

**Arquitectura de software**

**Carlos González Vives**

**Grupo B**

**Paradigmas de Programación**

**3º Grado en Ingeniería Matemática e Inteligencia Artificial**

# Índice

[Introducción 3](#_Toc178845383)

[Desarrollo 4](#_Toc178845384)

# Introducción

Github: <https://github.com/carlosgonzalezcomillas/paradigmas.git>

UML:

A black background with white text

Description automatically generated

# Desarrollo

6. Principios SOLID bien aplicados

- Single Responsibility Principle (SRP):

Cada clase tiene una responsabilidad clara: PoliceCar gestiona las acciones del coche de policía, City gestiona taxis y comisaría, y SpeedRadar maneja las mediciones de velocidad. Esto hace que el código sea fácil de mantener y extender.

- Liskov Substitution Principle (LSP):

Las subclases como Taxi y PoliceCar pueden sustituir a Vehicle sin problemas, manteniendo el comportamiento esperado. Esto garantiza que se pueden usar de manera intercambiable sin romper el sistema.

- Interface Segregation Principle (ISP):

La interfaz IMessageWritter se utiliza correctamente para imprimir mensajes, separando la lógica de impresión del resto del código, lo cual mejora la modularidad.

7. Violación y corrección del SOLID

Si un coche de policía puede usar diferentes dispositivos de medida, violarías el Principio de Sustitución de Liskov (LSP) porque PoliceCar estaría acoplado a la implementación concreta de esos dispositivos. La solución es usar una interfaz común para los dispositivos, como IMeasuringDevice, lo que permite extender el sistema sin modificar PoliceCar, cumpliendo también con Inversión de Dependencias (DIP) y Abierto/Cerrado (OCP).