

**FACULTAD DE INGENIERÍA**

**Asignatura: Ingeniería de Software**

Título:

“Trabajo de investigación #01”

Docente:

Ing. Alexander Siguenza

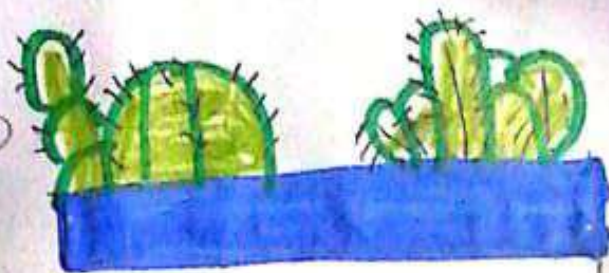
Presentado por:

Apellidos, Nombres	Carné	Carrera	Teoría
Katherine Lissette Sánchez Vila,	SV161855	Ing. en Ciencias de la Computación	01T
Salvador Alejandro González Meléndez	GM190689	Ing. en Ciencias de la Computación	01T
José Manuel Rosales Portillo	RP142494	Ing. en Ciencias de la Computación	01T
,Gerson Ernesto Lopez Chevez	LC151886	Ing. en Ciencias de la Computación	01T

Soyapango, 18 de Junio de 2021



# PATRONES DE DISEÑO

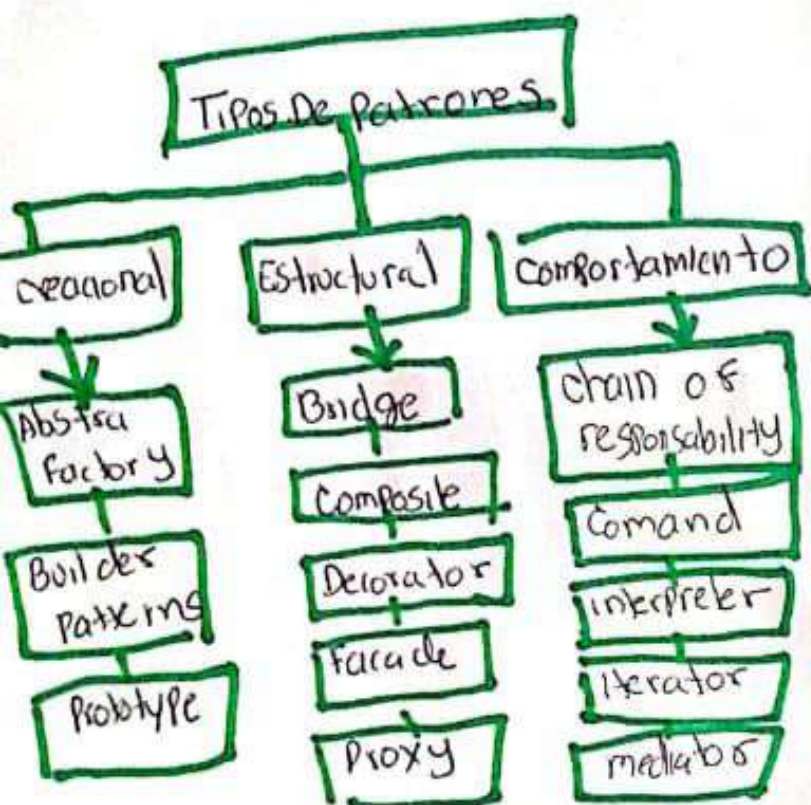


¿Que es?

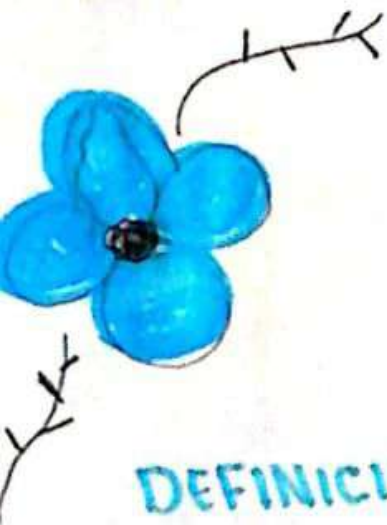


es una solución general y reutilizable aplicable a diferentes problemas. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales.

Estos patrones de diseño nos permiten crear nuestro código de manera más fácil y con estructuras de código que ya han sido probadas







# CLEAN Architecture

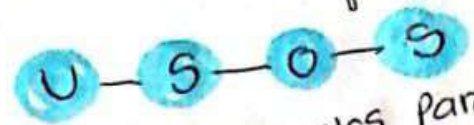


## DEFINICION

es un compendio de Principios y Patrones de desarrollo que tienen como objetivo facilitar el proceso de construcción de software así como su mantenimiento.

## BENEFICIOS

- ▶ Creación de aplicaciones desacopladas que son más fáciles de usar.
- ▶ Mayor Flexibilidad para añadir o remover funcionalidades del software.
- ▶ Diseño basándose en componentes con responsabilidades bien definidas.



Aplicaciones de negocios para proyectos.

Infraestructuras heterogeneas a nivel base de datos, servicios web, etc.

Aplicaciones pensadas para ser extendidas por terceros a través de plugins.



- enterprise Business rules
- Application Business rules
- Interface Adapters
- Frameworks y drivers

# SOLID

## ¿Qué es Solid?

Es el acrónimo que acuñó a Michael Feathers, basándose en los principios de programación orientada a objetos

SIGLAS

- **S**ingle responsibility Principle
- **O**pen/closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

Aplicar estos principios facilitará mucho el trabajo, tanto propio como ajeno



## Ventajas

- ✳ Mantenimiento del código más fácil y rápido.
- ✳ Permite añadir nuevas funcionalidades de forma más sencilla.
- ✳ Favorece una mayor reusabilidad y calidad de código, así como la encapsulación.

## Principios

### Responsabilidad única

Una clase debería tener una y solo una razón para cambiar.

**Principio Abierto/Cerrado** debemos ser capaces de extender el comportamiento de la clase sin tocarla.

**Sustitución de Liskov** Las clases derivadas deben poder sustituirse por sus clases base.

**Segregación de la interfaz** hacer interfaces que definan pocos métodos.

**Dependencias** dependen de abstracciones no de clases concretas.