



**ESCUELA PROFECIONAL:
INGENIERIA DE SISTEMAS Y COMPUTACION**

ALUMNO:AGUILAR QUISPE, Gabriela Marioret

DOCENTEN:BEJARANO FERNANDEZ, Raul

ASIGNATURA:ARQUITECTURA DE SOFTWARE

UNIVERSIDAD PERUANA LOS ANDES

ÍNDICE

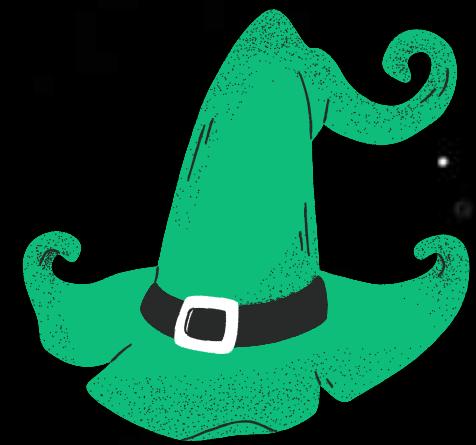
01 Introducción a la Arquitectura de Software

02 Principios y Conceptos Fundamentales de la Arquitectura de Software

03 Estándares Internacionales en Arquitectura de Software

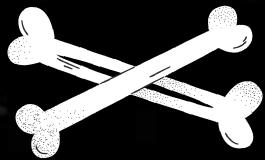
04 Estilos y Patrones Arquitectónicos

05 Importancia de la Arquitectura de Software en el Desarrollo de Sistemas

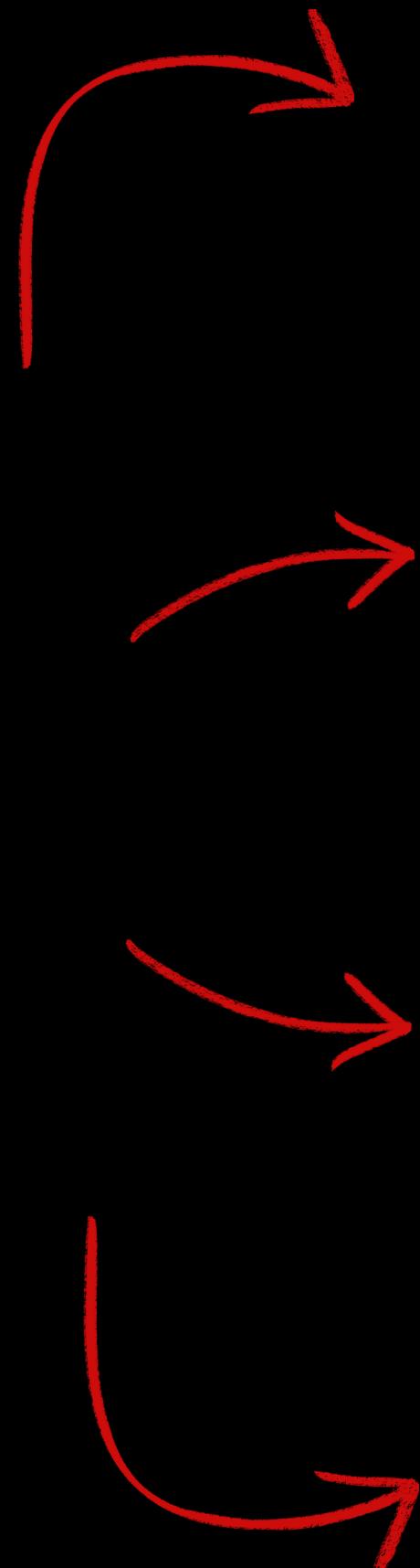


TCMA 1

Introducción a la Arquitectura de
Software



DEFINICION DE ARQUITECTURA DE SOFTWARE



ISO/IEC/IEEE
42010:2011

IEEE 1471
(predecesor)

Len Bass, Paul
Clements, Rick
Kazman

Philippe
Kruchten

De acuerdo con la norma internacional ISO/IEC/IEEE 42010:2011, la arquitectura no se limita a la organización de los módulos de software, también integra aspectos como la Comunicación entre componentes, la elección de tecnologías y los mecanismos que aseguran el cumplimiento de requisitos funcionales y no funcionales.

Son guías como ISO/IEC/IEEE 42010 que establecen cómo describir, documentar y evaluar arquitecturas de software.

Nos dice que la Arquitectura de software es el conjunto de estructuras necesarias para razonar sobre el sistema: incluye elementos de software, relaciones y propiedades (atributos de calidad). Introduce el concepto de vistas como representaciones de estructuras relevantes.

Practica la idea de que la arquitectura se describe mediante múltiples vistas concurrentes (lógica, desarrollo, procesos, despliegue + escenarios), enfocada en preocupaciones de distintos stakeholders

DIFERENCIAS ENTRE DISEÑO DE SOFTWARE Y ARQUITECTURA

Aspecto	Arquitectura de Software	Diseño de Software
Nivel	Alto nivel (visión general del sistema)	Nivel detallado (implementación interna)
Propósito	Definir la estructura general del sistema y cómo se relacionan los componentes principales	Especificar cómo se construyen internamente los componentes definidos en la arquitectura
Enfoque	Organización del sistema, módulos, capas y comunicación entre ellos	Lógica interna, clases, funciones, interfaces y algoritmos
Responsable	Arquitecto de software	Diseñador o desarrollador de software
Incluye	Selección de tecnologías, frameworks, patrones arquitectónicos, comunicación entre módulos	Diagramas UML, estructuras de clases, patrones de diseño (Singleton, Factory, etc.)
Ejemplo	Definir una arquitectura basada en microservicios que se comuniquen por API REST	Diseñar la clase Usuario que implementa la interfaz IRegistro y gestiona sesiones
Impacto en el sistema	Afecta toda la estructura del sistema y sus decisiones a largo plazo	Afecta el funcionamiento interno de partes específicas del sistema
Objetivo principal	Garantizar escalabilidad, rendimiento y mantenimiento global	Asegurar claridad, eficiencia y reutilización en el código
Representación	Diagramas de componentes, despliegue o capas	Diagramas de clases, secuencia o casos de uso

ROL DEL ARQUITECTO DE SOFTWARE EN EL CICLO DE VIDA DEL DESARROLLO

GUIA TECNICO

Es el responsable de tomar decisiones clave de arquitectura y velar porque el sistema cumpla los requisitos de calidad

RESPONSABILIDAD

El arquitecto selecciona tecnologías, frameworks y define los lineamientos que marcarán la evolución del sistema.

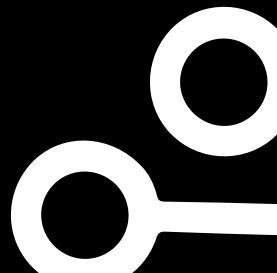
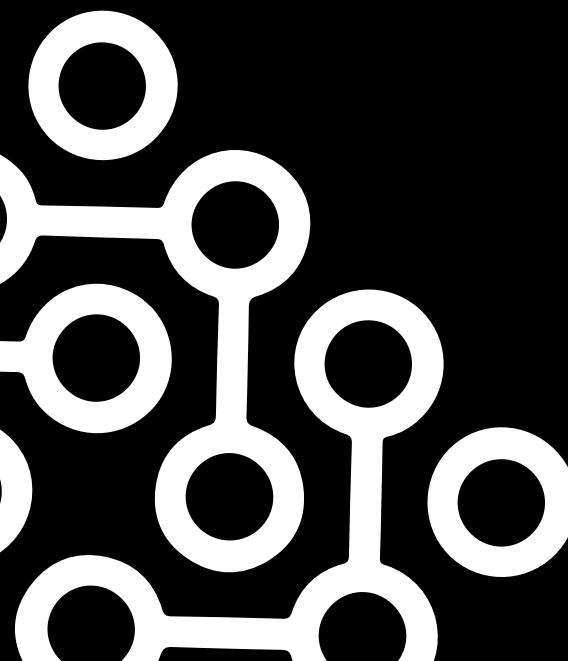


COLABORACION

Trabaja junto a analistas, desarrolladores y testers para alinear la arquitectura con las necesidades del negocio

LIDERAZGO

No solo debe tener conocimientos técnicos, también habilidades de comunicación y liderazgo para guiar al equipo

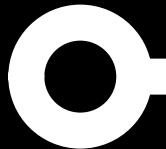


IMPACTO DE LA ARQUITECTURA EN LA CALIDAD Y EFICIENCIA DE LOS SISTEMAS

BUEN DISEÑO ARQUITECTONICO

Una buena arquitectura asegura que el sistema sea mantenible, escalable y confiable en el tiempo

1



2



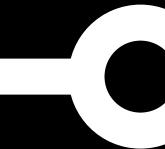
SOLUCIONES COMUNES

El uso de patrones arquitectónicos probados (microservicios, capas, hexagonal) mejora la calidad del sistema

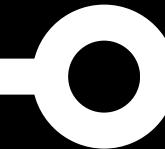
ARQUITECTURA EFICIENTE

Las decisiones arquitectónicas afectan directamente al rendimiento y la eficiencia de los recursos del sistema

3

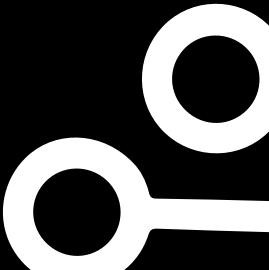
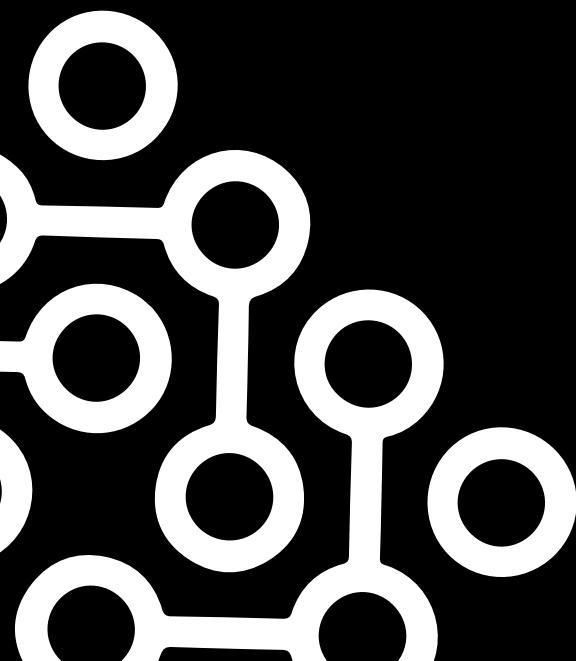


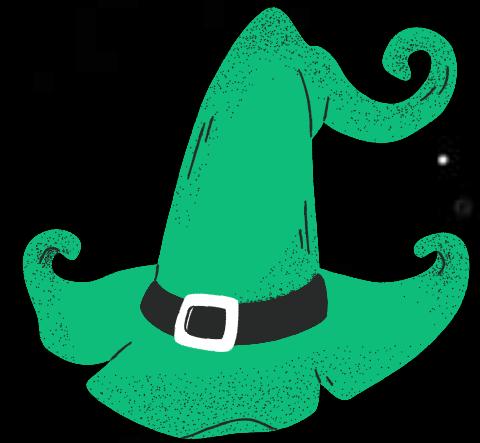
4



RIESGOS Y FALLOS

Una mala arquitectura puede llevar a retrabajo, aumento de costos y problemas graves en la operación.



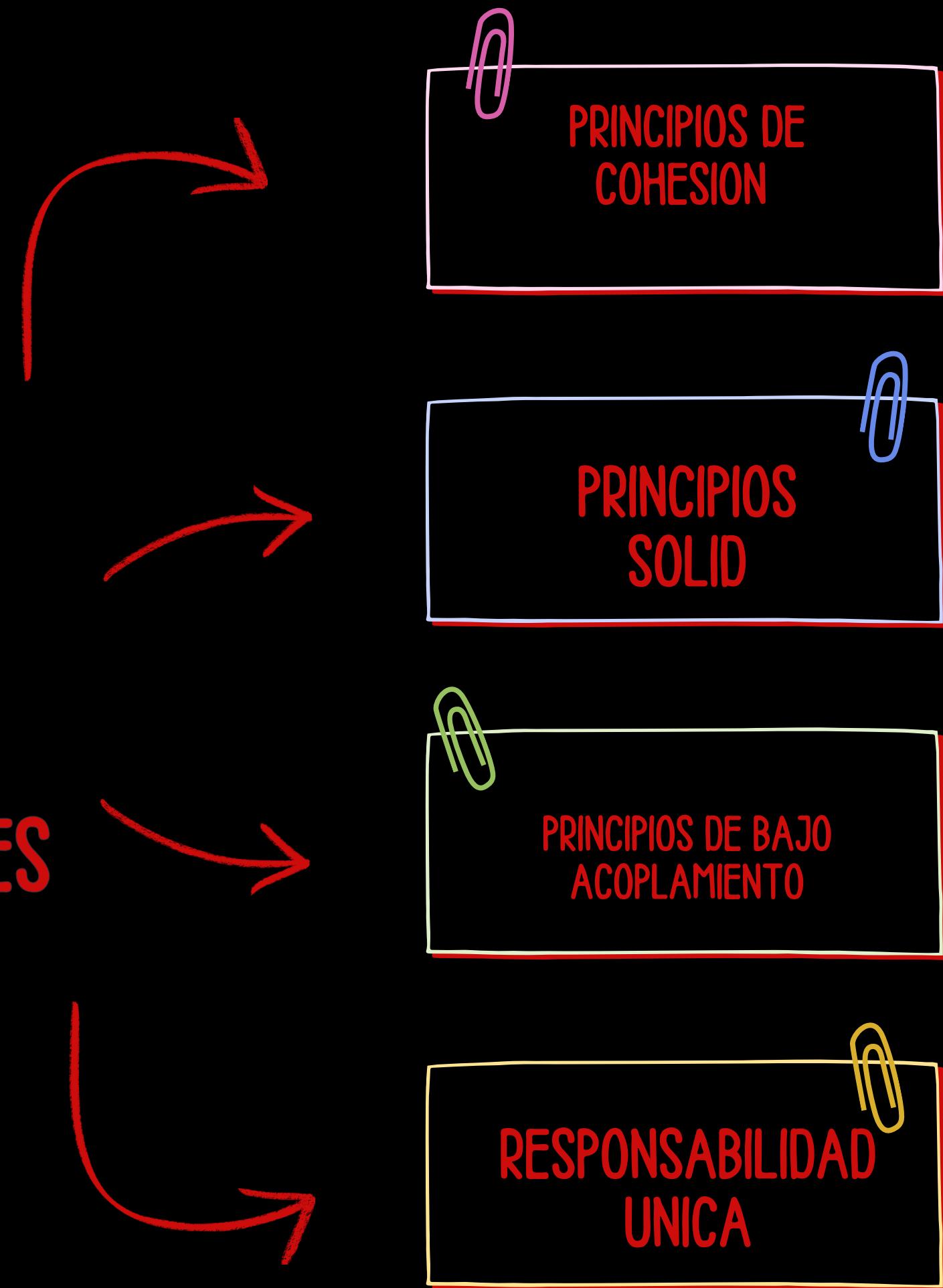


TCMA 2

* Principios y Conceptos Fundamentales de
la Arquitectura de Software *



PRINCIPIOS DE SEPARACION DE REPONSABILIDADES



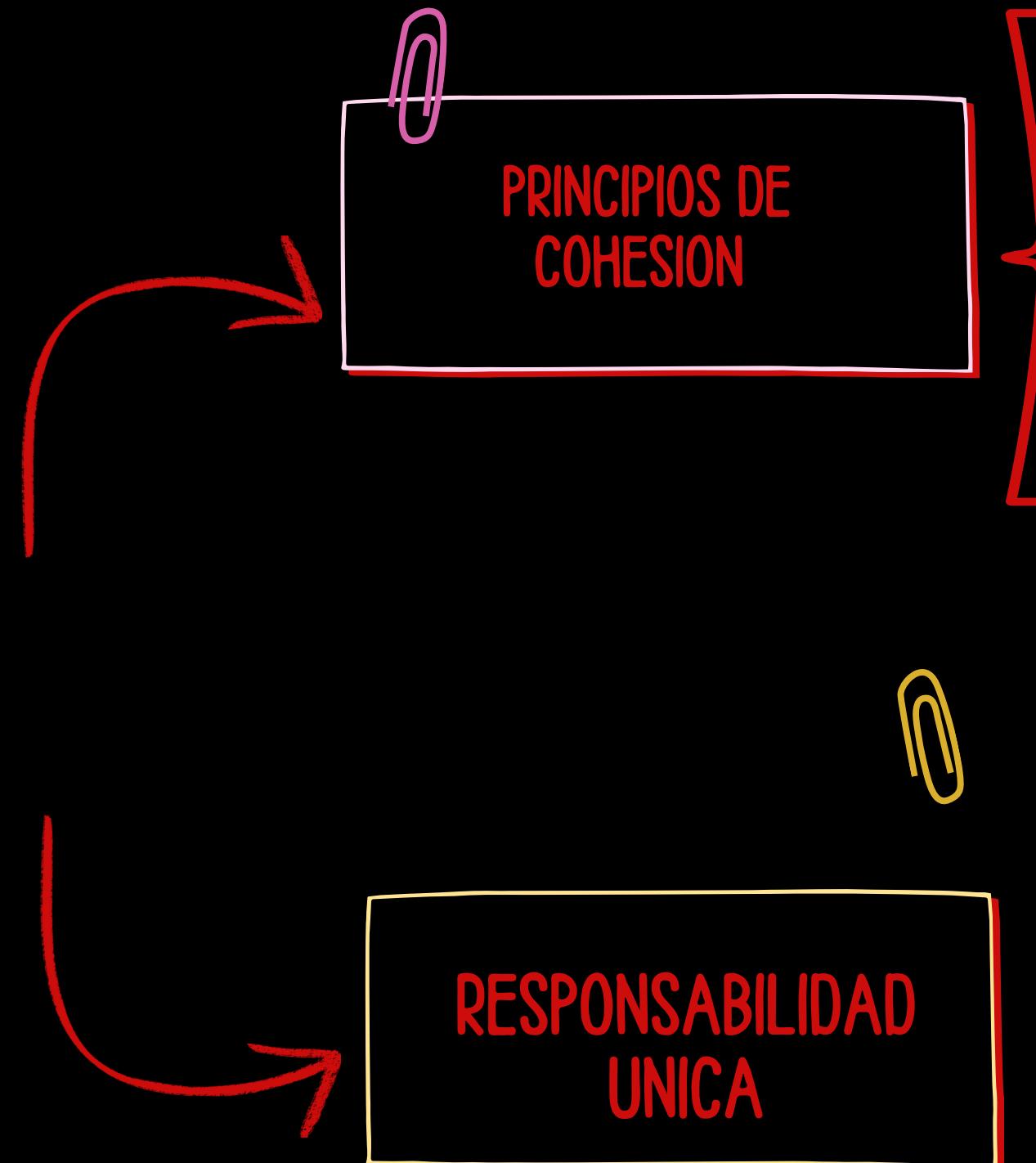
Cada modulo debe tener un unico propósito

Conjunto de reglas que facilitan mantenimiento y escalabilidad

Reducir dependencia entre componentes

Cada clase o modulo debe cumplir solo una función

COHESION Y ACOPLAMINETO EN LA ARQUITECTURA



Grado en que los elementos de un módulo están relacionados entre sí y cumplen una única responsabilidad

- Funcional (ideal)
- Secuencial
- Comunicacional
- Procedural

Cada clase o modulo debe cumplir solo una función

ABSTRACCIÓN, MODULARIDAD Y REUTILIZACIÓN



ABSTRACCIÓN DE DATOS

- Representar información sin exponer su complejidad.

MODULARIDAD

- Dividir el sistema en piezas pequeñas y manejables.

REUTILIZACIÓN DE COMPONENTES

- Usar módulos en diferentes proyectos.

INTERFACES

Definir contratos claros para la comunicación entre módulos.

ESCALABILIDAD VERTICAL

1

Aumentar recursos en un solo servidor.

ESCALABILIDAD HORIZONTAL

2

Agregar más servidores para distribuir la carga.

MANTENIBILIDAD

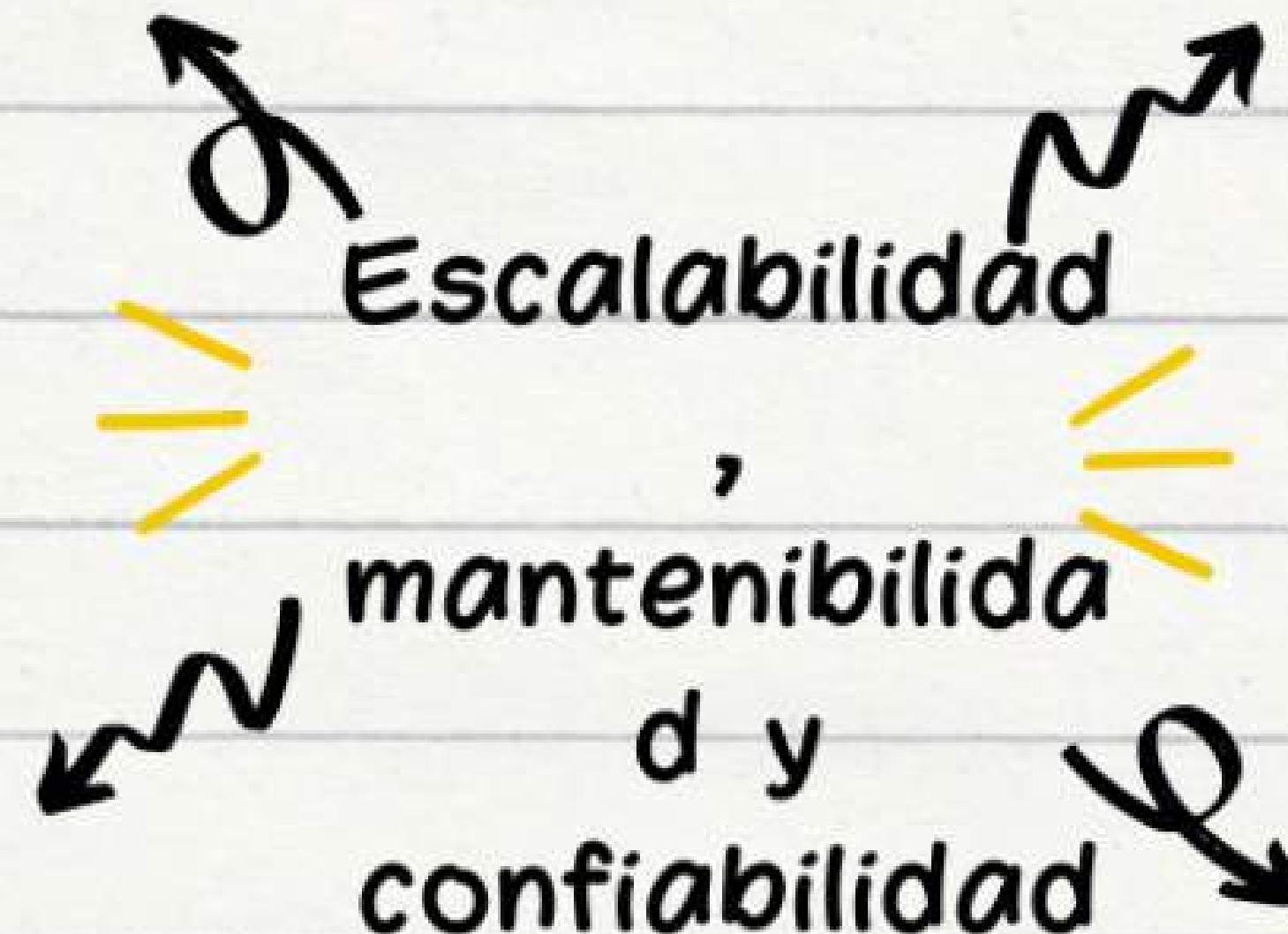
3

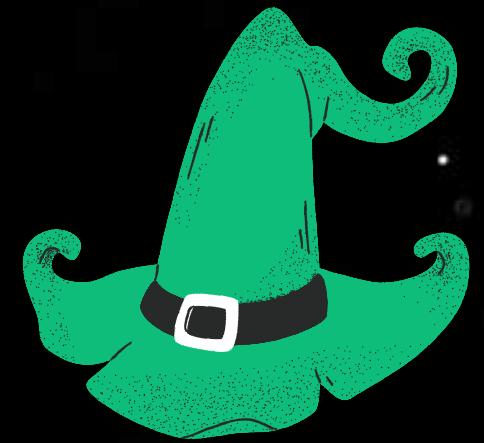
Facilidad para corregir, adaptar o mejorar un sistema.

CONFIABILIDAD

4

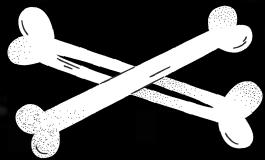
Garantizar que el sistema funcione de manera continua.



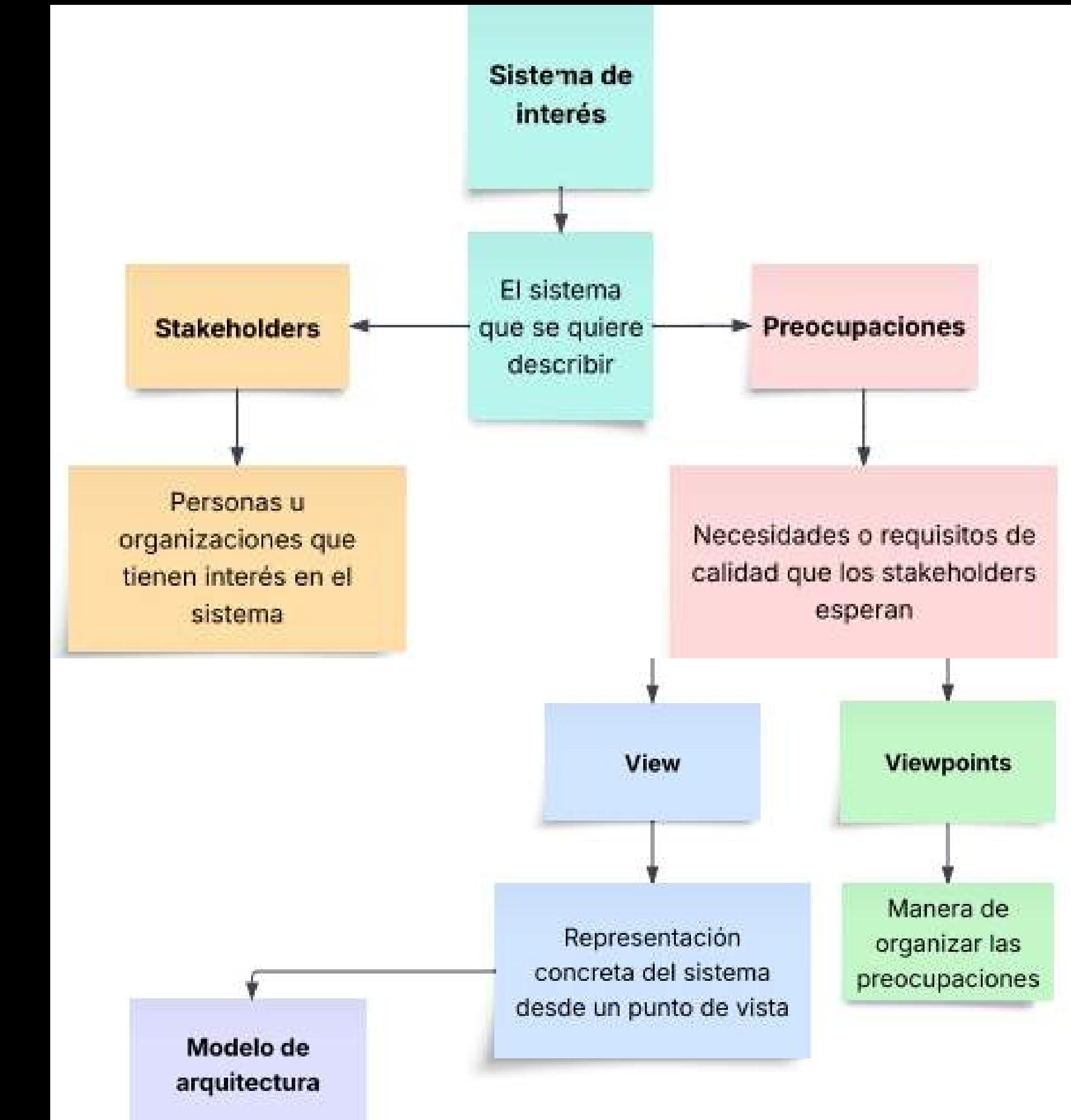


TCMA 3

Tema 3: Estándares Internacionales en
Arquitectura de Software



ISO/IEC/IEEE 42010:2011





ESTÁNDARES DE CALIDAD DE SOFTWARE

ISO/IEC 25010

- Usabilidad: Facilidad de uso del software para los usuarios finales.

- Portabilidad: Capacidad del software para adaptarse a distintos entornos.

- Eficiencia de desempeño: Uso óptimo de recursos como memoria y CPU.

- Seguridad: Protección frente a accesos no autorizados y ataques.



MARCOS DE REFERENCIA INTERNACIONALES (TOGAF, ZACHMAN)



TOGAF ADM

- Método de desarrollo arquitectónico en fases iterativas.

1

ZACHMAN FRAMEWORK

- Enfoque en perspectivas y niveles de detalle del sistema.

2

MODELADO EMPRESARIAL

Relación entre procesos de negocio y arquitectura TI.

3

GOBERNANZA

Asegurar que la arquitectura cumpla objetivos organizacionales.

4



Importancia de la estandarización en proyectos de software

Compatibilidad

Permite que diferentes sistemas trabajen juntos sin conflictos.

Comunicación clara

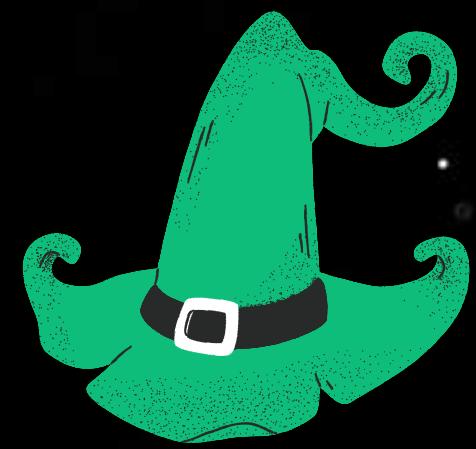
- Los estándares facilitan la interacción entre equipos y proveedores.

Reducción de errores

Un proceso estandarizado disminuye fallos en la implementación.

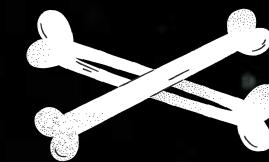
Productividad

Se ahorra tiempo al seguir guías comunes ya probadas.

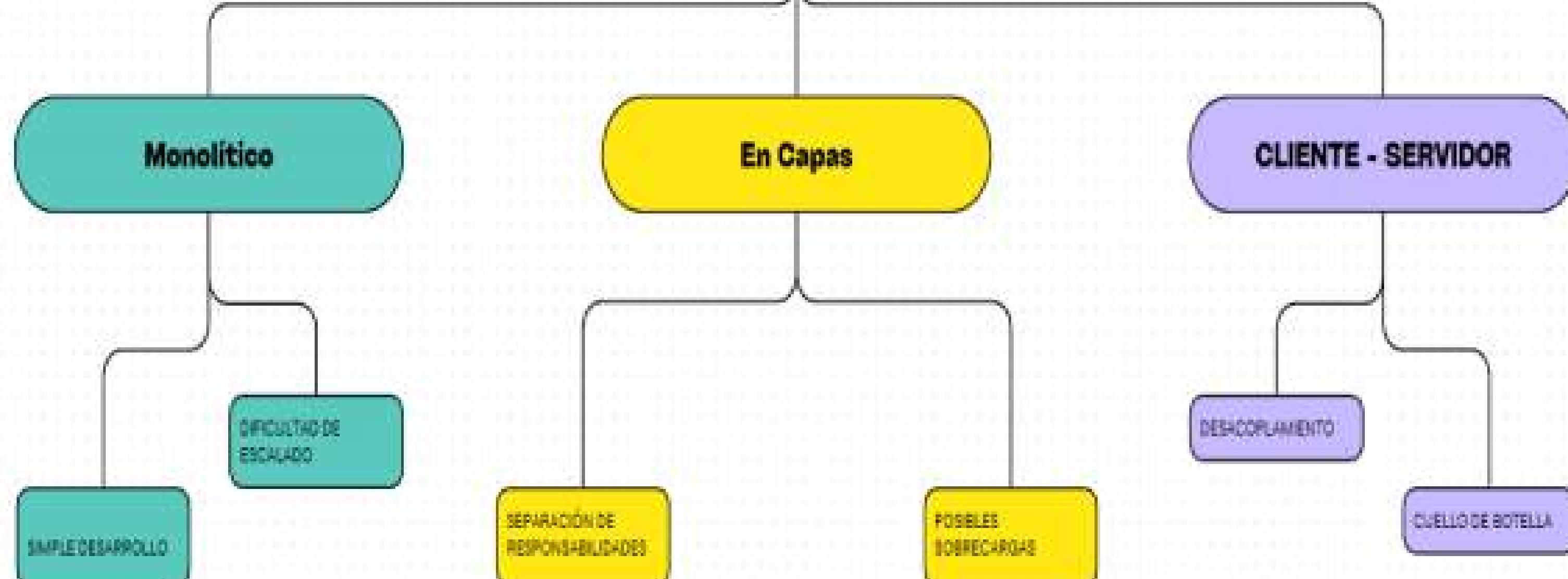


TCMA 4

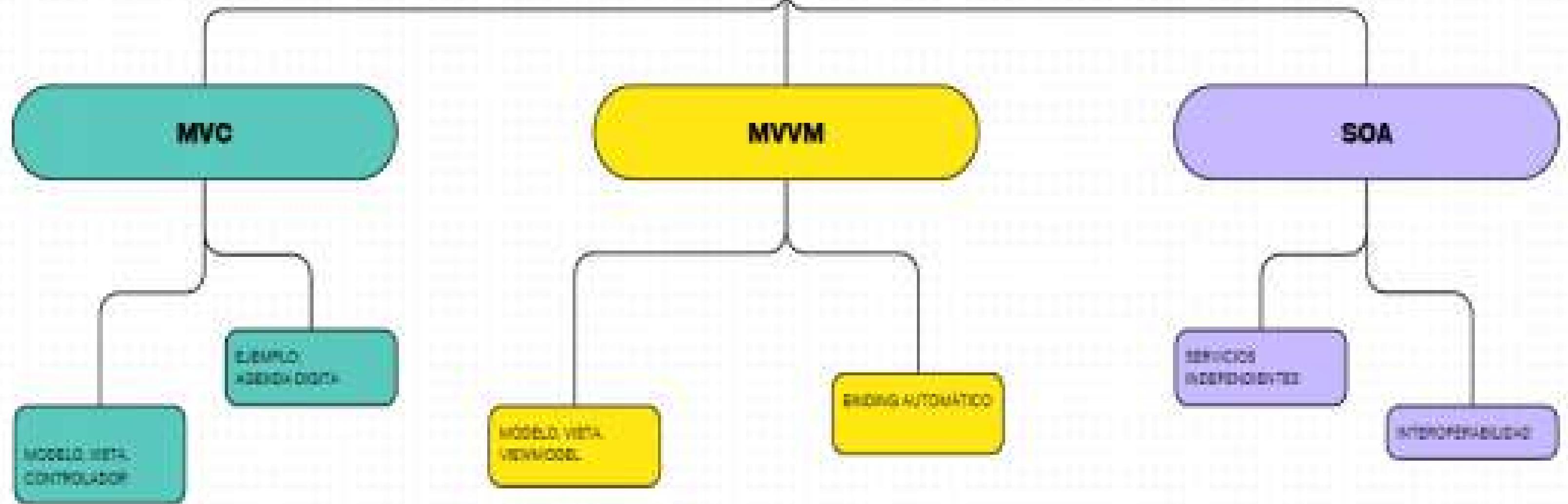
Estilos y Patrones Arquitectónicos



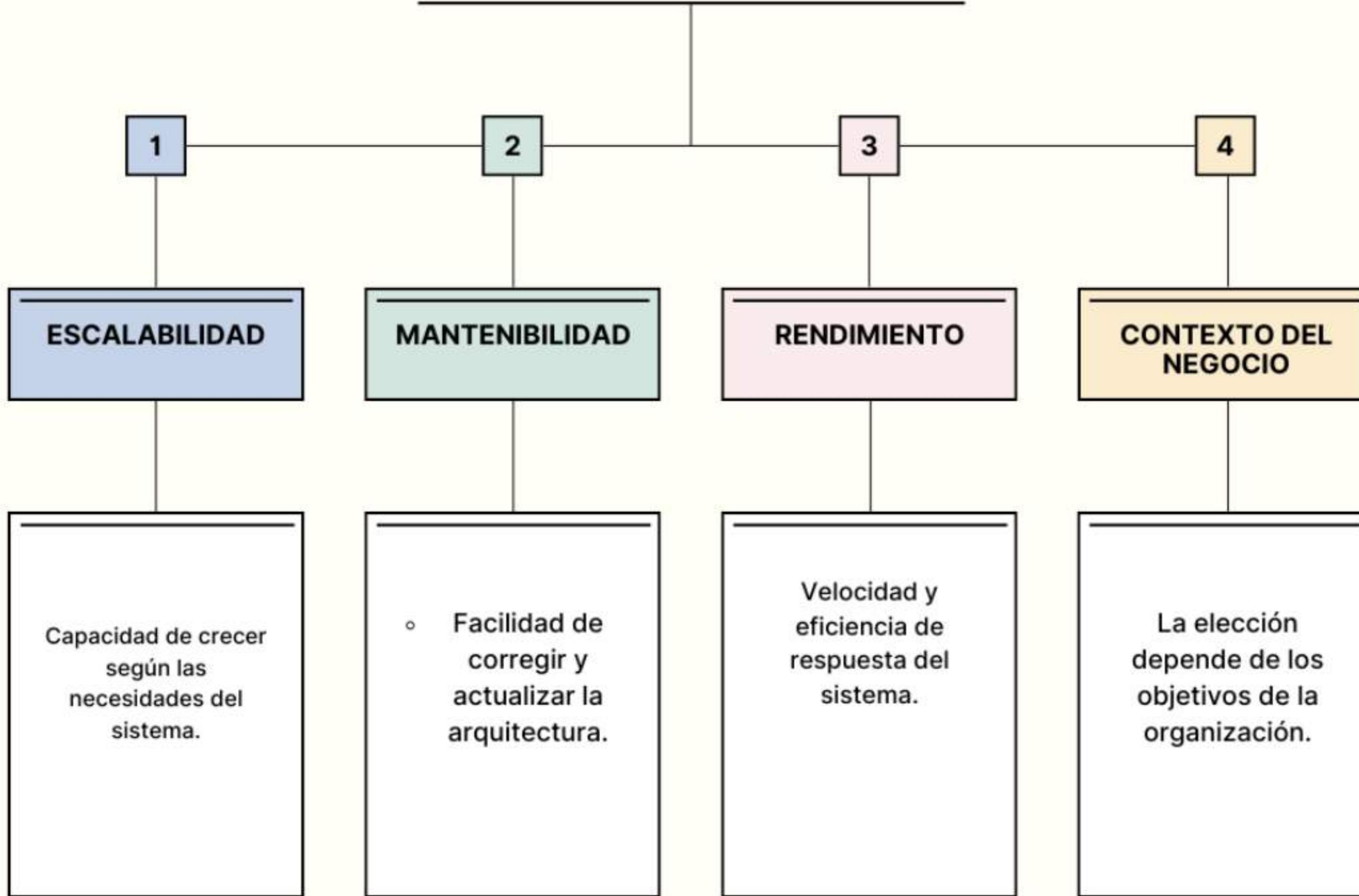
ESTILOS ARQUITECTÓNICOS



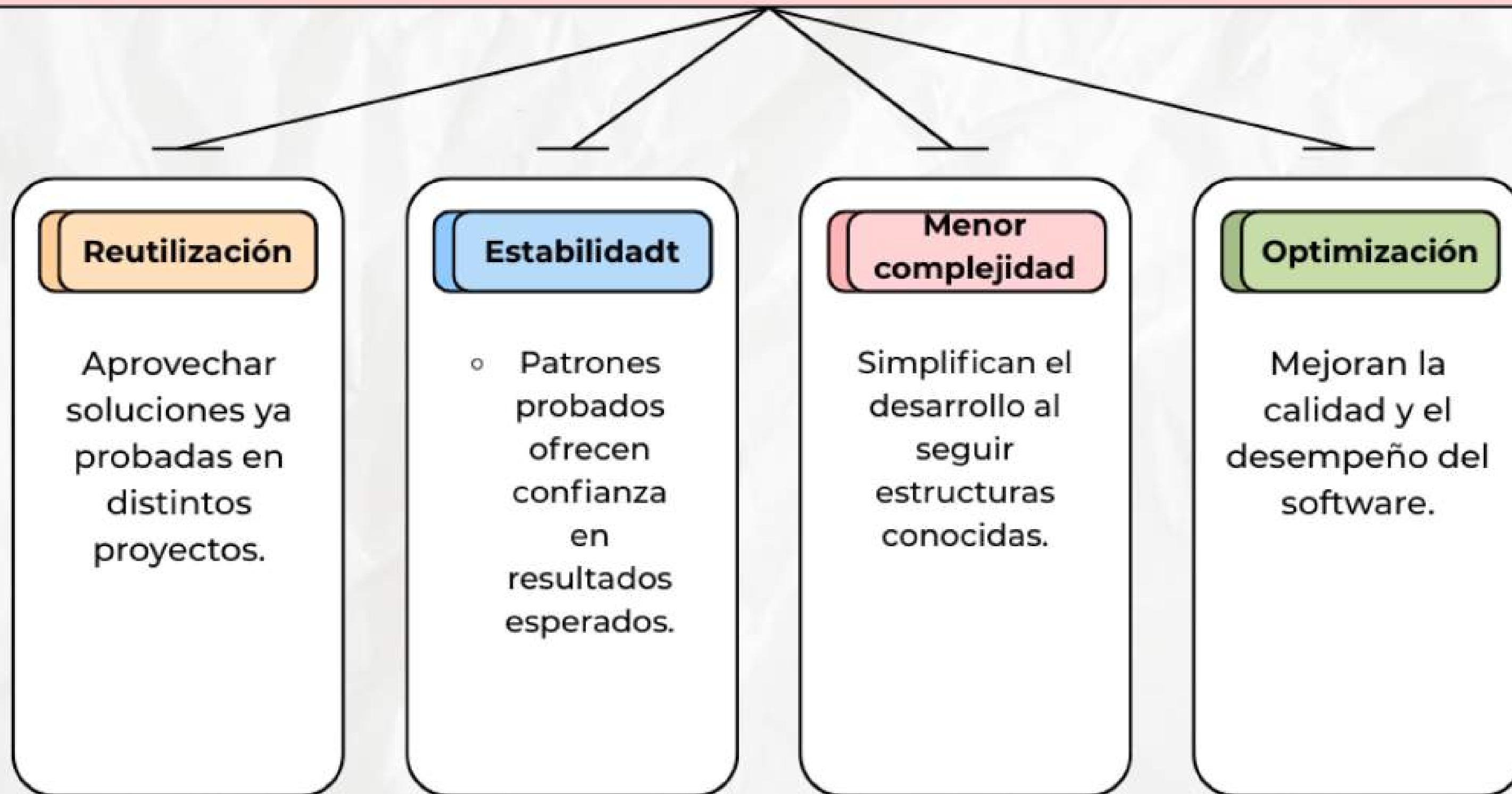
PATRONES ARQUITECTÓNICOS

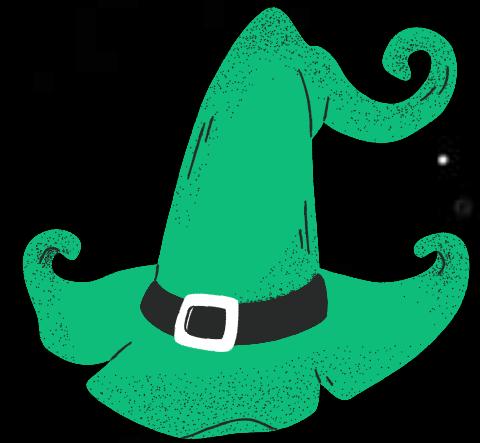


CRITERIOS DE SELECCIÓN DE UN ESTILO O PATRÓN ARQUITECTÓNICO



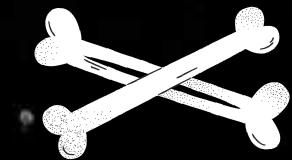
BENEFICIOS DE APLICAR PATRONES RECONOCIDOS EN LA EFICIENCIA DEL SISTEMA





TCMA 5

Importancia de la Arquitectura de
Software en el Desarrollo de Sistemas



DESARROLLO Y MANTENIMIENTO

Hoja de ruta



- La arquitectura marca el rumbo del desarrollo de software.

Control de calidad



Ayuda a garantizar que el sistema cumpla requisitos funcionales y no funcionales.

Facilidad de mantenimiento



Una buena arquitectura reduce el costo de cambios futuros.

Soporte a la evolución



Permite que el software se adapte a nuevas tecnologías o necesidades.

1 EFICIENCIA DE RECURSOS

- La arquitectura adecuada asegura un uso óptimo de CPU, memoria y red.

2 ESCALABILIDAD HORIZONTAL

Posibilidad de añadir servidores para manejar más usuarios o datos.

3 ESCALABILIDAD VERTICAL

Ampliar la capacidad de un solo servidor para aumentar rendimiento.

IMPACTO EN LA EFICIENCIA, ESCALABILIDAD Y SEGURIDAD DEL SISTEMA

4 SEGURIDAD ESTRUCTURAL

Diseñar barreras contra ataques mediante capas y protocolos.

Prevención de fallos

Reducir gastos evitando retrabajo o sistemas inefficientes.

Prevención de fallos

Anticipar errores mediante patrones de diseño robustos.

Reducción de riesgos y costos a través de una arquitectura bien definida

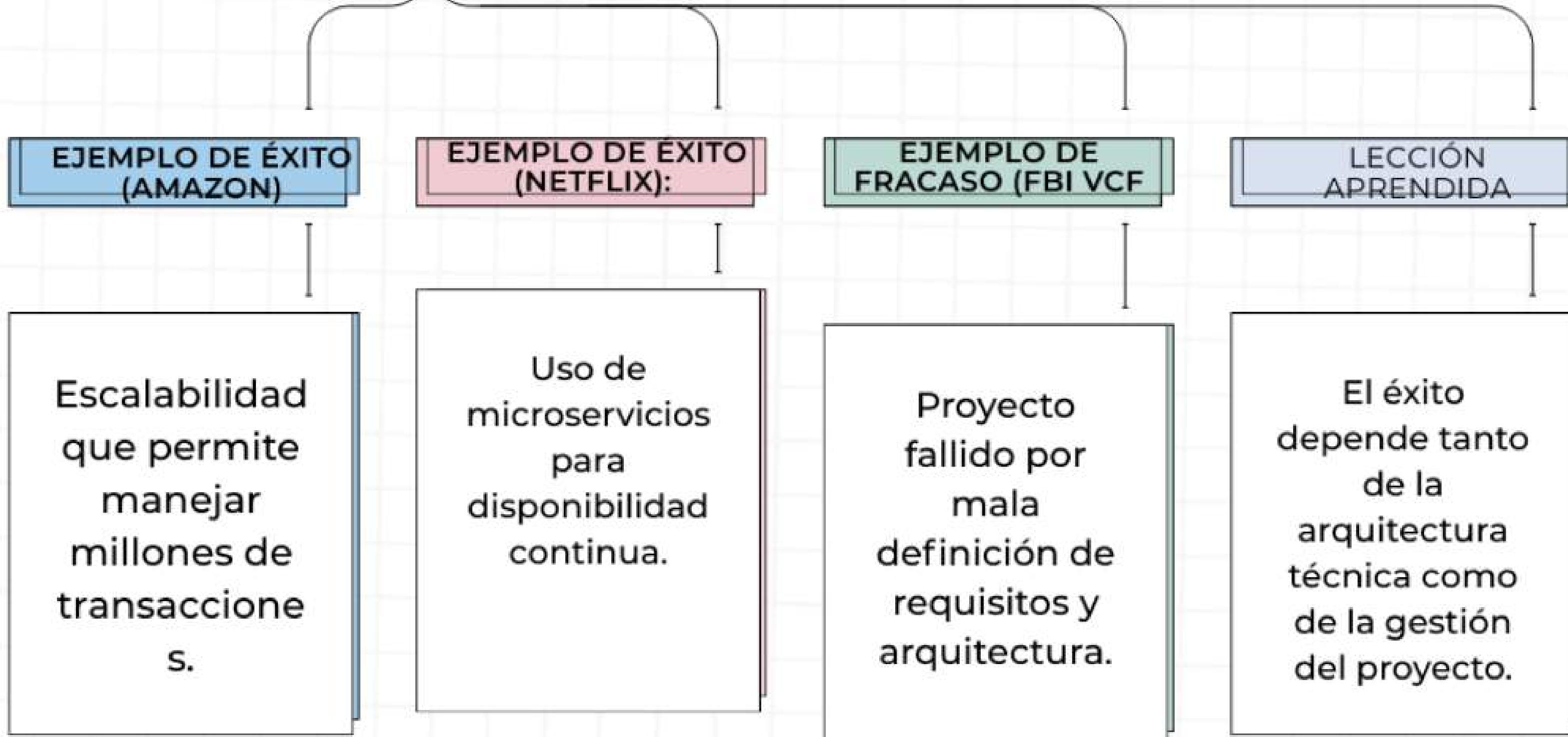
Mitigación de riesgos

Diseñar arquitecturas con redundancia y tolerancia a fallos.

Alineación con el negocio

Minimizar pérdidas asegurando que la solución soporte los objetivos empresariales.

CASOS PRÁCTICOS DE ÉXITO Y FRACASO RELACIONADOS CON LA ARQUITECTURA DE SOFTWARE



GRACIAS
