

Fundamentos de Software

Arquitectura, Metodologías y Futuro de la Ingeniería

Presentación Técnica Avanzada • 60 Diapositivas

AGENDA DE LA SESIÓN

- **01.** Orígenes y Conceptos Base
- **02.** Arquitectura de Sistemas
- **03.** Clasificación del Software
- **04.** Paradigmas de Programación
- **05.** Algoritmia y Estructuras
- **06.** El Ciclo de Vida (SDLC)
- **07.** Metodologías Ágiles y Tradicionales
- **08.** Calidad y Mantenimiento
- **09.** Ética y Licenciamiento
- **10.** Tendencias y Futuro

01. Orígenes y Conceptos

Definiendo la infraestructura invisible de la civilización digital.

¿QUÉ ES EL SOFTWARE REALMENTE?

Según el estándar [IEEE 610.12](#), el software es la suma de:



Programas

Instrucciones ejecutables por el hardware.



Documentación

Manuales, diagramas y especificaciones de diseño.



Datos

Información de configuración y procesamiento.

HARDWARE VS SOFTWARE: LA SINERGIA

Ninguno puede existir sin el otro. El hardware es la **capacidad de ejecución** y el software es la **inteligencia del propósito**.



Características del Software:

- No se fabrica, se desarrolla.
- No se "desgasta" físicamente, se deteriora lógicamente.
- Es infinitamente replicable sin costo de material.

PIONEROS DEL PENSAMIENTO

Ada Lovelace

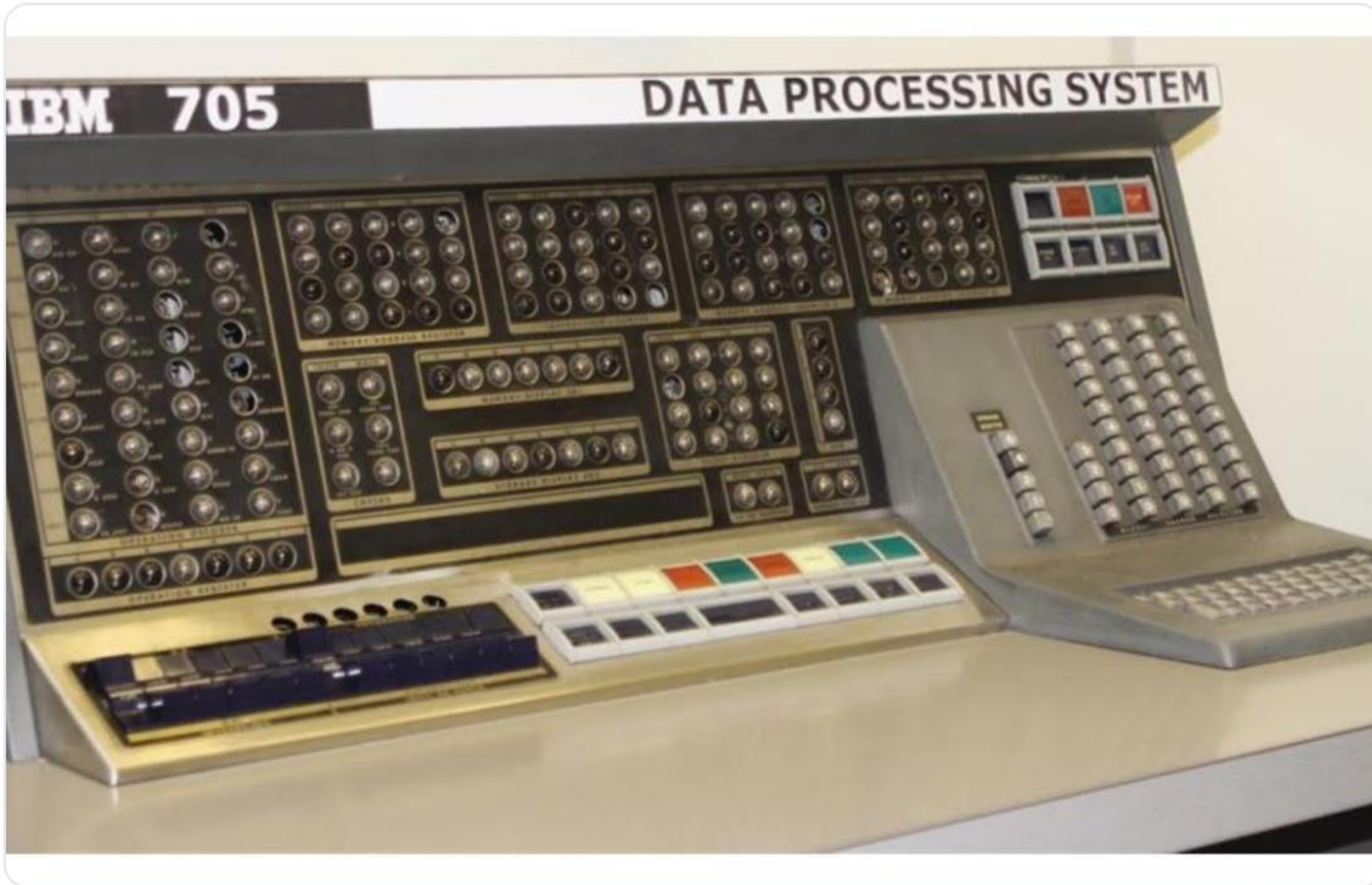
Considerada la primera programadora. Visualizó que las máquinas podían procesar más que solo números.

Alan Turing

Formalizó el concepto de algoritmo con la "Máquina de Turing", base de la computabilidad.

Ambos sentaron las bases matemáticas antes de que existiera el hardware electrónico moderno.

LA CRISIS DEL SOFTWARE (1968)



En la década de los 60, el hardware avanzó más rápido que nuestra capacidad de escribir programas fiables.

Problemas principales:

- Proyectos fuera de presupuesto.
- Software inestable y con bugs.
- Mantenimiento casi imposible.

Esto dio origen a la **Ingeniería de Software**.

02. Arquitectura de Sistemas

Cómo se organiza la interacción entre el silicio y la lógica.

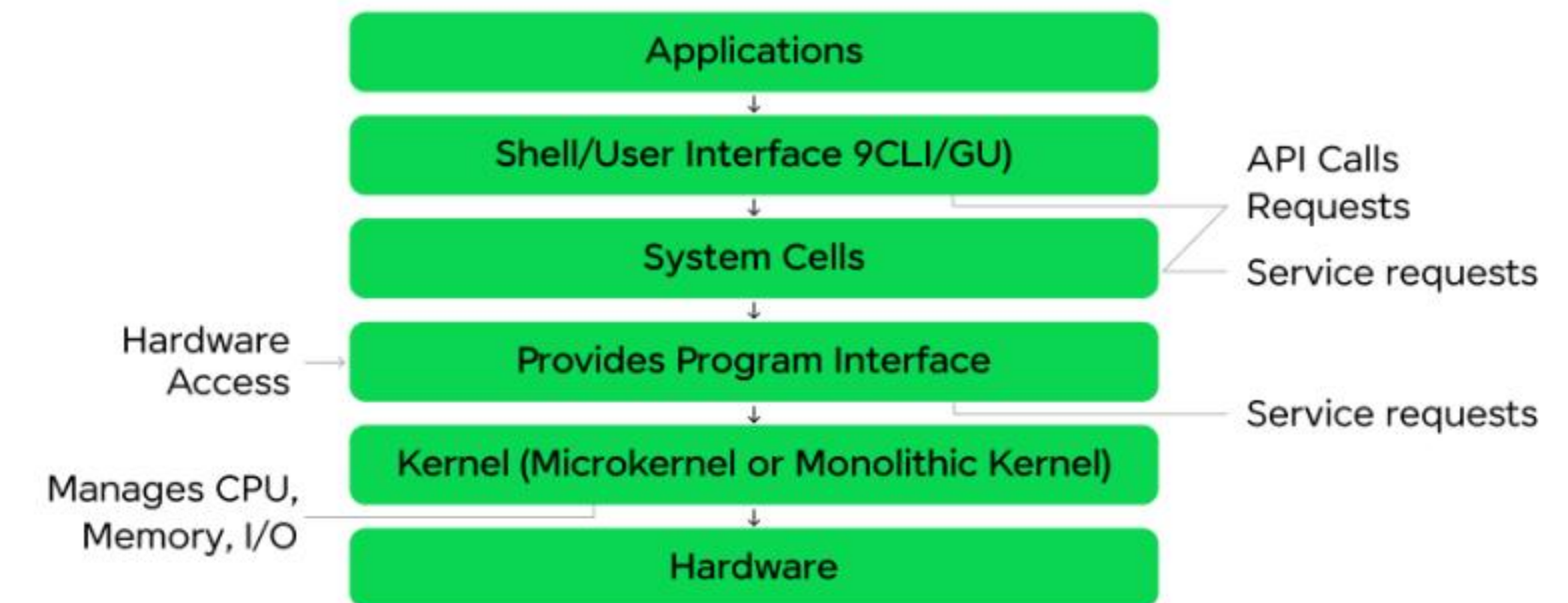
ARQUITECTURA DE CAPAS

Para gestionar la complejidad, el software se divide en niveles de abstracción:

1. **Hardware:** CPU, RAM, I/O.
2. **Kernel:** Control directo del hardware.
3. **Sistema Operativo:** Servicios base.
4. **Aplicaciones:** Lo que el usuario ve.



Key Components of Operating System Architecture



EL KERNEL: EL CORAZÓN

El núcleo es la parte del software que tiene control total sobre todo el sistema.

Gestión de Memoria

Asigna qué programa usa qué parte de la RAM.

Planificador

Decide qué proceso se ejecuta en la CPU y por cuánto tiempo.

Drivers

Traduce las peticiones del software a señales para el hardware.

CONCEPTO: MEMORIA VIRTUAL

El software hace creer a los programas que tienen más memoria de la que realmente existe físicamente.

Dirección Virtual → MMU (Unidad de Gestión de Memoria) → Dirección Física

Este nivel de abstracción permite que múltiples programas funcionen simultáneamente sin interferir entre sí (Protección de Memoria).

FIRMWARE: EL PUNTO MEDIO

Es un tipo de software específico que está grabado permanentemente en un chip de hardware.

- **BIOS/UEFI:** Arranca la placa base.
- **Firmware de Router:** Controla el tráfico de red.



Si el firmware falla, el hardware suele quedar "brickeado" (inutilizable), ya que es el nivel más bajo de control.

03. Clasificación del Software

Desde motores de búsqueda hasta sistemas de control de vuelos.

GRANDES CATEGORÍAS



Software de Sistema

Sirve para que el hardware funcione correctamente (OS, Drivers, Utilidades).

Software de Aplicación

Diseñado para que el usuario realice tareas específicas (Navegadores, Excel, Juegos).

SOFTWARE DE PROGRAMACIÓN

Son las herramientas que usan los ingenieros para crear más software:



Editores de Código

VS Code, Vim, Sublime Text.



Compiladores

Traducen lenguajes humanos a lenguaje máquina.



Depuradores

Permiten rastrear errores en el código paso a paso.

SOFTWARE EMBEBIDO (IOT)

Software diseñado para una sola tarea específica dentro de un dispositivo no informático.

- **Sistemas Críticos:** Frenos ABS de un coche, marcapasos.
- **Consumo:** Smart TVs, Lavadoras inteligentes.

Reto principal: Recursos de memoria y CPU extremadamente limitados.

SOFTWARE WEB: FRONT VS BACK

Frontend

Lo que se ejecuta en el navegador del usuario (HTML, CSS, JS).

Backend

Lo que se ejecuta en el servidor (Bases de datos, Lógica de negocio).

La comunicación entre ambos suele ocurrir a través de **APIs**.

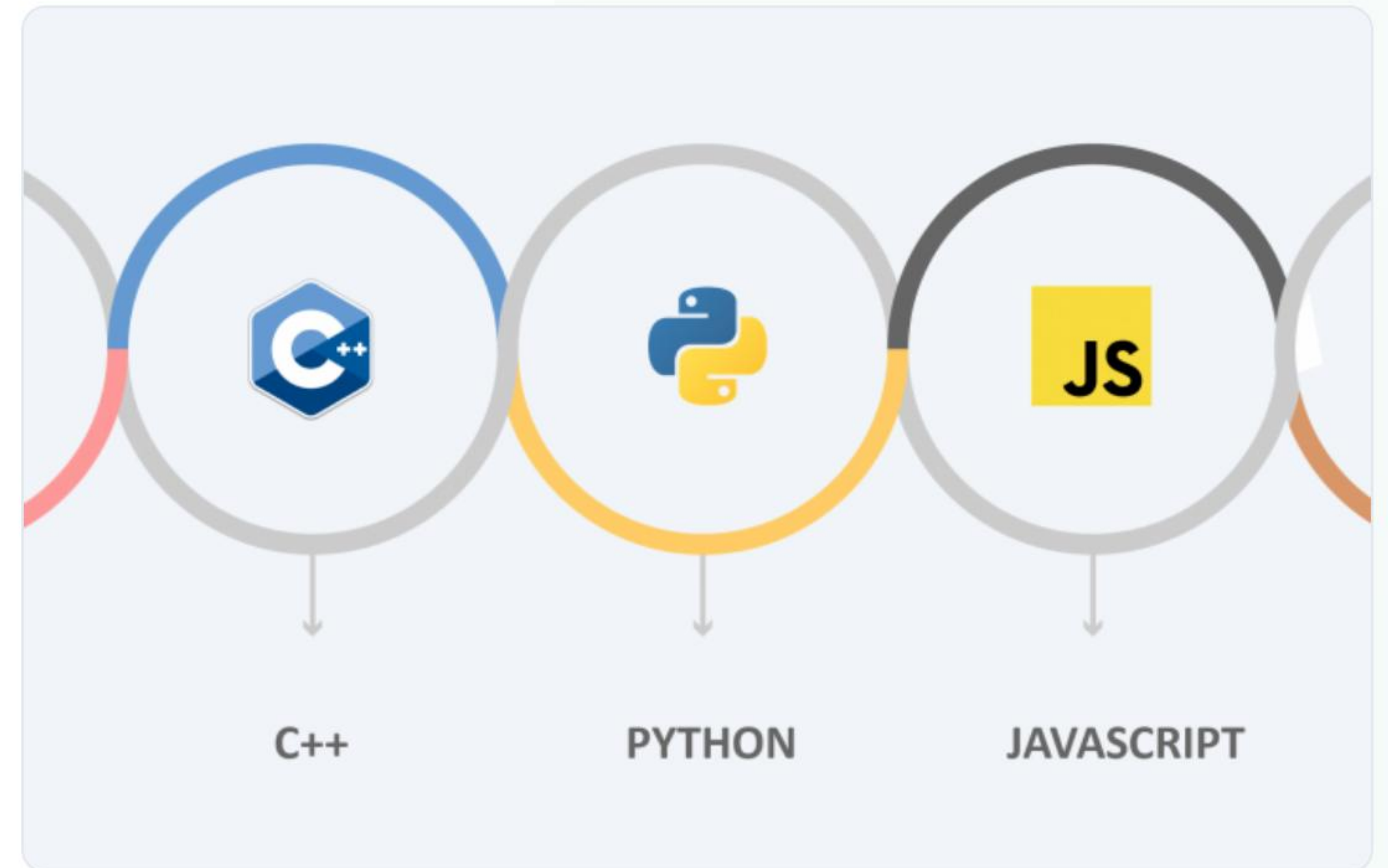
04. Paradigmas de Programación

Diferentes filosofías sobre cómo resolver problemas con código.

LENGUAJES DE PROGRAMACIÓN

Cada lenguaje tiene su fortaleza basada en su paradigma y gestión de memoria.

- **Python:** Versátil, fácil lectura.
- **Java:** Robusto, orientado a objetos.
- **C++:** Alto rendimiento, control de hardware.



IMPERATIVO VS DECLARATIVO

Imperativo

Dices **CÓMO** hacerlo paso a paso. (C, Java inicial).

Declarativo

Dices **QUÉ** quieres obtener. (SQL, HTML).

Ejemplo SQL: `SELECT name FROM users;` (No te importa cómo busca el motor, solo quieres el resultado).

ORIENTACIÓN A OBJETOS (POO)

El paradigma más usado en la industria. Se basa en 4 pilares:

Abstracción

Ocultar detalles complejos y mostrar solo lo necesario.

Encapsulamiento

Agrupar datos y métodos que los manejan en una unidad.

Herencia

Crear nuevas clases basadas en clases existentes.

Permite modelar el software como entidades del mundo real.

COMPILADO VS INTERPRETADO

Característica	Compilado (C, Rust)	Interpretado (Python, JS)
Velocidad	Muy rápida (Binario directo)	Más lenta (Capa intermedia)
Desarrollo	Ciclo más lento (hay que esperar)	Instantáneo (escribes y corres)
Errores	Se detectan antes de ejecutar	Se detectan durante la ejecución

05. Algoritmia y Estructuras

La lógica matemática que sustenta el procesamiento de datos.

¿QUÉ ES UN ALGORITMO?

Es una serie de pasos finitos y no ambiguos para resolver un problema.

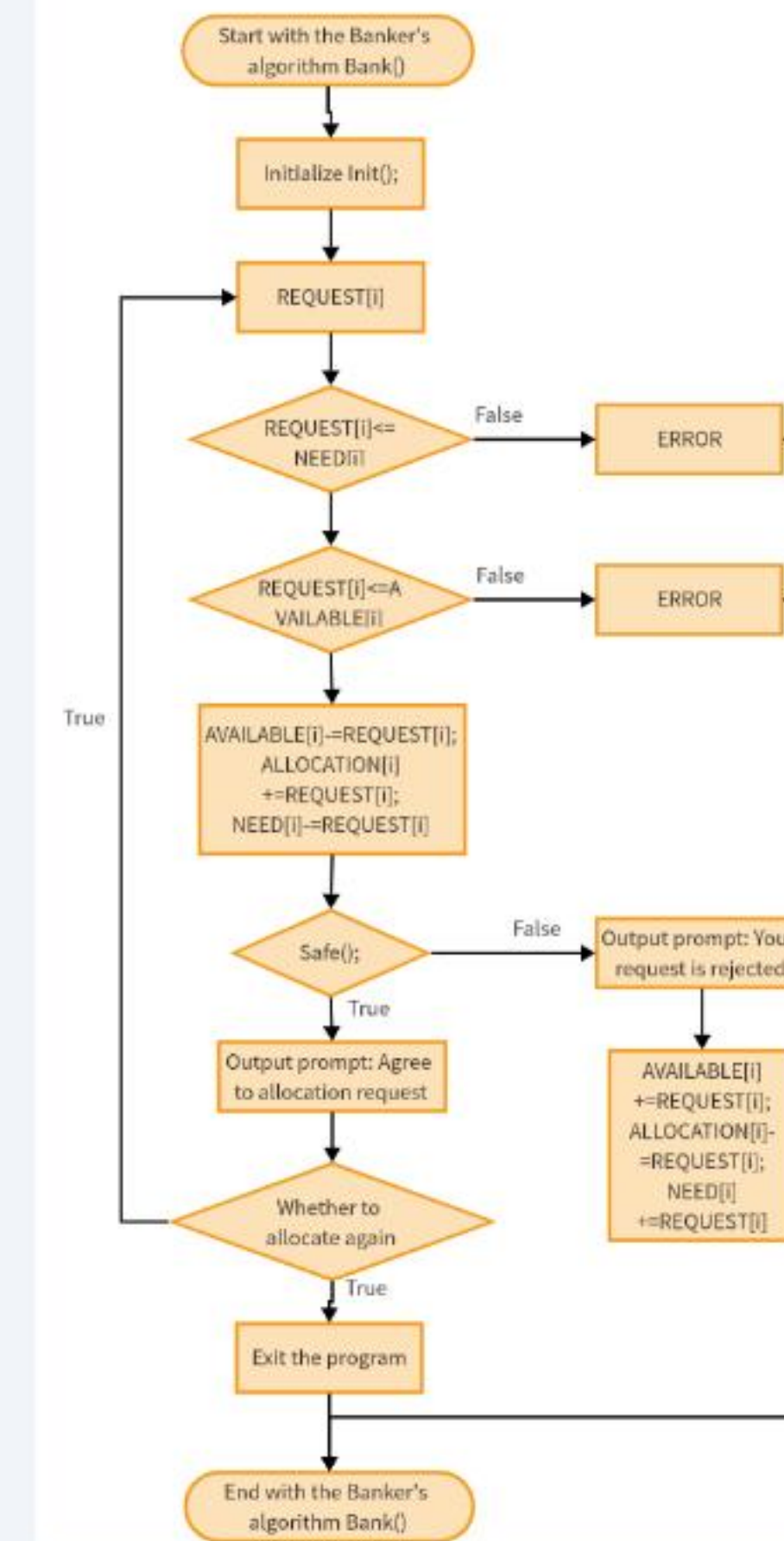
Entrada



Proceso



Salida



COMPLEJIDAD (NOTACIÓN BIG O)

Mide cómo escala un algoritmo cuando crecen los datos.

$O(1)$

Tiempo constante. No importa si hay 1 o 1 millón de datos.

$O(n)$

Tiempo lineal. Si doblas los datos, doblas el tiempo.

$O(n^2)$

Tiempo cuadrático. ¡Peligroso! El tiempo crece exponencialmente.

ESTRUCTURAS DE DATOS BÁSICAS

Arrays (Arreglos)

Elementos contiguos en memoria. Acceso rápido por índice.

Listas Enlazadas

Elementos dispersos conectados por punteros.
Flexibles para insertar.

La elección de la estructura afecta directamente al rendimiento del software.

PILAS (STACKS) Y COLAS (QUEUES)

Pila (LIFO)

Last-In, First-Out. Como una pila de platos. Usado en el botón "Atrás" del navegador.

Cola (FIFO)

First-In, First-Out. Como una fila del banco. Usado en impresoras o procesamiento de mensajes.

ESTRUCTURAS JERÁRQUICAS

Cuando los datos no son lineales:

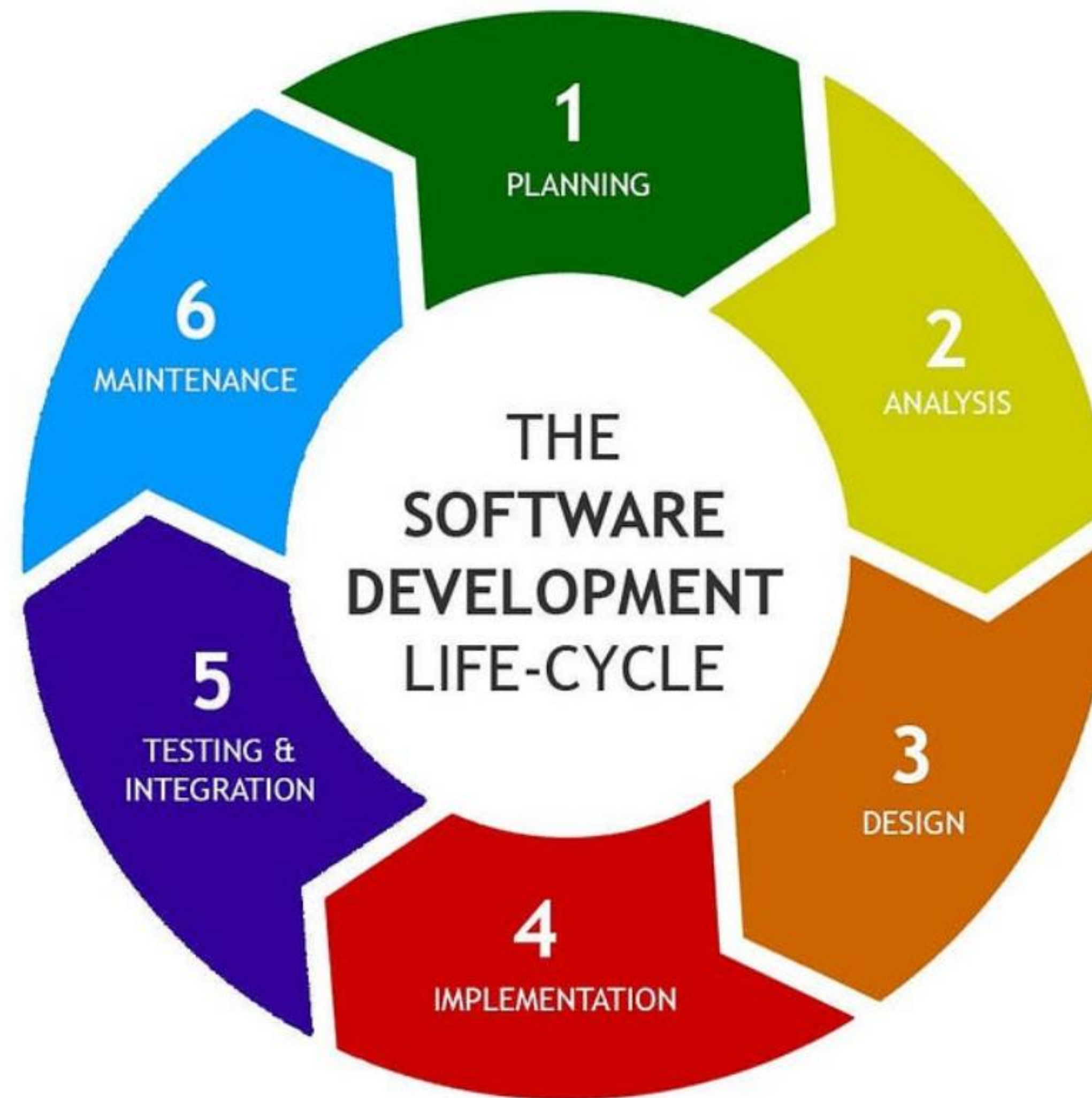
- **Árboles:** Organizan archivos en carpetas o el DOM de una web.
- **Grafos:** Representan redes sociales o rutas de GPS (Google Maps).



06. Ciclo de Vida (SDLC)

Del concepto al despliegue: El proceso de ingeniería.

FASES DEL DESARROLLO



Este ciclo asegura que el producto final cumpla con las expectativas del negocio y sea mantenible.

1. INGENIERÍA DE REQUISITOS

Es la fase más crítica. Un error aquí cuesta 100 veces más corregirlo al final.

Funcionales

¿Qué hace el sistema? (Ej: "Permitir pago con tarjeta").

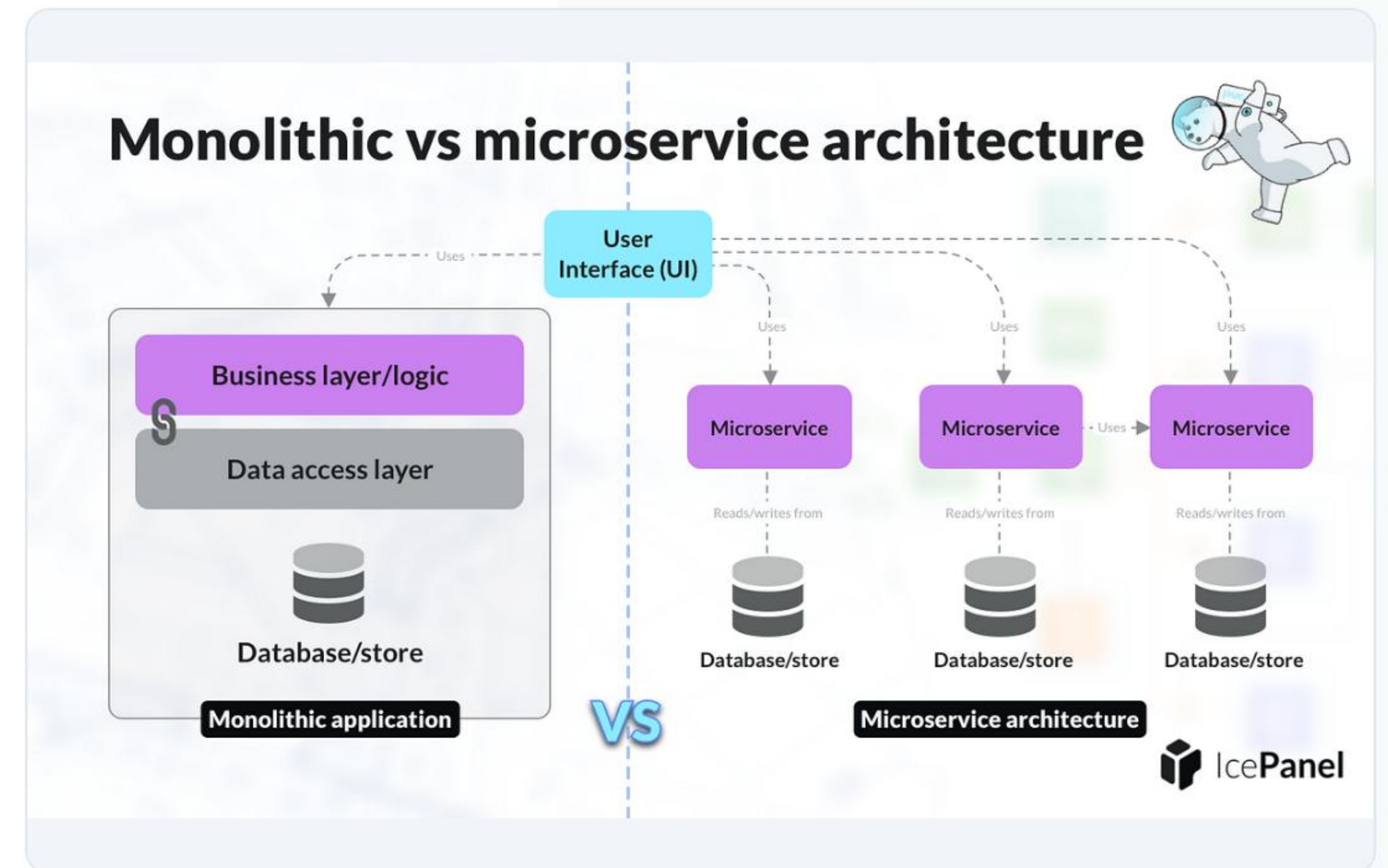
No Funcionales

¿Cómo se comporta? (Ej: "Cargar en menos de 2 segundos").

2. DISEÑO Y ARQUITECTURA

Se definen los planos del sistema antes de escribir código.

- **Arquitectura Monolítica:** Todo el código en un solo bloque.
- **Microservicios:** El sistema se divide en servicios independientes.



3. IMPLEMENTACIÓN (CODIFICACIÓN)

Los desarrolladores transforman el diseño en código ejecutable.

Standard Coding

Seguir guías de estilo para que el equipo entienda el código.

Version Control

Uso de Git para rastrear cada cambio histórico.

Code Review

Revisión por pares para detectar errores temprano.

4. PRUEBAS (V&V)

- **Verificación:** "¿Estamos construyendo el sistema correctamente?" (Sigue el diseño).
- **Validación:** "¿Estamos construyendo el sistema correcto?" (Satisface al cliente).

Dato: Se recomienda que el 30% del presupuesto de un proyecto se dedique exclusivamente a pruebas.

07. Metodologías de Desarrollo

Cómo organizamos a las personas y al tiempo.

METODOLOGÍA CASCADA (WATERFALL)

Enfoque lineal y secuencial. No puedes pasar a la siguiente fase sin terminar la anterior.

Requisitos > Diseño > Código > Test

Pros: Fácil de gestionar, hitos claros.

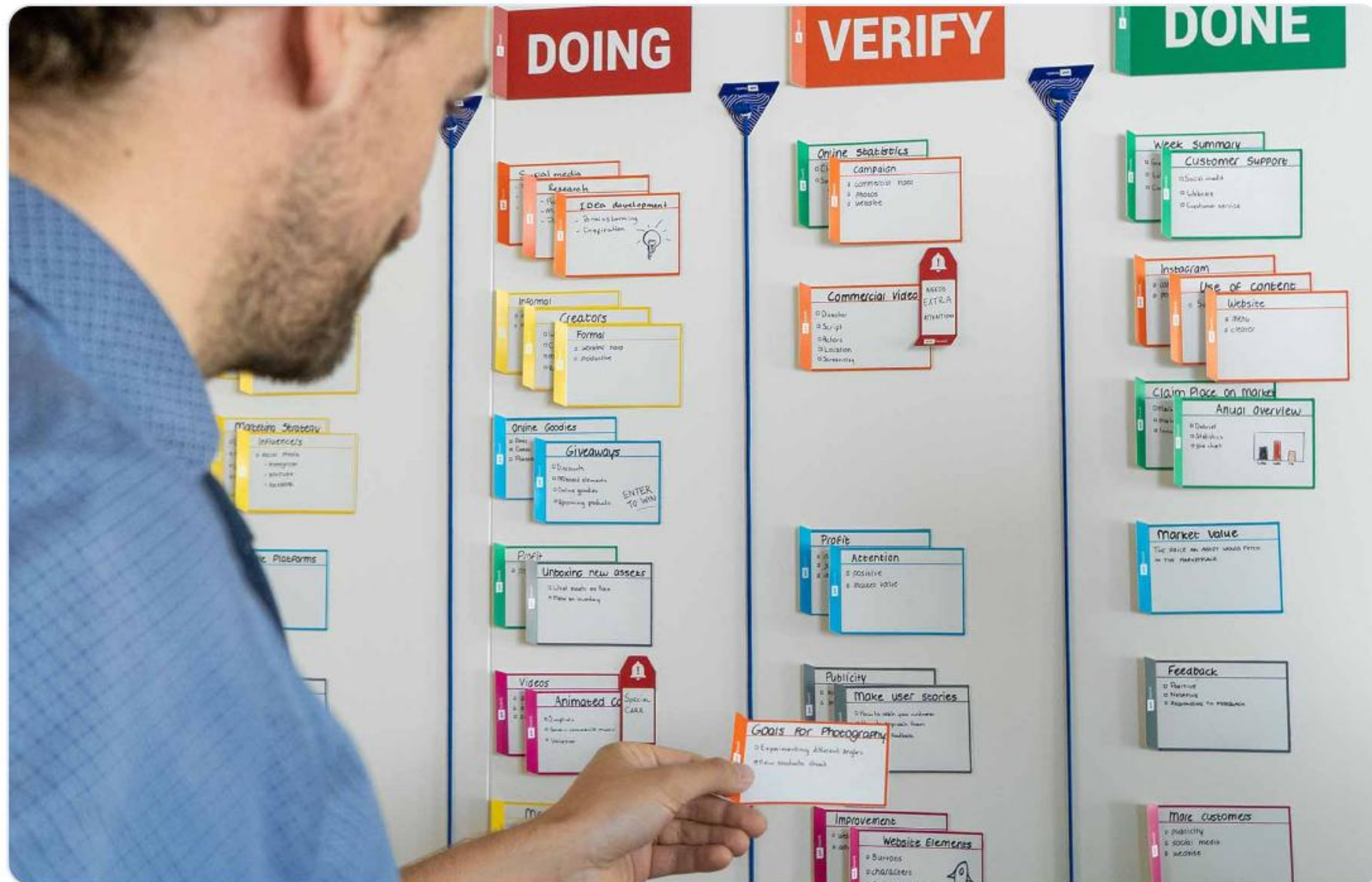
Contras: Muy rígido. El cliente no ve nada hasta el final.

EL MANIFIESTO ÁGIL (2001)

Cambió el mundo del software priorizando:

- **Individuos e interacciones** sobre procesos y herramientas.
- **Software funcionando** sobre documentación extensiva.
- **Colaboración con el cliente** sobre negociación contractual.
- **Respuesta ante el cambio** sobre seguir un plan.

SCRUM: EL REY DE LA AGILIDAD



Divide el trabajo en ciclos llamados **Sprints** (1-4 semanas).

- **Product Owner:** Define prioridades.
- **Scrum Master:** Facilita el proceso.
- **Daily Scrum:** Reunión de 15 min de sincronización.

KANBAN: FLUJO CONTINUO

Se basa en la visualización del flujo de trabajo y la limitación del Trabajo en Curso (WIP).

Por Hacer

Haciendo

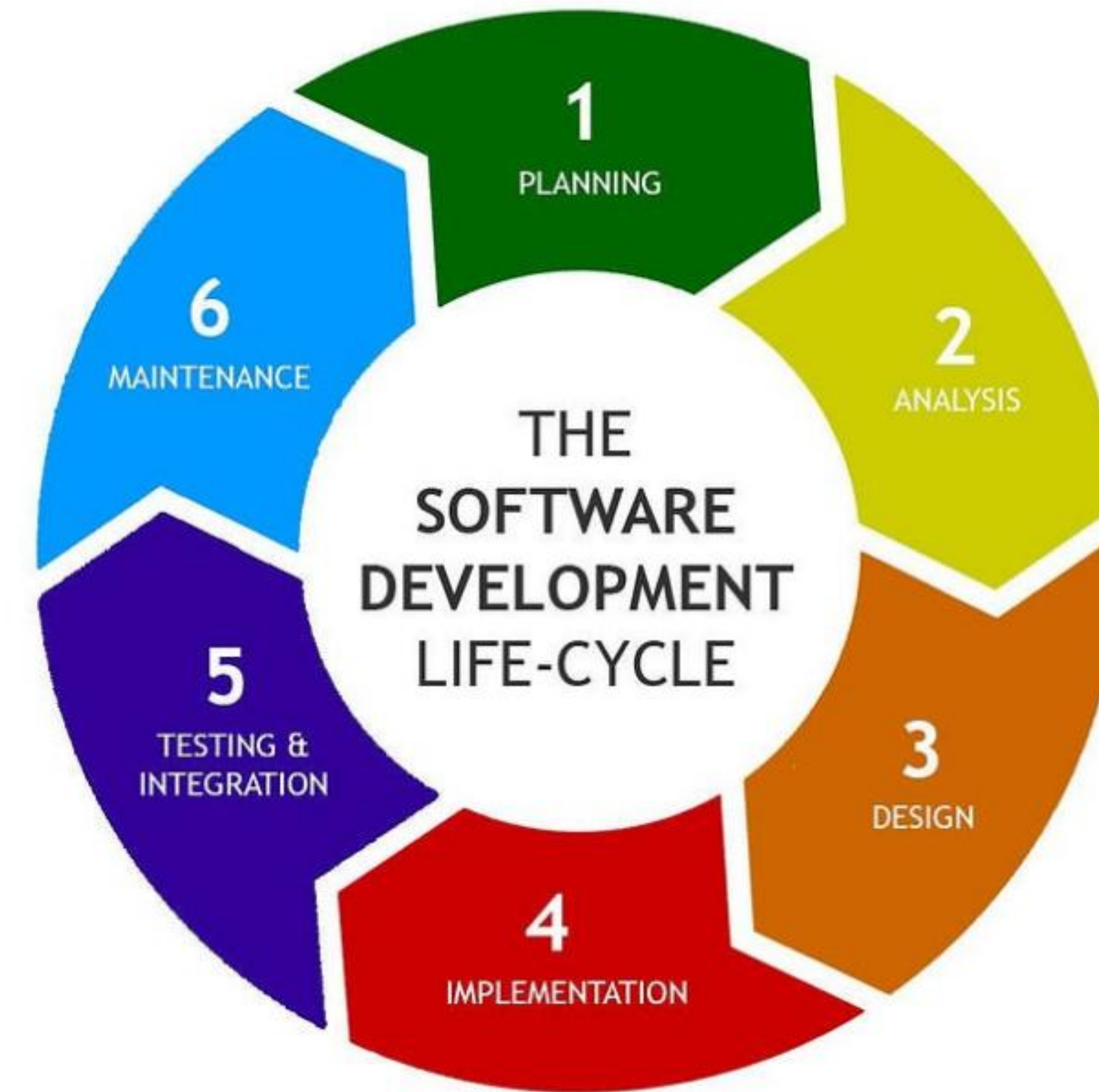
Hecho

Ideal para equipos de mantenimiento o soporte donde las prioridades cambian diariamente.

CULTURA DEVOPS

Une **Desarrollo** (Dev) y **Operaciones** (Ops). Su objetivo es reducir el tiempo entre que se escribe el código y se despliega al usuario.

Clave: Automatización total de pruebas y despliegues.



08. Calidad y Mantenimiento

Porque el software que no evoluciona, muere.

CALIDAD (ISO 25010)

Un buen software no solo "funciona", también debe ser:

Mantenible

Fácil de cambiar o arreglar.

Seguro

Protege los datos de intrusos.

Usable

Fácil de entender para el humano.

NIVELES DE PRUEBA

- **Unit Testing:** Prueba una sola función.
- **Integration Testing:** Prueba cómo se hablan dos módulos.
- **System Testing:** Prueba el software completo de punta a punta.
- **Acceptance Testing:** El cliente dice "Sí, esto es lo que pedí".

SEGURIDAD POR DISEÑO



No se debe añadir seguridad al final. Debe estar en el ADN del software:

- Principio de menor privilegio.
- Validación de todas las entradas.
- Cifrado de datos sensibles.

TIPOS DE MANTENIMIENTO

Correctivo

Arreglar bugs reportados.

Adaptativo

Cambiar para que funcione en un nuevo OS o hardware.

Evolutivo

Añadir nuevas funciones que pide el mercado.

09. Ética y Licenciamiento

La responsabilidad social del creador de código.

MODELOS DE LICENCIA

Tipo	Descripción	Ejemplo
Propietario	Código cerrado, pagas por usarlo.	Windows, Photoshop
Open Source	Código abierto, libre de estudiar y cambiar.	Linux, Android
SaaS	Software como Servicio (Nube).	Netflix, Gmail

ÉTICA DEL INGENIERO

El código tiene el poder de influir en vidas:

- **Privacidad:** ¿Qué haces con los datos del usuario?
- **Sesgo:** ¿Tu algoritmo discrimina a ciertos grupos?
- **Impacto Social:** ¿Cómo afecta tu software a la democracia o la salud?

"Tu código es tu compromiso con la sociedad."

10. Tendencias y Futuro

Lo que viene en la próxima década.

IA Y COPILOTS

La inteligencia artificial no reemplazará al ingeniero, pero el ingeniero que usa IA reemplazará al que no la usa.

- Generación automática de tests.
- Detección de vulnerabilidades en tiempo real.
- Explicación de código complejo.

COMPUTACIÓN CUÁNTICA

Un cambio total de paradigma. Ya no son bits (0 o 1), sino qubits.

Impacto: Romperá el cifrado actual pero resolverá problemas de química y medicina en segundos que hoy tardarían milenios.

CONCLUSIONES

1. El software es el tejido de la sociedad moderna.
2. La ingeniería es necesaria para gestionar la complejidad.
3. La calidad y la ética son tan importantes como la lógica.
4. El aprendizaje debe ser continuo.

FIN



¿Preguntas?

Contacto: ingenieria@example.com

IMAGE SOURCES



https://static.vecteezy.com/system/resources/previews/006/098/506/non_2x/abstract-hi-tech-technology-background-with-hexagons-creative-digital-futuristic-internet-template-graphic-modern-computer-banner-design-business-card-minimal-blue-screen-wallpaper-illustration-free-vector.jpg

Source: www.vecteezy.com



<https://i.ytimg.com/vi/I9Luktdq9pc/maxresdefault.jpg>

Source: www.youtube.com



<https://www.guvi.in/blog/wp-content/uploads/2025/12/Key-Components-of-Operating-System-Architecture-1200x630.png>

Source: www.guvi.in



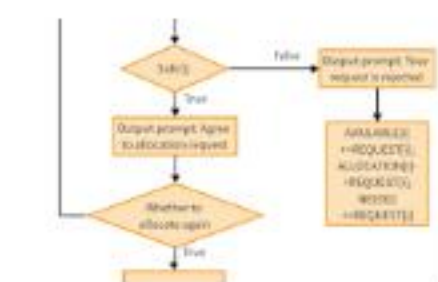
<https://www.bitstudios.com/wp-content/uploads/2022/09/types-of-software-development-infographics-scaled.jpg>

Source: www.bitstudios.com



https://miro.medium.com/0*9UWrSd7cRhDnwWw2.png

Source: medium.com



<https://cms.boardmix.com/images/articles/what-is-algorithm-flowchart.png>

Source: boardmix.com

IMAGE SOURCES



https://miro.medium.com/v2/resize:fit:1400/1*GU-YWDqM-ZljYHG51iDNPg.jpeg

Source: medium.com



https://miro.medium.com/v2/resize:fit:1200/1*aSdnOJNT2UoiaAhy-vuV_Q.png

Source: icepanel.medium.com



<https://www.patboard.com/wp-content/uploads/2022/04/kanban-board-magnets-magnetic-full-tool-set-patboard.jpg>

Source: adimpletionum.fr



<https://static.vecteezy.com/system/resources/thumbnails/068/346/689/large/futuristic-digital-shield-with-lock-symbol-representing-cybersecurity-and-data-protection-free-video.jpg>

Source: www.vecteezy.com