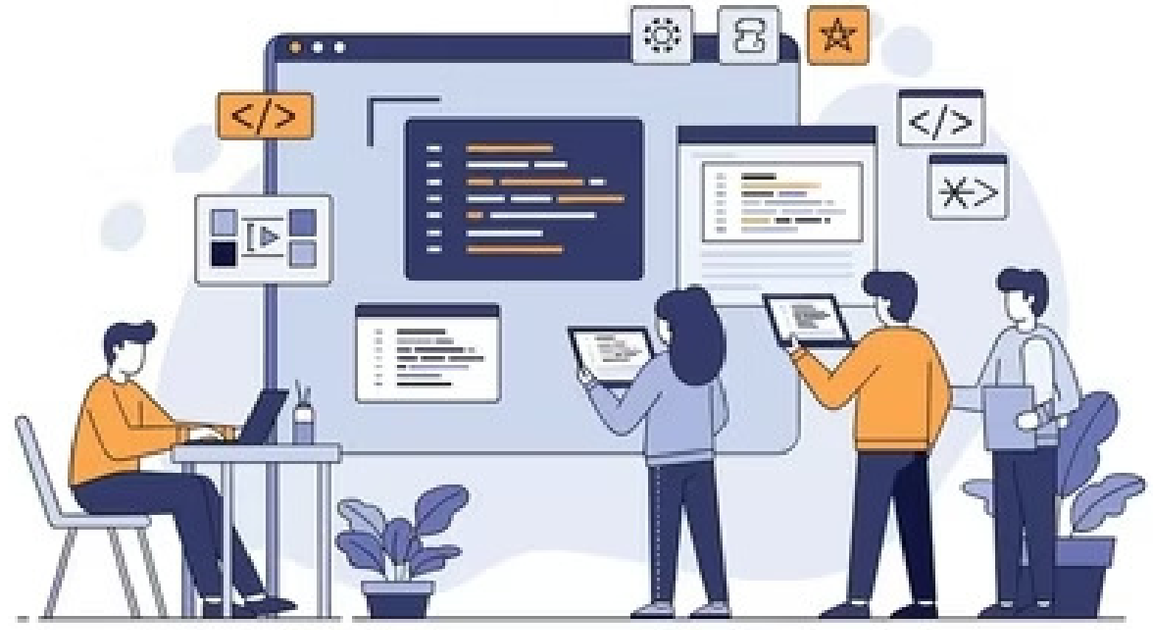
Metodologías para el Buen Desarrollo de Software

Prof. Katherine Molina

Desde los modelos tradicionales hasta los enfoques ágiles: guía completa para entender, elegir y aplicar la metodología correcta en cada proyecto de software y desarrollo web.



FUNDAMENTOS

El Ciclo de vida del software

Todo proyecto de software sigue un ciclo de vida estructurado. Conocerlo es el primer paso para gestionarlo con éxito.

Análisis y definiciones

Entender qué hay que construir

Diseño y planificación

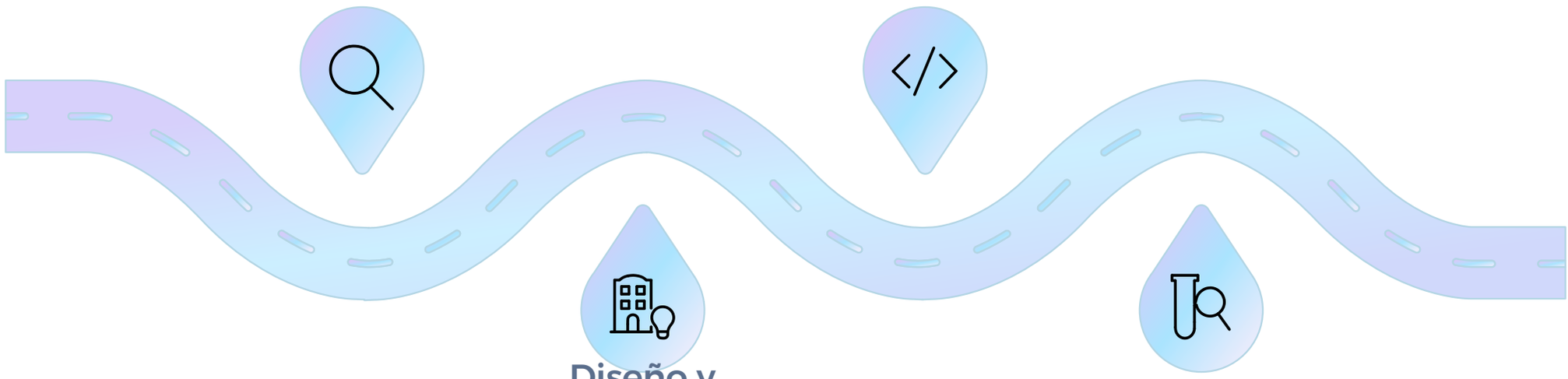
Arquitectura y planificación del sistema

Implementación

Desarrollo y construcción del código

Pruebas y Mantenimiento

Validación de calidad y corrección



Cada fase depende de la anterior. La calidad de cada etapa determina el éxito del producto final.

FUNDAMENTOS

Análisis y Definición de Requisitos

Esta fase es la brújula del proyecto. Aquí se sienta la base de lo que se construirá, asegurando que todos entiendan el objetivo y cómo llegar a él.

Definición del Alcance y Funcionalidades

Establecer claramente qué **hará** el sistema y, crucialmente, qué **no** hará. Delimita las funcionalidades esenciales y el comportamiento esperado para satisfacer las necesidades del usuario y los objetivos de negocio.

Estructura Técnica y Diseño

Determinar la arquitectura (p. ej., **one-page** vs. **multi-page** para web), el **stack tecnológico** (lenguajes, frameworks) y la **gestión de datos** (base de datos, JSON, estático). También, prever sistemas de control para la administración del contenido.

Operación y Mantenimiento

Planificar la **puesta en marcha**, los mecanismos de **despliegue** en servidores y las estrategias de **mantenimiento** futuro. Anticipar estos aspectos garantiza la longevidad y estabilidad del software.

FUNDAMENTOS

Diseño y Planificación

Una vez definidos los requisitos, se traza el mapa del sistema. Esta fase convierte las ideas en un plan detallado y ejecutable, sentando las bases para una implementación sólida.

Arquitectura del Sistema

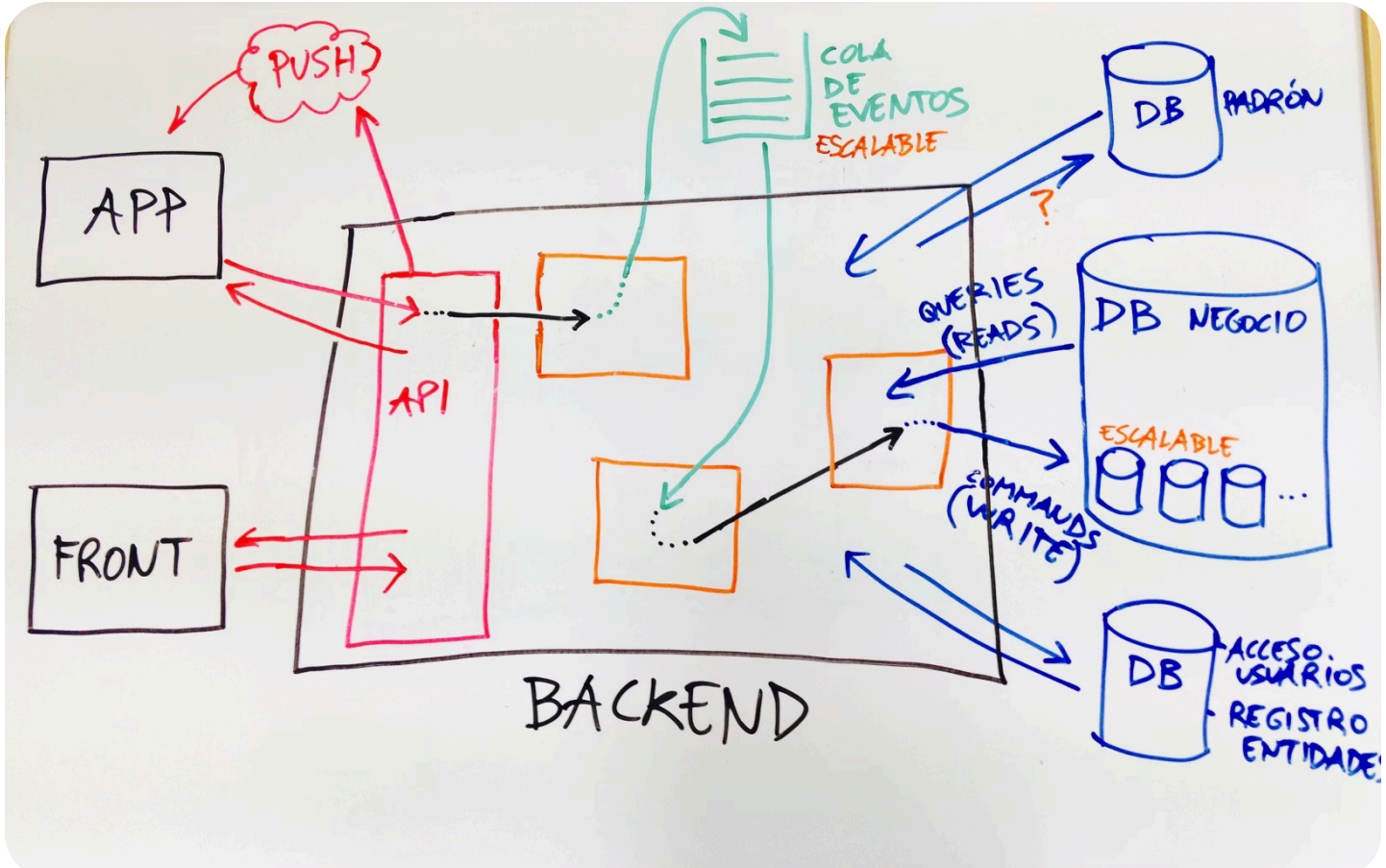
Define la estructura fundamental del software, la interconexión de componentes, la selección de la base de datos y la infraestructura de despliegue. Es el esqueleto técnico sobre el que se construirá todo el proyecto.

Diseño UX/UI

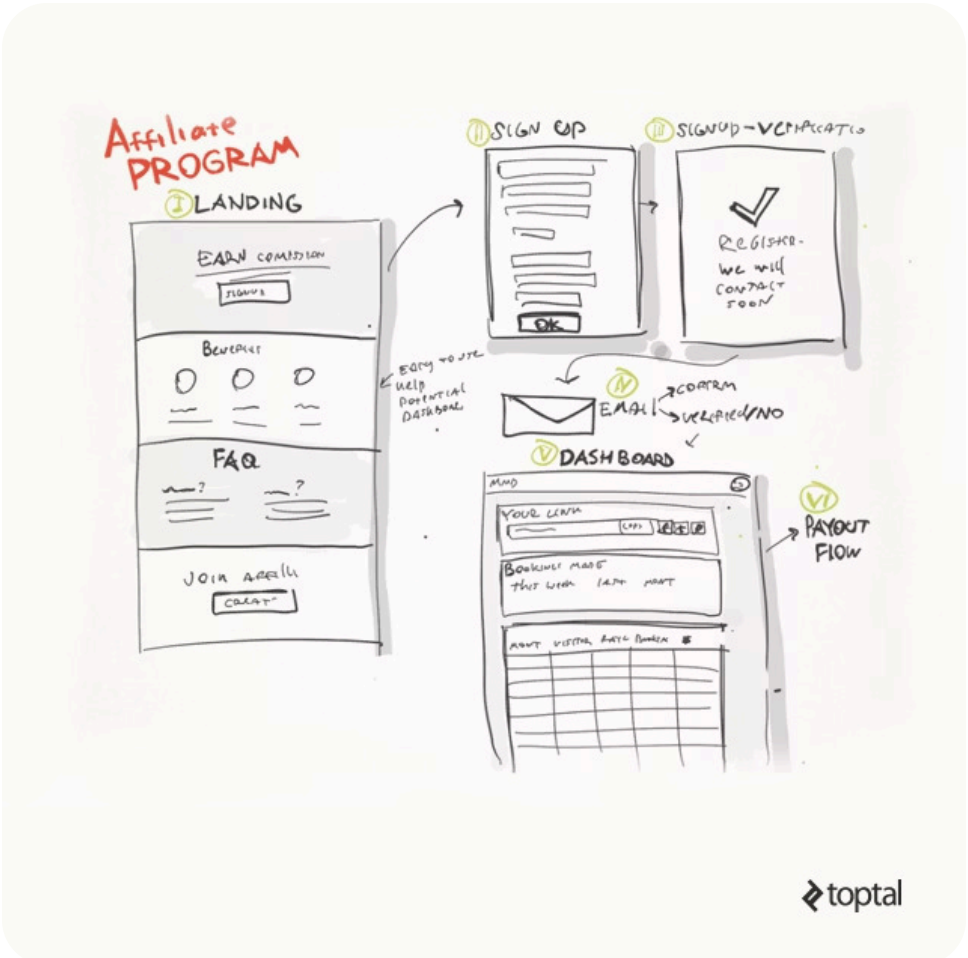
Se enfoca en crear una experiencia de usuario intuitiva y una interfaz visual atractiva. Incluye wireframes, mockups, prototipos y pruebas de usabilidad para garantizar que el producto sea fácil y agradable de usar.

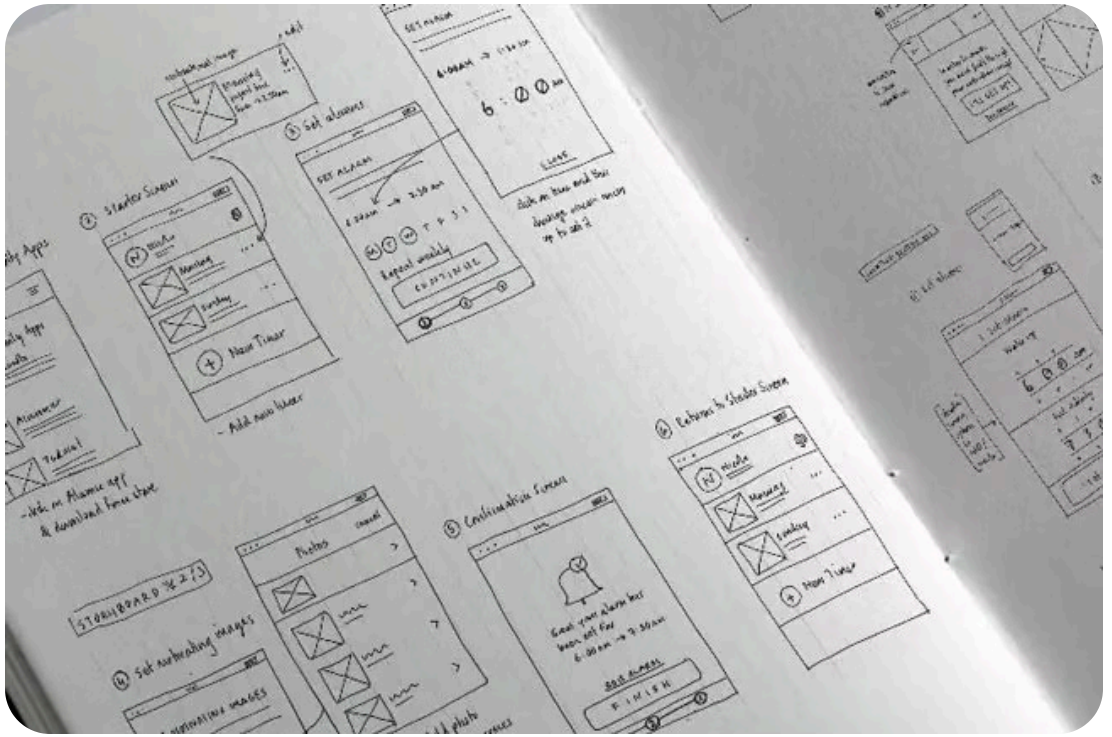
Planificación del Proyecto

Establece los cronogramas, asigna recursos, define los hitos clave y elabora estrategias de gestión de riesgos. Una planificación detallada es crucial para mantener el proyecto dentro del presupuesto y los plazos definidos.

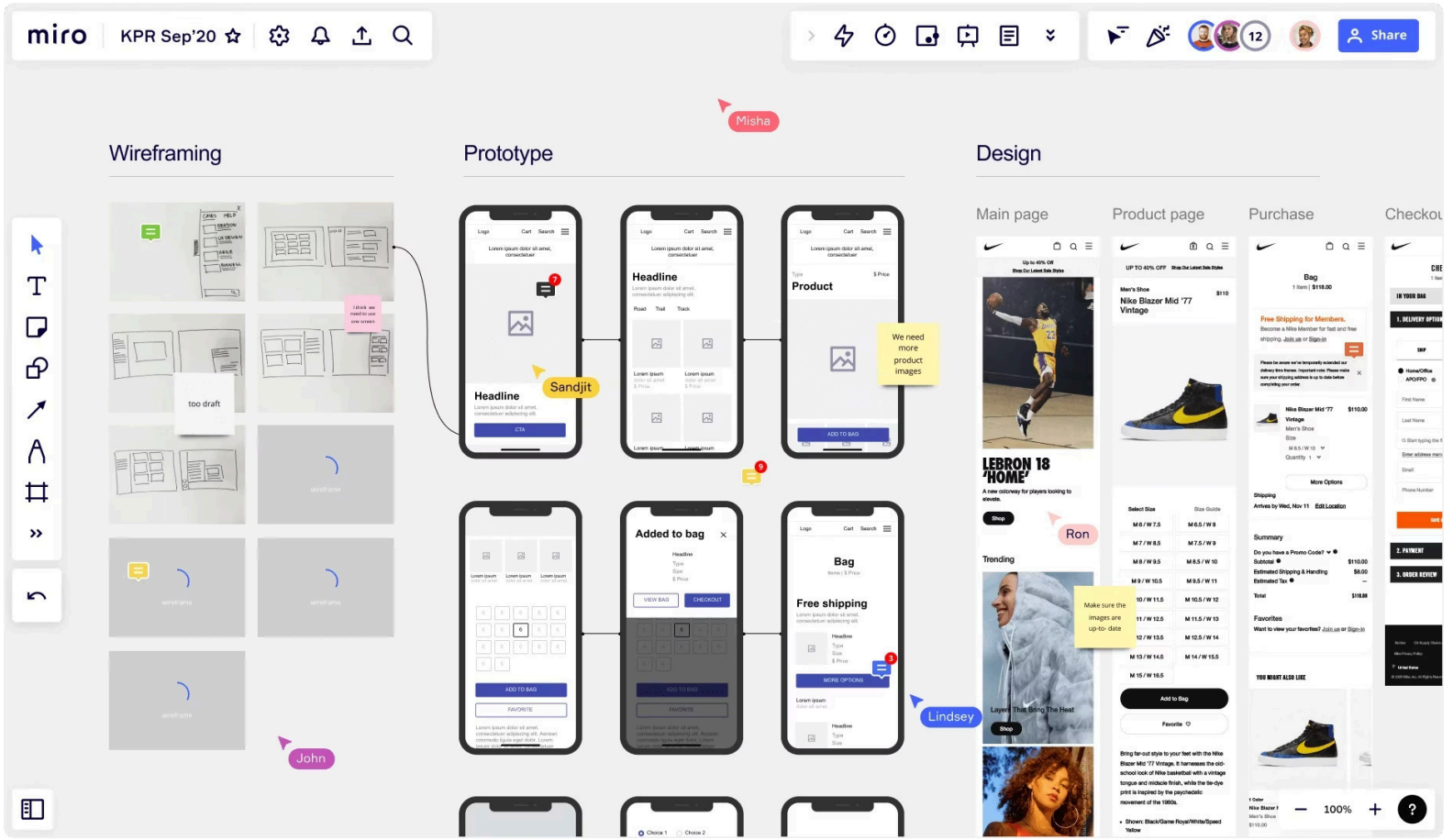


Arquitectura de un sistema





Diseño ux



Diseño UX/UI

GESTIÓN

Herramientas de Planificación y Seguimiento

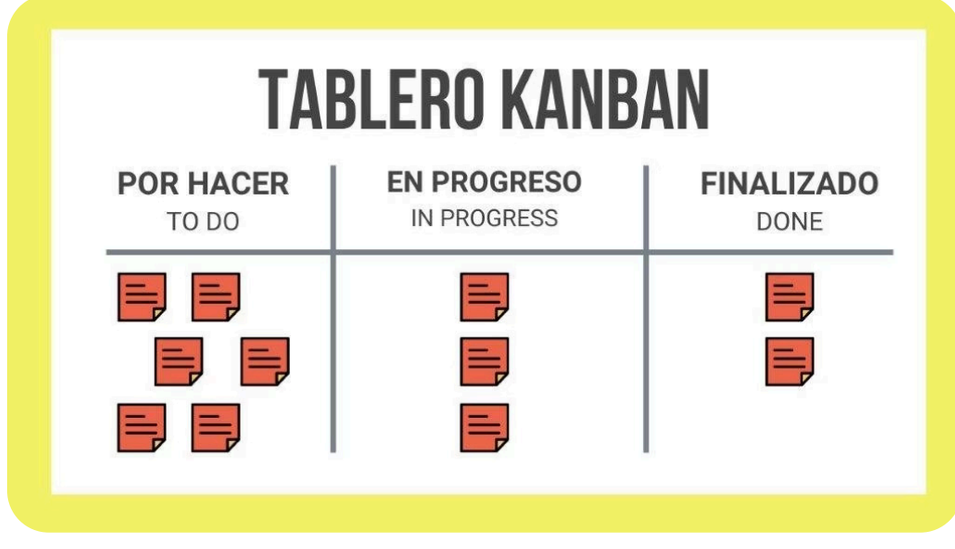
Una gestión efectiva del tiempo y las tareas es crucial. Los cronogramas y los tableros Kanban ofrecen enfoques visuales para mantener tu proyecto en marcha, garantizando visibilidad y control.

Cronogramas de Proyecto

Representaciones temporales que detallan las actividades, sus dependencias y plazos. Ideales para proyectos con secuencias fijas y entregables claros, como los que siguen el modelo en cascada.

Tableros Kanban

Herramientas visuales para gestionar el flujo de trabajo, limitando el trabajo en progreso y optimizando la eficiencia. Cada tarea avanza por columnas (Pendiente, En Progreso, Hecho) ofreciendo una vista clara del estado.



CICLO DE VIDA

Implementación y Desarrollo

Esta es la fase donde las ideas y diseños se transforman en código funcional. Es el corazón de la construcción del software, donde se materializa la visión del proyecto.

Codificación Estructurada

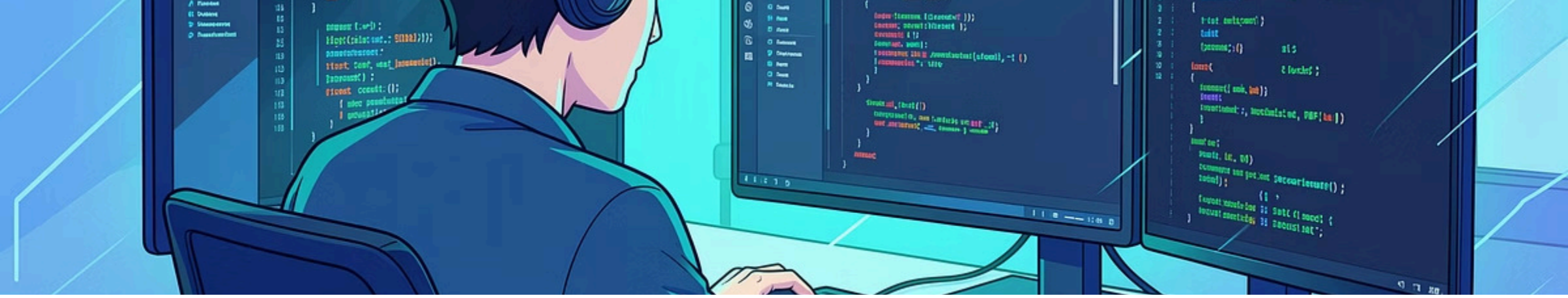
Escribir el código fuente siguiendo las mejores prácticas de programación, los estándares definidos y las especificaciones de diseño. Se enfoca en la legibilidad, eficiencia y escalabilidad.

Pruebas Rigurosas

Realizar pruebas unitarias y de integración para asegurar que cada módulo y su interacción funcionen como se espera. La detección temprana de errores es fundamental para la calidad.

Control de Versiones

Utilizar sistemas como Git para gestionar los cambios en el código, facilitando la colaboración entre equipos, el seguimiento de versiones y la resolución de conflictos.



CICLO DE VIDA

Pruebas y Mantenimiento

La calidad no es un accidente. La fase de pruebas garantiza que el software sea robusto y funcional, mientras que el mantenimiento asegura su longevidad y relevancia.

Validación Exhaustiva

Las pruebas no solo buscan errores; verifican que el sistema cumpla con los requisitos, se ejecute sin fallos en distintos dispositivos y en su entorno productivo.

Despliegue Controlado

Subir el código a producción es un paso crítico. Requiere estrategias de despliegue que minimicen interrupciones y permitan reversiones rápidas si surgen problemas.

Compatibilidad Multi-dispositivo

Asegurar que el sistema sea usable y se vea correctamente en una variedad de pantallas y navegadores es fundamental para la experiencia del usuario moderno.

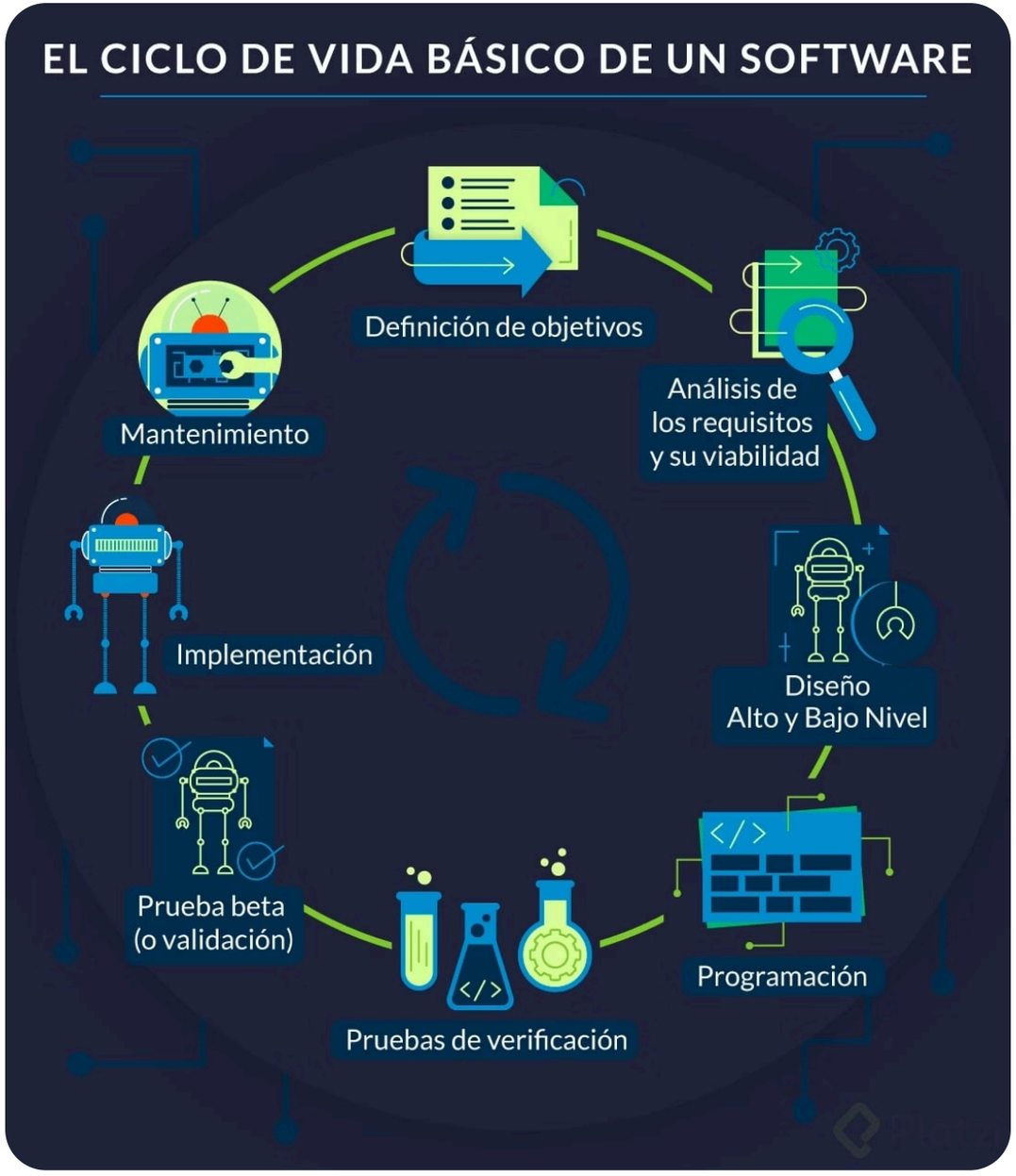
Retroalimentación Continua

Escuchar a los usuarios y al equipo es vital. Las opiniones y el feedback permiten iterar mejoras, identificar necesidades no cubiertas y corregir problemas de usabilidad.

Mantenimiento Proactivo

Determinar la frecuencia de mantenimiento (mensual, anual, por demanda) y asegurar el funcionamiento de sistemas de soporte (como un backoffice) es clave para la salud a largo plazo del software.

EL CICLO DE VIDA BÁSICO DE UN SOFTWARE



METODOLOGÍA TRADICIONAL

El Modelo en Cascada

Estructura lineal

Cada fase debe completarse antes de iniciar la siguiente. Sin retroceso posible.

Predecible

Presupuestos y cronogramas definidos desde el inicio del proyecto.

Documentación exhaustiva

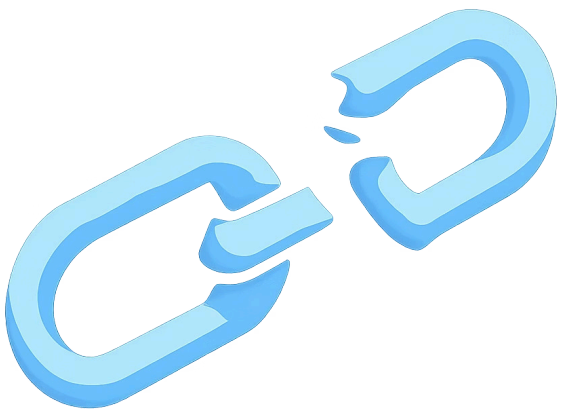
Requisitos detallados y firmados antes de escribir una sola línea de código.

El gran defecto

Los errores solo se detectan al final, cuando corregirlos es más costoso.

EL PROBLEMA

El punto de quiebre: ¿Por qué fallan los proyectos?



Las causas más comunes

Sin retroalimentación temprana: El cliente ve el producto solo al final, cuando los cambios son costosos.

Incapacidad de adaptación: Los mercados digitales cambian más rápido que la documentación.

Requisitos congelados: Lo que el cliente pedía al inicio ya no es lo que necesita al final.

METODOLOGÍAS ÁGILES

Agilidad: Adaptación Constante

Las metodologías ágiles nacieron como respuesta a los desafíos de los enfoques tradicionales, promoviendo la flexibilidad, la colaboración y la entrega continua de valor en entornos cambiantes.

Desarrollo Iterativo

El software se construye en ciclos cortos y repetitivos (sprints o iteraciones), permitiendo la adaptación constante a los cambios y la retroalimentación temprana del cliente.

Eventos Clave

Las ceremonias ágiles (planificación de sprint, stand-ups diarios, revisiones y retrospectivas) fomentan la comunicación transparente y la mejora continua del equipo y el producto.

Colaboración y Valor

Prioriza la interacción individual sobre los procesos, y la entrega de software funcional sobre la documentación exhaustiva, enfocándose siempre en el valor para el usuario final.

LOS 12 PILARES

Principios del Manifiesto Ágil

Estos principios detallan cómo los equipos ágiles colaboran y entregan valor, priorizando la adaptabilidad y la satisfacción del cliente.

1

Cliente Satisfecho

Entrega temprana y continua de software valioso.

2

Aceptar Cambios

Flexibilidad ante requisitos cambiantes, incluso al final del desarrollo.

3

Entrega Frecuente

Software funcional en ciclos cortos (semanas o meses).

4

Colaboración Diaria

Trabajo conjunto entre negocio y desarrolladores.

5

Individuos Motivados

Confianza y apoyo al equipo para hacer su trabajo.

6

Comunicación Directa

La conversación cara a cara es el método más eficiente.

7

Software Funcional

La medida principal del progreso es el software que funciona.

8

Ritmo Sostenible

Promover un desarrollo constante que todos puedan mantener.

9

Excelencia Técnica

Atención continua a la calidad y buen diseño.

10

Simplicidad

Maximizar el trabajo no realizado es esencial.

11

Equipos Autoorganizados

Las mejores soluciones emergen de equipos autónomos.

12

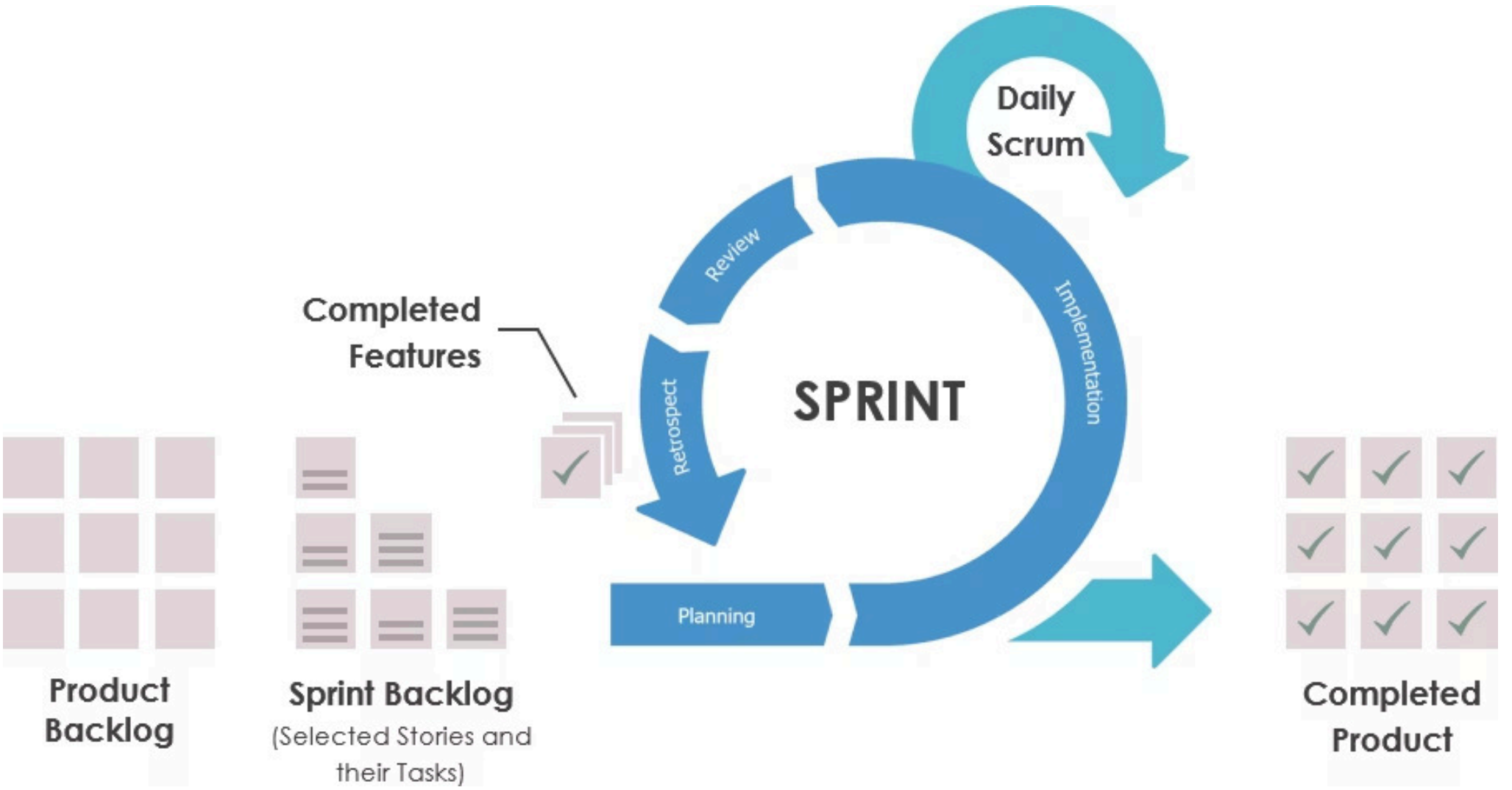
Reflexión Constante

El equipo mejora continuamente su eficacia y comportamiento.

Valores del manifiesto ágil

Scrum

Scrum es un marco de trabajo ágil para organizar equipos, entregar valor de forma incremental y adaptarse rápidamente al cambio.



Backlog

Lista priorizada de todas las funcionalidades y mejoras del producto.

Sprint

Ciclo de trabajo de 1-4 semanas donde el equipo desarrolla incrementos del producto.

Daily Standup

Reunión diaria de 15 minutos donde el equipo sincroniza el progreso.

Planning

Reunión al inicio del sprint donde se seleccionan tareas del backlog.

Review

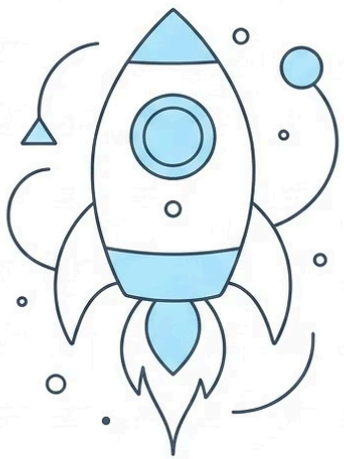
Demostración del trabajo completado al final del sprint.

Retrospectiva

Reunión para reflexionar sobre cómo mejorar el proceso del equipo.

COMPARATIVA

Tradicional vs Ágil

TRADICIONAL	ÁGIL
	
Modelo predictivo	**Modelo adaptativo**
Control centralizado	**Equipos autogestionados**
Documentación exhaustiva	**Entregas incrementales**
Resistencia al cambio	**Adopción del cambio**
Requisitos fijos	**Alta incertidumbre**

¿Cuál elegir?

No existe una metodología superior en términos absolutos. La elección depende de:

- La **claridad y estabilidad** de los requisitos
- La **velocidad** que exige el mercado
- La **complejidad** e incertidumbre del proyecto
- La **cultura organizacional** del equipo

Muchos equipos exitosos combinan elementos de ambos enfoques (híbrido).

TOMA DE DECISIONES

Gestionando el Proyecto: ¿Qué camino elegir?

Usa Tradicional cuando...

- Los requisitos son fijos, claros y documentables
- El proyecto es regulatorio o de alta criticidad
- El cliente no puede participar activamente
- El presupuesto y cronograma son inamovibles

Usa Ágil cuando...

- La incertidumbre es alta y los requisitos evolucionan
- El mercado exige velocidad de respuesta
- El cliente puede participar y dar feedback constante
- El producto es digital y requiere iteración continua

La agilidad no es falta de orden — es **disciplina flexible**.

Conclusión: Desarrollar es Evolucionar

El software nunca está terminado

Está en constante mejora. Cada versión es un aprendizaje, no un punto final.

La ventaja competitiva es aprender rápido

No gana el que planifica mejor, sino el que se adapta más rápido al cambio.

Construye hoy, refina mañana, entrega valor siempre

La metodología correcta es la que permite a tu equipo entregar valor real de forma consistente.

FASE CRÍTICA

Mantenimiento: La Fase Olvidada pero Crítica

Tras el lanzamiento, el software entra en su etapa más larga y a menudo subestimada. Un buen mantenimiento es sinónimo de longevidad y valor continuo.

Evolución Constante

El entorno, las necesidades del usuario y las amenazas de seguridad evolucionan. El software debe hacerlo también.

Deuda Técnica

Ignorar el mantenimiento acumula **deuda técnica**, haciendo que las futuras modificaciones sean más lentas y costosas.

Experiencia del Usuario

Un sistema sin atención constante frustra a los usuarios y afecta la reputación del producto.