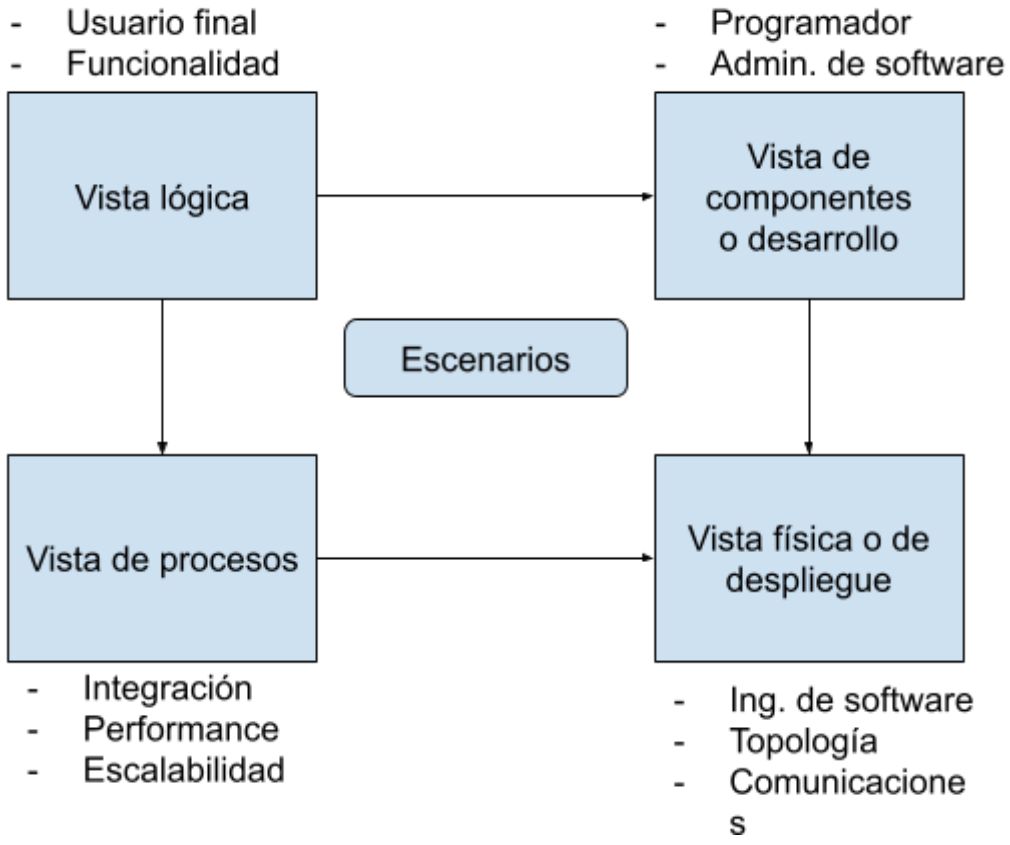
Modelo de Vistas 4 + 1 - Krutchen

Dado que hay muchos stakeholders que pueden tener distintos intereses y conocimientos, este modelo permite explicar la arquitectura propuesta para resolver el problema planteado.

El modelo 4+1 permite:

- Mejorar la comunicación entre stakeholders.
- Gestionar la complejidad del sistema dividiendo la arquitectura en aspectos específicos.
- Facilitar la adaptación a cambios y validar los requisitos funcionales.
- Asegurar que la arquitectura cumple con atributos clave de calidad.

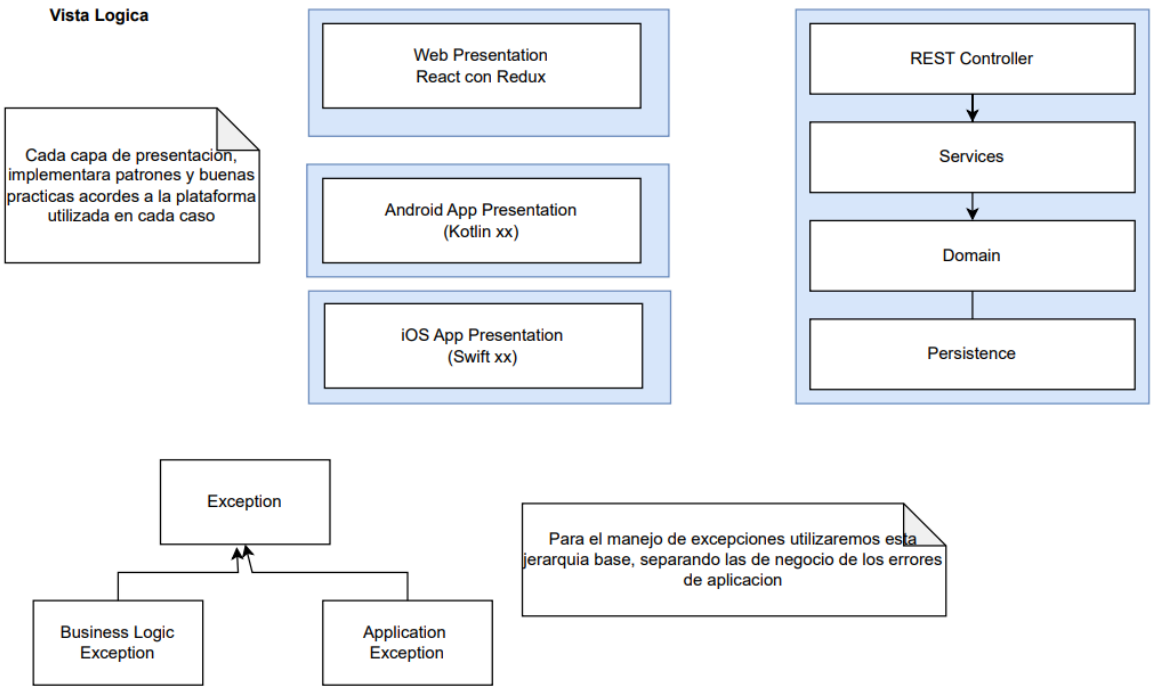


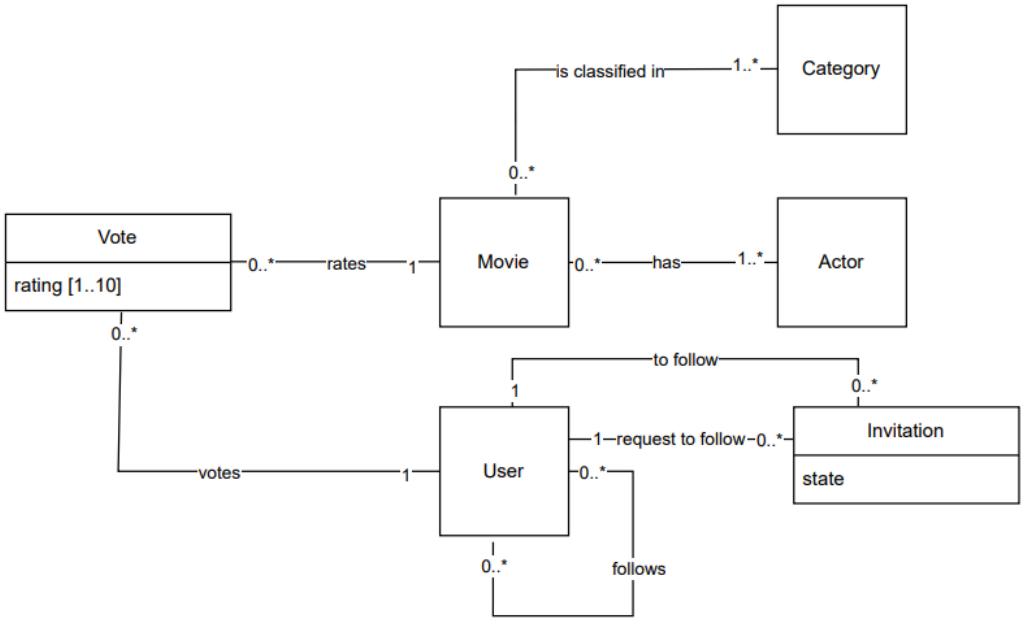
VISTA LÓGICA

Apoya los **requisitos funcionales** (lo que el sistema debe brindar en términos de servicios a los usuarios).

El sistema se descompone en **abstracciones clave**, en forma de objetos o clases a partir del dominio del problema. Se aplican los principios de abstracción, encapsulamiento y herencia.

Se busca identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.





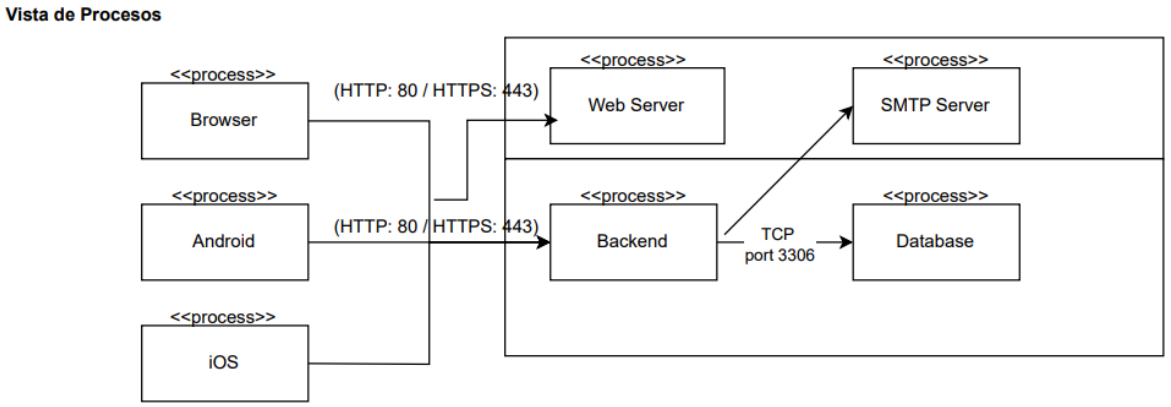
VISTA DE PROCESOS

Toma en cuenta algunos requisitos no funcionales como la **performance** y la **disponibilidad**.

Además, contempla:

- Concurrencia y distribución
- Integridad del sistema
- Tolerancia a fallas

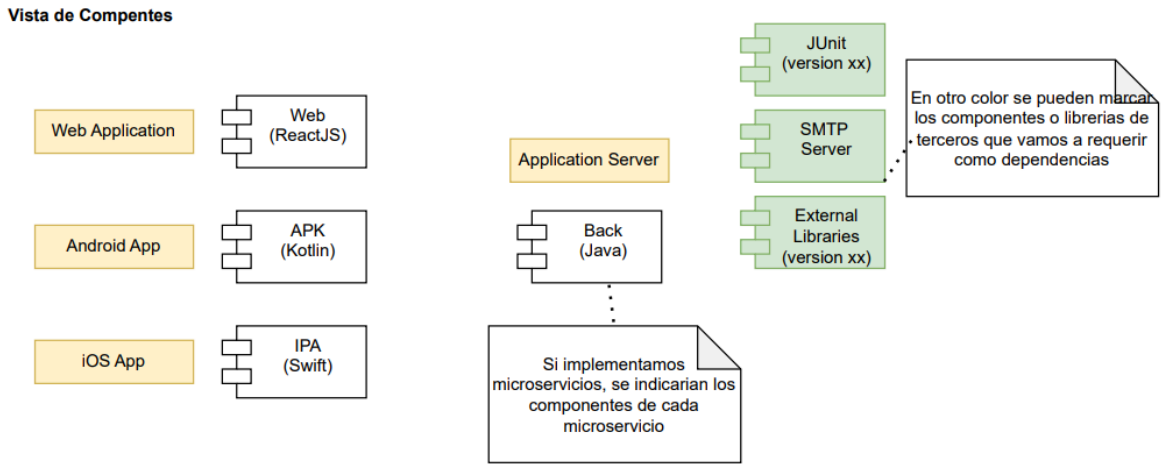
Especifica **en cuál hilo de control se ejecuta una operación** de una clase definida en la vista lógica.



VISTA DE DESARROLLO O DE COMPONENTES

Se centra en la **organización real de los módulos de software** en el ambiente de desarrollo.

Tiene en cuenta los requisitos internos relativos a la facilidad de desarrollo, administración, reutilización, elementos comunes y restricciones impuestas por las herramientas o el lenguaje de programación que se use.

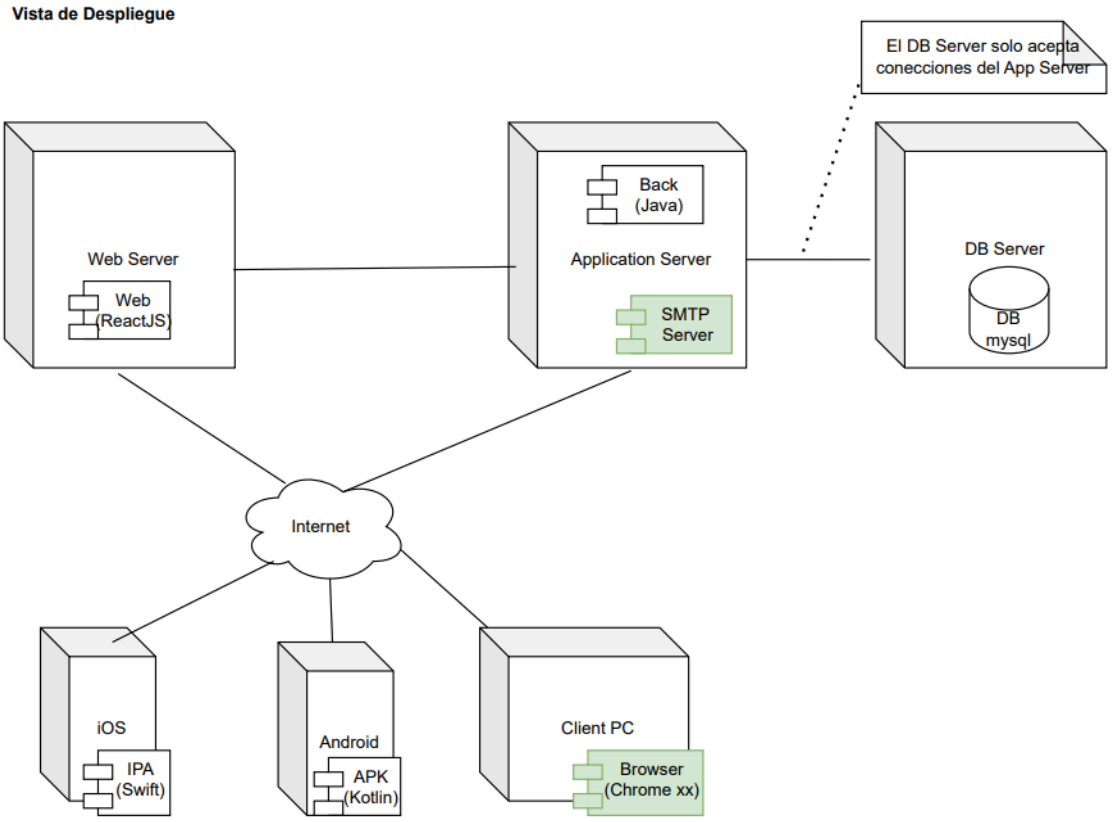


VISTA FÍSICA O DE DESPLIEGUE (software → hardware)

Detalla cómo los componentes del sistema se distribuyen en servidores, dispositivos, y otras infraestructuras físicas.

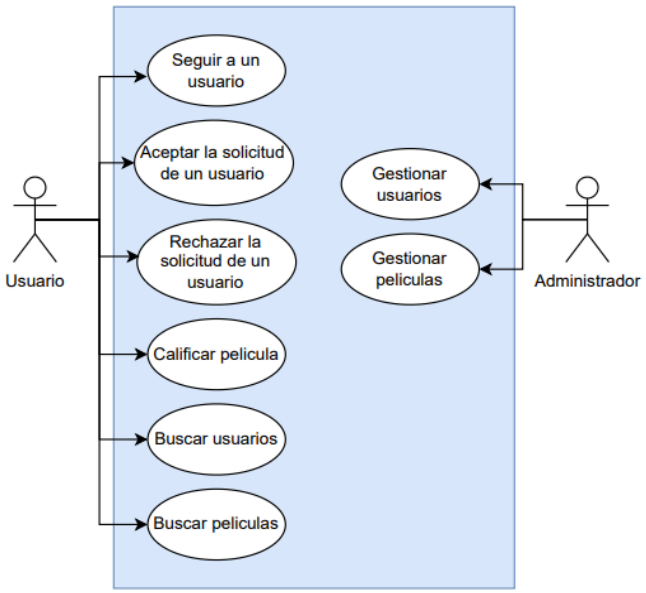
Toma los requisitos no funcionales como la disponibilidad, confiabilidad, performance y escalabilidad.

El software ejecuta sobre una red de computadoras o nodos de procesamiento. Los nodos se relacionan con elementos identificados en las vistas anteriores.



ESCENARIOS

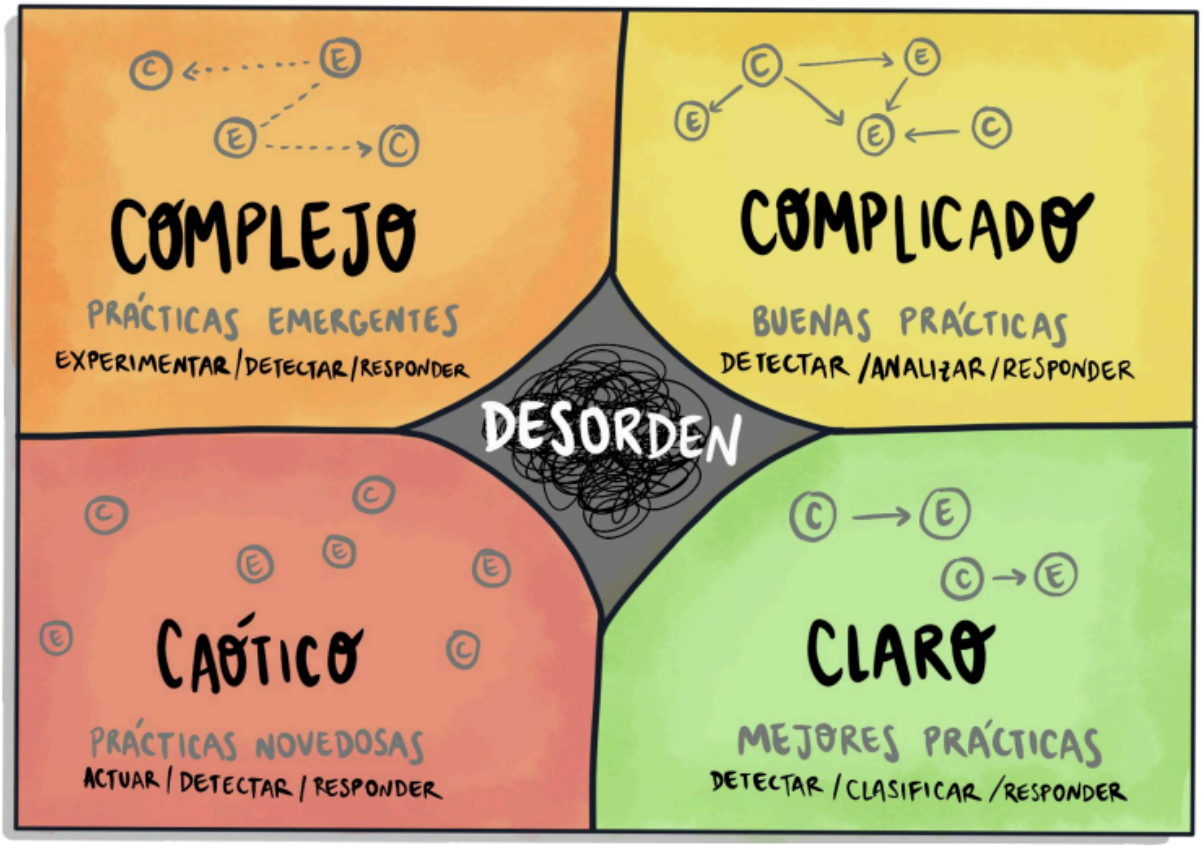
Son instancias de casos de uso más generales. Es la abstracción de los requisitos más importantes.



AGILE, SCRUM y XP

MARCO CYNEFIN

Sirve para entender la naturaleza de un problema y actuar de manera adecuada según su nivel de complejidad. Se divide en 5 dominios:



CLARO → lo obvio, problema intuitivo, fácil de resolver.
causa → efecto evidente

COMPLICADO → requiere análisis o experiencia para encontrar la solución adecuada.
causa → efecto no evidente de inmediato

COMPLEJO → hay que experimentar, observar y aprender del sistema. (prueba y error)

causa → efecto se identifica en retrospectiva

CAÓTICO → se requiere una respuesta inmediata para controlar la situación. Pensar y luego actuar.

causa → efecto no hay relación aparente

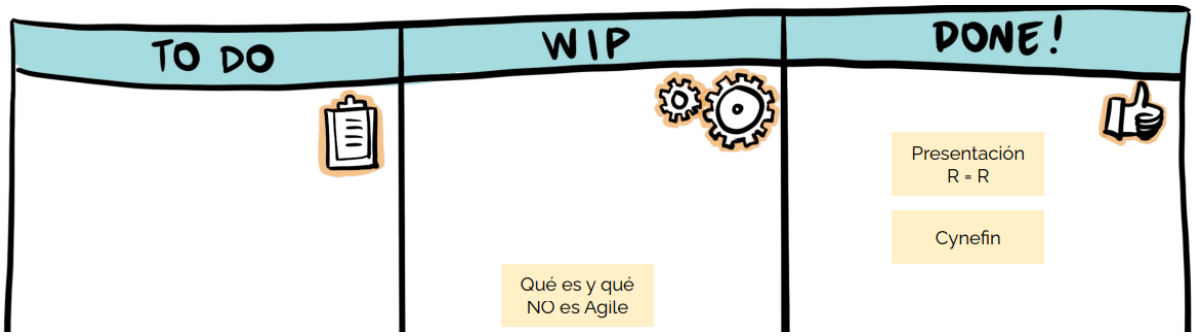
DESORDEN → estado inicial antes de clasificar un problema.

Diferencias entre AGILE, SCRUM, KANBAN y XP

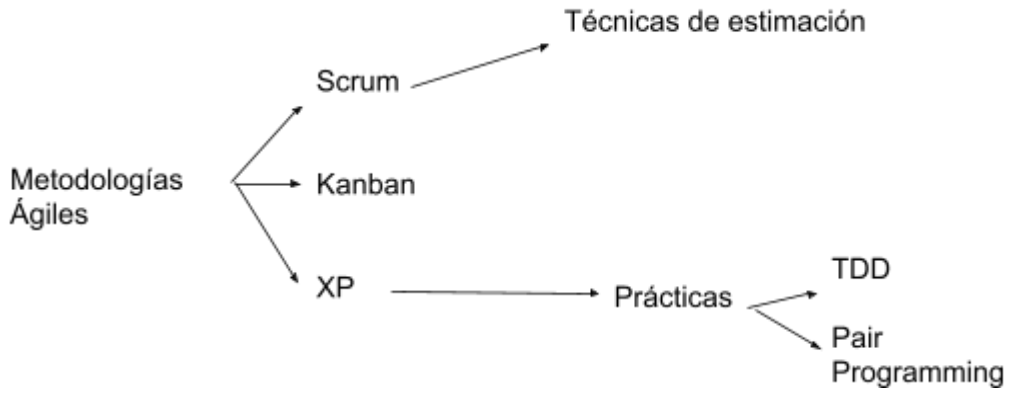
AGILE: marco general para el desarrollo iterativo e incremental enfocado en adaptabilidad y avance continuo, mejora en la colaboración.

SCRUM: gestión de proyectos mediante iteraciones (sprints) con roles definidos y enfoque en la planificación.

KANBAN: visualización del flujo de trabajo y limitación del trabajo en curso para maximizar la eficiencia.



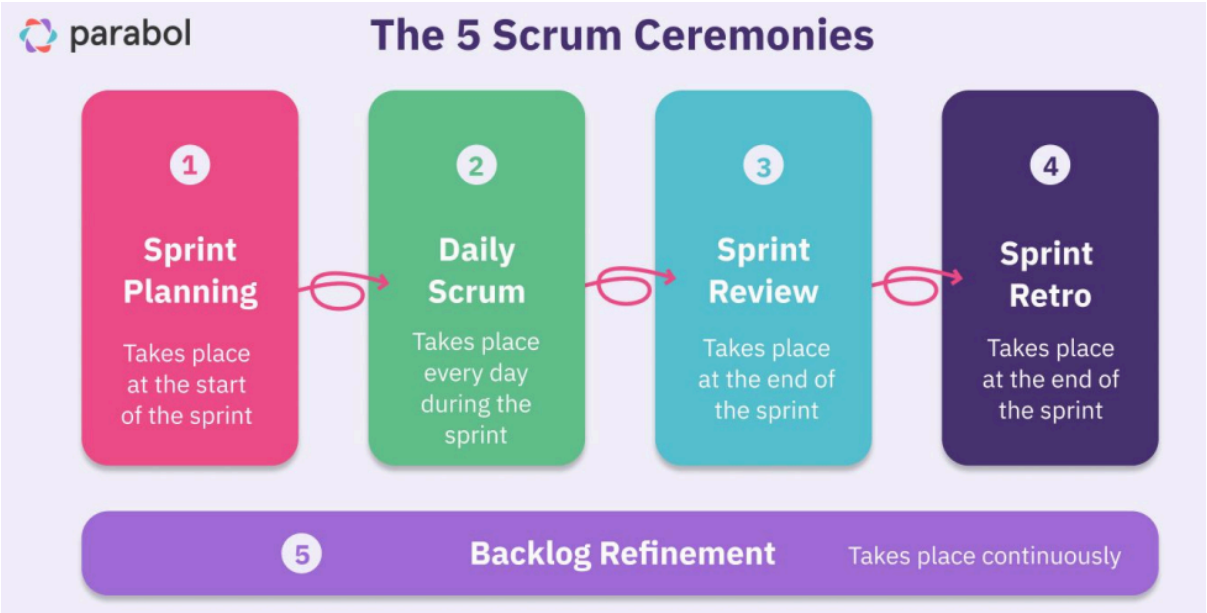
XP: prácticas técnicas como TDD, Pair Programming y entregas frecuentes de software funcional. Se basa en cómo hacer el trabajo, a diferencia de la mayoría de las metodologías ágiles.



SCRUM

Propone resolver un problema complejo en pedazos más pequeños. Se realizan 5 ceremonias:

1. Sprint Planning
2. Daily
3. Retro
4. Sprint Review
5. Backlog refinement



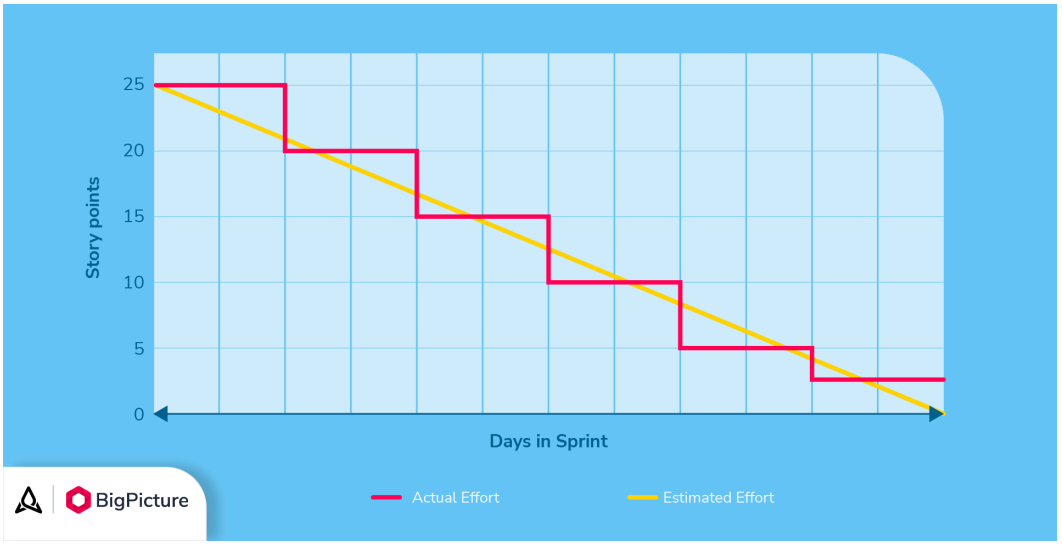
Para el sprint planning se utilizan métodos de estimación.

- Método delphi: estimación anónima → recopilación y discusión → otra ronda de estimación →
- Poker planning: presentación de la user story → c/ miembro elige una carta que representa su estimación según esfuerzo → se muestran las cartas → si hay discrepancias, se vuelve a rotar hasta el consenso.
- Por afinidad: las tareas se agrupan por similitud de esfuerzo, agrupándolas entre sí.
- Con talles de camisetas
- Por puntos: se le asigna un número y se suman los de todos los participantes para estimar el esfuerzo de una user story.

MÉTRICAS

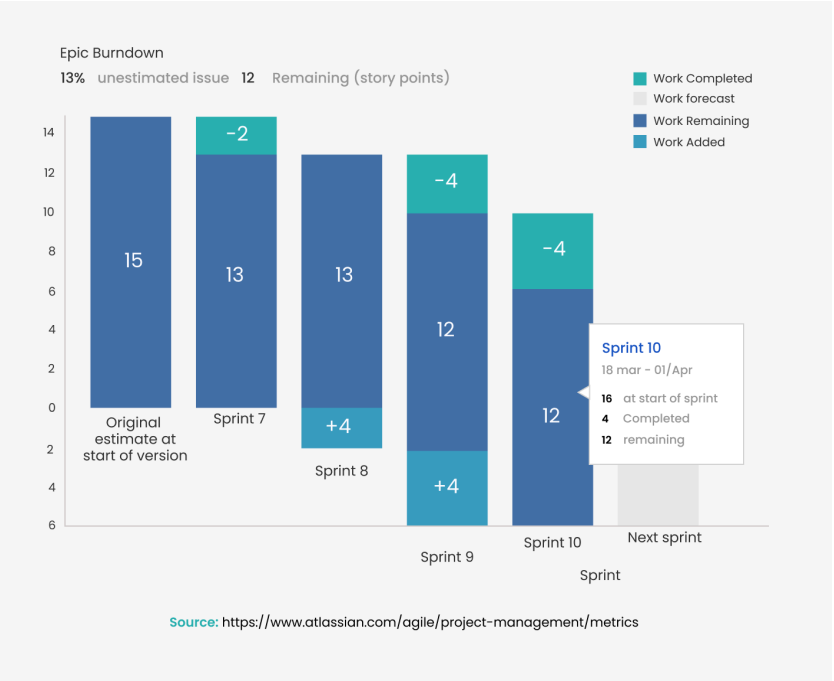
Burndown Chart
Cantidad de trabajo restante (Y) vs Tiempo (X)

- Trabajo restante real → Barras
- Trabajo restante ideal → Línea recta



Epic Burndown Chart

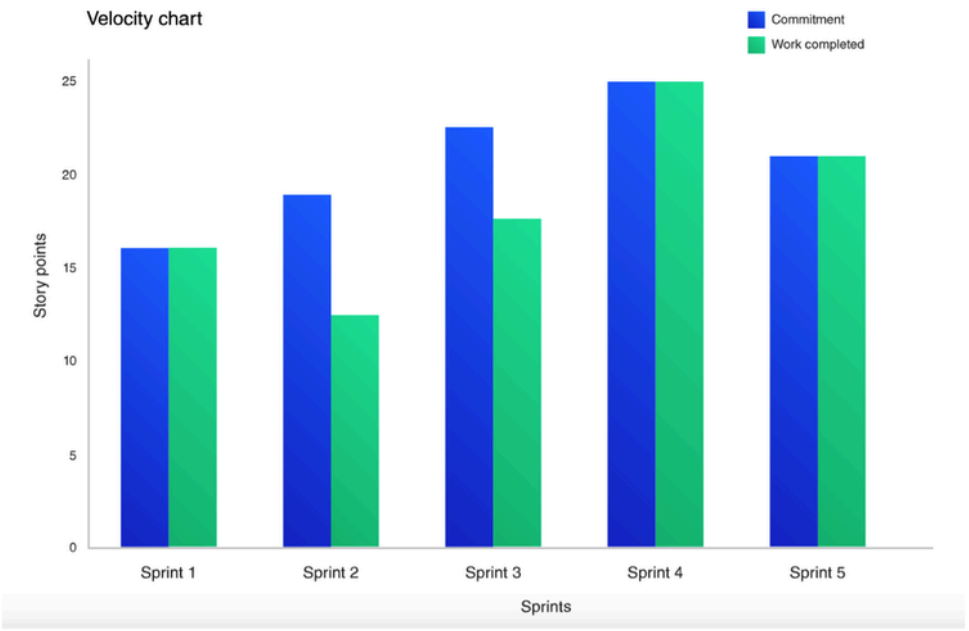
Abarca más tiempo que por sprint. Voy marcando lo completado a partir de la estimación original y lo que se me agrega en el camino.



Velocity Chart

Cumplido vs Objetivo (por Sprint)

https://www.nimblework.com/es/agile/programacion-extrema-xp/


Velocity chart
Commitment
Work completed
Story points
25
20
15
10
5
0
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprints