TESTING-US

VALIDÁ TU APRENDIZAJE

CUADERNILLO DE ESTUDIO · NIVEL INTERMEDIO

Proyectos ágiles

Scrum, Kanban y planificación

El Manifiesto Ágil, Scrum según la Guía 2020, Kanban y las técnicas para planificar y medir el avance, con casos de la práctica.

1 Valores y principios

2 Equipo Scrum

3 Eventos y artefactos

4 Kanban y flujo

5 Planificar y estimar

\+ Ejercicios con respuestas

Testing-US · Validá tu aprendizaje
testing-us.com

Edición octubre 2026

Antes de empezar

Los métodos ágiles nacieron en el desarrollo de software y hoy se usan en marketing, educación, recursos humanos y cualquier equipo que trabaje con incertidumbre: en lugar de planificar todo al principio, se entrega en partes, se mira el resultado y se ajusta. Este cuadernillo explica el Manifiesto Ágil, Scrum (según la Guía Scrum 2020), Kanban y las técnicas de planificación más usadas, con ejemplos de la práctica.

CÓMO APROVECHARLO

- Pensá cada concepto en un proyecto que conozcas: tu trabajo, un evento, una mudanza.
- Hacé los ejercicios del final de cada tema antes de mirar las respuestas.
- Cuando termines, rendí la evaluación **Gestión de proyectos ágiles** en testing-us.com.

TIP La Guía Scrum oficial es corta (unas 14 páginas) y gratuita en español: vale la pena leerla completa después de este cuadernillo.

Contenido

1 **Agilidad: valores y principios**
Predictivo y ágil, el Manifiesto, los principios y el empirismo.

2 **Scrum: equipo y responsabilidades**
Product Owner, Scrum Master, Developers y valores de Scrum.

3 **Scrum: eventos, artefactos y compromisos**
Sprint, Planning, Daily, Review, Retrospective, backlogs e Incremento.

4 **Kanban y métricas de flujo**
Tablero, límites de WIP, lead time, cycle time, throughput y ley de Little.

5 **Planificación, estimación y seguimiento**
Historias de usuario, puntos, velocidad, burndown, MoSCoW y MVP.

TEMA 1

1 Agilidad: valores y principios

LO QUE VAS A APRENDER

- Distinguir el enfoque predictivo del ágil.
- Conocer los valores del Manifiesto Ágil.
- Aplicar los principios ágiles a casos concretos.
- Entender el empirismo.

Predictivo y ágil

- En un enfoque **predictivo** (o "en cascada") se define todo al principio y las etapas van en secuencia: requisitos, diseño, construcción, pruebas y entrega. Funciona bien cuando el resultado está claro y casi no va a cambiar (por ejemplo, una obra con planos aprobados).
- En un enfoque **ágil** se trabaja en ciclos cortos: se construye una parte que ya sirve, se muestra, se aprende y se ajusta el plan. Conviene cuando hay **incertidumbre**: no se sabe bien qué necesita el cliente o cómo resolverlo.
- Trabajo **iterativo** (se repiten ciclos y se mejora) e **incremental** (cada ciclo agrega algo que funciona). El valor llega antes y los errores se descubren temprano, cuando corregirlos cuesta menos.

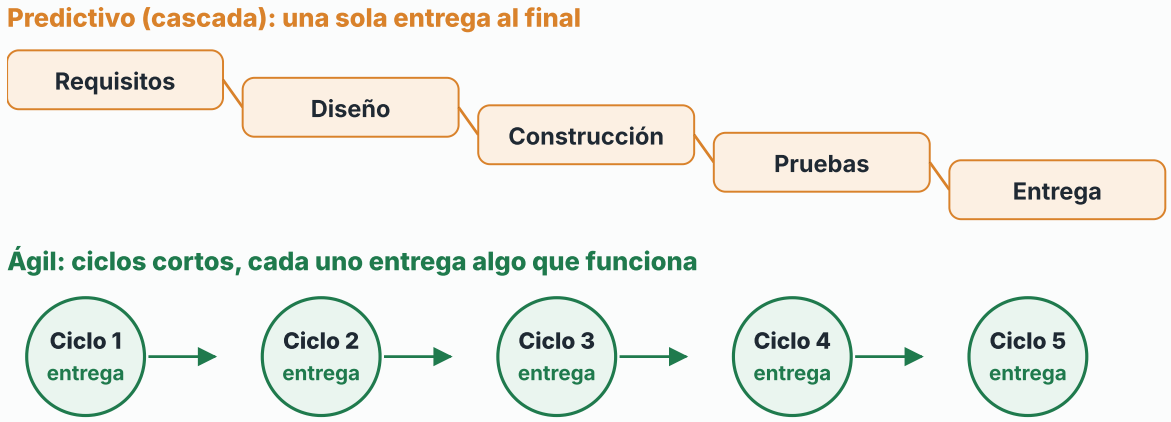


Figura 1.1 Una entrega al final frente a entregas en cada ciclo.

El Manifiesto Ágil (2001)

Diecisiete profesionales del software acordaron cuatro **valores**. Valoran:

- **Individuos e interacciones** por sobre procesos y herramientas.
- **Software que funciona** por sobre documentación extensa.
- **Colaboración con el cliente** por sobre negociación de contratos.
- **Responder al cambio** por sobre seguir un plan.

El propio manifiesto aclara que lo de la derecha también tiene valor, pero que valoran más lo de la izquierda: no dice "no documentar" ni "no planificar".

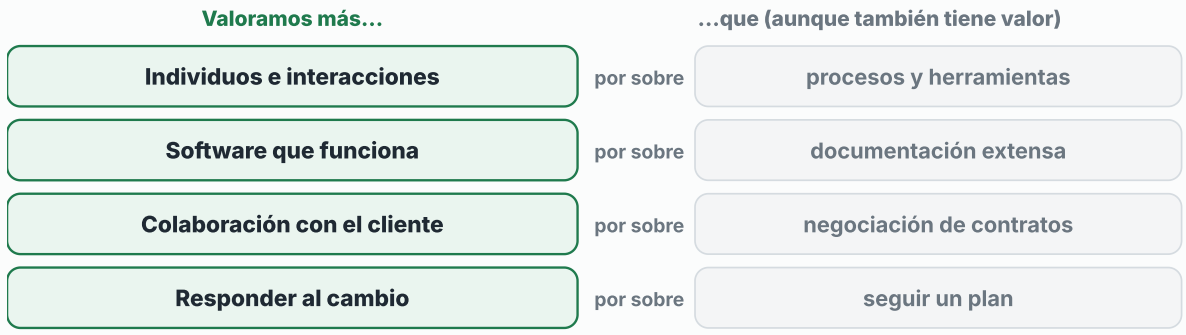


Figura 1.2 Los cuatro valores: lo de la derecha también importa, pero menos.

Algunos de los doce principios

- Satisfacer al cliente con entregas **tempranas y frecuentes** de algo que funcione (de un par de semanas a un par de meses, mejor más corto).
- Aceptar cambios en los requisitos, **aun tarde**, si le dan una ventaja al cliente.
- Gente de negocio y equipo técnico trabajando **juntos todos los días**.
- Confiar en personas motivadas y darles el entorno que necesitan; la conversación **cara a cara** es la forma más eficaz de comunicarse.
- El producto que funciona es la **medida principal de avance**.
- Un **ritmo sostenible**, que se pueda mantener en el tiempo (sin horas extra permanentes).
- La **simplicidad**, entendida como el arte de maximizar la cantidad de trabajo que **no** se hace (no construir lo que no aporta valor).
- Equipos **autoorganizados** y reflexión periódica para mejorar.

Empirismo

- Las decisiones se toman a partir de lo que se observa, no de suposiciones. Se apoya en tres pilares: **transparencia** (el trabajo y su estado están a la vista), **inspección** (se revisa con frecuencia) y **adaptación** (se ajusta cuando algo no va bien).

Para practicar

1. ¿Conviene un enfoque ágil para construir un puente con planos aprobados? ¿Por qué?
2. ¿Qué responderías a quien dice que en un proyecto ágil no se documenta?
3. Un equipo hace horas extra todas las semanas. ¿Qué principio no cumple?
4. Nombrá los tres pilares del empirismo y da un ejemplo de cada uno.

LAS RESPUESTAS ESTÁN AL FINAL DEL CUADERNILLO.

TEMA 2

2 Scrum: equipo y responsabilidades

LO QUE VAS A APRENDER

- Conocer las tres responsabilidades del Equipo Scrum.
- Distinguir qué le corresponde a cada una.
- Entender equipo multifuncional y autogestionado.
- Reconocer los valores de Scrum.

Qué es Scrum

- Un **marco de trabajo** liviano para resolver problemas complejos entregando valor en ciclos cortos llamados **Sprints**. Define pocas reglas: responsabilidades, eventos, artefactos y compromisos. No dice cómo programar ni qué herramientas usar.
- Se basa en el empirismo y en el pensamiento *lean* (reducir desperdicio y enfocarse en lo esencial).

El Equipo Scrum

- Está formado por **un Product Owner, un Scrum Master y los Developers**. No hay subequipos ni jerarquías dentro.
- Es **multifuncional** (tiene todas las habilidades para crear valor en cada Sprint) y **autogestionado** (decide internamente quién hace qué, cómo y cuándo).
- Suele tener **10 personas o menos**: los equipos chicos se comunican mejor. Si crece mucho, conviene dividirlo en varios equipos que trabajen sobre el mismo producto.

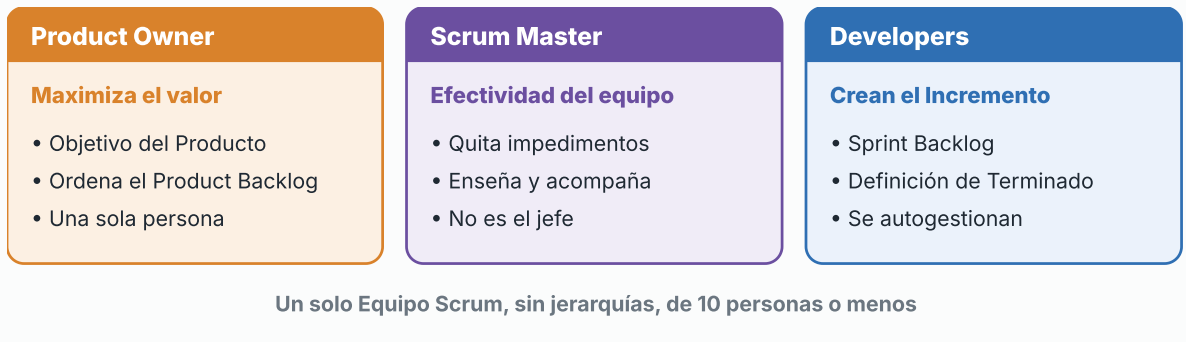


Figura 2.1 Product Owner, Scrum Master y Developers.

Product Owner

- Es responsable de **maximizar el valor** del producto.
- Gestiona el **Product Backlog**: define y comunica el **Objetivo del Producto**, crea y ordena los elementos y se asegura de que se entiendan.
- Es **una sola persona**, no un comité, aunque puede representar a muchos interesados. Puede delegar tareas, pero sigue siendo responsable.
- Es el único que puede **cancelar un Sprint** (si su objetivo dejó de tener sentido).

Scrum Master

- Es responsable de que Scrum se entienda y se aplique, y de la **efectividad** del equipo.

- Ayuda a eliminar **impedimentos**, facilita los eventos cuando hace falta, enseña y acompaña al equipo y a la organización.
- **No es el jefe** del equipo ni le asigna tareas: es un líder que sirve al equipo.

OJO Un error frecuente es convertir al Scrum Master en un jefe que asigna tareas o en un secretario que toma notas. Su trabajo es que el equipo sea cada vez más efectivo y autónomo.

Developers

- Son las personas que crean el **Incremento** en cada Sprint (no solo programadores: pueden ser diseñadores, analistas, docentes...).
- Arman el plan del Sprint (**Sprint Backlog**), cumplen la **Definición de Terminado**, adaptan el plan cada día y se hacen responsables entre ellos.

Valores de Scrum

Compromiso, foco, apertura, respeto y coraje. Por ejemplo, tener el coraje de decir que algo no va a llegar, o la apertura de mostrar los problemas.

Para practicar

1. ¿Quién ordena el Product Backlog? ¿Puede hacerlo un comité?
2. Nombrá dos tareas del Scrum Master y una que no le corresponde.
3. ¿Por qué una diseñadora puede ser Developer?
4. ¿Qué valor de Scrum muestra quien dice a tiempo que algo no va a llegar?

LAS RESPUESTAS ESTÁN AL FINAL DEL CUADERNILLO.

TEMA 3

3 Scrum: eventos, artefactos y compromisos

LO QUE VAS A APRENDER

- Conocer el Sprint y sus cuatro eventos.
- Saber para qué sirve y cuánto dura cada evento.
- Relacionar cada artefacto con su compromiso.
- Usar la Definición de Terminado.

El Sprint

- Es un período fijo de **un mes o menos** (muchos equipos usan 2 semanas) en el que se crea un Incremento. Termina uno y empieza el siguiente enseguida.
- Contiene a todos los demás eventos.
- Durante el Sprint **no se hacen cambios que pongan en riesgo el Objetivo del Sprint**, la calidad no baja y el Product Backlog se refina. El alcance se puede aclarar y renegociar con el Product Owner a medida que se aprende.

Los eventos (con su duración máxima para un Sprint de un mes)

- **Sprint Planning** (hasta 8 horas): todo el Equipo Scrum define **por qué** es valioso el Sprint (el Objetivo del Sprint); los Developers, conversando con el Product Owner, eligen **qué** se puede terminar, y deciden ellos **cómo** se va a hacer.
- **Daily Scrum** (15 minutos, todos los días hábiles): los Developers revisan el avance hacia el Objetivo del Sprint y ajustan el plan del día. No es una reunión para informar al jefe; los problemas largos se tratan después. Si el Product Owner o el Scrum Master trabajan en elementos del Sprint Backlog, participan como Developers.
- **Sprint Review** (hasta 4 horas): el equipo y los **interesados** inspeccionan el resultado y deciden qué sigue; puede cambiar el Product Backlog. Es una sesión de trabajo, no solo una presentación.
- **Sprint Retrospective** (hasta 3 horas): el equipo revisa **cómo trabajó** (personas, interacciones, procesos, herramientas, Definición de Terminado) y elige mejoras. Es el último evento del Sprint.
- Para Sprints más cortos, los eventos suelen ser más cortos.
- El **refinamiento** del Product Backlog (dividir, detallar y estimar elementos) es una actividad continua, no un evento.

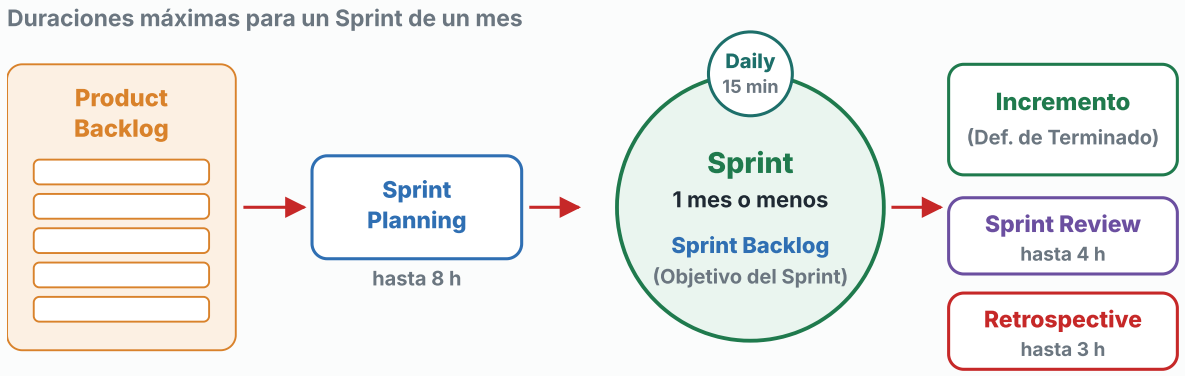


Figura 3.1 El Sprint contiene a todos los eventos.

Artefactos y compromisos

Cada artefacto tiene un compromiso que da foco:

- **Product Backlog** → **Objetivo del Producto**. Es la lista ordenada y cambiante de todo lo que podría mejorar el producto. Los elementos de arriba están más detallados.
- **Sprint Backlog** → **Objetivo del Sprint**. Tiene el objetivo, los elementos elegidos y el plan para entregarlos. Es de los Developers y lo actualizan durante el Sprint.
- **Incremento** → **Definición de Terminado**. Es un paso concreto hacia el Objetivo del Producto, que se puede usar. Un trabajo que no cumple la Definición de Terminado **no es parte del Incremento** y vuelve al Product Backlog.
- Se pueden entregar varios Incrementos dentro de un mismo Sprint: no hace falta esperar a la Sprint Review.

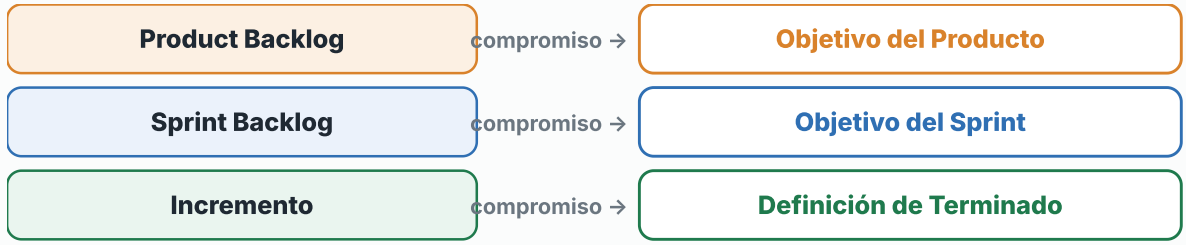


Figura 3.2 Cada artefacto tiene un compromiso que le da foco.

Definición de Terminado

- Describe la **calidad** que tiene que tener un trabajo para considerarse terminado (por ejemplo: probado, revisado por otro, documentado y publicado).
- Da transparencia: todos entienden lo mismo por "terminado".

Para practicar

1. ¿Qué diferencia hay entre la Sprint Review y la Sprint Retrospective?
2. A mitad del Sprint llega un pedido urgente que pone en riesgo el objetivo. ¿Qué se hace?
3. Un trabajo está hecho pero no probado y la Definición de Terminado pide pruebas. ¿Es parte del Incremento?
4. ¿Cuánto dura la Daily Scrum y quiénes participan?

LAS RESPUESTAS ESTÁN AL FINAL DEL CUADERNILLO.

TEMA 4

4 Kanban y métricas de flujo

LO QUE VAS A APRENDER

- Armar un tablero con columnas y políticas.
- Usar límites de trabajo en curso (WIP).
- Medir lead time, cycle time y throughput.
- Detectar cuellos de botella.

Qué es Kanban

- Un método para **visualizar y mejorar el flujo** de trabajo. Se puede empezar con el proceso que ya se tiene: no exige roles nuevos ni iteraciones fijas.
- El **tablero** muestra las etapas del trabajo en columnas (por ejemplo: Por hacer, En curso, En revisión, Hecho) y cada tarea es una tarjeta que avanza.
- Las **políticas explícitas** dejan claro cuándo una tarjeta puede pasar de una columna a la siguiente (por ejemplo, "pasa a Revisión solo con las pruebas hechas").

Limitar el trabajo en curso (WIP)

- Las columnas de trabajo activo tienen un **límite de WIP** (*work in progress*): la cantidad máxima de tarjetas al mismo tiempo. Si la columna En curso tiene límite 3 y ya hay 3, no se empieza nada nuevo: primero se termina o se ayuda a destrabar.
- Es un sistema de **tirar** (*pull*): cada persona toma trabajo cuando tiene lugar, en vez de que se lo empujen.
- Menos trabajo empezado a la vez = menos cambios de tarea, menos esperas y entregas más rápidas. "Dejá de empezar, empezá a terminar."
- Cuando una columna se llena y la siguiente está vacía, se ve un **cuello de botella**.

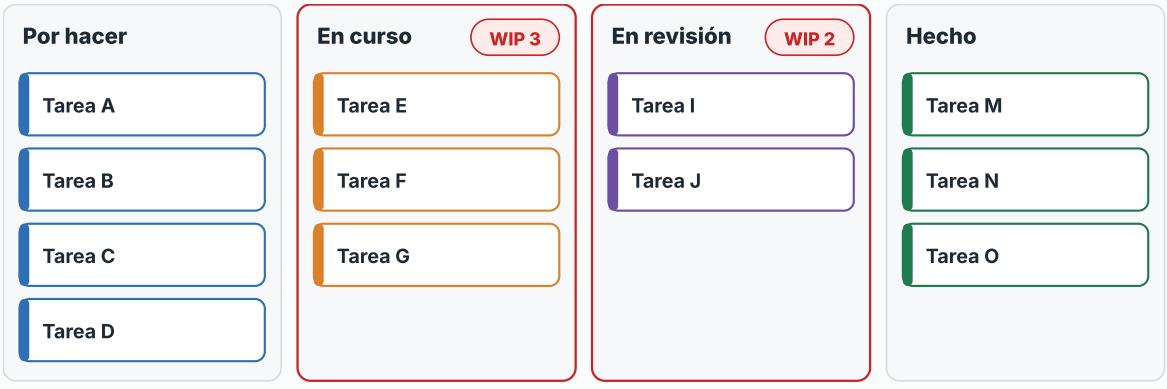


Figura 4.1 Un tablero Kanban con dos columnas en su límite.

Métricas de flujo

- **Lead time**: el tiempo desde que se pide algo hasta que se entrega (lo que vive el cliente).
- **Cycle time**: el tiempo desde que se **empieza** a trabajar en algo hasta que se termina.
- **Throughput**: cuántos elementos se terminan por período (por ejemplo, 12 tarjetas por semana).
- **WIP**: cuántos elementos hay empezados y sin terminar.

- La **ley de Little** las relaciona en promedio: cycle time = WIP ÷ throughput. Con 10 tareas en curso y 5 terminadas por semana, cada una tarda en promedio 2 semanas.
- El **diagrama de flujo acumulado** (CFD) muestra cuántos elementos hay en cada etapa a lo largo del tiempo: una franja que se ensancha indica que ahí se acumula trabajo.

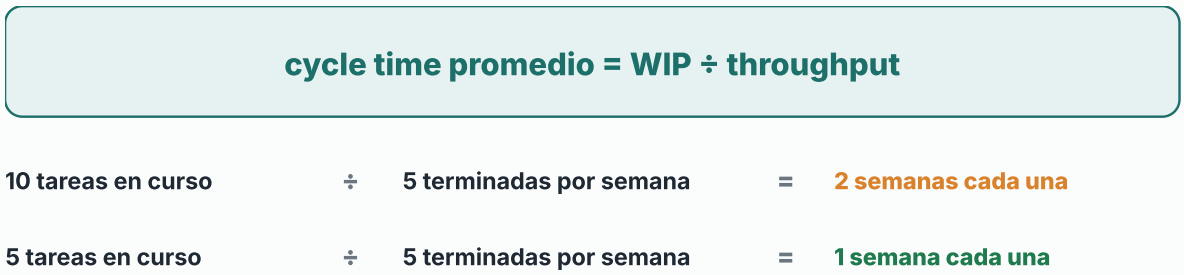


Figura 4.2 Menos trabajo en curso, con el mismo ritmo, es menos tiempo de espera.

Scrum y Kanban juntos

- Muchos equipos Scrum usan un tablero con límites de WIP para gestionar el Sprint (a veces se lo llama *Scrumban*).
- Kanban suele convenir para trabajo que llega de forma continua e impredecible (soporte, mesa de ayuda, mantenimiento).

Para practicar

1. ¿Qué hacés si te desocupás y la columna En curso está en su límite?
2. ¿Qué diferencia hay entre lead time y cycle time?
3. Con 12 tareas en curso y 4 terminadas por semana, ¿cuánto tarda en promedio cada una?
4. ¿Para qué tipo de trabajo conviene Kanban?

LAS RESPUESTAS ESTÁN AL FINAL DEL CUADERNILLO.

TEMA 5

5 Planificación, estimación y seguimiento

LO QUE VAS A APRENDER

- Escribir historias de usuario con criterios de aceptación.
- Dividir épicas en partes que entreguen valor.
- Estimar en forma relativa y usar la velocidad.
- Leer gráficos burndown y burnup, y priorizar.

Historias de usuario

- Describen una necesidad desde el punto de vista de quien la usa: **"Como** [tipo de usuario], **quiero** [algo], **para** [beneficio]". Ejemplo: "Como alumna, quiero descargar mi certificado en PDF para enviarlo con mi CV".
- Una historia es una invitación a conversar, no una especificación completa.
- Los **criterios de aceptación** dicen cuándo la historia está cumplida. Se suelen escribir como escenarios: **Dado** [contexto], **cuando** [acción], **entonces** [resultado].
- La sigla **INVEST** resume cómo debería ser una buena historia: Independiente, Negociable, Valiosa, Estimable, pequeña (*Small*) y Testeable.
- Una **épica** es una historia demasiado grande para un Sprint: se divide en historias más chicas, preferentemente "en rebanadas" que atraviesen todo (de punta a punta) y entreguen algo usable.

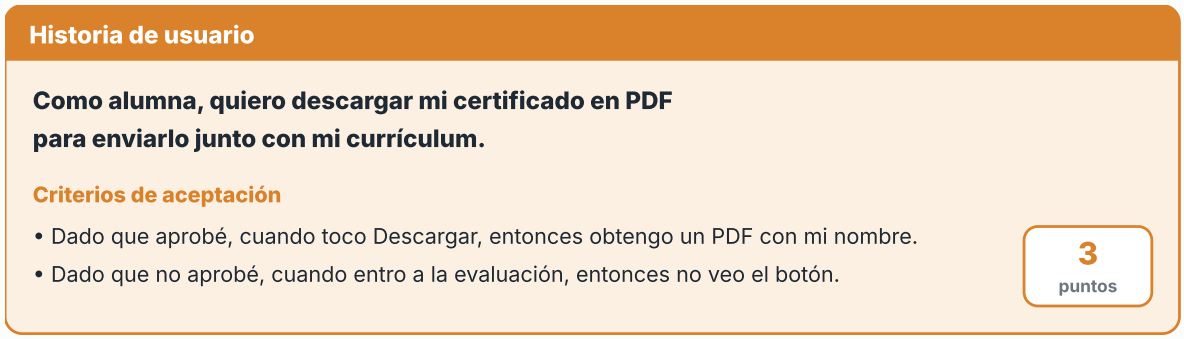


Figura 5.1 Una historia, sus criterios de aceptación y su estimación.

Estimar

- Muchos equipos estiman en **forma relativa**: comparan el esfuerzo de un elemento con otros ya conocidos, en lugar de calcular horas.
- Los **puntos de historia** miden tamaño relativo (esfuerzo, complejidad e incertidumbre). Se usan escalas como 1, 2, 3, 5, 8, 13: los números se separan a medida que crecen porque la incertidumbre también crece.
- En el **Planning Poker** cada persona elige su estimación en secreto y se muestran todas juntas; si hay mucha diferencia, quienes pusieron el valor más alto y el más bajo explican por qué y se vuelve a votar.
- Estiman quienes hacen el trabajo (los Developers), no el Product Owner ni un jefe.

Velocidad y gráficos de avance

- La **velocidad** es la cantidad de puntos terminados por Sprint. Sirve para que **el mismo equipo** pronostique: no sirve para comparar equipos, porque cada equipo usa su propia escala.
- Los puntos de un trabajo que **no cumple la Definición de Terminado** no suman a la velocidad.
- El **burndown** muestra el trabajo que **falta** a lo largo del Sprint (baja hacia cero). El **burnup** muestra el trabajo **terminado** y el total: deja ver si el alcance creció.

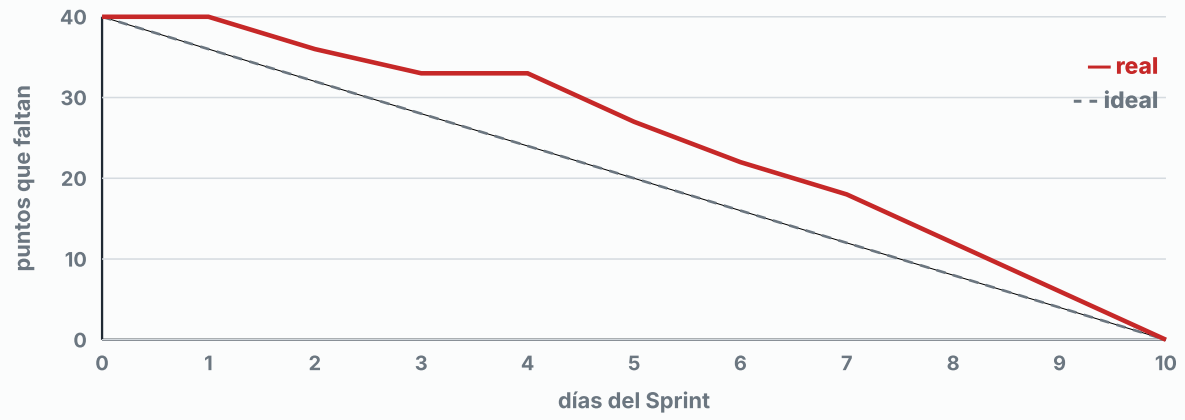


Figura 5.2 Burndown: la línea real se compara con la ideal.

Priorizar y entregar valor

- El Product Backlog se ordena por **valor**, riesgo, dependencias y esfuerzo. Una técnica simple es **MoSCoW**: Must (imprescindible), Should (importante), Could (deseable), Won't (no por ahora).
- El **MVP** (producto mínimo viable) es la versión más chica que permite aprender de usuarios reales con el menor esfuerzo.
- La **deuda técnica** es el costo futuro de los atajos tomados hoy (código desprolijo, falta de pruebas): si no se paga, el equipo va cada vez más lento.

Para practicar

1. Escribí una historia de usuario para recibir un aviso por mail al aprobar una evaluación.
2. ¿Por qué no conviene comparar la velocidad de dos equipos?
3. En un Planning Poker salen 3, 3, 5 y 13. ¿Qué hacés?
4. Clasificá con MoSCoW: login, modo oscuro, pago con tarjeta, exportar a Excel.

LAS RESPUESTAS ESTÁN AL FINAL DEL CUADERNILLO.

PARA CONTROLAR

Respuestas de los ejercicios

Tema 1

1. No: el resultado está definido y casi no va a cambiar; conviene un enfoque predictivo.
2. Que el Manifiesto valora más el producto que funciona, pero la documentación también tiene valor.
3. El ritmo sostenible.
4. Transparencia (tablero a la vista), inspección (revisiones frecuentes) y adaptación (cambiar el plan si algo no va).

Tema 2

1. El Product Owner; es una sola persona, no un comité.
2. Quitar impedimentos y enseñar Scrum; no le corresponde asignar tareas ni ordenar el Product Backlog.
3. Porque Developer es cualquier persona que crea el Incremento, no solo quien programa.
4. Coraje (y también apertura).

Tema 3

1. La Review mira el producto con los interesados; la Retrospective mira cómo trabajó el equipo.
2. No se cambia el Sprint: el pedido va al Product Backlog, salvo que el Product Owner cancele el Sprint.
3. No: vuelve al Product Backlog.
4. 15 minutos; participan los Developers (el Product Owner y el Scrum Master, si trabajan en elementos del Sprint).

Tema 4

1. Ayudar a terminar o destrabar lo que está en curso, sin empezar nada nuevo.
2. Lead time: desde el pedido hasta la entrega. Cycle time: desde que se empieza hasta que se termina.
3. 3 semanas (12 ÷ 4).
4. Trabajo que llega de forma continua e impredecible, como soporte o mantenimiento.

Tema 5

1. Como validante, quiero recibir un mail cuando apruebo, para tener el link del certificado a mano.
2. Porque cada equipo usa su propia escala de puntos.
3. Quien puso 3 y quien puso 13 explican sus motivos y se vuelve a votar.
4. Must: login y pago con tarjeta. Should: exportar a Excel. Could: modo oscuro (puede variar según el producto).

Validá lo que aprendiste

Rendí la evaluación **Gestión de proyectos ágiles** en **testing-us.com**. Es gratis: en cada intento te tocan 10 preguntas al azar, dos de cada tema de este cuadernillo. Tenés 15 minutos y aprobás con 70%.

Si aprobás, recibís un certificado con código de verificación que podés sumar a tu currículum y compartir con quien te contrate.

MIS NOTAS

© 2026 Testing-US. Material de estudio propio. Se permite imprimirlo y compartirlo sin modificaciones y sin fines comerciales.

Referencias: Manifiesto por el Desarrollo Ágil de Software (2001, agilemanifesto.org) y La Guía Scrum de Ken Schwaber y Jeff Sutherland (noviembre de 2020, licencia CC BY-SA 4.0, scrumguides.org), explicados con palabras propias; prácticas difundidas del método Kanban. Este material no está respaldado por los autores de esas obras. Scrum.org y Scrum Alliance ofrecen certificaciones oficiales de Scrum; el certificado de Testing-US no las reemplaza.

El certificado de Testing-US acredita la aprobación de una evaluación de conocimientos. No es un título oficial ni habilita para el ejercicio profesional.