Ingeniería de Software II

Metodologías Ágiles

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología

Lic. Paolo Orundés Cardinali
orundescardinali.paolo@uader.edu.ar

Lic. Damián Cian
cian.damian@uader.edu.ar

Tabla de contenido

Historia

En marzo de 2001 diecisiete críticos de los modelos de mejora del desarrollo de software basados en procesos, convocados por Kent Beck, quien había publicado un par de años antes Extreme Programming Explained, libro en el que exponía una nueva metodología denominada Extreme Programming, se reunieron en Salt Lake City para tratar sobre técnicas y procesos para desarrollar software. En la reunión se acuñó el término "Métodos Ágiles" para definir a los métodos que estaban surgiendo como alternativa a las metodologías formales (CMMI, SPICE) a las que consideraban excesivamente "pesadas" y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. Los integrantes de la reunión resumieron los principios sobre los que se basan los métodos alternativos en cuatro postulados, lo que ha quedado denominado como Manifiesto Ágil.

Manifiesto de cuatro puntos

Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- ✓ A los individuos y su interacción, por encima de los procesos y las herramientas.
- ✓ El software que funciona, por encima de la documentación exhaustiva.
- ✓ La colaboración con el cliente, por encima de la negociación contractual.
- ✓ La respuesta al cambio, por encima del seguimiento de un plan.

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.

¿Qué son las metodologías agiles?

Las metodologías ágiles consisten en una serie de estrategias, cuyo principio es la mejora continua empresarial, principalmente aplicadas en el sector del desarrollo de software. La característica que las define es la agilidad de los procesos productivos. Este conjunto recibe el nombre formal de **Agile Project Management**, es decir**, Gestión de Proyectos Ágil**.

Su aplicación en el mercado tecnológico supone la adaptación de los procesos productivos a las necesidades del mercado del momento. Por lo tanto, su estrategia se traduce en **innovación y en una constante renovación** de las fórmulas empleadas.

Estas metodologías se implementan **durante la ejecución de las actividades** que contribuyen a la producción de un producto o a la prestación de un servicio. Se toman y se aplican decisiones durante la marcha, por lo que su enfoque es totalmente práctico y activo.

Además, **su punto de focalización es el cliente**, el indicador de la rentabilidad y la base para la producción de una empresa. Responder a las necesidades y demandas del mercado es uno de los objetivos de las metodologías ágiles, lo que conlleva a su vez la capacidad de adaptarse a las circunstancias de manera flexible.

En efecto, la base de las metodologías ágiles se encuentra en la constante retroalimentación y la comunicación con los potenciales clientes. A partir del frecuente lanzamiento de productos experimentales, se analizan los comentarios y el recibimiento de los mismos para aplicar mejoras a los procedimientos de manera periódica.

En resumen:

- ✓ Las metodologías ágiles consisten en una serie de estrategias, cuyo principio es la mejora continua empresarial, llamadas Agile Project Management.
- ✓ Aplicado a un mercado tecnológico se traduce en constante innovación y renovación.
- ✓ Se implementan durante la ejecución de las actividades.
- ✓ Su punto de focalización es el cliente

Las Metodologías Ágiles (AMs) valoran:

- Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas
- Desarrollar software que funciona más que conseguir una buena documentación.
- Minimalismo respecto del modelado y la documentación del sistema
- La colaboración con el cliente más que la negociación de un contrato
- Responder a los cambios más que seguir estrictamenteuna planificación

Los 12 principios de Manifiesto Ágil

1. **El cliente como objetivo**: la base de todo proyecto debe ser el cliente. Por esta razón, es importante conocer sus opiniones a través del lanzamiento de breves funciones para mejorarlas en la producción del producto final.
2. **Flexibilidad**: los procesos están sujetos a cambios constantemente, de acuerdo a las dificultades que se presenten o a la variación de las demandas y requisitos del proyecto.
3. **Producción de prototipos**: los prototipos o sprints de un proyecto son una serie de funciones presentadas individualmente al público, con el fin de implementar los cambios constantemente en su integración con el producto final.
4. **Integración de los miembros del equipo**: la productividad de una empresa depende también de la correcta integración de los miembros del equipo y sus tareas. En este caso, las relaciones son un aspecto a tener en cuenta.
5. **Motivación**: un equipo motivado garantiza el cumplimiento de los objetivos del proyecto y la calidad de sus resultados.
6. **Comunicación**: este elemento es fundamental con respecto al trabajo en equipo, pues se hace necesario que todos los miembros del proyecto conozcan las decisiones y las necesidades que se presenten para ejecutar, oportunamente, los cambios pertinentes.
7. **Funcionalidad como indicador de progreso**: que el producto funcione correctamente y que se adapte a las necesidades del público, constituye un síntoma de progreso con respecto al desarrollo del proyecto.
8. **Soporte**: establecer medios de soporte y mantenimiento de los productos y servicios garantiza la correcta funcionalidad de los mismos después de haber sido comercializados.
9. **Excelencia**: la excelencia de los procesos productivos propicia la calidad de los resultados, así como la efectividad de los mismos procesos.
10. **Sencillez**: los procesos simples permiten un mayor nivel de control sobre ellos, así

como su análisis de manera rápida, identificando los cambios pertinentes que se requieran realizar.

11. **Coordinación**: la interdependencia de las tareas dentro de los procesos productivos de una empresa implica la coordinación de las mismas para el cumplimiento de los objetivos del proyecto.
12. **Aprendizaje**: cada etapa productiva implica un periodo de reflexión sobre la misma, con el fin de analizar qué funcionó y qué no. El aprendizaje constante tanto de los procesos como del entorno permite proponer soluciones y cambios que optimicen la productividad empresarial.

Tipos de metodologías Ágiles

Las metodologías ágiles son, como se ha mencionado, un conjunto de estrategias, cuyos enfoques particulares hacen que unas se adapten de mejor manera y con resultados más óptimos que otras. La siguiente es una lista de metodologías que se desempeñan de una u otra manera en el campo productivo empresarial:

Método Scrum

Enfocada en la gestión

El método Scrum se caracteriza por la división de las responsabilidades. Uno de los elementos fundamentales de este método son los **Sprints**, proyectos breves que conforman el proyecto definitivo.

Método Lean

Enfocada en maximizar el valor para el cliente y minimizar el desperdicio en los procesos de producción

En este método se identifican aquellos procesos que no contribuyen a la realización de los objetivos, o que no afectan de manera positiva la productividad, con el fin de enfocarse en aquellos que sí lo hacen, de manera que se da un ahorro de los recursos.

Extreme Programming XP

Enfocada en la parte técnica. Uso de iteración.

El enfoque del método Extreme Programming XP está dirigido a establecer relaciones perdurables entre clientes y trabajadores, así como entre los mismos integrantes del proyecto. En este método prevalece el trabajo en equipo, pues su esquema estratégico está diseñado para implementarse en proyectos informáticos, donde es necesario que exista una reciprocidad laboral.

Método Kanban

La metodología Kanban divide el desarrollo de un proyecto en diversas etapas dispuestas de manera que destaquen las más importantes por terminar. Para esto, se recurre al tablero Kanban definido por tres columnas (que se va a realizar, que se está realizando, que se realizó) y no

necesariamente se usa ni sprints ni iteraciones, (se hacen tareas cortas y de excelencia), donde sí se utilizan tarjetas que representan la importancia de una tarea dentro del contexto de proyecto.

Diez personas trabajando en dos tareas (para evitar cuello de botella) a diferencia de scrum y XP que diez personas trabajando en diez tareas.

Agile Inception

El método Agile Inception propicia la iniciativa de las actividades y proposición de los objetivos por parte de todos los integrantes del proyecto, con el fin de unificar las aspiraciones individuales en una visión y misión que caracterizarán los procesos productivos, lo que implica la planificación y comunicación.

Design Sprint

Esta metodología se trata de diseñar un prototipo, o sprints, del producto final, con el fin de mostrarse al público esperando de él la retroalimentación necesaria para la continuación del proyecto.

Pasos y etapas de las metodologías Ágiles

Las metodologías ágiles están diseñadas para implementarse de manera periódica y cíclica, de manera que la empresa establezca nuevos objetivos y mejore sus procesos de manera constante. Existen una serie de pasos recomendados para la implementación de estas metodologías, los cuales son:

- **Análisis de los requisitos del proyecto**: se analizan las demandas y necesidades del mercado, y se proponen los objetivos y enfoques del proyecto. Uno de los elementos más importantes en esta etapa y cuyo análisis es necesario es las opiniones de los potenciales clientes.
- **Planificación de las actividades**: en esta etapa se dividen las responsabilidades del proyecto, los tiempos límites para la entrega de cada componente del producto, y su posterior análisis de acuerdo a los comentarios de los usuarios.
- **Reunión del equipo**: el equipo analiza el plan diseñado, y cada miembro propone posibles cambios, expone sus objetivos y se unifican en una sola misión.
- **Producción**: es la etapa productiva del proyecto, en la que se empieza el desarrollo de sprints o prototipos. Durante esta etapa, se pueden implementar modificaciones de acuerdo a las dificultades de los procesos en marcha.
- **Análisis, documentación y lanzamiento**: por último, y con el sprint una vez terminado, se analizan y documentan la eficacia de las líneas de producción en el desarrollo de proyecto. Se realiza el lanzamiento del sprint y se estudian los comentarios de los clientes.

Aunque las metodologías ágiles no presentan un esquema específico a seguir, con el objetivo de propiciar la flexibilidad a la hora de realizar cambios pertinentes a los procesos, el anterior esquema se suele implementar para la implementación de estas estrategias.

Al finalizar la implementación de un ciclo, se repite el proceso nuevamente, verificando los resultados y proponiendo nuevos cambios de acuerdo al contexto actual.

Principales roles en las metodologías Ágiles

Dentro de la implementación de las metodologías ágiles existen ciertos roles cuyas funciones individuales permiten que los procesos y las estrategias se ejecuten de manera correcta. Estos son:

- **Agile Coach**: está relacionado con la capacitación de capital humano y su organización. Garantiza la calidad y excelencia de los medios de trabajo para que los miembros del equipo dispongan de ellos correctamente, agilizando de esta manera los procesos.
- **Scrum Máster**: es el garantizador de que se cumpla el esquema planificado de actividades y objetivos. Gestiona la calidad de los procedimientos y se enfoca en la producción de sprints. Es aquel que guía al equipo en la metodología aplicada.
- **Project Manager**: es aquel que gestiona el proyecto en ejecución, lo que implica conocer cada uno de los procesos y actividades para responder a las situaciones que se presenten. Es aquel que dirige al equipo.
- **Product Owner**: se encarga del correcto desarrollo de un producto en específico. Es aquel que está en contacto permanente con el equipo y el cliente y quien determina las prioridades.

Ventajas y desventajas de las metodologías Ágiles

Ventajas	Desventajas
✓ Flexibilidad para adaptarse a cambios del mercado y del proyecto.	× Requiere implementación periódica y seguimiento constante.
✓ Entrega continua de valor y agilidad en la producción.	× Puede generar incertidumbre en proyectos con alcance poco claro.
✓ Fomenta la innovación y la mejora continua.	× Requiere análisis constante del escenario del mercado.
✓ Incrementa la competitividad de la empresa.	× El cambio continuo puede afectar la estabilidad de la producción y actividades.
✓ Mejora la calidad de productos y servicios.	× Menor documentación formal puede dificultar el mantenimiento a largo plazo.
✓ Mejora la comunicación, colaboración y motivación del equipo.	× Depende del compromiso y madurez del equipo de trabajo.
✓ Detecta errores tempranamente y permite aplicar cambios rápidamente.	× Puede ser difícil de escalar en organizaciones grandes o poco maduras.

¿Cómo aplicar la metodología Ágil a tu proyecto o empresa?

Lo principal es identificar las necesidades del mercado y de los clientes. Las metodologías ágiles están diseñadas para implementarse en el sector informático, por lo que en este caso se analizan las nuevas tecnologías y funcionalidades presentes en el sector, así como las soluciones en desarrollo para los problemas actuales. Sin embargo, su aplicación se extiende hasta cualquier ámbito empresarial.

Siguiendo el esquema previamente previamente expuesto, la aplicación de la metodología ágil se podría resumir en los siguientes pasos:

- Identificar las demandas del público.
- Proponer soluciones.

- Reunir las ideas como un solo proyecto.
- Establecer los objetivos de acuerdo a las necesidades del cliente.
- Planificar las actividades y reunir al equipo.
- Coordinar a los miembros del equipo y sus procesos enfocándose en el objetivo.
- Verificar los procesos y proponer cambios siempre teniendo en cuenta los requisitos del proyecto.
- Lanzamiento del prototipo.
- Análisis de la retroalimentación.
- Aplicación de cambios.

¿Cuál es la diferencia entre una metodología tradicional y ágil?

La principal diferencia entre las metodologías tradicionales y las metodologías ágiles, radica en que en las primeras el proceso es lineal y secuencial, mientras que en las segundas es proceso es iterativo.

En las metodologías en cascada los requisitos han de estar bien definidos desde el principio, mientras que en las metodologías agiles los requisitos se toman de una manera más dinámica.

Uno nos aporta una mayor precisión, mientras que el otro nos da una mayor flexibilidad y adaptabilidad.

Enfoque predictivo vs enfoque adaptativo

Criterio	Cascada	RUP (Rational Unified Process)	Ágil (Scrum, XP, etc.)
Enfoque	Secuencial, fase por fase	Iterativo e incremental, con fases definidas	Iterativo e incremental, basado en ciclos cortos
Planificación	Rígida, definida desde el inicio	Planificada por fases, con revisiones entre iteraciones	Flexible, se adapta en cada iteración o sprint
Gestión del cambio	Difícil y costosa una vez iniciada la ejecución	Moderadamente flexible entre fases	Altamente flexible, el cambio es parte del proceso
Entrega del producto	Una sola entrega al final	Entregas incrementales en cada fase	Entregas frecuentes en cada iteración
Participación del cliente	Limitada (inicio y validación final)	Presente en revisiones, pero no constante	Activa y constante durante todo el proceso
Documentación	Muy detallada y extensa	Moderada a extensa, con foco en artefactos definidos	Ligera, solo la necesaria para el desarrollo
Detección de errores	Tarde, al final del proceso	Se detectan en cada iteración o fase	Temprana y continua en cada sprint o ciclo
Adaptabilidad	Muy baja	Media, permite ajustes entre iteraciones	Muy alta, se adapta continuamente
Tamaño de proyectos	Ideal para proyectos grandes con requisitos estables	Apto para proyectos medianos a grandes con cierto nivel de cambio	Ideal para proyectos pequeños a medianos con alto nivel de cambio

Y vimos que RUP es una metodología tradicional, pero con enfoque Iterativo e incremental, como la definimos entonces:

RUP (Rational Unified Process) es un caso **intermedio**, pero se lo considera generalmente una **metodología tradicional adaptable** o **híbrida**, **no ágil en su forma original**, aunque **puede incorporar prácticas ágiles**.

¿Por qué?

- RUP es **iterativo e incremental**, lo cual se parece a lo ágil.
- Pero también es **pesado en documentación**, roles definidos, fases bien marcadas (inicio, elaboración, construcción, transición) y planificación estructurada, lo cual se alinea más con metodologías tradicionales como **Cascada**.

Entonces:

- **RUP en su versión clásica ➤ Tradicional estructurado** (aunque iterativo).
- **RUP adaptado con prácticas ágiles (Agile RUP) ➤** Se puede usar en entornos más ágiles, pero no es ágil "puro" como Scrum o XP.

Scrum, kanban y XP

Scrum

Scrum es un marco (framework) de gestión de proyectos de metodología ágil que ayuda a los equipos a estructurar y gestionar el trabajo mediante un conjunto de valores, principios y prácticas. Scrum anima a los equipos a aprender a través de las experiencias, a auto organizarse mientras aborda un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente.

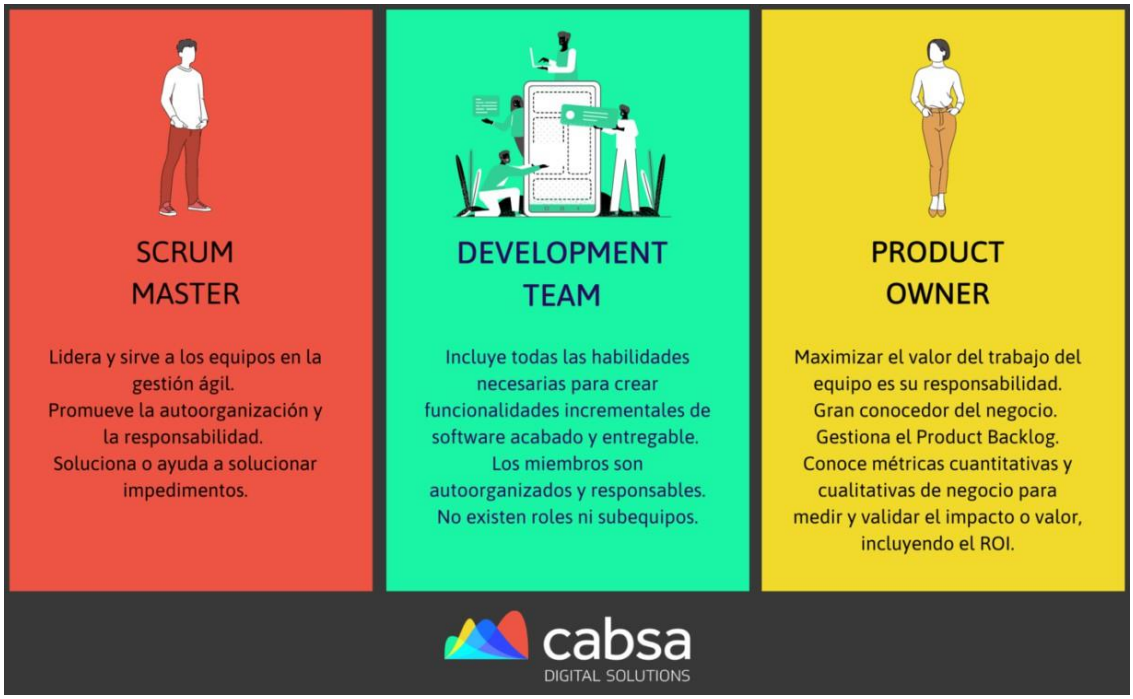
Scrum incluye un conjunto de reuniones, herramientas y funciones que, de forma coordinada, ayudan a los equipos a estructurar y gestionar su trabajo.

El software está en constante evolución y cambio por lo que Scrum enmarca estos cambios en forma ágil en una serie de iteraciones conocidas como Sprints que dividen proyectos grandes y complejos en partes más pequeñas.

Roles

Scrum, como metodología ágil, tiene 3 roles principales. Cada uno de ellos tiene unas responsabilidades muy definidas y, a través de su correcto desempeño, podemos llevar a cabo con éxito nuestros proyectos de software a medida para nuestros clientes.

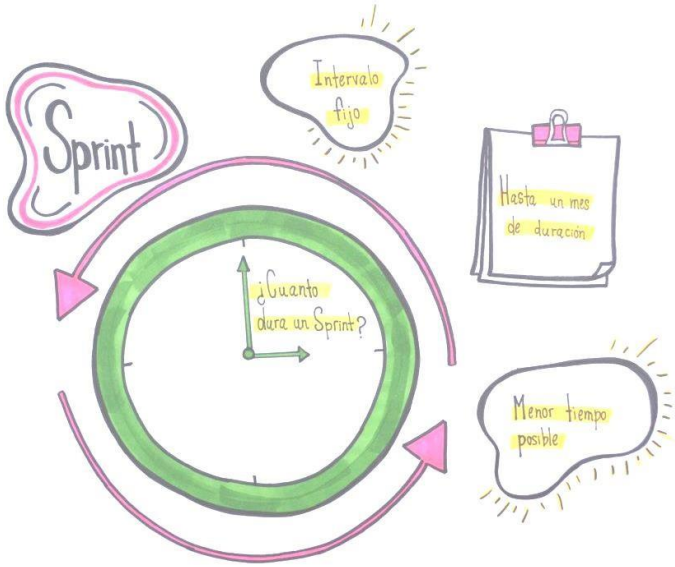
Averigua un poco más sobre la labor del Scrum Master, el Development Team y el Product Owner con nuestra infografía.



Sprints

Un sprint es un período breve de tiempo fijo en el que un equipo de scrum trabaja para completar una cantidad de trabajo establecida.

Los sprints ayudan a los equipos a seguir el principio de la metodología ágil de "entregar software de trabajo con frecuencia", así como vivir el valor ágil de "dar respuesta a un cambio por encima del seguimiento de un plan". Los valores de transparencia, inspección y adaptación de la metodología scrum son complementarios a los ágiles y centrales al concepto de sprints.



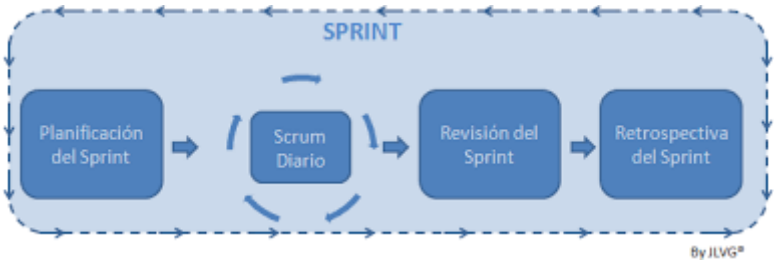
Qué es una planificación de Sprints, cómo planificar y ejecutarlos

La planificación de sprints es un evento en scrum que inicia el sprint. El objetivo de la planificación de sprints es definir lo **que se puede entregar en el sprint y cómo se conseguirá** ese trabajo. La planificación de sprints se hace en colaboración con todo el equipo de scrum, el cual responde a dos preguntas básicas:

- Qué trabajo se puede llevar a cabo en este sprint.
- Cómo se realizará el trabajo elegido.

Preparar y ejecutar un sprint

- Para preparar un sprint tienes que decidir cuánto tiempo va a durar, el objetivo del sprint y por dónde vas a empezar.
- La sesión de planificación de sprints inicia el sprint definiendo el orden del día y el punto de atención.
- La planificación de sprints debería limitarse a no más de dos horas por cada semana del sprint.



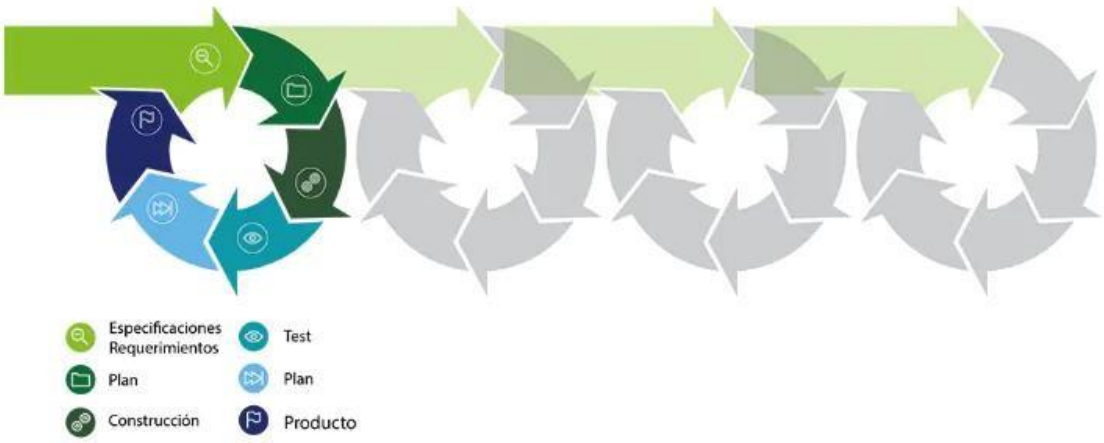
Elegir los elementos de trabajo adecuados para un sprint es un esfuerzo de colaboración entre el propietario del producto, el experto en scrum y el equipo de desarrollo.

- **El qué**: El propietario del producto analiza el objetivo que debería conseguir el sprint y los elementos del backlog del producto que, una vez finalizados, contribuirían a alcanzar dicho objetivo.
- **El cómo**: A continuación, el equipo crea un plan sobre cómo crearán y finalizarán los elementos del backlog antes del fin del sprint. Los elementos de trabajo elegidos y el plan sobre cómo completarlos se conocen como el backlog del sprint. Al terminar de planificar un sprint, el equipo está listo para empezar a trabajar en el backlog del sprint, es decir, completará todos los elementos del backlog que tengan el estado "En curso".
- **Los elementos de trabajo**: En un sprint, el equipo comprueba durante el scrum diario (o reunión rápida) cómo progresa el trabajo. El objetivo de esta reunión es descubrir los impedimentos y retos que afectarían a la capacidad de los equipos para alcanzar el objetivo del sprint.
- Después de un sprint, el equipo demuestra el trabajo que ha completado durante la revisión. De este modo, el equipo tiene la oportunidad de exponer su trabajo ante las partes interesadas y compañeros antes de poner en marcha la producción.
- **El resultado**: Completa tu ciclo de sprints con una reunión**: la retrospectiva del sprint**. De esta forma, los equipos tienen la oportunidad de identificar áreas de mejora para el próximo sprint.

Protocolos

Dentro del marco scrum existen cinco protocolos o eventos a saber conocidas como ceremonias scrum

1. Sprint Planning (Planificación del sprint)
2. Daily Scrum (Reunión rápida diaria)
3. Sprint Review (Revisión del sprint)
4. Sprint Retrospective (Retrospectiva del sprint)
5. Sprint Grooming o Refinement (Refinamiento del sprint)



1. Sprint Planning

Cuando se practica el scrum, la reunión de planificación del sprint se celebra al principio del sprint y, en ella, los equipos identifican lo que se puede ofrecer en el sprint y cómo se va a llevar a cabo el trabajo. Al final de la reunión de planificación, cada miembro del scrum debe tener claro qué se puede entregar en el sprint y cómo se puede entregar el incremento.

- **Asistentes**: Equipo de desarrollo, experto en scrum y propietario del producto.
- **Cuándo**: Al empezar un sprint.
- **Duración**: normalmente, hasta una hora por semana de iteración. Por ejemplo, un sprint de dos semanas se inicia con una reunión de planificación de dos horas.
- **Marco ágil**: scrum.
- **Propósito**: la planificación del sprint prepara a todo el equipo para el éxito durante todo el sprint. Al llegar a la reunión de scrum, el propietario del producto tendrá un backlog del producto priorizado. Hablan de cada elemento con el equipo de desarrollo y el grupo estima colectivamente el esfuerzo que implica. A continuación, el equipo de desarrollo hará una previsión del sprint en la que se indicará cuánto trabajo puede realizar el equipo del backlog del producto. Ese conjunto de trabajos se convierte entonces en el backlog del sprint.

2. Daily Scrum

El Daily Scrum, conocido comúnmente sólo como “La Daily”, es una reunión diaria de 15 minutos en la que participa exclusivamente el Development Team.

En esta reunión todas y cada una de las personas del Development Team responden a las siguientes preguntas:

1. ¿Qué hice ayer para contribuir al Sprint Goal?
2. ¿Qué voy a hacer hoy para contribuir al Sprint Goal?
3. ¿Tengo algún impedimento que me impida entregar?

Mucha gente confunde el Daily Scrum con el Standup Meeting, sin embargo, este último es una práctica de eXtreme Programming (XP) orientada a controlar y gestionar el trabajo motivada por un manager, mientras que el primer término, Daily Scrum, hace referencia a la práctica que permite la inspección y adaptación a través de la auto-organización del equipo.

3ª ceremonia: Sprint Review

El Sprint Review es la reunión que ocurre al final del Sprint, generalmente el último viernes del Sprint, donde el product owner y el Develpment Team presentan a los stakeholders el incremento terminado para su inspección y adaptación correspondientes. En esta reunión organizada por el product owner se estudia cuál es la situación y se actualiza el Product Backlog con las nuevas condiciones que puedan afectar al negocio.

El equipo ha pasado hasta cuatro semanas desarrollando un incremento terminado de software que ahora mostrará a los stakeholders. No se trata de una demostración, sino de una reunión de trabajo. El software ya ha sido mostrado y validado junto con el product owner previamente a esta reunión.

Por un lado, se revisará el incremento terminado. Se mostrará el software funcionando en producción y los stakeholders tendrán la oportunidad de hacer cuantas preguntas estimen oportunas sobre el mismo. El software funcionando ha sido validado previamente por el product owner, que se ha encargado de trabajar con el equipo durante el Sprint para asegurarse que cumple con la Definition of Done y, efectivamente, hace que el Sprint Goal sea válido. Si no existe software funcionando, el Sprint Review carece de sentido, aunque en ciertas ocasiones y oportunidades se sigue manteniendo.

El Development Team tiene que tener un papel importante en esta reunión. Muchas veces no es el product owner quien demuestra el incremento producido, sino que son los propios miembros del Development Team quienes lo hacen. Es una buena práctica no sólo el que lo lleven a cabo, sino también el que lo hagan de forma rotatoria y, tras varios Sprints, hayan participado todos.

El equipo de desarrollo comenta posteriormente qué ha ocurrido durante el Sprint, los impedimentos que se han encontrado, así como soluciones tomadas y actualizan a los stakeholders con la situación del equipo. Por último, el product owner actualiza -con la información de negocio recibida en esta reunión- el Product Backlog para el siguiente Sprint.

En contra de lo que comúnmente se cree, el Sprint Review no se trata de una demo para un cliente o para los stakeholders o incluso para el product owner, ni es tampoco una reunión para felicitar al Equipo de Desarrollo. Es una reunión de trabajo, una de las más importantes porque sirve para marcar la estrategia de negocio. La duración estimada en el estándar para un Sprint Review es de 8 horas para un Sprint de 4 semanas, aunque habitualmente estas reuniones se ejecutan en un entorno de entre 2 y 3 horas.

4ª ceremonia: Sprint Retrospective

La retrospectiva ocurre al final del Sprint, justo después del Sprint Review. En algunos casos y por comodidad de los equipos, se realiza conjuntamente con el Sprint Planning, siendo la retrospectiva la parte inicial de la reunión.

El objetivo de la retrospectiva es hacer de reflexión sobre el último Sprint e identificar posibles mejoras para el próximo. Aunque lo habitual es que el Scrum Master sea el facilitador, es normal que distintos miembros del equipo Scrum vayan rotando el rol de facilitador durante la retrospectiva.

Un formato común es analizar qué ha ido bien durante el Sprint, qué ha fallado y qué se puede mejorar. Este formato se puede facilitar pidiendo a los miembros del equipo Scrum que escriban notas –en post-its- para luego agruparlas y votar aquellos ítems más relevantes, dando la oportunidad a todos de hablar y expresar sus inquietudes.

También se utiliza el formato de retrospectiva basado en cinco fases:

- **Preparar el ambiente**: un pequeño ejercicio para romper el hielo.
- **Recolectar información**: durante esta fase, se utilizan actividades para intentar construir una imagen de lo que ha sido el último Sprint, resultando una imagen conjunta de equipo.
- **Generación de ideas**: el equipo intenta generar ideas para identificar acciones que ayuden a mejorar el rendimiento del equipo durante el siguiente Sprint.
- **Decidir qué hacer**: de las ideas generadas, se proponen acciones que el equipo pueda implementar en el próximo Sprint.
- **Cierre**: Una pequeña actividad de cierre, normalmente unida a una evaluación de la propia retrospectiva, ayuda al equipo a decidir hacia dónde dirigirse en próximas ocasiones. Un recordatorio de la mejora continua.

La duración recomendada por Scrum para un Sprint de 4 semanas es de un máximo de 3 horas, aunque habitualmente se destina entre 1 y 2 horas a este evento.

5ª ceremonia: Sprint Grooming o Refinement

El refinamiento del Product Backlog es una práctica recomendada para asegurar que éste siempre esté preparado. Esta ceremonia sigue un patrón similar al resto y tiene una agenda fija específica en cada Sprint. Se estima su duración en 2 horas máximo por semana del Sprint. Es responsabilidad del product owner agendar, gestionar y dirigir esta reunión.

Los participantes de esta reunión son todo el equipo Scrum, así como cualquier recurso adicional que considere necesario el PO y que pueda contribuir a aclarar el requerimiento. Es necesario, por tanto, que antes de la reunión todos conozcan los requerimientos o historias de usuario que van a ser tratados en la misma y sólo asistan aquellos cuya presencia sea estrictamente relevante.

Workflow de un Sprint – Scrum

Backlogs

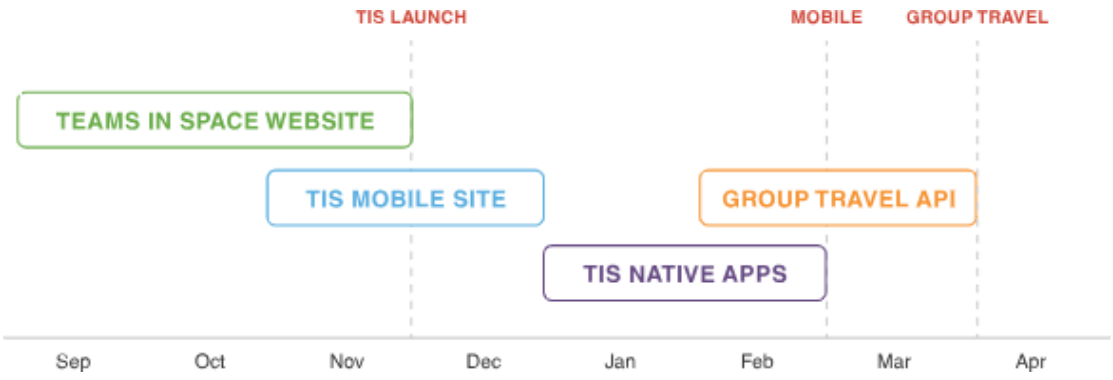
Un backlog ágil bien priorizado no solo simplifica la planificación de iteraciones y publicaciones, sino que también refleja todo a lo que tu equipo dedique tiempo, incluido el trabajo interno que el cliente nunca sabrá. Esto ayuda a establecer las expectativas con las partes interesadas y otros equipos, especialmente cuando te traen trabajo adicional, y consigue que el tiempo de ingeniería sea un activo fijo.

¿Qué es un backlog del producto?

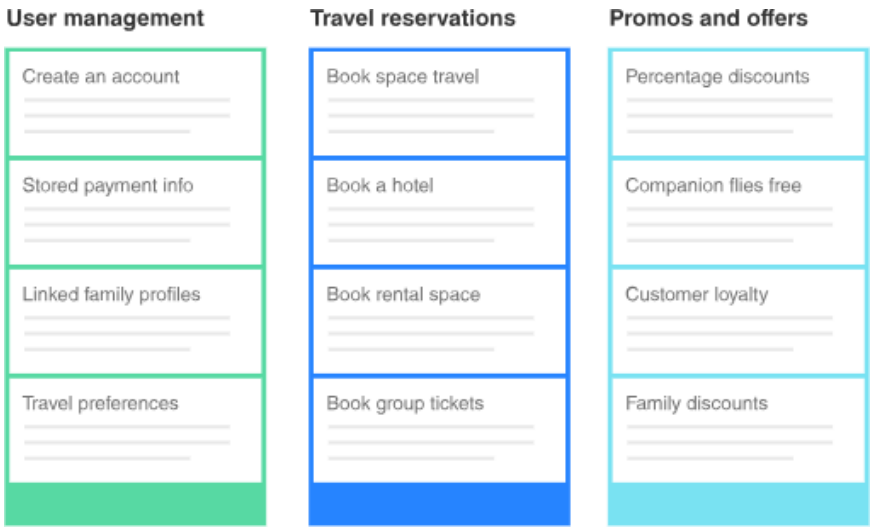
El backlog de un producto es una lista de trabajo ordenado por prioridades para el equipo de desarrollo que se obtiene de la hoja de ruta y sus requisitos. Los elementos más importantes se muestran al principio del backlog del producto para que el equipo sepa qué hay que entregar primero. El equipo de desarrollo no trabaja con el backlog al ritmo que dicta el propietario del producto, y este no presiona al equipo de desarrollo para que saque el trabajo adelante. En su lugar, el equipo de desarrollo saca trabajo del backlog del producto en la medida de sus capacidades, ya sea de forma continua (kanban) o por iteraciones (scrum).

Comienza por las dos erres

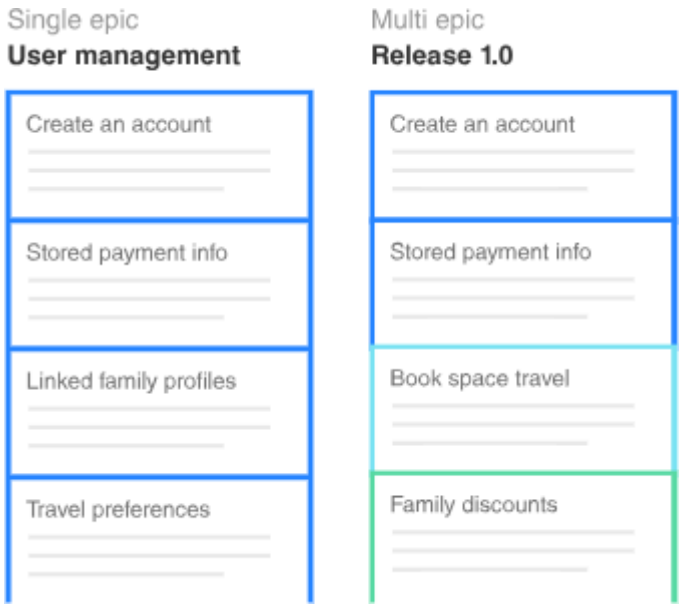
La hoja de ruta y requisitos de un equipo son la base para el backlog del producto. Las iniciativas de hoja de ruta se dividen en varios epics, y cada epic tendrá varios requisitos e historias de usuario. Veamos la hoja de ruta de un producto ficticio llamado Equipos en el Espacio.



Dado que el sitio web de Equipos en el Espacio es la primera iniciativa en la hoja de ruta, dividiremos esa iniciativa en epics (que se muestran aquí en verde, azul y turquesa) y en historias de usuario para cada uno de esos epics.



A continuación, el propietario del producto organiza las historias de usuario en una única lista para el equipo de desarrollo. El propietario del producto puede optar por entregar primero un epic completo (izquierda), p puede que sea más importante para el programa reservar un vuelo barato que requiera historias de diversos epics (derecha). A continuación, puedes ver ambos ejemplos.



¿Qué puede influir en la priorización del propietario del producto?

- Prioridad del cliente
- Urgencia de tener feedback
- Dificultad relativa de la implementación
- Relaciones simbióticas entre elementos de trabajo (por ejemplo, B es más fácil si primero terminamos A)

Si bien el propietario del producto es el encargado de priorizar el backlog, esta tarea no se hace de forma aislada. Los propietarios del producto eficaces buscan los comentarios y las opiniones de los clientes, los diseñadores y el equipo de desarrollo para optimizar la carga de trabajo de todos y la entrega del producto.

Cómo gestionar de forma eficaz el backlog del producto

Una vez creado el backlog del producto, es importante mantenerlo periódicamente para que siga el ritmo del programa. Los propietarios del producto deben revisar el backlog antes de cada reunión para planificar la iteración con el objetivo de asegurar que la priorización es correcta y que se ha incorporado el feedback de la iteración anterior. La revisión periódica del backlog se denomina "limpieza del backlog" en entornos ágiles (hay quien usa también el término "mejora del backlog").

Cuando el backlog aumenta, los propietarios del producto deben agruparlo en elementos a corto y a largo plazo. Los elementos a corto plazo deben estar completamente descritos antes de

etiquetarlos como tales. Esto significa que se han diseñado historias de usuario completas, se ha acordado la colaboración con diseño y desarrollo, y el equipo de desarrollo ha realizado estimaciones. Los elementos a largo plazo pueden seguir siendo algo abstractos, aunque es una buena idea tener una estimación aproximada del equipo de desarrollo para ayudar a su priorización. La palabra clave aquí es "aproximada": las estimaciones cambiarán cuando el equipo entienda completamente esos elementos a largo plazo y empiece a trabajar en ellos.

El backlog funciona como conexión entre el propietario del producto y el equipo de desarrollo. El propietario del producto puede cambiar la prioridad del trabajo en el backlog en cualquier momento debido a los comentarios de los clientes, los ajustes de las estimaciones y los nuevos requisitos. No obstante, una vez que el trabajo se está realizando, los cambios deben ser mínimos, ya que interrumpen el trabajo del equipo de desarrollo y afectan a la concentración, el ritmo y el ánimo.

Antipatrones ante los que estar alerta

- El propietario del producto prioriza el backlog al inicio del proyecto, pero no lo ajusta a medida que llega el feedback de desarrolladores y partes interesadas.
- El equipo limita los elementos en el backlog a aquellos orientados a clientes.
- El backlog se mantiene como un documento almacenado localmente y no es frecuente compartirlo, evitando que las partes interesadas reciban esa información.

Los backlogs del producto ayudan a mantener la metodología ágil de los equipos

Los propietarios del producto con experiencia cuidan rigurosamente el backlog del producto de su programa, convirtiéndolo en un esquema fiable de los elementos de trabajo de un proyecto que se puede compartir.

Las partes interesadas cambiarán las prioridades, y eso está bien. Fomentar las conversaciones sobre qué es importante sincroniza las prioridades de todo el mundo. Estas conversaciones crean una cultura de priorización colectiva que asegura que todos comparten la misma idea del programa.

El backlog del producto también sirve como base para planificar iteraciones. Todos los elementos de trabajo deben incluirse en el backlog: historias de usuario, errores, cambios de diseño, deuda técnica, solicitudes de clientes, elementos de acción de la retrospectiva, etc. Así se garantiza que los elementos de trabajo de todos se incluyan en la conversación general de cada iteración. Los miembros del equipo pueden negociar con el propietario del producto antes de iniciar una iteración con un conocimiento completo de todo lo que debe realizarse.

Kanban

¿Qué es Kanban?

Kanban es una forma de ayudar a los equipos a encontrar un equilibrio entre el trabajo que necesitan hacer y la disponibilidad de cada miembro del equipo. La metodología Kanban se basa en una filosofía centrada en la mejora continua, donde las tareas se "extraen" de una lista de acciones pendientes en un flujo de trabajo constante.

La metodología Kanban se implementa por medio de tableros Kanban. Se trata de un método visual de gestión de proyectos que permite a los equipos visualizar sus flujos de trabajo y la carga de trabajo. En un tablero Kanban, el trabajo se muestra en un proyecto en forma de tablero organizado por columnas. Tradicionalmente, cada columna representa una etapa del trabajo. El tablero Kanban más básico puede presentar columnas como Trabajo pendiente, En progreso y Terminado. Las tareas individuales —representadas por tarjetas visuales en el tablero— avanzan a través de las diferentes columnas hasta que estén finalizadas.

Breve historia de la metodología Kanban

Kanban fue desarrollado por Taiichi Ohno, ingeniero japonés de Toyota, a fines de la década de 1940. Ohno se dio cuenta de que podía mejorar el sistema de producción de Toyota al incorporar elementos de la producción ajustada: en lugar de fabricar productos nuevos en función de la demanda anticipada, el marco Kanban de Ohno produjo y reaprovisionó productos como resultado de la demanda del consumidor; a esto se lo conoce también como el método de producción "justo a tiempo" (just in time o jit). El marco Kanban transformó el proceso de fabricación de Toyota, de un proceso de "empuje" (los productos se introducen en el mercado) a un proceso de "extracción" (los productos se crean según la demanda del mercado). Esto significó que Toyota podía tener un nivel de inventario más bajo sin que afectara su competitividad en el mercado.

El marco de fabricación ajustada que construyó Ohno se basó en tarjetas Kanban. De hecho, "Kanban" es una combinación de dos palabras japonesas: 看 (Kàn), que significa "signo" o "señal visual", y 板 (Bǎn), que significa "tablero". En Toyota, las tarjetas Kanban eran tarjetas de papel que indicaban que se necesitaba un producto nuevo, una pieza nueva o un inventario, y que disparaban el proceso de producción de dicho artículo.

Aunque la metodología Kanban todavía se usa en muchos procesos de fabricación, fue adaptada para el desarrollo de software a principios de la década de 2000. Kanban para el desarrollo de software se inspiró en el método de fabricación ajustada de Ohno y usa el mismo "proceso de extracción".

En el Kanban moderno, los equipos comienzan con una lista de tareas pendientes. El trabajo se "extrae" de las tareas pendientes, según la carga laboral y capacidad de cada miembro del equipo. Luego, los miembros pueden hacer un seguimiento visual del trabajo a medida que avanza a través del ciclo de vida de las tareas, representado por etapas en un tablero Kanban, hasta su finalización. En su configuración actual, Kanban funciona como un método visual de

gestión de proyectos que permite a los equipos encontrar un equilibrio entre la demanda de trabajo y la disponibilidad de los recursos del equipo.

¿Es Kanban lo mismo que Scrum?

Probablemente hayas oído hablar de Kanban junto con Scrum. De hecho, la mayoría de los equipos de trabajo que ejecutan Scrum lo hacen en tableros Kanban. Sin embargo, aun siendo compatible con Kanban, Scrum es un marco diferente. Si bien Kanban se centra en la mejora de procesos, Scrum generalmente se implementa para ayudar a los equipos a finalizar más trabajos y más rápido. Para hacerlo, Scrum organiza "sprints", sesiones de trabajo de dos semanas con reuniones diarias y una cantidad determinada de trabajo a finalizar durante el ciclo de Scrum.

¿Es Kanban una metodología ágil?

Sí, Kanban para el desarrollo de software es una subcategoría de gestión ágil de proyectos. La filosofía ágil fomenta la planificación adaptativa, el desarrollo evolutivo, la entrega temprana y la mejora continua, para ayudar a los equipos a responder de manera flexible al cambio.

Cómo funciona la metodología Kanban

Hoy en día, los tableros Kanban son en su mayoría tableros virtuales con columnas que representan las etapas del trabajo (¡aunque aún se pueden dibujar tableros Kanban en pizarras blancas y dar seguimiento al trabajo con post-it!). En un tablero, una "tarjeta Kanban" representa una tarea, y esta tarjeta de tarea avanza a través de las etapas del trabajo a medida que se finaliza. Los equipos que usan un sistema Kanban tienden a colaborar en un único tablero Kanban, aunque las tareas generalmente se asignan a miembros individuales del equipo.

Los principios Kanban

Existen cuatro principios básicos que te ayudarán a guiar a tu equipo al momento de implementar la metodología Kanban:

Empieza con lo que haces ahora

Puedes implementar el marco Kanban a cualquier proceso o flujo de trabajo actual. A diferencia de algunos procesos de gestión ágil más definidos como Scrum, el marco Kanban es lo suficientemente flexible como para adaptarse a las prácticas centrales de tu equipo de trabajo.

Comprométete a buscar e implementar cambios progresivos y evolutivos

Los grandes cambios pueden ser perjudiciales para tu equipo y, si intentas cambiar todo a la vez, es posible que el nuevo sistema no funcione como esperabas. Ese es el motivo por el cual el marco Kanban está diseñado para fomentar la mejora continua y el cambio progresivo. En lugar de cambiar todo de una vez, empieza por buscar cambios progresivos para lograr que los procesos de tu equipo realmente evolucionen con el tiempo.

Respeta los procesos, los roles y las responsabilidades actuales

A diferencia de otras metodologías Lean, Kanban no tiene roles integrados y puede funcionar con la estructura y los procesos actuales de tu equipo. Además, tu proceso actual podría tener

elementos excelentes, que se perderían si intentaras modificar completamente tu sistema de trabajo de un momento a otro.

Impulsa el liderazgo en todos los niveles

Con el espíritu de mejora continua, el método Kanban reconoce que el cambio puede provenir de cualquier dirección y no solo “de arriba abajo”. Con Kanban, se alienta a los miembros del equipo a participar, proponer nuevas formas para lograr que los procesos evolucionen y emprender nuevas iniciativas de trabajo.

Las 6 prácticas de la metodología Kanban

Los principios básicos de Kanban te ayudan a guiar la mentalidad de tu equipo al momento de abordar un flujo de trabajo Kanban. Para implementar un proceso Kanban, sigue estas seis prácticas que usan las grandes empresas y que te permitirán ayudar a tu equipo a implementar una mentalidad de mejora continua y a lograr un crecimiento progresivo: los principios esenciales del marco Kanban.

Visualizar el trabajo

Una de las principales ventajas de Kanban es que puedes visualizar cómo el trabajo “avanza” a través de las etapas. Una tarjeta Kanban de tarea comenzará su viaje en el lado izquierdo de tu tablero y, a medida que tu equipo trabaja en ella, recorrerá lentamente las siguientes etapas hasta que aterrice en la columna Finalizadas. Esta práctica no solo te brinda una idea general de cómo el trabajo avanza a través de las etapas, sino que también te permite obtener información en tiempo real y apreciar de un vistazo el estado de los proyectos.

Limitar el trabajo en curso

Como metodología ágil, Kanban se centra en un principio de entrega temprana, lo que implica que las tareas deben moverse rápidamente de una columna a otra en lugar de estancarse en un estado ambiguo de “trabajo en progreso” (wip). Aunque no existe un requisito establecido sobre cuántas tareas deben estar “en progreso” en un momento dado, es importante establecer los límites del trabajo; por lo que anima a tu equipo a centrarse en finalizar tareas individuales y a evitar realizar varias tareas a la vez.

Gestionar el flujo de trabajo

La práctica n.° 2 recomienda limitar la cantidad de trabajo en curso, y la mejor manera de hacerlo es con la optimización del flujo de tareas dentro del tablero Kanban. Gestionar y mejorar el flujo de trabajo te permitirá controlar el tiempo predestinado para el trabajo y así poder reducir el tiempo de entrega (el tiempo que pasa entre el inicio de una tarea hasta que llega a la columna Finalizadas de tu tablero Kanban) y garantizar que estás entregando tareas o enviando nuevos productos mientras siguen siendo relevantes.

Implementar políticas de procesos explícitas

Debido a la rapidez con la que se mueven las tareas en Kanban, asegúrate de que tu equipo haya establecido y comunicado claramente las convenciones. Las políticas de tu proceso deben guiar a tu equipo en la implementación de la metodología Kanban. Además, se debe alentar a todos en el equipo a participar e innovar las políticas Kanban, tal como se establece en el cuarto principio básico de Kanban: Impulsar el liderazgo en todos los niveles.

Implementar ciclos de comentarios

En Kanban, necesitas recopilar comentarios de dos grupos distintos: tus clientes y tu equipo.

Recopila comentarios de tus clientes sobre la calidad y eficacia de la solución que produjo el equipo. ¿Fue el producto adecuado? ¿Hubo algún problema? En el caso de que haya surgido algún problema, como errores en un código o cualquier otro defecto del producto, revisa tu flujo Kanban y agrega más tiempo para la revisión, los ajustes y la evaluación.

Realiza consultas frecuentes con el equipo sobre el proceso de ejecución de un marco Kanban.

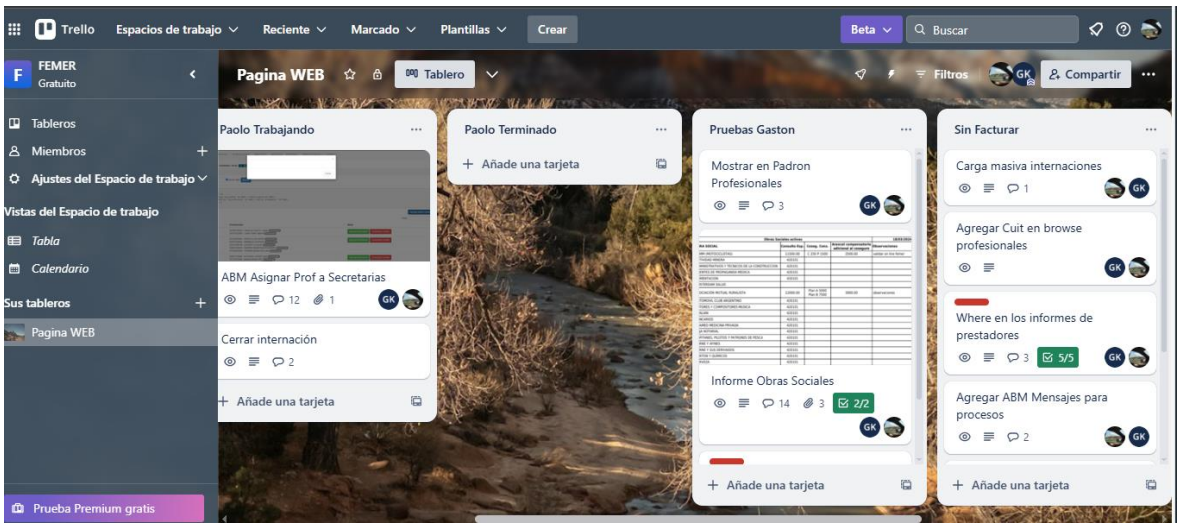
¿Cómo se sienten con los resultados? Aquí tienes otra oportunidad para fomentar el liderazgo en todos los niveles y mejorar las políticas de procesos del equipo.

Mejorar colaborando y evolucionar experimentando

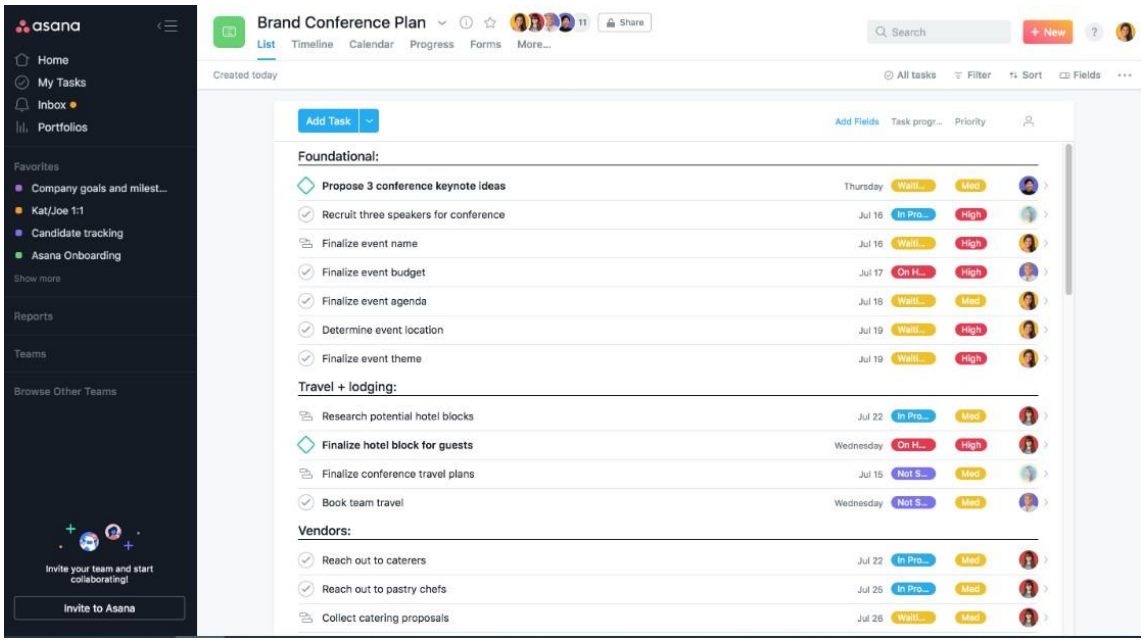
En esencia, Kanban se trata de una mejora continua. Sin embargo, también significa que otros sistemas podrían funcionar bien junto con Kanban. Ya sea Scrum o alguna otra metodología, debes estar siempre dispuesto a colaborar, experimentar y desarrollar tus procesos si es necesario.

Tableros Kanban

Trello: https://trello.com/



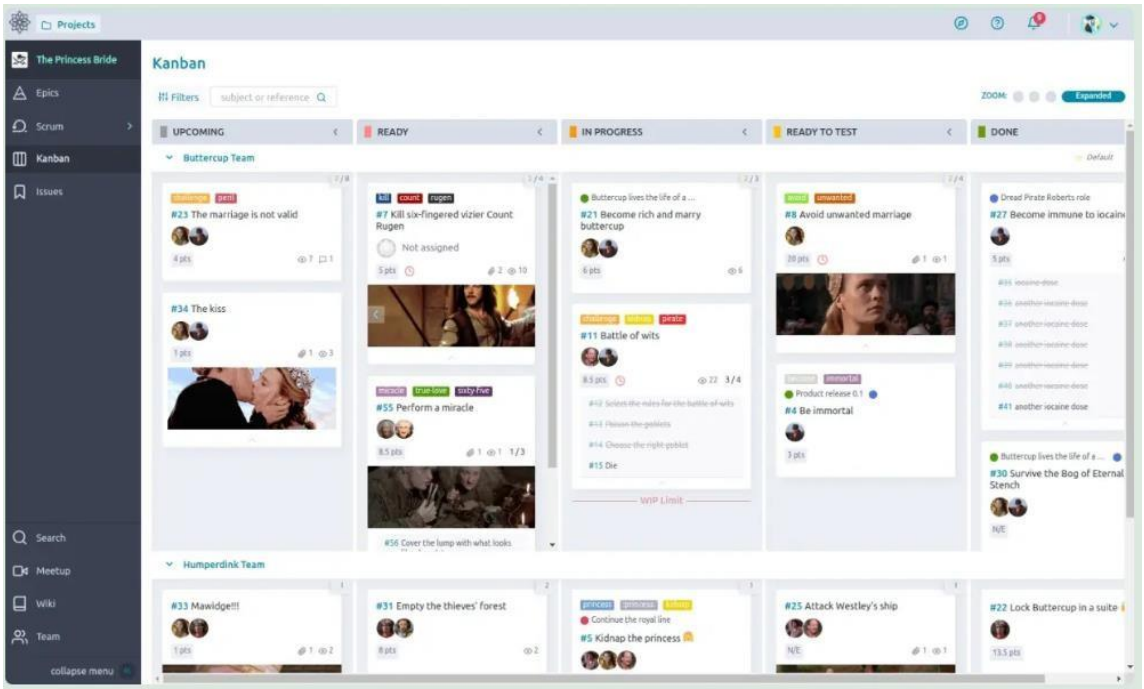
Asana: https://asana.com/



Monday: https://www.monday.com/

Taiga

https://taiga.io/



Programación XP

La programación extrema (XP) es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos. Esta metodología se basa en

5 valores, 5 reglas y 12 prácticas de programación. Si bien tiene una estructura rígida, el resultado de estos sprints altamente centrados y las integraciones continuas buscan dar como resultado un producto de mayor calidad.

¿Qué es la programación extrema (XP)?

Al igual que otras metodologías ágiles, la programación extrema es un método de desarrollo de software dividido en sprints de trabajo. Los marcos ágiles siguen un proceso iterativo, en el que se completa y revisa el marco al final de cada sprint, refinándolo para adaptarlo a los requisitos cambiantes y alcanzar la eficiencia máxima. Al igual que otros métodos ágiles, el diseño de la programación extrema permite a los desarrolladores responder a las solicitudes de los clientes, adaptarse y realizar cambios en tiempo real. Sin embargo, la programación extrema es mucho más disciplinada; realiza revisiones de código frecuentes y pruebas unitarias para realizar cambios rápidamente. Además, es muy creativa y colaborativa, ya que promueve el trabajo en equipo durante todas las etapas de desarrollo.

¿Quién desarrolló la programación extrema?

Los orígenes de XP se remontan a fines de la década de 1990, cuando Kent Beck la creó para gestionar el desarrollo de un sistema de software de nómina para Chrysler llamado Proyecto C3. El objetivo al implementar la programación extrema era (y sigue siendo) eliminar la resistencia a cambiar el código en un proyecto de desarrollo. En los métodos de desarrollo de software más tradicionales, es muy común que el código no se cambie una vez que está escrito (excepto para la depuración). Con la programación extrema, en cambio, el código se examina con tanto detalle que los desarrolladores pueden decidir modificarlo por completo luego de una sola iteración.

¿Cuándo deberías implementar la programación extrema?

Como la programación extrema se centra en el desarrollo de software, suele ser implementada solamente por los equipos de ingeniería. Incluso los equipos de software, suelen usarla únicamente para determinadas configuraciones. Para obtener el máximo beneficio de la programación extrema, recomendamos usarla en los siguientes casos:

Para gestionar un equipo más pequeño. Debido a su naturaleza altamente colaborativa, la programación extrema funciona mejor en equipos pequeños de menos de diez personas.

Si estás constantemente en contacto con tus clientes. La programación extrema incorpora los requisitos de los clientes a lo largo del proceso de desarrollo y también se basa en ellos para las pruebas y aprobaciones.

Si trabajas con un equipo flexible que pueda aceptar el cambio (sin resentimientos). Dada su propia naturaleza, la programación extrema a menudo requerirá que todo el equipo deseche todo su arduo trabajo. Algunas reglas también permiten que algunos miembros del equipo

realicen cambios en cualquier momento, lo que supondría un problema si los demás compañeros del equipo se lo toman como algo personal.

Si dominas los aspectos técnicos de la codificación. La programación extrema no es para principiantes ya que necesitas poder trabajar e implementar cambios rápidamente.

Ciclo de vida de la programación extrema (XP)

El ciclo de vida de XP fomenta la integración continua, ya que requiere que los miembros del equipo trabajen casi constantemente, cada hora o todos los días. Sin embargo, el ciclo de vida completo se estructura de la siguiente manera:

- Extraer trabajos sin finalizar de las historias de usuarios
- Priorizar los elementos más importantes
- Comenzar con la planificación iterativa
- Incorporar un plan realista
- Mantener una comunicación constante con todas las partes interesadas y empoderar al equipo
- Presentar el trabajo
- Recibir comentarios
- Regresar a la etapa de planificación iterativa y repetir si es necesario

Los 5 valores de la programación extrema (XP)

La programación extrema está impulsada por el valor. En lugar de usar motivadores externos, la programación extrema permite que tu equipo trabaje de una manera más sencilla (priorizando la simplicidad y la colaboración sobre diseños complejos), basándose siempre en estos cinco valores.

1. **Simplicidad**

Antes de empezar cualquier trabajo de programación extrema, debes hacerte la siguiente pregunta: ¿Cuál es el proceso más simple y que también funciona? El concepto “que también funciona” es un diferenciador clave ya que lo más simple no siempre es práctico o efectivo. En la programación extrema, tu atención se centra en realizar primero el trabajo más importante. Esto significa que debes buscar un proyecto simple que sabes que puedes lograr.

2. **Comunicación**

XP se basa en una respuesta rápida y una **comunicación efectiva**. Para trabajar de manera efectiva, el equipo debe ser abierto y honesto entre sí. Cuando surgen problemas, se espera que todos aporten sus comentarios e ideas, ya que probablemente alguno de ellos ya tenga una solución adecuada. Y si no la tiene, podrán resolver el problema más rápidamente como grupo de lo que podrían hacerlo solos.

3. **Comentarios**

Al igual que otras metodologías ágiles, el método XP incorpora comentarios e **historias de usuarios** directamente en el proceso. El enfoque de XP es producir trabajo de forma rápida y sencilla, para luego compartir los resultados para obtener comentarios de forma casi inmediata.

Por eso, los desarrolladores están en contacto casi constante con los clientes durante todo el proceso. En la programación extrema, se lanzan varias versiones con regularidad para obtener información nueva desde el primer momento y con frecuencia. Podrás adaptar el proceso (en lugar del proyecto) para incorporar los comentarios recibidos de clientes. Por ejemplo, si los comentarios ayudan a reducir el tiempo de retraso innecesario, podrás ajustar el proceso para que los desarrolladores trabajen para mejorar el tiempo de retraso en lugar de ajustar el proyecto completo.

4. **Valentía**

Para implementar la programación extrema, se requiere de mucha valentía. Siempre se espera que seas honesto al brindar actualizaciones al equipo sobre tu progreso, lo que puede dejarte en una posición de vulnerabilidad. Si no cumples con una fecha de entrega en la programación extrema, es probable que al líder de tu equipo no le interese analizar los motivos. En cambio, le dirías que no cumpliste con la fecha de entrega, te responsabilizarías por ello y te pondrías a trabajar nuevamente.

Si eres líder de un equipo, tu responsabilidad al comienzo del proceso de XP será establecer las expectativas de éxito y definir lo que será "el trabajo terminado". A menudo suele haber poca planificación para lidiar con el fracaso ya que el equipo se centra principalmente en el éxito. Sin embargo, puede ser un poco alarmante ya que sabemos que las cosas no siempre suceden como se planifican. Pero si surgen cambios durante el proceso de XP, se espera que el equipo se adapte y acompañe estos cambios.

5. **Respeto**

Teniendo en cuenta que las comunicaciones y la honestidad son prioridad en el método XP, tiene sentido que el respeto sea una virtud esencial. Para que los equipos se comuniquen y colaboren de manera efectiva, deben aprender a estar en desacuerdo. Pero hay maneras de hacerlo amablemente. El respeto es una base importante que promueve la bondad y la confianza, incluso cuando se expresan las opiniones con total honestidad. Para la programación extrema, estas son las expectativas:

- Respeto mutuo entre los clientes y el equipo de desarrollo.
- Respeto mutuo entre los miembros del equipo.
- El reconocimiento de que todos en el equipo aportan algo valioso al proyecto.

Reglas de la metodología de programación extrema (XP)

Los valores de la programación extrema son los aspectos más filosóficos. Las reglas, por otro lado, son los usos prácticos de cómo se realiza el trabajo. Necesitarás contar con ambos aspectos para gestionar un equipo XP efectivo.

1. **Planificación**

Durante las **etapas de planificación** de la programación extrema, deberás determinar si el proyecto es viable y si se adapta al método XP. Para hacer esto, deberás evaluar:

- Las historias de usuarios para confirmar que coinciden con el valor de simplicidad y garantizar

que el cliente esté disponible para participar del proceso. Si la historia del usuario es más compleja o fue creada por un cliente anónimo, es probable que no sea adecuada para el método XP.

- El valor comercial y el nivel de prioridad del proyecto para asegurarte de que esté alineado con la idea de "realizar el trabajo más importante primero".
- La etapa de desarrollo en la que te encuentras. La programación extrema se adapta mejor al desarrollo en sus etapas iniciales y no será un método tan efectivo para iteraciones posteriores.

Una vez que hayas confirmado que el proyecto es viable para implementar la programación extrema, te recomendamos crear un cronograma de lanzamiento. Sin embargo, es importante tener en cuenta que debes lanzarlo tan pronto sea posible y con frecuencia para poder obtener comentarios. Para hacer esto:

- Divide el proyecto en iteraciones y crea un plan para cada una.
- Establece plazos realistas y un ritmo de trabajo sostenible.
- Comparte actualizaciones a medida que ocurren para permitir que el equipo sea lo más honesto y transparente posible.
- Comparte actualizaciones en tiempo real para ayudar al equipo a identificar, adaptar y realizar cambios más rápidamente.
- Usa una **herramienta de gestión de proyectos** para crear un **tablero Kanban** o un cronograma para poder dar seguimiento al progreso en tiempo real.

2. Gestión

Uno de los elementos clave de la programación extrema es el espacio de trabajo. Los puristas de XP recomiendan usar un espacio de trabajo abierto donde todos los miembros del equipo puedan trabajar juntos. Debido a que la programación extrema es altamente colaborativa, es una ventaja contar con un espacio donde puedan reunirse todos los colaboradores físicamente. Sin embargo, en estas épocas, esta no siempre es una solución práctica o factible. Si trabajas en un equipo remoto, considera usar una **plataforma** que fomente el **trabajo asincrónico** para la **colaboración remota**. De esta manera, todos los miembros podrán trabajar juntos en el proyecto, incluso si no se encuentran en el mismo espacio físico.

Al igual que con otros métodos ágiles, puedes organizar **reuniones diarias de actualización** para verificar el estado del trabajo y fomentar las comunicaciones abiertas y constantes. Te recomendamos implementar un ciclo semanal y un ciclo trimestral. Durante el ciclo trimestral, tu equipo y tú revisarán las historias que guiarán el trabajo a realizar. También analizarán los procesos de XP, para identificar brechas u oportunidades para realizar cambios. Luego, trabajarás en ciclos semanales, que comenzarán con una reunión con el cliente. El cliente elegirá la historia de usuario para que los programadores trabajen esa semana.

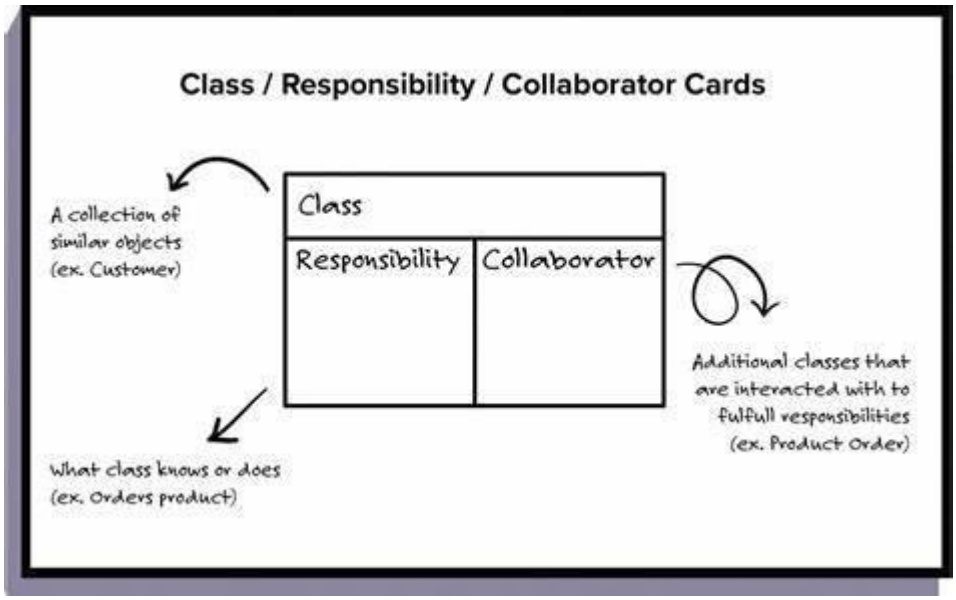
Como gerente o líder de equipo, tu atención deberá centrarse en mantener el progreso constante del trabajo, supervisar el ritmo, gestionar la carga de trabajo del equipo para que puedan trabajar en los errores o problemas a medida que surjan, o cambiar el proceso de XP para que se ajuste a tu proyecto e iteración actuales. Recuerda que el objetivo de la programación extrema es ser flexible y ágil, por lo que tu trabajo estará muy centrado en el trabajo actual del equipo y en poder responder rápidamente a cualquier cambio.

3. Diseño

Si recién incursionas en la programación extrema, te recomemos empezar con el diseño más simple posible, ya que las iteraciones posteriores incrementarán su complejidad. Intenta no agregar ninguna funcionalidad temprana durante esta etapa para mantener el proceso lo más sencillo posible.

Los equipos de metodología XP a menudo usan tarjetas de clase, responsabilidad y colaboración (CRC, por sus siglas en inglés) para mostrar cómo interactúa cada elemento en el diseño global. Al completar cada campo de la tarjeta, podrás obtener una imagen visual de cómo se relacionan e interactúan todas las funciones. Las tarjetas CRC incluyen:

- Clase (conjunto de elementos similares)
- Responsabilidades (según la clase)
- Colaboradores (clase que interactúa con el colaborador en cuestión)



Tarjetas CRC

Las tarjetas CRC son útiles para simular el proceso y detectar problemas potenciales. Independientemente del tipo de diseño, te recomendamos usar un sistema que permita reducir los cuellos de botella potenciales. Para esto, asegúrate de prevenir los riesgos de manera proactiva. Tan pronto como descubras una amenaza potencial, asigna uno o dos miembros del equipo para encontrar una solución en caso de que suceda.

4. Codificación

Uno de los aspectos clave de la programación extrema es el contacto permanente que mantendrás con el cliente durante todo el proceso de codificación. Esta interacción te permite probar e incorporar comentarios dentro de cada iteración, en lugar de esperar hasta el final de cada sprint. Sin embargo, las reglas de codificación son bastante estrictas en el método XP. Algunas incluyen:

- Todo el código debe cumplir con el estándar de programación.
- Se deben realizar pruebas unitarias para definir los requisitos y desarrollar todos los aspectos del proyecto.
- La programación se realiza en parejas: dos desarrolladores trabajan juntos y simultáneamente en la misma computadora. El tiempo de programación sigue siendo el mismo; sin embargo, se duplica el enfoque para lograr resultados de la más alta calidad posible.
- Se deben usar integraciones continuas para agregar código nuevo y probarlo de inmediato.
- Solo un par de desarrolladores puede actualizar el código en un momento dado para reducir los errores.
- Propiedad colectiva del código: cualquier miembro del equipo puede cambiar el código en cualquier momento.

5. Prueba

Deberás realizar pruebas durante todo el proceso de programación extrema. Todo el código deberá someterse a pruebas unitarias antes de su lanzamiento. Si detectas errores durante estas pruebas, necesitarás crear pruebas adicionales para corregirlos. Más adelante, incorporarás la misma **historia de usuario** en la que has estado trabajando en una prueba de aceptación. Durante esta prueba, el cliente examinará los resultados para comprobar que has implementado correctamente la historia de usuario al producto.

¿Cuáles son las 12 prácticas de la programación extrema?

Para perfeccionar aún más el proceso, la programación extrema también implementa un conjunto de 12 prácticas a lo largo del proceso. Se basan en el **Manifiesto ágil**, pero se adaptan a las necesidades de la programación extrema.

- **El juego de planificación**: La planificación XP se usa para guiar el trabajo. Los resultados de la planificación deben ser los objetivos que pretendes alcanzar, los plazos previstos y los pasos a seguir.
- **Pruebas de clientes**: Cuando finalices una función nueva, el cliente desarrollará una prueba de aceptación para determinar si has cumplido con la historia de usuario original.
- **Pequeñas entregas**: La programación extrema realiza entregas pequeñas y periódicas para obtener información importante durante todo el proceso. A menudo, las entregas se envían directamente a los clientes, aunque también pueden enviarse internamente.
- **Diseño simple**: El sistema XP está diseñado para ser simple, producirá solo lo necesario y nada más.
- **Programación en parejas**: Toda la programación la realizan simultáneamente dos desarrolladores que se sientan físicamente uno al lado del otro. No hay trabajo individual en la programación extrema.
- **Desarrollo guiado por pruebas (TDD):** Debido a que la programación extrema se basa en los comentarios, se requieren pruebas exhaustivas. A través de ciclos cortos, los programadores realizan pruebas automatizadas para luego reaccionar de inmediato.
- **Refactorización**: Aquí es donde se deberá prestar especial atención a los detalles más finos del código base, para eliminar los duplicados y asegurarse de que el código sea coherente. De esta manera obtendrás diseños simples y de alta calidad.
- **Propiedad colectiva**: Cualquier par de desarrolladores puede modificar el código en cualquier

momento, independientemente de que lo hayan desarrollado o no. En la programación extrema, la codificación se realiza en equipo, y el trabajo de todos se lleva a cabo según los estándares colectivos más altos.

- **Integración continua**: Los equipos de XP no esperan a que se completen las iteraciones, sino que se integran constantemente. A menudo, un equipo de XP se integrará varias veces al día.
- **Ritmo de trabajo sostenible**: La intensidad de los trabajos de XP requiere que se establezca un ritmo de trabajo sostenible. Los equipos deben determinar cuánto trabajo pueden producir a este ritmo por día y por semana, y usarlo para establecer plazos de trabajo.
- **Metáfora**: La metáfora es, literalmente, una metáfora. Se decide en equipo y se usa un lenguaje para expresar cómo debe funcionar el equipo. Por ejemplo, somos hormigas trabajando en colectivo para construir el hormiguero.
- **Estándares de codificación**: Los equipos de XP siguen un estándar. De la misma manera que un grupo de escritores necesita adoptar el tono de una marca para que parezca que siempre está escribiendo una misma persona, los desarrolladores de XP deben codificar de la misma manera unificada para que parezca que el código esté escrito por un solo desarrollador.

Programación extrema vs. Scrum

Scrum es otro tipo común de metodología ágil gestionada por un Scrum master. Al igual que la programación extrema, organiza sprints basados en historias de usuarios para desarrollar funciones nuevas de productos o de software. Sin embargo, el método XP es mucho más rígido que el método Scrum, ya que tiene reglas y pautas estrictas que promueven intercambios constantes entre desarrolladores y clientes. Además, puedes aplicar Scrum para cualquier proceso que requiera iteración y aportes del cliente, mientras que solo usarías la programación extrema para la programación.

Criterio	Programación Extrema (XP)	Scrum
Enfoque principal	Mejora de la calidad del código y respuesta rápida al cambio.	Gestión del proyecto enfocada en entregas incrementales.
Tamaño del equipo recomendado	Pequeños equipos (hasta 10 personas).	Equipos pequeños a medianos (5-9 personas).
Duración de iteraciones	Iteraciones cortas de 1-2 semanas.	Sprints típicos de 2-4 semanas.
Rol del cliente	Cliente participa activamente en el equipo, incluso diariamente.	Product Owner representa al cliente, no siempre está presente a diario.
Prácticas destacadas	Programación en parejas, desarrollo guiado por pruebas (TDD), integración continua.	Daily Scrum, Sprint Planning, Review y Retrospective.
Documentación	Mínima, se prioriza la comunicación directa y el código bien escrito.	También ligera, pero algo más estructurada que XP.
Planificación	Planificación continúa basada en historias de usuario.	Planificación por sprint con backlog priorizado.
Adaptación a cambios	Muy alta; el cambio se considera parte natural del desarrollo.	Alta; se reevalúa el backlog en cada sprint.
Control del proceso	Basado en pruebas automáticas y feedback constante.	Basado en revisiones de sprint y métricas de progreso (burndown).
Enfoque técnico	Muy técnico, centrado en prácticas de codificación.	Más centrado en la gestión del equipo y el proyecto.