4. Product Owner: Gestión ágil de productos

La responsabilidad del Product Owner es crucial en Scrum, fungiendo como el principal enlace entre las partes interesadas y el equipo Scrum, con la meta de maximizar el valor generado a través del desarrollo del producto. Un Product Owner posee la autoridad y el conocimiento profundo necesarios para articular y defender las necesidades y expectativas de los stakeholders con precisión. En este capítulo, exploraremos las responsabilidades, habilidades y actividades de un Product Owner. Esto brindará una comprensión ampliada que complementa lo ya discutido en el modelo Core de Scrum, orientándote en cómo un Product Owner puede liderar efectivamente la visión del producto a través de colaboración efectiva, liderazgo estratégico y toma de decisiones basada en datos. Puntos clave de este capitulo: - Comprender la importancia estratégica de la responsabilidad del Product Owner en el desarrollo de productos ágiles. - Identificar las habilidades cruciales que un Product Owner necesita cultivar para liderar con eficacia. - Profundizar en las actividades específicas y diarias que definen la labor del Product Owner. - Analizar cómo un Product Owner puede funcionar como un puente eficaz entre el equipo Scrum y las partes interesadas, garantizando una entrega de valor maximizada.

4.1 - Gestión ágil de producto

La gestión de productos, o "Product Management" en inglés, es una disciplina que se centra en guiar el desarrollo, la estrategia y la entrega de productos.

La gestión de producto se pueden manejar con enfoque tradicional o con enfoque ágil: El Agile Product Management (Gestión ágil de producto) está diseñado para entornos y mercados que cambian rápidamente, donde la adaptabilidad y la respuesta rápida son esenciales. Mientras que el Product Management tradicional puede ser más adecuado para industrias o productos donde los requisitos son menos volátiles y el desarrollo puede planificarse con bastante antelación.

> Muchas empresas modernas están adoptando un enfoque ágil, al reconocer los beneficios de la adaptabilidad y la iteración rápida.

La gestión ágil de producto permite establecer estrategias para crear productos en un entorno ágil, manteniendo un enfoque adaptable para que la organización y el equipo propiamente puedan responder rápidamente a la retroalimentación de los clientes y crear mejores productos.

A diferencia de las técnicas de desarrollo tradicionales, la gestión ágil se centra en la adaptabilidad, permitiendo que el equipo responda rápidamente a la retroalimentación de los clientes.

Tip para el Product Owner
Valora más una entrega temprana y frecuente que esperar hasta que el producto esté "perfecto".

Beneficios de la gestión ágil de producto

La gestión ágil proporciona un enfoque más flexible en comparación con la planificación y el desarrollo tradicional, lo que permite que se construyan productos en incrementos cortos.

Algunos beneficios clave de la gestión ágil de productos:

- Ajustar continuamente la hoja de ruta del producto (abordado más adelante).
- Responder rápidamente a los requisitos nuevos y cambiantes, y entregar valor de manera incremental.
- Colaborar para entregar el trabajo rápidamente y con calidad.

- Aprender de los clientes y mejorar el producto.

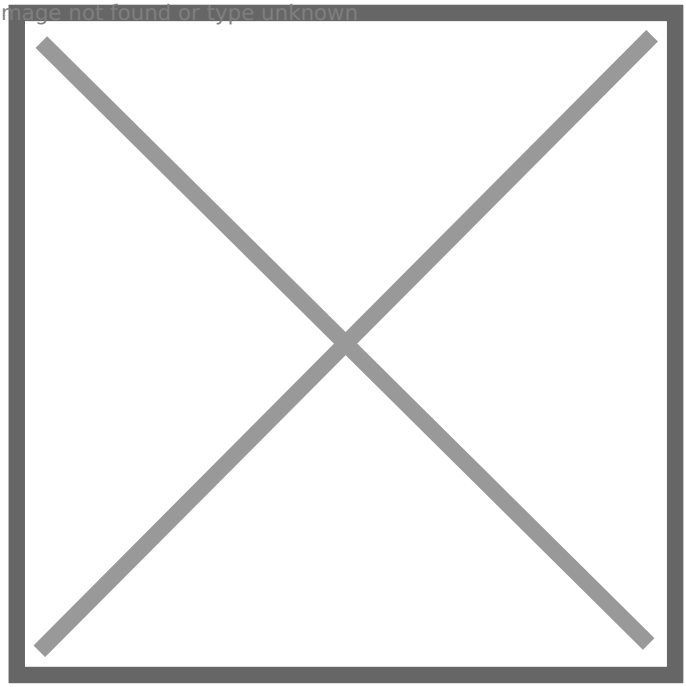
Tip para el Scrum Master

Anima al equipo a adoptar una mentalidad de mejora y aprendizaje constante, utilizando cada sprint y retroalimentación como una oportunidad de crecimiento.

Ecosistema de la gestión ágil de producto

Cada producto requiere un enfoque único y personalizado, tiene sus propios objetivos y retos, lo que se traduce en una gestión de productos basada en lograr la conexión entre las **expectativas de negocio**, una gran **experiencia de usuario** y el aprovechamiento de la **tecnología.**

Image not found or type unknown



- **UX (Experiencia de Usuario):**
  - Investigación centrada en el usuario a través de entrevistas, encuestas y pruebas de usabilidad, se recolecta feedback directo de los usuarios, lo que permite iterar y mejorar el producto continuamente.
  - Los prototipos y entregables se desarrollan, prueban y refinan de forma cíclica para garantizar una experiencia óptima para el usuario.
  - Los diseñadores UX trabajan en estrecha colaboración con los equipos de desarrollo para asegurar que la visión del producto y las necesidades del usuario se integren adecuadamente.
- **Expectativas de Negocio:**
  - En el Agile Product Management, los objetivos de negocio se definen con claridad y se comunican al equipo. Estos objetivos guían las decisiones y prioridades del equipo.
  - Se establecen indicadores clave de rendimiento para medir el éxito del producto en términos de negocio.

    - A diferencia de las estrategias rígidas tradicionales, en un entorno ágil, las estrategias y objetivos pueden adaptarse en función de la evolución del mercado y del feedback del cliente.
- **Tecnología:**
    - Se aplican técnicas como Integración Continua y Entrega Continua (CI/CD) para asegurar que las características y mejoras del producto se desarrollen, prueben y desplieguen rápidamente y de manera eficiente.
    - Se usan herramientas colaborativas (como JIRA o Trello) que permiten gestionar y priorizar las tareas y seguir el progreso en tiempo real.
    - La tecnología implementada debe ser adaptable y escalable, permitiendo que el producto crezca y se modifique según las necesidades cambiantes.

> El diseño y la investigación de la **Experiencia de Usuario** aseguran que el producto esté alineado con las necesidades y expectativas del usuario y del negocio. Las **Expectativas de Negocio** definen la dirección y los objetivos, proporcionando un marco dentro del cual el diseño y la tecnología deben operar. La **Tecnología** es el medio que permite que las expectativas respecto a la Experiencia de Usuario y del negocio se materialicen, proporcionando soluciones que cumplan con las expectativas del usuario y los objetivos de negocio.

Tip para el equipo
En escenarios de **desarrollo de software** es importante implementar técnicas de Integración Continua y Entrega Continua **(CI/CD)** para acelerar el proceso de desarrollo y despliegue.

La gestión ágil de producto en una organización

El enfoque jerárquico en comparación con el enfoque de la gestión ágil de producto abarca diferencias sustanciales en cómo se organizan, operan y toman decisiones las organizaciones:

	Enfoque Jerárquico	**Enfoque de gestión ágil de producto**
Toma de decisiones	Las decisiones vienen desde los niveles superiores de la organización y se transmiten hacia abajo. Demora en la toma de decisiones por necesidad de aprobación en múltiples niveles.	Decisiones descentralizadas tomadas por Product Owner en colaboración con su equipo y stakeholders. Se promueve la autonomía y la toma de decisiones basadas en datos.
Estructura	Departamentos o divisiones con roles claramente definidos.	Equipos multifuncionales centrados en productos o características específicas.
Comunicación	Comunicación vertical (de arriba hacia abajo y viceversa). Silos de información y desafíos en la comunicación.	Comunicación abierta y transparente entre todos los niveles y departamentos.

	Enfoque Jerárquico	Enfoque de gestión ágil de producto
Respuesta al cambio	Los cambios suelen ser más lentos y requieren aprobación en varios niveles.	Los cambios pueden implementarse rápidamente en función de la retroalimentación del usuario y las métricas del producto.
Cultura	Centrada en procesos, estructuras y cumplimiento de tareas asignadas.	Centrada en el valor para el cliente y negocio, y la entrega continua de características útiles.

Principios de la gestión ágil de producto

- **Valor sobre Funciones:** En lugar de centrarse simplemente en la cantidad de características, se enfoca en entregar el máximo valor al usuario.
- **Iteración sobre Perfección:** Liberar versiones tempranas y obtener feedback es más valioso que esperar a tener el producto "perfecto".
- **Colaboración sobre Silos:** Equipos multidisciplinarios trabajando juntos producen mejores resultados que departamentos trabajando en aislamiento.

Riesgos de no adoptar la gestión ágil de producto

- **Desarrollo Lento:** Sin un enfoque ágil, los equipos pueden quedar atrapados en largos ciclos de desarrollo sin entregas reales.
- **Desalineación con el Mercado:** Sin feedback constante, el producto puede desviarse de las necesidades reales del mercado.
- **Gastos Ineficientes:** Los recursos pueden desperdiciarse en características que no son valiosas para los usuarios.

¿Qué debo evitar en el Agile Product Management (Gestión ágil de producto)?

El Product Management es un área crucial que determina en gran medida el éxito de un producto. Sin embargo, hay ciertas trampas en las que como Product Owner puedes caer y que se deben evitar:

- **Convertirse en el 'Camarero de las funcionalidades':** Es un error común que los Product Owners se transformen en meros **intermediarios,** recogiendo y sirviendo todas las peticiones de características sin cuestionarlas. Esto puede conducir a un **ciclo interminable de desarrollo** que no resuelve los problemas reales ni cumple con los objetivos estratégicos.
- **Evitar la retroalimentación:** Ignorar el feedback de los usuarios es un **grave error.** No es obligatorio actuar sobre cada comentario, pero sí es vital escuchar, analizar y considerar la retroalimentación para mejorar el producto continuamente.
- **Visión y estrategia:** Mantén clara y presente tu visión y estrategia. **NO permitas que tus clientes decidan las soluciones**; ellos identifican problemas, pero tú (como Product

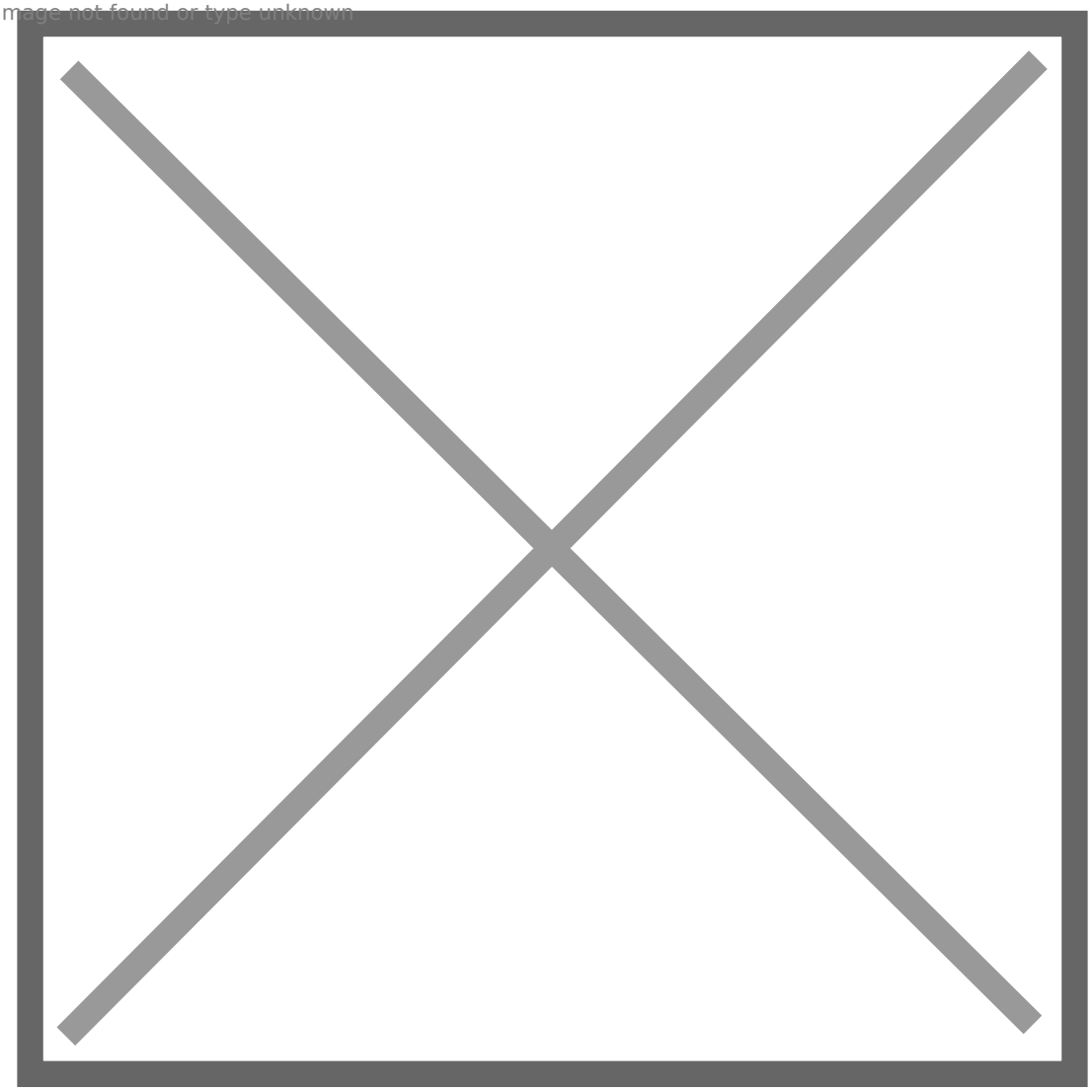
Owner) debes diseñar las soluciones.

- **Descuidar la comunicación:** Un Product Owner debe ser el puente entre diferentes equipos. La falta de comunicación clara y abierta puede llevar a malentendidos y retrasos.
- **No Establecer métricas claras:** Si no sabes qué estás midiendo, no sabrás si estás teniendo éxito. Establece KPIs claros desde el principio y monitorea el avance del equipo con respecto a lo que se espera del producto.
- **Experimenta antes de implementar:** Experimenta y valida tus ideas antes de implementarlas. Sé selectivo y prioriza las funcionalidades que realmente aportan valor.
- **No aprender del fracaso:** No todas las características o productos serán un éxito. En lugar de verlo como un fracaso, utilízalo como una oportunidad de aprendizaje.
- **Tratar de ser todo para todos: NO** puedes satisfacer todas las necesidades de todos los usuarios. Define el público objetivo y enfoca tus esfuerzos en resolver sus problemas y satisfacer sus necesidades.
- **No respetar los plazos:** Aunque la flexibilidad es una ventaja en la gestión ágil de producto, es esencial establecer y respetar plazos realistas.
- **Ignorar el mercado:** Mantente informado sobre las tendencias del mercado y la posición de tus competidores. Un producto, por más bueno que sea, si no cumple con una necesidad del mercado o no se diferencia de la competencia, puede no tener éxito.

Actores involucrados en la gestión ágil de producto

En la gestión ágil de producto existen actores complementarios al equipo Scrum, que son necesarios y usualmente definidos en las organizaciones para cubrir todas las necesidades y requerimientos de las partes interesadas:

Image not found or type unknown



Equipo Scrum:

En su concepción más básica, Scrum se centra en cómo un equipo puede trabajar de manera eficiente para entregar productos de alta calidad en ciclos iterativos y rápidos. Para hacerlo, Scrum establece tres responsabilidades: Product Owner, Developers y Scrum Master. Cada uno contribuye al flujo ágil de trabajo.

Sin embargo, cuando se trabaja con Scrum en un entorno empresarial real, con múltiples productos, interdependencias y expectativas de diversas partes interesadas, rápidamente nos damos cuenta que existen muchas otras consideraciones que no son cubiertas por el equipo Scrum.

Actores organizacionales o del cliente:

El equipo Scrum no trabaja totalmente aislado: existen roles adicionales, a menudo anclados en la estructura organizacional o en las necesidades del cliente, que desempeñan funciones críticas para asegurar que los esfuerzos del equipo Scrum se mantengan alineados con los objetivos del negocio y las expectativas del cliente.

Estos roles organizacionales o del cliente actúan como **puentes,** facilitando la comunicación, alineación y toma de decisiones entre el equipo Scrum y el ecosistema empresarial más amplio en el que operan. Entre los roles adicionales comúnmente podemos encontrar:

- *Business Manager (Gestor de negocio):* Se enfoca en la viabilidad comercial del producto, asegurándose de que el producto no solo sea funcional sino también rentable y alineado con los objetivos estratégicos de la organización.
- *Line Manager (Gestor de líneas de negocio):* Supervisa segmentos específicos o líneas de negocio y en conjunto con el Product Owner garantiza que el producto cumpla con las necesidades y objetivos de esa línea en particular.
- *Program Manager (Gestor de programas):* Coordina múltiples proyectos o equipos ágiles para garantizar que trabajen hacia objetivos comunes y que se gestionen las dependencias entre ellos.
- *APMO (Oficina de proyectos ágiles):* Se encarga de establecer, promover y asegurar las mejores prácticas ágiles a lo largo de la organización. Funciona como un centro de excelencia para las metodologías ágiles.
- *UX/UI Designers:* Aunque no siempre se mencionan en la estructura de un equipo Scrum, desempeñan un papel crucial en la gestión ágil del producto. Se encargan de diseñar una interfaz que sea agradable y funcional para el usuario.
- *Analistas de negocio:* Trabajan en estrecha colaboración con el Product Owner y el equipo Scrum para definir requisitos, entender las necesidades del negocio y traducirlas en características del producto.

□□ Tip para el Product Owner

Asegúrate de mantener una comunicación fluida con los actores externos al equipo (Business Managers, Line Managers, Analistas de Negocio) para garantizar que la visión del producto esté alineada con los objetivos de negocio.

4.2 - El Product Owner y los eventos

En Scrum, cada evento (ceremonia) está diseñado con un propósito específico para maximizar la transparencia, la colaboración y la eficiencia. **El Product Owner,** es fundamental para representar la voz del cliente y los intereses del negocio, y desempeña un papel crucial en estos eventos.

Su responsabilidad va más allá de simplemente definir y priorizar el trabajo; es el vínculo esencial que asegura que el equipo esté trabajando en las características más valiosas, alineando la entrega de producto con los objetivos del negocio.

En esta sección, exploraremos en detalle cómo el Product Owner se involucra en cada evento, qué se espera de él y cómo su aporte activo puede influir positivamente en la dinámica y los resultados del equipo Scrum.

El Product Owner en la Planificación del Sprint

- Facilita el evento, asegurando que durante la sesión se discuten 3 temas: ¿por qué este Sprint es valioso?, ¿qué se puede terminar (done) en este Sprint?, ¿cómo se realizará el trabajo elegido?
- Brinda información clara y precisa sobre el trabajo que se pueda terminar este Sprint
- Aclara las dudas en torno a las expectativas de las partes interesadas en el trabajo de este Sprint
- Aclara la información y detalles sobre elementos del Product Backlog.
- Define el Sprint Backlog y el Sprint Goal según la sesión con el equipo durante este evento
- Negocia el alcance del Sprint con los Developers.

> **Caso práctico:** Durante la planificación del sprint, el equipo está indeciso sobre qué funcionalidades priorizar. El Product Owner, Carlos, comparte recientes feedbacks de clientes y argumenta: "Nuestro principal competidor acaba de lanzar una funcionalidad similar, y debemos responder". Desglosa los elementos del Product Backlog, aclarando cualquier duda. Luego, al definir el Sprint Backlog, surge un debate sobre la carga de trabajo. Carlos, con base en la prioridad y la visión del producto, guía al equipo para acordar un Sprint Goal centrado en el impacto y valor para el usuario.

□□ Tip para el Product Owner

La preparación es clave: Asegúrate de entender muy bien los elementos del Product Backlog que has pre-seleccionado para discutir en la Planificación del Sprint, y ten a mano toda la información relevante para aclarar cualquier duda que puedan manifestar los Developers.
También asegúrate de que las historias de usuario y las tareas estén bien definidas, claras y priorizadas. Las más prioritarias deben tener suficiente detalle para que el equipo pueda trabajar en ellas.

El Product Owner en el Daily

Nota aclaratoria: Si el Product Owner está trabajando activamente en los elementos del Sprint, entonces participa como un Developer, de lo contrario no participa en este evento.

En caso que el Product Owner participe del Daily, es posible que pueda:

- Brindar aclaraciones sobre algún elemento del Product Backlog.
- Recopilar feedback para posibles reajustes del Product Backlog.

> **Caso práctico:** Durante un Daily, el equipo de desarrolladores discute un desafío relacionado con una historia de usuario. No están seguros de cómo implementar una funcionalidad específica. Laura, la Product Owner, quien ocasionalmente asiste a los Dailies, rápidamente comprende la confusión. Después del Daily, aparta a los miembros del equipo involucrados y proporciona una aclaración detallada con ejemplos del comportamiento esperado, resolviendo el desafío. Además, toma nota para mejorar la descripción de historias similares en el futuro.

□□ Tip para el Product Owner

Si decides asistir al daily, escucha activamente pero evita interrumpir o desviar la conversación a menos que sea estrictamente necesario. Si escuchas sobre desafíos o problemas relacionados con el Product Backlog, toma notas para refinar y mejorar el Backlog después del daily.
Si ves que hay confusiones o malentendidos sobre el Product Backlog, busca un espacio después del Daily para resolver las dudas.

El Product Owner durante el desarrollo del Sprint

- Agrega elementos al Product Backlog de acuerdo a las prioridades y expectativas de las partes interesadas.
- Realiza el refinamiento y priorización de los elementos del Product Backlog a partir de las sesiones de trabajo con el cliente y/o partes interesadas (esto suele hacerlo en paralelo mientras los Developers estan construyendo el producto).
- Resuelve las dudas a los Developers sobre el trabajo que esta en desarrollo (WIP).
- Colabora con las partes interesadas, identificando necesidades y ajustando expectativas.

> **Caso práctico:** Mientras el equipo está a mitad de camino en el Sprint actual, Tomás, el Product Owner, ya está en conversaciones con un stakeholder sobre las características de un futuro elemento que se desarrollará para el siguiente sprint. Tras obtener un entendimiento claro, programa una reunión de refinamiento con dos Developers para abordar este elemento. Laura, una de las Developers, sugiere una implementación que podría simplificar el trabajo. Tomás ajusta el elemento en el Product Backlog. Cuando llega el momento de la planificación del siguiente sprint, este elemento se discute con mayor fluidez y claridad, gracias a la planificación en paralelo que Tomás llevó a cabo.

□□ Tip para el Product Owner

Inicia el refinamiento y priorización de los elementos de los próximos sprints con anticipación, esto garantiza que estén listos cuando llegue el momento de la Planificación. Al trabajar en paralelo, es crucial que tanto el equipo como las partes interesadas tengan una visión clara del producto y cómo encajan las piezas.

El Product Owner en la Revisión del Sprint

- Facilita y dirige la sesión, garantizando un flujo estructurado.
- Asegura que el progreso respecto al Objetivo de Producto sea el foco de discusión de esta sesión.
- Invita y asegura que las partes interesadas clave estén presentes y activamente involucradas.
- Discute el Product Backlog tal y como está.
- Proyecta el objetivo y las fechas de entrega probables basándose en el progreso hasta la fecha (si es necesario).
- Verifica que el incremento cumpla con lo planificado, considerando tanto los criterios de aceptación como la Definición de Terminado (DoD).
- Facilita la discusión sobre posibles mejoras o ajustes en el producto.
- Confirma o redefine prioridades del Product Backlog basándose en la demostración realizada por los Developers, la deuda técnica (si existe) y en las necesidades cambiantes del negocio.
- Aborda la deuda técnica (si existe) y discute su impacto y posibles estrategias para gestionarla o reducirla.
- Indaga sobre las causas de la deuda técnica (para identificar qué aspectos deben mejorarse o ajustarse para la siguiente Planificación del Sprint).

> **Caso práctico:** Durante la revisión del sprint se presenta una funcionalidad de una aplicación de e-commerce. Un stakeholder clave menciona que, aunque la funcionalidad es lo que se pidió, el diseño podría ser más intuitivo para usuarios no técnicos. Laura, la Product Owner facilita el diálogo, para entender mejor la situación y añade esta observación al Product Backlog. Adicionalmente se

> encuentra un elemento en deuda técnica, por lo que Laura, reajusta el Product Backlog según el feedback recibido.

Tip para el Product Owner

Si se identifica deuda técnica, trata de comprender su origen y considera estrategias para gestionarla en futuros sprints. También asegúrate de registrar las observaciones y feedback para que puedan ser considerados en futuras decisiones y planificaciones.

El Product Owner en la Retrospectiva del Sprint

- Participa en la inspección de cómo fue el último Sprint con respecto a los individuos, las interacciones, los procesos, las herramientas, y la Definición de terminado (DoD).
- Participa de la identificación de lecciones aprendidas, compartiendo perspectivas desde el lado del negocio y el feedback del cliente.
- Participa en identificar las suposiciones erróneas que pudieron desviar al equipo y busca entender la causa raíz, los problemas que encontró el equipo, y cómo esos problemas fueron (o no fueron) resueltos.
- Reconoce y destaca los logros y las áreas de mejora observadas durante el Sprint.
- Puede sugerir la ampliación de la Definición de terminado (DoD) para mejorar la calidad del producto.

> **Caso práctico:** En la retrospectiva, Miguel, el Product Owner, resalta cómo el equipo logró superar un desafío técnico particular durante el Sprint. Sin embargo, surge que algunos Developers sintieron que no se comunicaron suficientemente bien los objetivos de negocio detrás de ciertas tareas. Patricia, una Developer, menciona que esto llevó a algunas decisiones técnicas que, de haber tenido toda la información, habrían sido diferentes. Miguel reconoce esta brecha y propone sesiones breves de alineación antes de cada Sprint para garantizar que el contexto del negocio esté claro. El equipo acuerda intentarlo en el próximo Sprint y evaluar su efectividad en la siguiente retrospectiva.

Tip para el equipo

La Retrospectiva **NO** es un espacio para culpar, sino para identificar oportunidades de mejora.

4.3 - Las Historias de Usuario

Las Historias de Usuario (User Story) son la forma recomendada en entornos ágiles para escribir los requisitos del producto desde la perspectiva del usuario. Dado que son afirmaciones breves, simples y fáciles de entender, resultan en una mejor comunicación entre los Stakeholders y el equipo Scrum.

Una de las principales ventajas de las Historias de Usuario es su estructura simplificada y orientada a la acción. Debido a que se centran en entender "¿quién solicita la funcionalidad?", "¿qué necesita/requiere?" y "¿por qué o para qué lo necesita?", las Historias de Usuario proporcionan una vista clara y contextualizada de lo que se espera, eliminando ambigüedades comunes en los requisitos tradicionales.

Los criterios de aceptación, que acompañan a cada Historia de Usuario, definen las condiciones específicas que deben cumplirse para que la historia se considere satisfactoriamente implementada o desarrollada. Actúan como una guía para el equipo, aterrizando las expectativas del Product Owner sobre el comportamiento o funcionalidad que debe tener esa historia. Mientras que, la Definición de Terminado (DoD) es la que establece cuándo una historia o tarea se considera completa en su totalidad.

Las Historias de Usuarios centran la atención en los usuarios finales, por lo que no utilizan un lenguaje técnico, con el fin de facilitar el entendimiento para los usuarios y asimismo ofrecer un contexto suficiente para guiar el esfuerzo de los Developers.

> Mike Cohn, uno de los pioneros en el desarrollo ágil, menciona que "Las historias de usuario permiten a los equipos trabajar en un lenguaje compartido, y eso hace que toda la experiencia de desarrollo sea más orientada al usuario y al valor. En lugar de ser una mera lista de tareas, las historias son llamadas a la acción para resolver desafíos específicos para los usuarios".

Estructura de una Historia de Usuario

Las Historias de Usuario suelen expresarse como una frase simple y breve que cumple con la siguiente estructura:

Como: Describe el ***Rol*** del Stakeholder que solicita (o usaría) la funcionalidad o requerimiento.

Quiero: Describe la ***necesidad o requerimiento*** del usuario, por lo general, es una frase corta.

Para: Describe el ***beneficio esperado*** por el Stakeholder una vez se desarrolle el requerimiento

Ejemplos de Historias de Usuario

> **Como:** Cliente, **Quiero:** pagar mi suscripción mensual vía sitio web por medio de transferencia bancaria o tarjeta de crédito, **Para:** evitar el desplazamiento hasta la sucursal de pago.

> **Como:** Supervisor de ventas, **Quiero:** consultar un listado de los pedidos de venta que han sido registrados y aún no han sido procesados, **Para:** Tener un reporte detallado a la mano.

> **Como:** Ejecutivo de cuenta, **Quiero:** consultar los datos de un cliente suministrándole al sistema únicamente su documento de identidad o código de cliente, **Para:** facilitar el proceso de búsqueda y reducir el tiempo de atención.

> **Como:** asesor de ventas, **Quiero:** disponer de una diadema inalámbrica, **Para:** poder levantarme regularmente de mi puesto y disminuir el estrés y el cansancio.

Componentes adicionales de una Historia de Usuario

Una Historia de Usuario puede tener componentes adicionales que permitan mejorar su entendimiento tanto para el usuario como para los Developers. Estos componentes también garantizan una ejecución precisa y acorde a las expectativas. Usualmente estos componentes son:

- Criterios de aceptación (detalle técnico)
- Estimaciones
- Prototipos, Diagramas, etc.

1. Criterios de aceptación (detalle técnico)

> Microsoft define a los criterios de aceptación como **las condiciones que un producto de software debe satisfacer para ser aceptado por un usuario, cliente o stakeholder.**

> Para Google, son **estándares pre-establecidos o requerimientos que un producto o proyecto debe satisfacer.**

Cada historia de usuario debe ir acompañada de sus criterios de aceptación. Estos criterios son esenciales para garantizar que todos en el equipo tengan el mismo entendimiento de lo que se espera. Ayudan a delimitar el alcance de una historia y a definir cómo se verificará que se ha cumplido con la necesidad o requerimiento descrito.

Concretamente en Scrum, se los define como un conjunto de sentencias redactadas de tal manera que conduzcan a una respuesta clara de "aceptado/rechazado". **Detallan las especificaciones técnicas de cada Historia de usuario o Tarea.**

El Product Owner es el responsable de redactar los criterios de aceptación para cada Historia de Usuario en conjunto con el cliente y/o los interesados, para posteriormente confirmarlos con los Developers antes de iniciar el desarrollo. Esta confirmación se realiza durante la Planificación del Sprint.

Los criterios de aceptación ayudan a los Developers a entender el funcionamiento del producto, de manera que estimarán mejor el trabajo necesario, además de servir como guía para tomar decisiones más acertadas de acuerdo con lo que se espera de cada Historia de Usuario.

Un criterio de aceptación debe poder probarse o testearse. Cuando no es posible probar algún criterio de aceptación es un claro indicio de una mala redacción.

Se debe tener en cuenta que al probar una Historia de Usuario frente a un criterio de aceptación se obtiene una respuesta binaria: Cumple o No cumple; **no existe el concepto de cumplir parcialmente un criterio de aceptación.**

Algunos beneficios de contar con buenos criterios de aceptación:

- Permite que los Developers entiendan una funcionalidad desde la perspectiva del usuario.
- Reduce las ambigüedades al desarrollar las Historias de Usuario.
- Promueve la calidad del producto permitiendo que los Developers se adhieran a los criterios de aceptación antes de llegar a una demostración de resultados en la Revisión del Sprint.
- Permiten confirmar si una Historia de Usuario se comporta de acuerdo con lo esperado.

Tip para el Developer

No esperes hasta la Planificación del Sprint para hacer preguntas. Si un criterio de aceptación no especifica algo, no lo asumas. Es mejor preguntar y obtener claridad.
Si durante el desarrollo encuentras que un criterio de aceptación no es viable o requiere mucho más trabajo del estimado, comunica esto al Scrum Master lo antes posible.

Una manera **opcional** de construir los criterios de aceptación es la siguiente:

Cuando (Rol) hace (Acción) consigue (Resultado / Comportamiento esperado)

Veamos un ejemplo:

> **Historia de Usuario:** *Como* cliente, *quiero* pagar mi suscripción mensual vía sitio web por medio de transferencia bancaria o tarjeta de crédito, *para* evitar el desplazamiento hasta la sucursal de pago.
>
> **Criterios de Aceptación:**
>
> - Cuando el cliente ingrese al portal de pagos del sitio web, visualiza una opción (botón) para iniciar el proceso de pago de sus suscripciones.
> - El cliente tiene la opción de seleccionar su método de pago preferido: transferencia bancaria o tarjeta de crédito.
> - Si el cliente selecciona transferencia bancaria, se le solicitan los datos bancarios y se presentan las instrucciones. Si elige tarjeta de crédito, se le solicitan los datos de la tarjeta.
> - Al ingresar la información del pago, el sistema valida que todos los campos estén completos y que los datos correspondan a formatos válidos (por ejemplo, número de tarjeta válido).
> - Una vez realizado el pago, el cliente recibe un mensaje de confirmación en la pantalla y, adicionalmente, un correo electrónico con el recibo de pago.
> - Si ocurre algún problema durante el proceso de pago (como una tarjeta declinada), el cliente recibe un mensaje claro sobre el error y las recomendaciones para solucionarlo.

Tip para el Product Owner

En la discusión de las historias de usuario, puedes emplear la técnica de las 3 C's: Card (Tarjeta), Conversation (Conversación) y Confirmation (Confirmación). Esta técnica ayuda a mantener la simplicidad, ya que la información se mantiene ligera (tarjeta), se clarifica a través del diálogo (conversación) y se verifica (confirmación).

2. Estimaciones

Las estimaciones se refieren al tiempo o esfuerzo necesario para completar una Historia de Usuario o una tarea. Los equipos Scrum a menudo usan **puntos de historia, horas o días** para estimar la

complejidad y el esfuerzo necesario.

Es muy importante realizar la estimación del trabajo requerido para desarrollar las Historias de Usuario y las tareas relacionadas, de esta manera se logran planificaciones más precisas.

La estimación **NO** es determinada por el cliente o el Product Owner, sino que son **los Developers en conjunto con el Scrum Master y el Product Owner,** quienes definen las estimaciones ya que ellos son quienes ejecutarán directamente el trabajo necesario.

□□ Tip para el Product Owner

Recuerda que las estimaciones son relativas y no deben interpretarse como compromisos rígidos. Sirven como guías basadas en la información y experiencia actual del equipo.

Para garantizar una estimación más precisa, se pueden considerar elementos como:

- Dificultad / complejidad: Hace referencia a qué tanto esfuerzo se requiere para ejecutar determinado trabajo. Se utilizan los Puntos de historia como unidad de medida.
- Duración: Hace referencia a cuánto tiempo se requiere para ejecutar determinado trabajo. Se puede utilizar horas o días como unidad de medida.
- Costo: Hace referencia al dinero requerido para la ejecución de un determinado trabajo, el cual se calcula a partir del trabajo de los developers más los recursos necesarios para su ejecución.

Uno de los problemas más comunes con la estimación de las Historias de Usuario, es sólo se estimar la dificultad (puntos de historia), o sólo estimar la duración (horas / días); por lo que es altamente recomendable estimar los dos. Muchos equipos prefieren unificar esto mediante una **relación entre los puntos de historia y la duración,** es decir que para cada punto de historia se asigna un tiempo determinado.

□□ Tip para el Scrum Master

Si sientes que las estimaciones consistentemente no reflejan el esfuerzo real de los Developers, aborda el tema en la retrospectiva del sprint. Analiza y ajusta el proceso de estimación en conjunto con el equipo según sea necesario.

El manejo que se da a los puntos de historia es **específico de cada equipo,** y es crucial entender que las estimaciones están íntimamente ligadas a la experiencia, conocimientos y dinámicas específicas de cada equipo, por lo que **NO** es recomendable comparar diferentes equipos basados en sus estimaciones: Incluso si dos equipos trabajan en el mismo producto o en productos similares, su ritmo de trabajo puede variar.

En Scrum el ritmo de trabajo también se conoce como la "velocidad" y se basa en cuántos puntos de historia, en promedio, un equipo puede completar en un Sprint. La velocidad no solo se basa en la capacidad de producción; también se ve influenciada por la comunicación dentro del equipo, las herramientas disponibles, las interrupciones externas, entre otros factores.

> **¿Por qué no comparar equipos basados en estimaciones?** Comparar equipos basados en sus puntos de historia o velocidad puede llevar a interpretaciones erróneas. Si un **equipo A completa 50 puntos** de historia en un Sprint y el **equipo B completa 30 puntos**, no significa necesariamente que el equipo A sea más productivo o eficiente. Puede que el equipo B esté manejando tareas más complejas o enfrentando desafíos que el equipo A no tiene. Comparar equipos de esta manera puede llevar a decisiones injustas y a presiones innecesarias, lo que puede afectar negativamente la moral y la productividad.

A continuación se muestra un ejemplo de ello:

Supongamos que tenemos dos equipos, Equipo A y Equipo B, ambos trabajando en diferentes funcionalidades del mismo producto.

El equipo A, con base en sus experiencias anteriores y en la comprensión que tienen sobre su capacidad y velocidad de trabajo, definió la siguiente tabla de estimaciones:

estimacion1
Image not found or type unknown

Por otro lado, el equipo B, que puede tener una composición diferente en términos de habilidades, experiencia y conocimientos técnicos, definió la siguiente tabla de estimaciones:

estimacion2
Image not found or type unknown

Durante la sesión de estimación, el Equipo A asigna 5 puntos a una historia de usuario específica, mientras que el Equipo B asigna 8 puntos a una historia similar. Esto no significa necesariamente que el Equipo B vea la historia como más complicada o que sean menos eficientes. Es posible que el Equipo B, basado en experiencias pasadas, considere ciertos riesgos o complejidades adicionales que el Equipo A no ha experimentado. Por otro lado, el Equipo A puede tener un experto en una herramienta o tecnología relevante que reduce la percepción de esfuerzo.

Si bien ambos equipos trabajan en el mismo producto, tienen distintas experiencias, habilidades y contextos que influyen directamente en cómo ven y estiman el trabajo.

□□ Tip para el Developer

Primero la calidad: No sacrifiques la calidad de tu trabajo para cumplir con una estimación. Si una historia de usuario necesita más tiempo del estimado para ser completada adecuadamente, es mejor discutir esto con el Scrum Master para ajustar el plan que entregar un producto inferior.

3. Prototipos, Diagramas, etc

Los prototipos, diagramas o esquemas de una Historia de Usuario nos sirven para mostrar un acercamiento al resultado que se espera obtener cuando los Developers ejecuten su trabajo, de esta manera se puede probar el producto antes de construirlo para identificar posibles errores y generar un mejor entendimiento para los Developers.

> **Prototipo:** Es una representación preliminar de un producto, que muestra cómo se verá y funcionará, sin necesidad de que esté completamente funcional. Puede ser desde un boceto en papel hasta una versión interactiva digital.

> **Diagrama:** Son representaciones gráficas que muestran las relaciones entre diferentes componentes, actores o flujos de un sistema o proceso.

Estos elementos visuales sirven como herramientas de comunicación entre el equipo, el Product Owner y los stakeholders, y pueden ser vitales para reducir la ambigüedad y mejorar la calidad del producto final.

Tip para el equipo

Un prototipo no tiene que ser perfecto ni estar completo: La idea es que sea una representación aproximada que pueda modificarse fácilmente en función de los comentarios recibidos.

Beneficios de usar prototipos:

- Permiten a los Developers desarrollar su trabajo en torno a una guia clara sobre las expectativas del cliente/usuario final.
- Permiten iterar y hacer cambios con rapidez antes de entrar en el desarrollo.
- Constituyen un gran apoyo para lograr mejores estimaciones de trabajo.
- Son más fáciles de abordar con los usuarios finales.
- Permite que el usuario se sienta involucrado en construcción del producto, ya que puede "verlo por adelantado".
- Se reduce el riesgo o la incertidumbre sobre el resultado de producto esperado.

Tip para el equipo

No todos los elementos o características necesitarán un prototipo o diagrama. La decisión de cuándo usarlos dependerá de la complejidad de la historia, los recursos disponibles y las necesidades del equipo y los stakeholders. Es importante recordar que, aunque estas herramientas son valiosas, deben usarse de manera estratégica para maximizar su valor sin desperdiciar esfuerzos en detalles innecesarios.

Épicas

Una épica es una gran pieza de trabajo que se puede desglosar en un conjunto de tareas más pequeñas (historias de usuario) que estan relacionadas entre sí. Puede considerarse como un **contenedor para historias de usuario que están relacionadas** y que, una vez completadas, cumplirán un objetivo o función más grande y más complejo.

Algunas ideas de las que se podrían definir las épicas son: • Módulos. • Componentes. • Hitos. • Entregables. • Funcionalidades.

> Se puede decir que una épica es un nivel de agrupación por encima de las historias de usuario que permite clasificarlas por funcionalidades, módulos, subsistemas, etc.

Las épicas:

- Son más grandes y complejas que las historias de usuario individuales.
- Se pueden descomponer en múltiples historias de usuario que, una vez completadas, satisfarán la épica en su totalidad.
- A menudo se requieren de múltiples sprints o ciclos de desarrollo para completarlas.

> El propósito de identificar épicas es ayudar a gestionar y organizar el trabajo del Product Backlog de manera más efectiva. Se trata de simplificar la complejidad y proporcionar una visión clara del objetivo final, permitiendo a los equipos trabajar de manera más coordinada y eficiente.

Recomendaciones sobre las épicas:

- Las épicas suelen **identificarse** en etapas tempranas del desarrollo, cuando el equipo se encuentra con grandes piezas de trabajo que son demasiado amplias para ser consideradas como una historia de usuario única.
- A medida que se acerca el momento de trabajar en una épica, el Product Owner en conjunto con el equipo **descompondrá la épica** en historias de usuario más pequeñas y manejables.
- Al igual que con las historias de usuario, **las épicas se priorizan.** Las épicas de alta prioridad se descompondrán antes que las épicas de baja prioridad.
- Dado que una épica se puede abarcar en múltiples sprints, se trabajará en las historias de usuario individuales que la componen a lo largo del tiempo.

> **Ejemplo de épica:** Renovar la experiencia de pago del sitio web para simplificar el proceso de compra y reducir el abandono del carrito.
>
> **Historias de Usuario derivadas de la Épica:**
>
> - *Como* cliente, *quiero* ver un resumen claro de mi pedido antes de confirmar el pago, *para* evitar errores.

> - *Como* cliente, *quiero* recibir un correo electrónico de confirmación inmediatamente después de hacer mi compra, *para* tener trazabilidad de la compra
> - *Como* cliente, *quiero* opciones de pago flexibles (tarjeta de crédito, PayPal, transferencia bancaria), *para* elegir dependiendo de mis necesidades y disponibilidad
>
> ... (y así sucesivamente, hasta que todas las facetas de la renovación de la experiencia de pago estén cubiertas).

Guía para identificar épicas sin morir en el intento ☝

- Las épicas suelen estar vinculadas a **objetivos estratégicos o de alto nivel.** Pregúntate: ¿Cuál es el gran objetivo o funcionalidad que queremos alcanzar?
- Si una historia de usuario parece **demasiado grande o ambigua,** es probable que sea una épica. Si al escribir una historia de usuario te encuentras agregando muchos **"y"** o **"o",** considera si puede ser desglosada en múltiples historias de usuario más pequeñas y, por lo tanto, convertirse en una épica.
- Si crees que una tarea o historia de usuario puede **extenderse más allá de un sprint**, es probable que estés frente a una épica.
- Discute las funcionalidades y objetivos con stakeholders y miembros del equipo. Ellos podrán proporcionar una perspectiva sobre la complejidad y el alcance, ayudando a identificar épicas.
- Si puedes desglosar un elemento en varias historias de usuario que son independientes entre sí pero que trabajan juntas para un objetivo común, entonces tienes una épica. (esto es que cada historia de usuario puede ser trabajada de forma independiente, pero que le apuntan al mismo objetivo).
- Utiliza herramientas como mapas de historias (story maps) para visualizar el flujo y las grandes áreas de funcionalidad. Estas áreas suelen ser buenas candidatas para épicas.
- Al identificar épicas, enfócate en el **"qué" (lo que se quiere lograr)** en lugar del **"cómo" (la solución técnica).** Las soluciones técnicas deben surgir al desglosar la épica en historias más pequeñas.
- A medida que el equipo avanza y se adquiere más conocimiento, es posible que algunas historias de usuario se expandan en alcance y se conviertan en épicas, o que algunas épicas se reduzcan y se conviertan en historias de usuario. Es esencial revisar y refinar regularmente.
- Si al intentar estimar una historia de usuario encuentras que el equipo tiene muchas incertidumbres o considera que llevará mucho tiempo, podría ser una señal de que estás tratando con una épica.
- Si un elemento tiene muchas dependencias o debe ser completado en un orden específico, podría ser una épica que necesita ser desglosada en historias de usuario que puedan ser trabajadas en paralelo o en un orden específico.

4.4 - Refinamiento del Product Backlog

> Estamos a puertas de iniciar con el desarrollo del producto, tenemos una comprensión general del producto y contamos con un Product Backlog que contiene elementos cortos y concretos. **Tú como Product Owner los entiendes claramente, pero: ¿el resto del equipo también los entienden?**

El refinamiento del Product Backlog es una actividad continua en la que el equipo Scrum colabora con las diferentes partes interesadas para crear un entendimiento compartido sobre lo que el producto hará y no hará (historias de usuario y tareas), sobre el esfuerzo que requerirá su implementación (estimaciones) y el orden en que lo hará (priorización).

El equipo puede llevar a cabo el refinamiento paralelamente al desarrollo del actual sprint, con esto se logrará que la planificación del siguiente sprint sea muchísimo más eficiente.

> Si bien el Product Owner lidera el proceso de refinamiento del Product Backlog, es vital que cuente con la colaboración activa de los Developers y el Scrum Master. Esta colaboración multidisciplinaria no solo enriquece el proceso con diversas perspectivas y habilidades, sino que también promueve un entendimiento compartido y una toma de decisiones más informada, lo que es esencial para el desarrollo exitoso del producto.

¿Por qué es importante el refinamiento del Product Backlog?

- Sin un entendimiento compartido, se corre el riesgo de implementar algo incorrecto, desperdiciar esfuerzos y tener que volver a trabajar en la implementación para hacerlo bien.
- Si no se dimensiona cada elemento, no se tiene en cuenta el “esfuerzo” de los elementos y se corre el riesgo de sobrevalorar los elementos de alto valor y alto coste, e infravalorar los de menor valor y menor coste.

- Si no se ordena el Product Backlog según su prioridad, se corre el riesgo de trabajar en elementos que no son tan importantes y de pasar por alto otros importantes.

> **Caso de estudio: ¿Por qué deberíamos refinar?** En una reconocida empresa de software, la omisión del refinamiento del Product Backlog llevó a consecuencias graves. Sin una adecuada claridad y detalle en los ítems del backlog, el equipo de desarrollo frecuentemente enfrentaba obstáculos en la implementación, resultando en retrasos significativos y una acumulación de tareas mal definidas. Esto no solo generó frustración entre los miembros del equipo, sino que también impactó negativamente en la satisfacción del cliente debido a entregas de productos por debajo del estándar esperado. Este caso resalta la necesidad vital de la inclusión activa y continua del Product Owner, el Scrum Master y los Developers en el proceso de refinamiento del backlog para asegurar un flujo de trabajo fluido y entregas exitosas.

◫ Tip para el Product Owner: Mantener una lista de "ítems de refinamiento listos para discusión

Estos ítems pueden ser abordados en las sesiones con el equipo. Esto garantiza que siempre haya un flujo constante de ítems listos para ser trabajados y que las sesiones de refinamiento sean eficientes y focalizadas. Además, no dudes en retirar elementos que ya no añaden valor, manteniendo así un Backlog limpio y centrado en las prioridades actuales.

¿Qué se hace en el refinamiento del Product Backlog?

Algunas de las cosas que se puedan hacer en el refinamiento son:

- Recopilar información (puede ser mediante entrevistas, investigaciones, etc.) para redactar nuevas historias de usuario.
- Explicar el contexto de las próximas funcionalidades.
- Desglosar el trabajo (historias de usuario y tareas).
- Agregar detalles a historias de usuario ya escritas (criterios de aceptación y prototipos).
- Revisar y ajustar las prioridades de los ítems del backlog en función de la retroalimentación y aprendizajes de los sprints anteriores.
- Identificar y eliminar posibles impedimentos que puedan surgir durante la implementación de ciertas tareas, facilitando una planificación más realista y evitando futuros bloqueos.

◫ Tip para el Product Owner: No olvides involucrar a todo el equipo Scrum en el proceso de refinamiento.

Aunque como Product Owner tienes la responsabilidad de mantener y priorizar el Product Backlog, el conocimiento y la perspectiva de los Developers y del Scrum Master pueden ayudar a enriquecer los ítems del backlog, proporcionando detalles técnicos y apuntando posibles obstáculos desde una etapa temprana, facilitando una entrega más fluida y exitosa del producto.

¿Cuándo es el mejor momento para el refinamiento del Product Backlog?

El refinamiento del Product Backlog es una actividad continua, no solo para el Product Owner, sino para **todo el equipo.** El Product Owner puede refinar los elementos del Product Backlog en cualquier momento, dentro o fuera de una reunión, dependiendo de lo que sea más conveniente para el equipo.

- **En una reunión (sincrónica):** El equipo se reúne y puede invitarse a otras partes interesadas si fuese necesario. Esta reunión **NO hace parte de los eventos** de Scrum.
- **En un documento (asincrónica):** El Product Owner prepara un documento colaborativo y el resto del equipo va agregando detalles y comentarios (no se requiere reunión).

Nota: La forma y momento en que se haga el refinamiento depende del equipo Scrum.

> **Es vital destacar que la planificación en Scrum ocurre de manera paralela** Mientras los Developers están inmersos en la construcción del actual incremento de producto, el Product Owner se encuentra activamente trabajando en el refinamiento de los próximos elementos del backlog, identificando nuevas oportunidades y ajustando prioridades, garantizando así un flujo constante y eficiente de trabajo para futuros sprints.

4.5 - Técnicas y herramientas útiles para el Product Owner

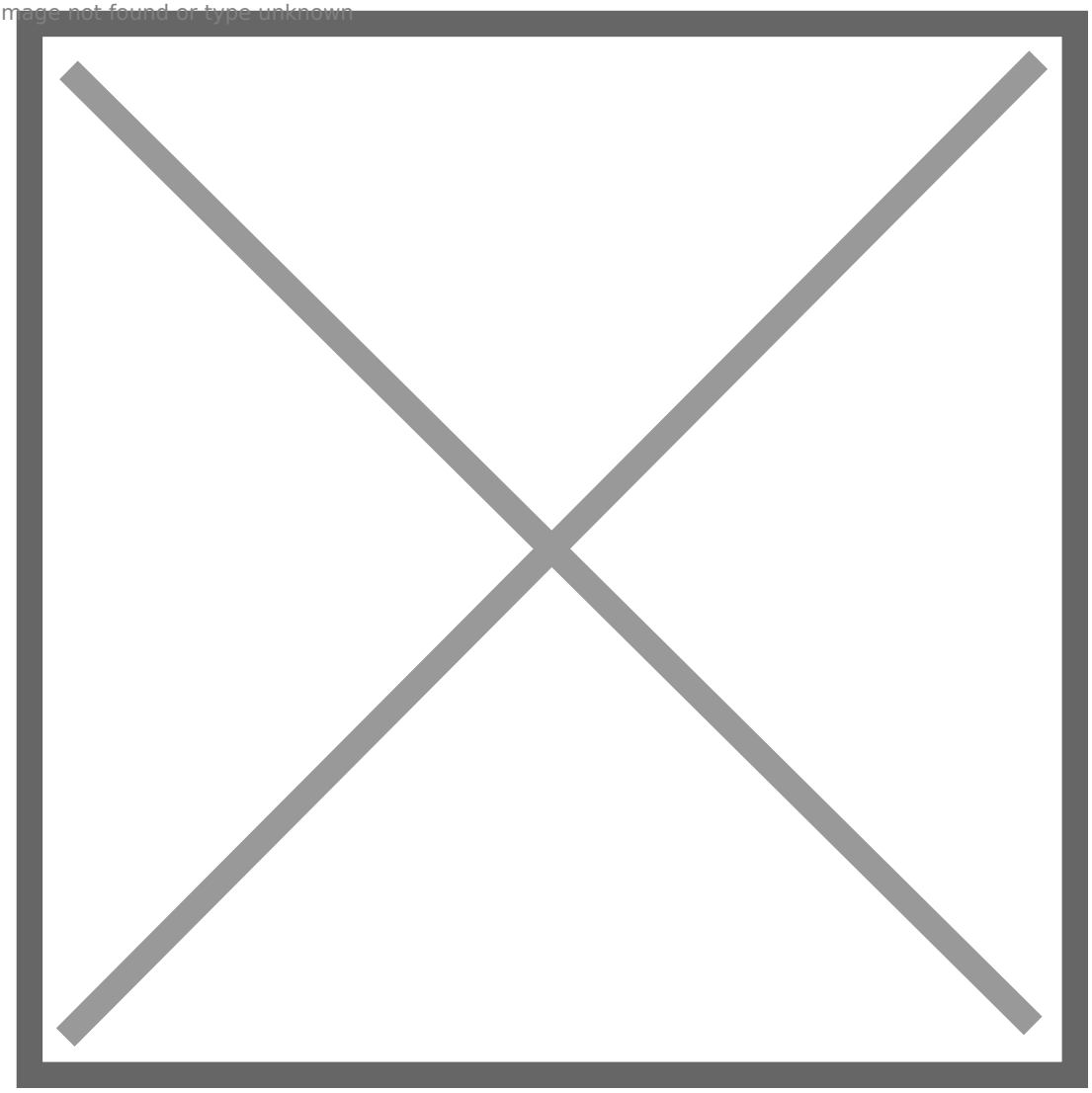
Herramienta 1: User stories mapping (Mapeo de historias de usuario)

Esta técnica es vital para organizar y visualizar de manera clara y estructurada los elementos del Product Backlog, agrupándolos según su funcionalidad y dependencia, lo que a su vez facilita su proyección a lo largo del tiempo.

El proceso de mapeo se desarrolla típicamente en sesiones de trabajo que duran entre 2 y 4 horas. Durante estas sesiones, se utiliza un lienzo y post-its para facilitar la categorización y priorización de los elementos que integrarán el producto.

Una de las grandes ventajas de este método es que asiste en la programación de los sprints, proporcionando una perspectiva clara para los stakeholders sobre qué se realizará y cuándo, aunque sea de manera aproximada.

Image not found or type unknown



> La técnica de Mapeo de Historias de Usuario (User Stories Mapping) sirve como una herramienta crucial para estructurar y definir el MVP (Producto Mínimo Viable), permitiendo una visualización clara de las funcionalidades esenciales que deben ser implementadas en primera instancia para satisfacer las necesidades centrales del cliente.

¿Cómo construir el mapa de historias?

Antes de comenzar, es fundamental establecer el estado objetivo del producto (Product Goal), ya que esto determinará el contexto dentro del cual se desplegará el producto. Posteriormente, es necesario identificar las funcionalidades o componentes principales que constituirán el producto, conocidos como "épicas".

Pasos para crear un Mapa de Historias de Usuario

1. **Definición del Product Goal:** Establece claramente el objetivo principal del producto que guiará todas las decisiones futuras, recuerda basarte en el problema que se pretende

solucionar.
2. **Identificación de Épicas:** Determina las grandes funcionalidades o componentes que conformarán el producto.
3. **Creación de Historias de Usuario:** Desarrolla historias de usuario que describan cada función desde la perspectiva del usuario.
4. **Organización de Historias:** Organiza las historias de usuario identificadas en una estructura visual, categorizándolas según su funcionalidad y dependencia.
5. **Priorización:** Establece las prioridades de las historias de usuario, decidiendo cuáles deben abordarse primero basándote en su valor y complejidad.
6. **Revisión y Ajustes:** Durante las sesiones de refinamiento del Product Backlog, revisa y ajusta el mapa según sea necesario, incorporando los aprendizajes y cambios que surjan durante el desarrollo.

Herramienta 2: Técnica para priorizar - MoSCoW (Must have, Should have, Could have, Would have)

La técnica MoSCoW es un método efectivo para **priorizar los elementos del Product Backlog**, ayudando a discernir lo esencial de lo deseable. Aquí está desglosada en sus componentes:

- **Must have (M):** Características indispensables, sin las cuales el producto no puede lanzarse.
- **Should have (S):** Características importantes que agregan valor significativo pero no son críticas para el lanzamiento.
- **Could have (C):** Características deseables pero no necesarias, pueden posponerse sin afectar gravemente el producto.
- **Would have (W):** Características que serían buenas tener en algún momento, pero no son una prioridad en este momento.

Este enfoque facilita la toma de decisiones durante el refinamiento del backlog, permitiendo una distribución más estratégica de los recursos y esfuerzos del equipo.

¿Cuándo aplicar MoSCoW en Scrum?

La técnica MoSCoW puede aplicarse en varias etapas del ciclo de Scrum:

- **Durante el Refinamiento del Product Backlog:** Para ayudar a definir y ajustar las prioridades de los elementos en el backlog.
- **En la Planificación del Sprint:** Para decidir cuáles elementos deben abordarse en el próximo Sprint, basándose en su prioridad y dependencia.

- **En las Revisiones del Sprint:** Para evaluar si las características "Must have" se han implementado correctamente y para re-priorizar características para los siguientes Sprints si es necesario.

Herramienta 3: Modelo Kano

El Modelo Kano es una herramienta útil para identificar y categorizar las necesidades del cliente en el desarrollo del producto. Se clasifica en las siguientes categorías:

- **Características Básicas:** Elementos que los clientes dan por sentado. Su ausencia causaría insatisfacción, pero su presencia no aumenta la satisfacción.
- **Características de Rendimiento:** Aspectos que pueden incrementar la satisfacción del cliente según su grado de implementación.
- **Características de Excitación:** Innovaciones que los clientes no esperan, pero que podrían deleitarlos y generar una ventaja competitiva.

Al aplicar el Modelo Kano, el Product Owner puede obtener una comprensión más profunda de lo que realmente desea el cliente, lo que puede ayudar a priorizar de manera efectiva las funcionalidades en el Product Backlog.

Herramienta 4: Customer Journey Maps (Mapas del Viaje del Cliente)

El Customer Journey Map es una herramienta visual que representa la experiencia total del cliente con un producto o servicio desde el primer contacto hasta la interacción final. Ayuda a comprender las necesidades, las emociones y los obstáculos que el cliente podría enfrentar en cada etapa. Aquí está su estructura general:

- **Puntos de contacto:** Donde y cuando el cliente interactúa con el producto o servicio.
- **Motivaciones:** ¿Por qué el cliente se siente motivado para continuar o abandonar?
- **Preguntas:** Dudas o inquietudes que el cliente pueda tener en cada etapa.
- **Emociones:** Cómo se siente el cliente en cada punto de contacto.

El uso de los Customer Journey Maps permite al Product Owner:

- Identificar áreas de mejora en la experiencia del cliente.
- Priorizar características basadas en momentos críticos del viaje del cliente.
- Diseñar soluciones que se alineen mejor con las expectativas y necesidades del cliente.

Otras herramientas / técnicas útiles para el Product Owner

- **Técnica de Estimación Poker (explicada en la sección de Developer):** Utilizada para estimar el esfuerzo requerido para cada tarea.
- **Impact Mapping:** Una técnica estratégica de planificación que visualiza las metas y el impacto esperado de un proyecto.
- **Herramientas de Gestión de Proyectos Ágiles (como JIRA, Trello, Asana):** Facilitan la organización y seguimiento de las tareas y progresos del equipo.
- **Lean Canvas:** Una herramienta que ayuda a desarrollar y afinar el modelo de negocio de un producto.
- **Analytics Tools:** Herramientas como Google Analytics para entender mejor el comportamiento del usuario y tomar decisiones informadas.
