PLANEANDO SER ÁGIL

# Destino desconocido, la diferencia entre planear todo desde el principio a realizar planeamiento iterativo

*Douglas Adams dijo una vez: «Me encantan los plazos. Me encanta el silbido que*

*hacen al pasar volando. ¡Whoosh! Ahí va otra fecha límite».*

Esto nos pasa todo el tiempo, ponemos una fecha límite y no la cumplimos. La pregunta es, ¿por qué sucede esto? Pero lo que es más importante, ¿qué podemos hacer para evitarlo?

A esto me gusta llamarlo navegar por lo desconocido, por ejemplo, si te dijera que tienes que navegar a través de este campo de pingüinos, ya sabes, podrías mirar hacia abajo y decir: «Bueno, si puedo, puedo poner mi pie aquí y tal vez entrar y dar una vuelta».

Pero cuando llegas al centro, ¿cómo trazas un rumbo hacia el otro lado? Ah, y por cierto, los pingüinos van a seguir moviéndose a medida que tú avanzas por ellos, como ocurre con el

desarrollo de software, en el que el sistema operativo se va parcheando y remendando. Y los

paquetes se están parcheando. Y ya sabes, las cosas se mueven, se mueven, pero sabes que si te metes de puntillas en medio de esos pingüinos y miras hacia abajo, se parecerá mucho a

lo que era al principio, cuando tú, ya sabes más, estés en un punto de vista diferente.

Y esa es la idea, claro, desde este punto de vista, probablemente puedas seguir conspirando.

Y a medida que os vayáis acercando, tendréis el siguiente punto de vista en el que podréis trazar el mapa.

**Así que el mensaje aquí es que no lo decidas todo en el punto que menos sabes.** Hacemos esto todo el tiempo. Al principio del proyecto, no sabemos prácticamente nada, sabemos muy poco sobre el proyecto. Y ahí es cuando planificamos todo, ¿no?, como si pudiéramos imaginarnos lo que va a pasar hacia el final. De eso se trata la metodología ágil, de la planificación iterativa. **No decidas todo al principio, cuando menos sabes.** Lo que quieres hacer es planificar lo que sabes y, a medida que avanzas, decides que sabes más, ajustas el plan.

Por lo tanto, sus estimaciones serán más precisas de esta manera. Porque ya sabes, si alguien te pregunta qué vas a hacer dentro de tres meses. Dígales: «Bueno, puedo decirles que podemos tener una precisión del 50%, ¿verdad? Pero puedo decirles con una precisión de casi el 100% lo que voy a hacer dentro de dos semanas, bien y dos semanas después».

**Así que la idea aquí es que no intentes ser omnipotente, porque no tienes razón. No intentes planificar todo por adelantado.** Desea planificar sobre la marcha y, a medida que vaya aprendiendo más de lo que podría, añadir más al plan y obtener estimaciones cada vez mayores del lugar en el que se encuentra y del tiempo que tardará en llegar hasta allí.

# Roles ágiles y la necesidad de entrenamiento

Algunas cosas organizativas que las personas hacen cuando intentan ser ágiles, que conducen al fracaso, y una de ellas es **colocar a las personas en un nuevo puesto sin capacitarlas.**

Por lo tanto, **el gerente de producto se convierte en el propietario del producto.** Aquí hay una gran diferencia, el gerente de producto es un título de trabajo. El propietario del producto es un rol de scrum. Puede que sean la misma persona, puede que no sean la misma persona.

Luego veo que **el gerente del proyecto se convierte en el maestro de scrum** sin capacitación sobre cómo ser un maestro de scrum.

Y, por último, **el equipo de desarrollo se convierte en un equipo de scrum.** Pero solo son ingenieros de software. No hay evaluadores en el equipo. No hay personal de operaciones en el equipo. No hay analistas de negocios ni nada en el equipo. Así que analicemos un poco más por qué hacer estas cosas conduce al fracaso.

En primer lugar, el gerente de producto se convierte en propietario del producto. El gerente de producto suele ser una persona de negocios que administra el presupuesto, puede o no tener una visión de cómo van las cosas. Pero el **propietario del producto es el visionario que dirige al equipo en una serie de experimentos diseñados para lograr cualquiera que sea el objetivo del sprint.** Y estos no suelen ser los mismos conjuntos de habilidades, **hay que centrarse más en dónde llevar el producto que en cómo gestionar el presupuesto**. Además, los gerentes de producto suelen ser la persona, se centran en los aspectos operativos del negocio, ¿verdad? En lugar de ser un conducto entre las partes interesadas y el equipo, traduzca esos requisitos empresariales en los objetivos técnicos que el equipo debe alcanzar. Así que ten cuidado. Algunos gerentes de productos de proyectos son buenos para ser propietarios de productos, otros no solo entienden por qué uno de ellos es un puesto de trabajo. El otro es un rol en scrum, y puede que no sea la misma persona.

Lo siguiente es que el director del proyecto se convierta en el scrum master. Es clave entender cómo trabajan los gerentes de proyectos. Nuestro director de proyectos es un administrador de tareas, hace que todos sigan un plan. Sabes, cuando enseño Agile, muy a menudo tengo a alguien al final de la sala que me está viendo planificar nuestros sprints y crear nuestro tablero Kanban y dice: «Disculpe, ¿cómo puedo saber si le he asignado demasiado trabajo a alguien?» Y la respuesta es que no se asigna trabajo a nadie. Por eso no hay forma de saberlo, ¿verdad? Son un equipo autogestionado que se asignan el trabajo a sí mismos. Por lo tanto, ser **un maestro de scrum es una función muy diferente, en la que eres un entrenador que mantiene al equipo concentrado en el sprint actual y en la autogestión.**

La otra gran diferencia entre un gestor de proyectos y un scrum master es que cuando el gestor de proyectos siente que tienes un impedimento, documenta el riesgo en su pequeña hoja de cálculo, ¿verdad? Esto es lo que hacen, documentan los riesgos. Así que si tengo un problema, tengo algún bloqueo, el director del proyecto va a decir: «¿Qué vas a hacer para desbloquearte?» A diferencia de un **scrum master que elimina los impedimentos y protege al equipo de las interrupciones.** Por lo tanto, **si el equipo dice: «Tengo un impedimento», el maestro de scrum debería decir: «Deja que yo me encargue de eso , tú ponte a trabajar en algo más productivo. Y déjame quitarte eso del camino».** Los directores de proyecto casi nunca dirán eso. Por lo tanto, realmente se necesita un tipo de persona diferente para asumir el papel de scrum master que el que se les enseñó a los gerentes de proyectos a gestionar los proyectos para planificar y documentar los riesgos.

Y, por último, es el equipo de desarrollo contra el equipo de scrum. Quiero decir, un equipo de desarrollo normalmente está formado por ingenieros de software, ¿verdad? Solo desarrolladores. Pero **un equipo de scrum es un equipo multifuncional. Incluye a desarrolladores y evaluadores, personal de seguridad y analistas y operaciones empresariales, a cualquier persona que necesitemos para crear un incremento.** Por eso es muy importante que se reorganice y cree equipos de desarrollo multifuncionales, no solo un grupo de ingenieros de software.

*cita de Bill Cantor: «Hasta que los líderes empresariales no acepten la idea de que ya no administran proyectos con funciones, plazos y costos fijos como lo hicieron con Waterfall, tendrán dificultades para usar Agile tal como fue diseñado para usarse».*

Por lo tanto, los roles han cambiado. No se puede poner a las personas en un nuevo rol sin capacitarlas, de lo contrario, un gerente de proyecto se esforzará por convertir un tablero Kanban en un diagrama de Gantt, porque eso es todo lo que sabe, es administrar proyectos con diagramas de Gantt. Es muy importante que esta mentalidad provenga de la alta dirección, ya que **la alta dirección no puede pedirte que predigas cosas en un futuro lejano. En lugar de preguntarte qué vas a hacer antes de fin de año, deberían decir qué vas a hacer en las próximas dos semanas. ¿Cómo vas a deleitar a mis clientes al final del próximo sprint?**

# ​Kanban y herramientas de planificación ágil

Existen muchas herramientas de planificación ágil. Pero una cosa que debes entender es

que una herramienta no te hará ágil, realmente tienes que tener una mentalidad ágil para ser ágil. **Las herramientas respaldarán su proceso ágil, pero primero debe tener el proceso**. He visto a muchas personas que no han recibido capacitación tratar de hacer que un tablero Kanban parezca un diagrama de Gantt. Son dos cosas diferentes, dos maneras diferentes de gestionar proyectos. Así que es muy importante que entiendas esto. Y hay muchísimas herramientas de planificación ágil. La mayoría hace lo mismo, pero algunos son un poco más complicados que otros.

Vamos a usar ZenHub para este curso. Entonces, ¿qué es ZenHub? Bueno, es un complemento para GitHub, esto permite que los desarrolladores permanezcan en GitHub y usan este complemento de ZenHub. Lo que hace ZenHub es que te permite agregar herramientas de administración de proyectos, un tablero Kanban, para administrar tus proyectos mientras permaneces en GitHub. La otra es que ZenHub es personalizable. comienza con un conjunto de columnas, llamadas canalizaciones, en tu tablero Kanban, pero puedes personalizarlas para que se adapten a lo que quieras. Puedes hacerlo tan complejo como simple. Preferiblemente simples.

Entonces, ¿por qué lo usamos? Bueno, una de las principales razones es porque usa problemas de GitHub. No es otra herramienta con otra cosa que tenga que editar. Uno de los

problemas que encuentro es que si utilizas demasiadas herramientas y un desarrollador tiene que ir a otro lugar para actualizar el estado, ese estado estará desactualizado el 100% del tiempo. Porque tan pronto como se actualicen, pasarán a hacer otra cosa, y eso no se reflejará en el estado. En cambio ZenHub Proporciona una manera fácil de entender lo que está pasando, mantiene una versión actualizada de la verdad, los desarrolladores trabajan en historias y en GitHub todo el día. Los problemas en GitHub, que son historias. Esos problemas son las cosas que abren y cierran, y no tienen que ir a otro lugar para actualizar el estado. El estado se mantiene directamente en su herramienta favorita, GitHub. Por lo tanto, permite a los desarrolladores ir a un solo lugar para hacer las cosas y no tener que actualizar varias herramientas.

¿Qué es un tablero Kanban? Hemos estado hablando de los tableros Kanban aquí. En pocas palabras, **hay cosas que debes hacer, cosas que estás haciendo y cosas que**

**ya has hecho.** No tiene por qué ser más complicado que eso. Además, **mueves las cosas por el tablero Kanban para mostrar el progreso que estás haciendo**, por lo que es una forma muy visual de ver exactamente dónde te encuentras en cualquier momento.

Así que hablemos de las canalizaciones o columnas predeterminadas que están en ZenHub.

Empieza con nuevos temas. **Los nuevos números son como tu bandeja de entrada. Cuando alguien abre un tema. El valor predeterminado es**

**ir a la columna de nuevos problemas.** Por eso no me gusta dejarlo ahí demasiado tiempo. Cuando refino mi cartera de pedidos y trabajo en mi tablero Kanban, normalmente hago primero una clasificación de las nuevas ediciones y digo **que es algo que acaba de llegar, ¿a dónde debería ir? Y lo paso a otro canal, o lo rechazo, pero no lo guardo en el nuevo número para que cuando llegue otro, sepa que debe ser muy nuevo porque lo acabo de limpiar como tu bandeja de entrada de correos electrónicos.**

**Luego está la nevera.** Esto es exclusivo de ZenHub, pero me gusta un poco. La nevera es de almacenamiento en frío. **Es donde pones las cosas que van a estar, en las que vas a trabajar a largo plazo.** Así que si no voy a conseguir algo durante un tiempo, lo tiraré a la

nevera. Así que no lo olvido. Pero tampoco estoy trabajando activamente en mis otros proyectos para que no se acumulen demasiado.

Después de la nevera está la **cartera de productos. La**

**cartera de productos es todo lo que siempre deseas hacer para siempre en tu producto y que aún no has hecho.**Correcto. No contiene las cosas que estás haciendo. **Contiene todas las cosas que no has incluido en un sprint y que quieres resolver en el futuro.** Y una vez más, para que la cartera de productos no sea demasiado desordenada, cosas que tienen un futuro a largo plazo, me gusta pasarme a la nevera.

Esto nos lleva al atraso de los sprints.

**La cartera de sprints es lo que vamos a hacer en las próximas dos semanas.** Así que he tomado cosas de la cartera de productos y las he trasladado a la cartera de sprints para

hacer un plan de sprints. Ahora los desarrolladores no tienen que preocuparse por ninguno de estos otros procesos, solo se concentran en la acumulación de sprints, porque ese es el trabajo que haremos en el próximo sprint.

**Luego, una vez que empezamos a trabajar en las cosas, las pasamos a estar en progreso.** Así que sé que cuando veo esas cartas, esas historias en la columna en progreso, sé que alguien está trabajando en ellas mientras se las asigna a sí mismo, aparece su pequeño avatar. Así puedo ver exactamente quién está trabajando en qué y en qué se está trabajando mientras hablamos.

Una vez que se completa el trabajo, **el desarrollador suele crear una solicitud de extracción para devolver su trabajo a la rama base.** Y para que se incluyan en el control de calidad de las revisiones, también puedes configurar GitHub y ZenHub para que creen automáticamente solicitudes de cambios en esa columna de control de calidad de las reseñas. Y para que los desarrolladores sepan, cuando vean que algo aparece allí, que tienen que ir a verlo y ayudar a otro desarrollador a revisar su historia y asegurarse de que lo entienden, cumple con los criterios necesarios para fusionarlo con el resto del código.

**Y, por último, está la columna «Listo». Cuando hayamos terminado, volvemos a fusionar nuestro código y, a continuación, movemos la historia a la columna Listo. Hecho significa que el desarrollador ha terminado. No significa que el**

**propietario del producto lo haya aceptado.** Eso es algo que ocurre durante la revisión del sprint. Pero sí significa que el desarrollador ha terminado con esa historia. Vuelven al trabajo pendiente del sprint, ponen la historia en marcha, se la asignan a sí mismos y siguen trabajando. Así que el flujo va de izquierda a derecha. Las nuevas historias aparecen por la izquierda y un incremento hecho termina saliendo por la derecha.

# CREANDO BUENAS HISTORIAS DE USUARIO

¿Qué es una historia de usuario?

En pocas palabras, **una historia de usuario representa una parte del valor empresarial que el equipo puede ofrecer en poco tiempo.** Solíamos llamarlos requisitos, pero las historias de usuario son mucho más. Por lo general, el requisito es: necesito esto, necesito aquello, pero **una historia de usuario incluye: ¿Para quién es? ¿Qué es lo que necesitan? Pero lo que es más importante, ¿por qué lo necesitan?** ¿Cuál es el valor empresarial que obtienen al obtener esta función o función?

Por lo tanto, las historias deben contener una buena descripción del valor empresarial: qué es. ¿Quién lo necesita? ¿Cuál es el valor que obtenemos de esa historia? Luego, me gusta **tener cualquier suposición y cualquier detalle documentado en la historia.** A veces, sabes que hay suposiciones, estás haciendo una historia y necesita algún tipo de persistencia.Y sabes que vas a usar una base de datos relacional, ponla ahí para darle al desarrollador una pista: «Oye, no busques bases de datos NoSQL, hemos decidido que vamos a usar una base de datos relacional. Así que cualquier cosa que sepas, cualquier suposición, aprovisionaremos algo en la nube, lo pondremos en la historia del usuario para que el desarrollador sepa cuáles son tus intenciones. Y lo más importante, **necesitas la definición de terminado, algunos criterios de aceptación. ¿Cómo puedo saber si esta historia está completa? ¿Cuándo esté hecho?** No querrás ver en la reseña del sprint que el propietario del producto dice: «Oye, eso no es lo que quería». Quieres documentar la definición de hecho para decir: «Bueno, esta es la definición en la que estuvimos de acuerdo. Esto es lo que hace la historia, y si no es lo que quieres, podemos escribir otra historia, pero hicimos lo que dijeron que debía hacer».

Entonces, la descripción de la historia, ¿qué pones en esa descripción? Una vez más, quieres documentar a una persona, solicitar alguna funcionalidad para conseguir algún objetivo o lograr algún objetivo, obtener algún valor empresarial. Por lo tanto, la forma en que lo hacemos es con este problema de sintaxis como rol. **¿Cuál es el papel? ¿Es el director de marketing? ¿Son los clientes? ¿El administrador del sistema?** ¿Para quién es esto? Y luego necesito algo de funcionalidad. ¿Qué es lo que necesitan? Este es el meollo de la historia del usuario, pero es muy importante, para obtener algún beneficio empresarial. ¿Cuál es el beneficio de ofrecer esa función? Cuando volvamos y prioricemos el trabajo atrasado, consideraremos el valor empresarial como uno de los criterios para preguntarnos: «¿Debería tener un volumen elevado de atrasos o uno más bajo?» **Así que es muy, muy importante, no solo tener lo que es, sino ¿cuál es el valor que obtenemos de ello?**

Luego están las suposiciones y los detalles. Así que cualquier cosa que sepas, claro, no tiene que ser en gran detalle. Pero cualquier cosa que sepas, como dije, ya sabes, si sabes que vas a usar una base de datos, algo de persistencia, ya sabes, simplemente ponla ahí. Cualquier cosa de la que hayas hablado mientras escribías la historia, ponla ahí. **Lo que estás intentando hacer es ayudar al desarrollador a entender lo que hay que hacer;** cualquier cosa que no fuera obvia para completar esta historia. Y luego están los **criterios de aceptación**, la definición de terminado. Esto es de una importancia crítica y crítica. Es muy importante entender qué hace que esta historia de usuario esté terminada, completa o terminada.

**Y para eso utilizamos una sintaxis llamada Gherkin,** que lleva el nombre del pepinillo. Además, Gherkin es una sintaxis que facilita tanto a las partes interesadas como a los clientes y a los desarrolladores la descripción de la definición de terminado. Describir el comportamiento del sistema. Y es más o menos así.

*Dadas algunas condiciones previas,* esto prepara la historia, dado que tengo cinco artículos en mi carrito de compras, Dado que ya he iniciado sesión en el sistema, sea lo que sea, se establece la historia. *¿Y entonces cuál es la acción que ocurre?* ¿Qué es lo que lo desencadena cuando esto ocurre, cuando agrego otro artículo a mi carrito o elimino un artículo de mi carrito? Y finalmente, *¿cuál es el resultado comprobable de eso?*Entonces debería ver esto, no debería ver aquello, debería tener seis artículos en mi carrito o debería ver cuatro artículos en mi carrito, ¿verdad? Hace que sea muy fácil de entender para todos. Dado que este es el caso base, cuando esto suceda, debería tener un resultado medible. Y si mido ese resultado, la historia está acabada. Así que veamos un ejemplo de historia.

***Como gerente de marketing,*** ahora sabemos cuál es su función. Esto no es para el cliente. Esto no es para el administrador del sistema. Esto es para el director de marketing, así que sabemos quién se va a beneficiar de ello. ***Necesito una lista de los nombres y correos electrónicos de los clientes.*** Ahora que sabemos cuál es la función, tienen que enumerar a

los clientes con sus correos electrónicos y, luego, el valor empresarial, ***para que pueda notificarles las promociones de marketing.***ahora puedo decir: «¿Qué importancia tiene notificar a los clientes sobre las promociones de marketing?» Y tal vez anteponga eso a otra historia que es menos importante. Así que ahora he capturado: ¿para quién es? ¿Qué es lo que necesito? ¿Y por qué lo necesito? ¿Qué valor obtengo de ello? Entonces me gusta hacer cualquier suposición. Estas suposiciones ayudan al desarrollador a crear el código que acompaña a la historia. ***Por lo tanto, una suposición es que mantenemos los***

***correos electrónicos de los clientes, tal vez no.*** Tal vez esta historia dependa de otra historia que realmente crea los correos electrónicos de los clientes. Quizás haya otra suposición: los clientes han optado por las promociones. Eso es bastante importante. Puede que alguien no haya pensado en eso. Oye, no podemos simplemente enviar correos electrónicos a todos nuestros clientes, tenemos que permitirles optar por recibir promociones.

Y luego tenemos los criterios de aceptación. ¿Cuál es la definición de terminado? ¿Cuál es el comportamiento que queremos que tenga esta historia? Y usaremos la sintaxis de Gherkin, ***dado que hay 100 clientes en la base de datos.*** Así que eso prepara el escenario. ***Y 90 han optado por las promociones por correo electrónico.*** Por lo tanto, esto es de vital importancia no solo para los 100, sino para los 100 que han optado por la promoción por correo electrónico, por lo que ahora los desarrolladores saben que tienen que filtrar el número de clientes de la base de datos según quién lo haya hecho.

***Cuando solicito una lista de correo electrónico de clientes,*** esa es la acción, eso es lo que ocurre, ***debería ver una lista de 90 correos electrónicos de clientes, no de 100 correos electrónicos de clientes,*** 90 correos electrónicos de clientes, porque solo 90 han optado por la promoción por correo electrónico. Como puedes ver que esto es algo que puedes regalar a una parte interesada, el gerente de marketing podría entender el comportamiento y decir: «Sí, ese es el comportamiento que quiero». El desarrollador puede leer esto y decir: «Sí, ese es el comportamiento que puedo ofrecer». Y que al final del sprint, siempre y cuando pidas clientes, solo recibas a los que se han suscrito. Y esa es la definición de hecho. Por lo tanto,

no hay discusiones sobre si la historia está hecha o no. Si tiene ese comportamiento, está hecho.

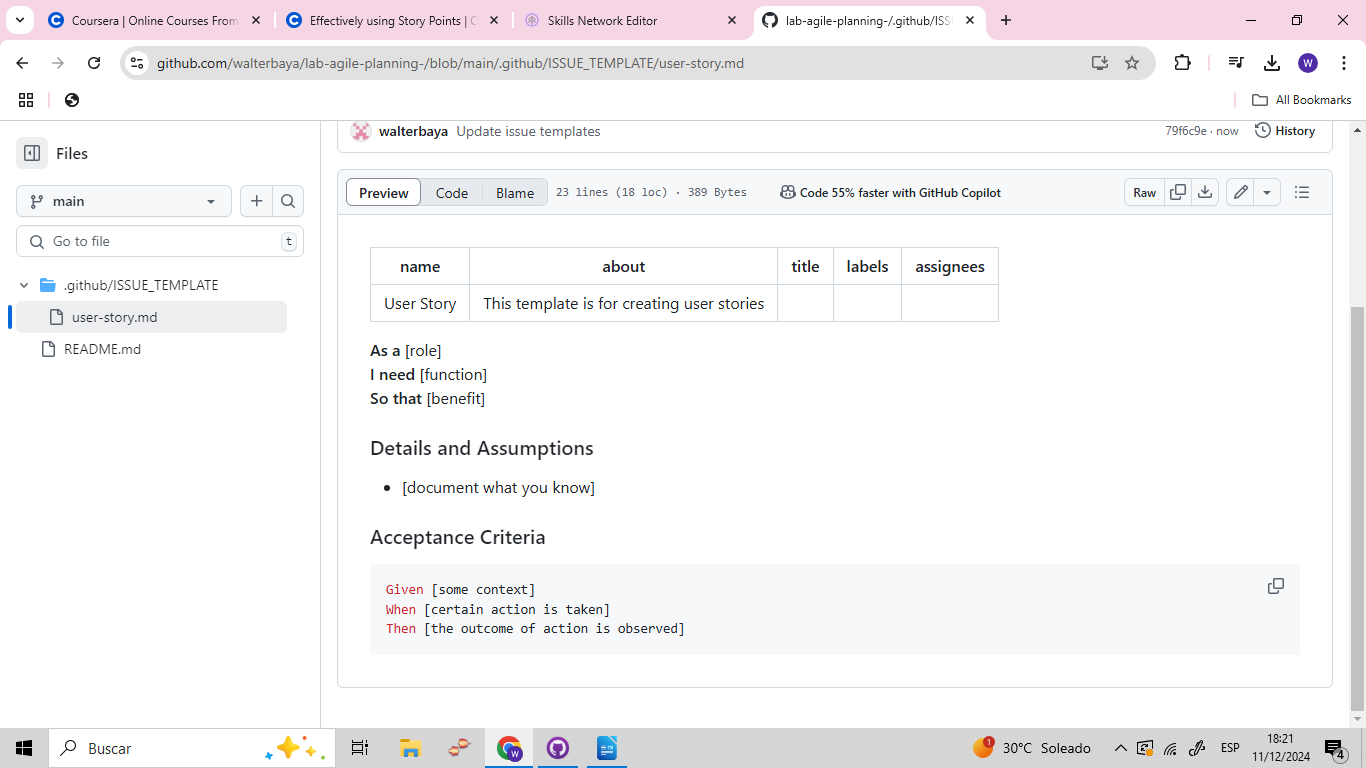
INVEST afirma ***«Las historias deben ser independientes, quiero poder clasificarlas en la lista de publicaciones pendientes, quiero poder moverlas de un lado a otro, quiero poder decir que esta es anterior a la otra».*** Ahora sé que no siempre pueden ser independientes, a veces hay dependencias, ¿verdad? En el ejemplo anterior, supusimos que había correos electrónicos de clientes en la base de datos y, si no los había, no se

trataba de una historia independiente, sino que dependía de la historia que pusiera los correos electrónicos en la base de datos. Pero, en su mayor parte, quieres intentar escribirlas para que sean independientes. ***También tienen que ser negociables***, los voy a poner en la lista de pedidos pendientes. Voy a clasificarlos más o menos. Tal vez voy a decir que deberíamos poner más valor, más funcionalidad o menos funcionalidad para negociar cuánto hay que hacer realmente. Por lo tanto, deberían ser negociables. No deberían estar demasiado

acoplados exactamente a lo que se necesita. ***Y luego tienen que ser valiosos,*** Necesito poder decir: «¿Qué tan valiosa es esta historia? ¿Qué valor obtiene el cliente de ello para

asegurarse de que realmente se trata de una historia de usuario?». Y no solo una historia de deuda técnica de que esto es algo que tengo que hacer y que el cliente nunca ve. ***Tiene que ser estimable, tengo que ser capaz de estimarlo.***¿Qué tan grande es? ¿Pequeña? ¿Medio? ¿Grande? Tiene que haber suficiente información ahí para poder decir: «Vaya, esta es una historia muy importante, teniendo en cuenta todo lo que hay aquí». Muy bien. Y no es solo una línea que parece simple, sino que se vuelve realmente difícil. Y debe ser pequeño. Quieres que sea algo en lo que alguien pueda trabajar en un sprint, ***por lo que tiene que ser más pequeño que un sprint.*** Y, por último, ***tiene que ser comprobable.*** Necesito poder comprobar si esa historia está lista. Quiero probar la definición de hecho para esa historia. Por lo tanto, las historias deben poder comprobarse.

¿Qué hay de las ideas realmente grandes? Bueno, las grandes ideas se llaman epopeyas ***(EPIC).*** Usamos epopeyas cada vez que tenemos una historia que es más grande que un solo sprint, porque la historia debe ser más pequeña que un sprint, por definición. Y cuando es más grande, se convierte en una gran idea que no podemos completar en un sprint, así que la convertimos en épica y luego añadimos historias más pequeñas que forman la epopeya a la épica. Así que una epopeya en la jerarquía es más alta que una historia. Las historias son consumidas por las epopeyas. Por lo general, cualquier historia que sea demasiado grande, la convertimos en épica, si no podemos estimarlo por sí solo, lo dividimos en cosas más pequeñas que podemos estimar. Entonces, ¿cuándo usamos una epopeya? Bueno, está claro que cuando una historia es demasiado grande para ponerla en un sprint, la convertimos en una epopeya. Por lo general, los artículos pendientes comienzan como grandes ideas, van a ser, van a ser muy grandes. y así comienzan como epopeyas. Y luego, a medida que refinamos el trabajo atrasado, hacemos que sean historias cada vez más pequeñas que podemos incluir en un sprint. Así que, para planificar los sprints, ***querrás dividir todas esas grandes historias en historias más pequeñas, dividir todas esas epopeyas en historias más pequeñas, cosas que normalmente vienen en los nuevos números, a veces son epopeyas,*** alguien ha pedido una función o una función. Suena bastante simple, pero va a llevar más de un sprint. Así que lo pones como una epopeya y, más adelante, lo desglosas en historias de usuario.



# Uso efectivo de los puntos de historia

¿Qué son los puntos de la historia?

Los puntos de historia son una métrica que se utiliza para estimar la dificultad que supone

entregar e implementar una historia de usuario. La clave es que se trata de una medida abstracta.

¿Qué incluye esa estimación? Hay un par de componentes.

El esfuerzo¿Qué tan difícil es hacerlo?

La complejidad¿Es muy complejo?¿Es muy fácil? Podrías esforzarte mucho por algo que es fácil de hacer, pero que es aburrido y laborioso.

La incertidumbre. ¿Has hecho esto antes? Cierto, tal vez si nunca lo has hecho antes,

calificas los puntos de la historia un poco más alto, porque hay más incertidumbre, realmente

no sabes lo complejo que es. Así que es una compensación entre lo que sabes y lo que no sabes.

Tradicionalmente, los humanos son muy, muy malos para estimar la hora de un reloj de pared. Recuerda, nos encanta ver pasar esas citas, nos comprometemos con algo y luego nos lo perdemos aprendemos que **lo único que lleva 30 minutos son 30 minutos,** **todo lo demás lleva más tiempo.**

Así que es muy, muy difícil para los humanos decir que esto va a llevar una hora, esto va a

llevar, esto va a llevar una semana. **Así que no hacemos eso.** Si no somos muy precisos al predecir de esa manera, **no predigamos de esa manera.** Así que, en su lugar, utilizamos tallas de camisetas, **los puntos de la historia son tallas de camisetas. Es una historia mediana, una historia pequeña, una historia grande, una historia extra grande.** Solo queremos dar un valor relativo de lo grande que es esta cosa, para poder compararla con otras cosas que están en el sprint actual. Pero porque no podemos sumar y obtener un número. Y queremos usar números para registrar la velocidad, que es la cantidad de puntos de la historia que podemos hacer en un sprint. En la mayoría de las herramientas, **utilizamos una secuencia de Fibonacci.** Nunca uso todos los números de la secuencia,

lo que recomiendo es que **cinco es mediano, y tres es pequeño y ocho es grande, tal vez**

**13 sea extra grande, y no voy mucho más allá,** Puedes usar toda la secuencia, está bien. Pero no seas demasiado granular. Recuerde, los humanos no son muy buenos para hacer este tipo de estimaciones. Por eso queremos que sea abstracto.

¿cómo lo hago abstracto? Bueno, tienes que estar de acuerdo en lo que significa medio, como los puntos de la historia son relativos, todos están de acuerdo en que está bien, esto es

medio. Así que decimos medios de cinco. Y esto es lo que es una historia de cinco pisos. Y luego podríamos tomar las otras historias y podríamos decir, evaluar a partir de ahí. ¿Es esta más pequeña que esa historia? ¿Es más grande que esa historia? Es del mismo tamaño que esa historia. Así es como aprendí los puntos de la historia.

Si te preguntan: ¿Qué altura tienen estos edificios? Puedes intentar contar los pisos. Cada piso mide unos 20 a 25 pies, un poco detrás de la servilleta, pero ¿y si te dijera: «Oye, esta historia, este edificio es un cinco? ¿Qué tan grande es el edificio de al lado, justo a la izquierda?» Tú dices: «Bueno, ese es probablemente un tres. Si ese es un cinco, ¿ese es un tres? Bueno, ¿qué tal el de la derecha que es un poco más alto? Quizá ese sea un ocho. Y quizás el más grande sea un 13». Son tamaños relativos, solo estás diciendo que si esta historia es mediana o pequeña, ¿la otra historia es más grande? ¿Es más grande? Y así es como aprendí a hacer y a racionalizar lo que son los puntos de la historia, es simplemente relativo. Todos están relacionados con algo que hace el equipo.

¿Cuál debería ser el tamaño de una historia? **La historia debería ser pequeña, relativamente pequeña, espero que sea algo que pueda hacer en un par de días,** no querrás que sean demasiado grandes. No querrás que se prolonguen una y otra vez. Así que quieres mantener esas historias pequeñas y estimar las cosas que, con suerte, puedes terminar en unos pocos días, las historias grandes deben dividirse en historias más pequeñas. Así que si tienes una historia, dices: «Este es un 21». Y tú decides «Vaya, eso es supergrande y no va a caber en un sprint, luego tiene que dividirse en historias más pequeñas». Y tal vez una parte se haga en este sprint, otra parte se haga en el siguiente sprint y quizás creemos una **epopeya** para registrar el hecho de que van a ser necesarios varios sprints.

Antipatrones

El primer antipatrón es evaluar las historias sobre la hora de los relojes de pared. Esto sucede todo el tiempo y se relaciona con nuestros antipatrones organizacionales. Así que tengo un maestro de Scrum, que no comenzó como un maestro de Scrum. Empezaron como gestores de proyectos y les encantan sus diagramas de Gantt, ¿y qué es un diagrama de Gantt? Es la hora del reloj ambulante, son las fechas de un calendario, por lo que solo pueden pensar en las fechas de un calendario. Y he visto a maestros de Scrum que fueron gerentes de proyectos decir al equipo: «Un uno es un día, un tres son tres días, un cinco son cinco días» o cinco horas o 12 horas o lo que sea. Eso es lo peor que se puede hacer. No se trata de la hora del reloj de pared, es solo del tamaño relativo, ¿verdad? Por lo tanto, una mediana puede llevarme de dos a tres días, tal vez cuatro días, una pequeña puede tardar un día menos de un día. Simplemente estás de acuerdo en un poco confuso al respecto. Pero nunca, nunca, **nunca le digas a tu equipo que esa cosa de Fibonacci secuencia esos puntos de la historia, son días o semanas.**

# Construyendo el Product Backlog (Cartera de productos)

¿Qué es una cartera de productos?

Una cartera de productos son todas las historias no implementadas, historias que no están en un sprint, o sea no se está trabajando en ellas, están esperando a que se trabaje en ellas.

Por lo general, volvemos, clasificamos esto en orden de clasificación. Ahora, si tienen una gran cartera de productos pendientes, **los que estén en la parte superior se clasificarán de forma un poco más precisa que los que están en la parte inferior,** y no pasa nada, solo tienes que clasificar **uno o dos sprints** siguientes y , luego, **el resto podría no estar relativamente clasificado**, pero **los primeros puestos de la cartera deberían clasificarse en orden de importancia.**

Por lo tanto, vamos a ver un ejemplo y, en este ejemplo, le daremos algunos ejemplos de requisitos para el servicio que estamos creando.

¿Qué estamos construyendo? Vamos a crear un contador de visitas, va a ser un servicio que cuente las cosas, así que podría ponerle un nombre al contador y, cada vez que lo incremente 1, 2, 3, 4, y luego decir: «¿Cuántos de esos tengo?» Y me devuelve el número 4. Queremos permitir varios contadores con el tiempo, por lo que uno de los requisitos que tenemos es tener varios contadores para poder contar varias cosas, además, los contadores deben persistir después de los reinicios así que esto simplemente no puede ser un contador de memoria, va a necesitar algún tipo de persistencia, algún tipo de base de datos para

que cuando el sistema se reinicie y vuelva a funcionar recuerde todos los contadores.

Y luego tenemos un requisito que dice: «Sabes, probablemente necesite poder restablecer un contador de vez en cuando para volver a cero», así podría empezar a contar esas cosas de nuevo y, por lo tanto, restablecer un contador es otro requisito que nos ha impuesto el cliente.

Si echamos un vistazo al tablero kanban de ZenHub, verás estas columnas, que se llaman Canalizaciones para nuevos problemas, Icebox, Product Backlog, Sprint Backlog, In Progress, Q&A, Review/Q&A y Listo.

Hablemos sobre la creación de nuevas historias. Tenemos nuevos números y está vacío, por lo que vamos a empezar a crear la primera historia: Se necesita un servicio para contar cosas. Así que volvamos a nuestra plantilla de historias, recordemos que, como función, necesito alguna función para obtener algún beneficio. Apliquemos esto a la primera historia, necesitamos un servicio para contar cosas, bueno, tal vez sea para el usuario. ***Como usuario, necesito un servicio que tenga un contador para poder hacer un seguimiento de cuántas veces se ha hecho algo.***

Veamos el siguiente requisito: ***Como usuario, necesito tener varios contadores para poder realizar un seguimiento de varios recuentos a la vez.***«¿Qué tan valioso es que

puedan hacer un seguimiento de varios a la vez? Tal vez al principio no, tal vez hagamos que uno funcione», así que esta es toda la información que necesita saber para clasificar este

atraso más adelante. Así que eso se convierte en una historia en los nuevos números.

Veamos el siguiente requisito: ***Necesito un servicio que mantenga el último recuento conocido para que los usuarios no pierdan la cuenta de sus recuentos una vez que se reinicie el servicio y podamos evaluar su importancia.***

veamos el cuarto requisito: ***los contadores se pueden restablecer. Así que lo revisamos de nuevo, este podría ser para un administrador del sistema.***

Ahora pasemos a priorizar el trabajo atrasado. Ahora que los tengo en números nuevos.

Quiero decidir si debo ponerlos en la nevera o ponerlos en la cartera de productos pendientes.

¿Qué hago con estos?

Recuerda que dije que me gusta usar los números nuevos como bandeja de entrada, así

que ahora que tenemos estas cuatro cosas en la bandeja de entrada, déjame hacer algo con ellas. Así que cogeré la primera y diré: «¿Sabes qué? Eso es algo fundamental». ¿Verdad?

Teníamos que poner en marcha el servicio, así que eso iba a ocupar el primer lugar de la cartera de productos pendientes. Luego miro el siguiente y lo leo, claro, necesito tener varios contadores. Y podría decir: «¿Sabes qué? Múltiples contadores, nos ocuparemos de eso más adelante. Eso es algo que vamos a hacer en el futuro, no lo voy a estropear ahora mismo». Luego veo que necesito que el servicio persista en los reinicios. Digo: «Probablemente sea algo justo después de que lo haga funcionar. Probablemente quiera crear una base de datos». Así que cuando hago mi producto mínimo viable, puede que no tenga ninguna persistencia detrás de él. Y luego tendrá persistencia. Y luego, como administrador, miro el último de los nuevos números y digo: «Sí, me gustaría poder restablecer el contador justo después de mantenerlo en la base de datos. Para poder empezar a contar de nuevo. Así que

lo que hice fue tomar estos nuevos requisitos que me envió el cliente, que eran solo una línea, y los escribí en la sintaxis de la historia. Utilizo la plantilla de la historia para decir: «Como usuario, necesito, cierto, un rol que «necesito» para obtener algo de valor y luego los paso al trabajo atrasado en algún tipo de orden de prioridad en el que creo que me gustaría ejecutarlos.

# Refinar el Product Backlog

Entonces, si volvemos a los pasos en el proceso Scrum, esta vez sólo estamos viendo el refinamiento del backlog.

¿Qué hacemos durante la reunión de refinamiento del backlog?

Así que estamos listos para tener nuestra reunión de refinamiento del backlog y hacer el refinamiento del backlog, este es el proceso de tomar el backlog y clasificarlo en orden de prioridad para que las cosas más importantes estén en la parte superior del backlog.

Y posteriormente desglosamos las historias más grandes en historias más pequeñas, además nos aseguramos de que todas las historias que están cerca de la parte superior son lo suficientemente pequeñas como para caber en un sprint y por último, nos aseguramos de que las historias cerca de la parte superior tienen suficiente detalle como para que un desarrollador

pueda simplemente tomarlas y empezar a trabajar en ellas una vez que las pongamos en un plan de sprint.

Ahora la reunión de refinamiento del backlog en sí...

¿Quién debería asistir?

Bueno, el propietario del producto es la persona clave (Product Owner), porque es quien debería estar escribir las historias, es el que tiene la visión.

El Scrum Master es otra persona a la que invitaría a la reunión de refinamiento del backlog

porque va a ayudar al propietario del producto a refinar ese backlog.

Y finalmente el equipo de desarrollo es opcional, por lo general, podría ser buena idea tomar un líder de desarrollo o arquitecto y tenerlos ahí para responder a preguntas técnicas.

¿Qué tan difícil es hacer esto? ¿Dónde decís que tenemos que clasificar esto?

Porque a veces no es una decisión de negocio, hay dependencias técnicas que necesitamos saber.

Entonces, ¿cuál es el objetivo? tener un backlog clasificadopor orden de importancia.

Entonces, lo que está arriba es lo más importante y después asegurarse de que la historia contiene suficiente información para que los desarrolladores puedan entonces tomar esa historia y ponerla en un sprint, no vamos a añadir estos detalles durante la reunión de planificación del sprint, sino que vamos a describir las historias y hacer que los desarrolladores las entiendan, pero no querrá escribir mucho en esa reunión, cuanto más pueda hacer en esta reunión de refinamiento del backlog para que la historia esté lista para el sprint, mejor le irá y más rápido irá su planificación.

Entonces, si miramos nuestro tablero Kanban, tenemos por ejemplo una historia en la nevera, tres historias en el backlog del producto,así que no tenemos que preocuparnos de nada más que de esas tres columnas por ahora, tenemos algunas cosas en la nevera y algunas cosas en el backlog pero, los clientes por ejemplo durante este proceso estuvieron escribiendo más cuestiones, eso es algo común. Así que han añadido un par de nuevas desde que tuvimos la última reunión de refinamiento del backlog, **"necesito la capacidad de eliminar un contador"** y **"desplegar el servicio en una nube"**, alguien quiere desplegarlo en la nube.

Por eso se llaman nuevas cuestiones (new issues), llegan cosas nuevas, nuevos requisitos, y por eso una de las primeras cosas que se pueden en la reunión de refinamiento del backlog es ocuparme de eso, queremos dejar esta (new issues) vacía, asegurándonos de que al final de la reunión de backlog no hay nada en la columna de nuevas cuestiones(new issues) porque

puede que haya algo realmente importante que deba hacerse en el siguiente sprint.

Hay que mirar las nuevas cuestiones y comprender: ¿Hay algo que quizá queramos poner justo al principio del backlog o no? ¿O hay cosas que pondremos y a las que llegaremos con el tiempo? En ese último caso las ponemos en la nevera.

Queremos que la columna de new issues se vacíe en cada reunión de refinamiento del backlog, por lo tanto tomamos la historia, algunas cuestiones nuevas, y decidimos:

¿Las movemos al backlog del producto? ¿Es algo que vamos a hacer tal vez en el próximo sprint, en un sprint a partir de ahora, en dos sprints a partir de ahora? Dependiendo de lo profundo que sea su backlog de producto, dependiendo de cuál sea su criterio, para poner algo en la nevera en lugar de ponerlo en el backlog de producto.

Si es algo en lo que no voy a trabajar durante un tiempo, entonces tal vez lo ponga en la nevera y diga: "Sabes, es una gran idea, pero no es nuestra máxima prioridad. La pondré en la nevera y la trataremos en un sprint futuro", o también puede pasar que digamos: "Sabes, esto no es hacia donde va este producto", y entonces la rechazamos sin más, porque es algo que nunca vamos a hacer, por alguna razón, no sentimos que vaya en la dirección del proyecto

Asi que entonces con las new issues hacemos lo siguiente:

Miramos la primera "necesitamos la capacidad de eliminar el contador".

Yo digo: "Sabes, eso es algo que podríamos hacer eventualmente pero ni siquiera tenemos contadores múltiples todavía." No quiero eliminar el único contador, así que lo muevo a la nevera, la segunda cosa era "desplegar servidores en la nube" y digo: "Sabes, quizás deberíamos hacer eso más pronto que tarde. De hecho, quiero hacerlo antes de que tengamos la capacidad de reiniciar un contador. Ya sabes, una vez que tengamos esto desplegado,

y tengamos algo de persistencia, vamos a ponerlo en la nube, ¿verdad?"

Así que, ahora he tomado una decisión, la pongo en el backlog del producto y la otra en la nevera.

El flujo de trabajo para el refinamiento del backlog.

El propietario del producto está tomando estas historias y las está

poniendo en el backlog en orden de importancia, tomando cosas que pensaban que no eran importantes y que podrían llegar a serlo y cambiándolas de sitio

¿Haria estimaciones? Así que, guardamos las estimaciones hasta que haces la planificación del sprint porque esas son las estimaciones reales del plan pero, ya sabes, está bien saber cuántos puntos de historia hay en mi backlog. Incluso si se trata de una estimación bruta, ya sabes un orden aproximado de magnitud.

Así que, a veces vamos a asignar **puntos de historia** en los backlogs.

Vemos cómo de grande es el backlog, cuántos puntos de historia hay allí, y luego cuando los ponemos en el plan, los refinamos. Conseguimos que el equipo esté de acuerdo, ¿era eso realmente un medio? O podrían decir, ya sabes, eso es realmente un grande pero al menos tengo alguna estimación allí.

Entonces, es posible que desee hacer eso.

Luego los grandes elementos vagos, tienen que ser divididos, ¿verdad? Los elementos vagos tienen que ser aclarados. Los elementos grandes tienen que ser divididos en elementos más pequeños.

El objetivo aquí es hacer las historias, lo que yo llamo sprint-ready. Están listos para poner en el sprint y no tengo que pasar mucho tiempo durante la reunión de planificación del sprint añadiendo cualquier detalle. Algo puede surgir en la reunión donde un desarrollador puede hacer una pregunta y usted documenta esa pregunta en las suposiciones, pero en su mayor parte de lo que sabemos ahora, en la reunión de refinamiento del backlog, quiero asegurarme de que estos están listos para el sprint.

Así que, vamos a utilizar esta plantilla de historia completa, hablamos sobre el principio de la misma. **como algún rol, necesito algunas funciones para que obtenga algún valor de negocio.**

**Pero ahora queremos volver atrás y queremos añadir criterios de aceptación, cualquier cosa que podamos saber y queremos añadir la definición de hecho utilizando la sintaxis Gherkin.**

**Cuando, ya sabe, dada alguna precondición cuando ocurre algún evento entonces veo algún resultado medible.**

Así que, vamos a hacer eso. Así que, tomemos este primer elemento en el backlog del producto en la parte superior, nos fijamos en él, dice como historia de usuario necesito un servicio que tenga un contador para que pueda hacer un seguimiento de cuántas veces se ha hecho algo.

Hay un par de anotaciones en el lateral cuando abra ZenHub en la página web, tendrá que actualmente no está asignado, no tiene etiquetas, no tiene hitos, no tiene estimaciones.

Así empiezan todas las historias, por ahora ignoramos esto y vamos a embellecer la historia un poco más ¿cuáles son algunas de las suposiciones? Las suposiciones necesitan una forma de incrementar el contador, quizás cuando tengamos este contador, tengo que ser capaz de incrementarlo y quizás quiero hacerlo en un solo paso. Tal vez no quiero hacer un get,

y luego un read, y luego un increment, y luego un put. Por ahí solamente quiero tener una función de incremento que diga, "Oye, en una transacción atómica, incrementa ese contador y necesito una forma de obtener el valor actual, entonces necesitamos una forma de incrementarlo, una forma de obtener el valor actual, debería ser bastante bueno para ir.

Posteriormente, escribimos los criterios de aceptación, ***como incrementé el contador a 2,***

***cuando hago una llamada para obtener el contador entonces debería devolver 2 es el valor del contador,***

Puede pensar en estos escenarios que dan al desarrollador los criterios de aceptación que dicen, "Este es el comportamiento que queremos que tenga el contador."