5. Developer: Desarrollo ágil de productos

Los Developers tienen la responsabilidad vital de dar vida a los incrementos generados en cada Sprint, construyendo los elementos que eventualmente conformarán el producto final. Es imperativo que posean las habilidades y competencias necesarias para realizar este trabajo de manera efectiva. En este capítulo, nos sumergiremos en el mundo de los Developers, destacando las actividades, habilidades y responsabilidades esenciales que delinean su función dentro del equipo Scrum, complementando lo ya discutido en el modelo core de Scrum. Puntos clave de este capítulo: - Identificar las responsabilidades centrales de los Developers en el proceso Scrum. - Explorar las habilidades y competencias que los Developers deben cultivar. - Analizar las actividades diarias y estrategias que facilitan un desarrollo de producto exitoso. - Comprender cómo los Developers colaboran para alcanzar los objetivos de los Sprint y contribuir al producto final.

- 5.1 - Desarrollo ágil de producto
- 5.2 - Los Developers y los eventos
- 5.3 - Técnicas y herramientas útiles para los Developers

5.1 - Desarrollo ágil de producto

En el dinámico mundo actual, la figura del Developer se transforma constantemente, abrazando roles más estratégicos y creativos. En este escenario, el desarrollo ágil de productos emerge como un enfoque que no sólo refina tus habilidades técnicas, sino que potencia tu capacidad para contribuir activamente en la creación de soluciones innovadoras.

En este módulo, te invitamos a explorar cómo, como developer, puedes ser un actor clave en este proceso. Desde participar activamente en la definición de la arquitectura del producto, hasta adoptar prácticas de documentación eficientes y ser un protagonista en la implementación de pruebas continuas, tu responsabilidad es esencial para el éxito del producto.

Prepárate para sumergirte en un camino donde tu conocimiento técnico se une con una visión estratégica, permitiéndote participar de manera significativa en la entrega continua de valor. ¡Adelante!

5.1.1 - Autogestión y Gestión del Tiempo: Claves en un buen Scrum Developer

Dentro de un equipo Scrum, se espera que los Developers sean proactivos y asuman un papel central en la autogestión de su tiempo y esfuerzos. Esto no solo fomenta la responsabilidad individual, sino que también promueve una cultura de colaboración y respeto mutuo.

> **Caso práctico:** Imagina un escenario donde un Developer, Carlos, se da cuenta de que una tarea específica está consumiendo más tiempo del esperado, lo cual puede afectar la entrega oportuna del incremento planeado para el Sprint. En lugar de esperar a ser dirigido, toma la iniciativa de comunicar la situación al equipo, proponiendo una sesión de brain-storming para encontrar una solución eficiente. Gracias a esta proactividad, el equipo pudo reajustar y repriorizar las tareas, garantizando la entrega exitosa del incremento.

Tips para una Efectiva Autogestión y Gestión del Tiempo

- **Establece Metas Claras:** Asegúrate de entender bien el objetivo de cada Sprint y cómo contribuyen tus tareas a alcanzarlo.
- **Comunicación Continua:** Fomenta la comunicación constante con tu equipo para identificar y abordar problemas a tiempo.
- **Priorización Efectiva:** Aprende a priorizar tareas basándote en su valor y urgencia, esto ayudará a evitar el desperdicio de tiempo en elementos menos críticos.
- **Feedback Continuo:** Promueve una cultura de feedback constante para facilitar la mejora continua y la adaptación ágil.

> La autogestión, combinada con una efectiva gestión del tiempo y priorización, no solo facilita la entrega oportuna de productos de alta calidad, sino que también contribuye a un ambiente de trabajo más armonioso y productivo.

5.1.2 - Consideraciones clave para el desarrollo ágil de productos

En el trayecto de construcción de productos ágiles, es imperativo tener en cuenta varios factores esenciales que influenciarán el éxito del producto final. Aquí te presentamos tres preguntas centrales que todo Developer debería tener en mente:

- ¿Por qué debería definirse la arquitectura del producto?
- ¿Qué debo considerar respecto a la documentación del producto?
- ¿Qué implican exactamente las pruebas del producto?

¿Por qué debería definirse la arquitectura de producto

> La "Arquitectura del Producto" puede definirse como el esquema estructural que ilustra cómo se organizarán y interactuarán las distintas partes o componentes de un producto. Esta estructura no solo comprende la organización de elementos técnicos, sino también la definición de las relaciones y dependencias que existen entre ellos. El establecimiento de una arquitectura sólida desde el principio, ayuda a prever desafíos y facilita el proceso de incorporación de cambios o ajustes en etapas posteriores del desarrollo. En pocas palabras, funciona como el mapa guía que orienta el desarrollo del producto de una manera organizada y estratégica, asegurando que todas las piezas encajen de manera óptima para cumplir con los objetivos propuestos.

Definir la arquitectura del producto no es solo una cuestión de organización, sino que establece una hoja de ruta clara que facilita la identificación y solución de posibles problemas antes de que surjan. Permite una colaboración fluida entre los miembros del equipo y garantiza que todos trabajen hacia una visión unificada, optimizando los recursos y el tiempo disponible.

> Antes de iniciar la construcción del producto es importante identificar la relación entre todos los componentes que harán parte del mismo. Definir una correcta arquitectura permitirá que los equipos de trabajo sepan como organizar sus entregables, se facilitará la integración y la mejora continua.

Es indispensable que, antes de iniciar con el desarrollo se defina el diseño técnico del producto o del incremento de producto a desarrollar. Esta actividad podría hacerse de forma iterativa antes de desarrollar los incrementos en los distintos Sprint, o de forma global al inicio del desarrollo.

Para garantizar que la arquitectura de producto cumple exactamente con los requerimientos y necesidades del cliente se debe hacer en conjunto entre los Developers y el Product Owner.

Tip para el Developer

Involúcrate activamente en las discusiones sobre la arquitectura del producto. Tu conocimiento técnico es vital para diseñar una arquitectura que no solo cumpla con los requisitos del cliente, sino que también sea sostenible y escalable a largo plazo. No dudes en compartir tus ideas y sugerencias.

> La arquitectura está estrechamente relacionada con el objetivo del producto (Product Goal). El diseño arquitectónico del producto o sistema sienta las bases para lograr los objetivos y metas del producto, asegurando que se puedan implementar las características y funcionalidades deseadas de manera efectiva.

Ejemplo de arquitectura 1: Construyendo un vehiculo electrico

Al embarcarse en la construcción de un vehículo eléctrico, es vital analizar cuidadosamente los siguientes aspectos cruciales y entender cómo interactuarán entre sí y quién será responsable de su desarrollo, algunos ejemplos que debes considerar al definir la arquitectura son:

- **Sistema de Propulsión:** ¿Motor eléctrico de eje central o trasero? ¿batería de iones de litio o plomo? (en el piso del auto).
- **Sistema de Frenado:** ¿Frenos regenerativos para recuperar energía o solo ABS tradicional?.
- **Sistema de Navegación:** ¿GPS integrado con asistentes de conducción autónoma?.
- **Interfaz de Usuario:** ¿Tablero digital o analógico? ¿Controles táctiles o de voz?.
- **Conectividad:** ¿Bluetooth, Wi-Fi? ¿Conectividad celular para actualizaciones y telemetría?.

Ejemplo de arquitectura 2: Construyendo un software LMS

Una empresa de desarrollo de software planea construir una aplicación para realizar cursos virtuales. Al ser un proyecto de software, en la definición de arquitectura se deberán tener en cuenta los siguientes elementos:

- **Front-end:** ¿Será HTML, CSS, JavaScript para la interfaz de usuario?.
- **Back-end:** ¿Se usará un framework de base Ej. Laravel PHP para la lógica del servidor??
- **Base de datos:** ¿Qué motor de base de datos? ¿Relacional o no relacional?
- **APIs:** ¿JSON Web Token para autenticación? RESTful APIs para interactuar con servicios externos, como plataformas de redes sociales o bases de datos de terceros.
- **Infraestructura:** ¿Qué proveedor? ¿Qué servicios? ¿Cómo será la red? Servidores en la nube con escalabilidad automática.

Ejemplo de arquitectura 3: Construcción: Diseño de una Casa Moderna

- **Cimientos:** ¿Será hormigón armado o pilotes?¿Madera?
- **Estructura Principal:** ¿Marco de acero o madera para las paredes y techos?
- **Sistema de Techo:** ¿Tejas de asfalto o metal?
- **Sistema Eléctrico:** ¿Cableado empotrado o visible? (puntos de luz, enchufes y switches).
- **Plomería:** ¿Tubos de PVC o cobre? (suministro de agua y desagüe).
- **Espacio Habitacional:** Distribución de habitaciones, baños, cocina, etc.
- **Sistema de Climatización:** ¿Aire acondicionado centralizado o radiadores?
- **Energía eléctrica:** ¿Se usará energía alternativa en el futuro? ¿Se van a instalar múltiples medidores?

¿Qué debo considerar respecto a la documentación del producto?

La documentación es una herramienta vital en el desarrollo de productos, ya que facilita la trazabilidad y comprensión de las funcionalidades y características del producto. Es importante considerar no solo qué documentar, sino cómo hacerlo de una manera que sea accesible y útil para todos los involucrados. Además, incentivar la documentación proactiva puede ser un método efectivo para garantizar que se mantenga actualizada y relevante.

La documentación es una herramienta indispensable que, aunque a veces despreciada, puede ser un salvavidas en el ciclo de vida del desarrollo de productos. Vamos a desglosar cómo puede realizarse de manera eficiente y sin dolor:

- Muchos trabajadores odian documentar.
- Se pueden definir incentivos por mantener una buena documentación (no necesariamente económicos).
- Definir las reglas detalladas de qué se debe documentar y cómo.
- Se puede crear una Wiki y allí almacenar todo el contenido de forma organizada.

- Se puede incluir la documentación como parte de los criterios de la Definición de Terminado (DoD).

Tip para el Developer

Considera la documentación como una herramienta que te ayudará, no como una tarea tediosa. Al mantener una documentación precisa y bien organizada, facilitas la gestión de la calidad y la colaboración efectiva en tu equipo.

> **Caso práctico** Pedro, uno de los developers, toma la iniciativa de crear una Wiki para almacenar toda la documentación del proyecto. A medida que el proyecto avanza, el equipo la mantiene actualizada, facilitando la incorporación de nuevos miembros al equipo y sirviendo como una valiosa referencia para revisar decisiones pasadas.

> **Caso Práctico: Motivando la Documentación** Laura, una developer experimentada, notó que el equipo evitaba la documentación, lo que causaba confusión y retrasos. Para remediarlo, instauró junto al Scrum Master sesiones dedicadas a la documentación en cada sprint, complementándolo con un sistema de recompensas para el miembro que más contribuyera. Este enfoque transformó la documentación de una tarea despreciada a una actividad valorada que mejoraba la eficiencia del equipo.

¿Qué implican exactamente las pruebas del producto?

Las pruebas no son una fase aislada en el proceso ágil, sino una integración continua que asegura la calidad y el valor del producto en cada etapa del desarrollo. Los Developers deben adoptar una postura proactiva hacia la realización de pruebas, asegurando que cada incremento cumpla con los estándares de calidad establecidos y contribuyendo al éxito a largo plazo del producto.

> **Una prueba es la forma de comprobar el correcto funcionamiento del producto o uno de sus incrementos.**

Tip para el Developer

Adopta un enfoque proactivo hacia las pruebas, viéndolo como una oportunidad para mejorar continuamente el producto, en lugar de una tarea de última hora. Incorpora pruebas regulares en tu flujo de trabajo para garantizar que cada incremento del producto sea de alta calidad.

Tip para el Developer

Recuerda que las pruebas no son una tarea aislada o exclusiva de un equipo de pruebas; eres co-responsable de la calidad de los entregables. Aprovecha tu conocimiento y habilidades para integrar pruebas continuas. Considera que las pruebas son exclusivas para proyectos de software, sino en todas las facetas del desarrollo de productos. Esto asegurará que el producto no solo funcione como se espera, sino que también cumpla con los estándares de calidad desde el inicio.

- En agilidad no se consideran a las pruebas como una fase separada, sino como parte integral del desarrollo de producto.
- En metodologías tradicionales, es una fase más dentro del desarrollo cascada o waterfall, lo cual no siempre resulta eficiente.
- Desde la agilidad se prueba continuamente, porque es una de las claves para mantener la calidad del producto.

> **Caso práctico: Cuando no se hacen pruebas o no son suficientes** En un equipo, inicialmente se pasaba por alto la fase de pruebas o se realizaban de manera muy superficial, lo que llevaba a múltiples errores en las versiones finales del producto. Pero, al establecer definiciones de terminado (DoD) centradas en procesos específicos, se creó una pauta clara para realizar pruebas exhaustivas en cada incremento. Esto no solo redujo los errores sino que también facilitó que cada miembro del equipo pudiera llevar a cabo pruebas efectivas de sus propios entregables, mejorando así la calidad general del producto.

5.2 - Los Developers y los eventos

El desarrollo de un producto exitoso es una jornada colaborativa, y los Developers están en el corazón de este viaje. En el contexto de Scrum, su participación activa en los diversos eventos es crucial, no solo por su papel en la construcción del producto sino también por su contribución a la planificación, adaptación y mejora continua que Scrum fomenta.

En esta sección, desglosaremos cómo los Developers se involucran y contribuyen en cada uno de los eventos de Scrum. Desde la planificación del Sprint, donde establecen las bases del trabajo a realizar, hasta la retrospectiva del Sprint, donde reflexionan sobre el progreso y buscan formas de mejorar. A través de su compromiso y colaboración en cada fase, garantizan que el producto en desarrollo esté alineado con los objetivos establecidos y se adapte eficazmente a las cambiantes demandas y expectativas.

□□ Tip para los Developers

Durante los eventos de Scrum, toma un papel activo en la comunicación y colaboración. No sólo compartas el progreso técnico, sino también busca feedback y aporta ideas para potenciar el valor del producto en desarrollo. La colaboración efectiva entre Developers y otras responsabilidades en el equipo Scrum puede ser la clave para la creación de soluciones innovadoras y de alto impacto.

Los Developers en la Planificación del Sprint

- Participan del evento, identificando los 3 objetivos: ¿por qué este Sprint es valioso?, ¿qué se puede terminar (done) en este Sprint?, ¿cómo se realizará el trabajo elegido?
- Indagan detalladamente sobre el trabajo que se pueda terminar este Sprint
- Aclaran los aspectos técnicos a las partes interesadas para lograr un entendimiento compartido del trabajo del Sprint
- Estiman el trabajo seleccionado para el Sprint de acuerdo a su experiencia y conocimiento técnico.

> **Caso práctico:** Durante la planificación del Sprint, el equipo se reúne para discutir las metas del próximo Sprint. Daniela, una Developer, sugiere que se enfoquen en un conjunto particular de historias de usuario que, según su análisis, pueden agregar un valor significativo al producto. Aprovecha este momento para clarificar dudas técnicas con el Product Owner y garantiza que todos los elementos están bien definidos. Juntos, como equipo, realizan una sesión de estimación basada en su experiencia y conocimientos técnicos,

> asegurando una comprensión compartida de la carga de trabajo y de los objetivos del Sprint.

Tip para los Developers

En la planificación del Sprint, no te limites a hacer caso a lo propuesto por el Product Owner, adopta una postura proactiva en identificar las historias de usuario que podrían agregar más valor al producto. Utiliza este tiempo no sólo para entender el "qué" sino también el "por qué" detrás de cada elemento del Product Backlog, alineándolos estratégicamente con el Objetivo del Sprint (Sprint Goal) y el Objetivo de Producto (Product Goal) a largo plazo.

Los Developers en el Daily

- Llevan a cabo el evento, cumplen con la agenda y respetan su duración
- Reportan los impedimentos que puedan encontrar en el trabajo diario del Sprint
- Realizan el Daily a la misma hora y en el mismo lugar
- Informan al resto del equipo sobre su progreso diario

> **Consejo:** Recuerda que el objetivo del Daily no es supervisar sino facilitar la colaboración y el flujo de información. Anima a los Developers a que se apropien de este espacio, compartiendo no sólo avances sino también aprendizajes y retos, promoviendo una cultura de transparencia y confianza mutua, y evitando la percepción de que se trata de una herramienta de supervisión.

Tip para los Developers

Evita la Monotonía: Para prevenir que los Daily se conviertan en encuentros monótonos, varía la estructura ocasionalmente. Por ejemplo, puedes introducir una breve sesión de brainstorming para solucionar un problema específico o facilitar una rápida actividad de team-building que fomente la colaboración y la energía positiva en el equipo.

Los Developers durante el desarrollo del Sprint

- Desarrollan los elementos que fueron identificados en la Planificación del Sprint
- Cumplen con los criterios de aceptación y definición de terminado al desarrollar su trabajo
- Participan del refinamiento de los elementos del Product Backlog a partir de las sesiones que gestione el Product Owner con el cliente y partes interesadas
- Se coordinan para abordar las dependencias que se puedan presentar en el Sprint

Consejos para Developers durante el Desarrollo de un Sprint

- **Colaboración Continua:** Fomenta una cultura de colaboración y apoyo mutuo. No dudes en pedir ayuda o brindar asistencia a tus compañeros de equipo si ves que pueden beneficiarse de tu experiencia o conocimientos.
- **Mantenimiento de la Calidad:** Dedica tiempo a revisar y mejorar la calidad de los incrementos en los que trabajas. Aprovecha las prácticas como la revisión con tus colegas

para asegurar que el producto mantenga un estándar de calidad elevado.
- **Gestión Efectiva del Tiempo:** Desarrolla una estrategia para gestionar tu tiempo eficientemente durante el Sprint. Esto podría incluir la utilización de técnicas como la técnica Pomodoro para mantener un enfoque sostenible y evitar el agotamiento.

> **Caso Práctico:** Durante el desarrollo del Sprint, Laura, una Developer, se da cuenta de que una funcionalidad que están implementando podría beneficiarse de una integración con una herramienta externa, lo que potencialmente podría ahorrar tiempo y esfuerzo en el futuro. **Acción:** Laura discute esta idea con el resto del equipo durante una sesión de trabajo. Juntos, evalúan los pros y los contras de la implementación de esta integración en este momento, teniendo en cuenta el Objetivo del Sprint y las capacidades del equipo. **Resultado:** Después de una discusión constructiva, deciden adaptar el plan de Sprint para incorporar esta mejora, ajustando las historias de usuario y las tareas asociadas según sea necesario. Esto les permite entregar un incremento de mayor valor al final del Sprint, demostrando una adaptabilidad y una perspectiva centrada en el valor.

Los Developers en la Revisión del Sprint

- Realizan la presentación del incremento de producto logrado durante el Sprint
- Discuten sobre el progreso respecto al Objetivo de Producto
- Discuten el Product Backlog tal y como está
- Generan compromisos frente al trabajo que no se terminó

Consejos para Developers durante la Revisión del Sprint

- **Preparación de Demostraciones:** Asegúrate de que las demostraciones estén bien preparadas y enfocadas en mostrar el valor añadido durante el Sprint, en vez de solo mostrar características técnicas (a veces los interesados no piensan tanto en las características técnicas si no es como su problema o necesidad es resuelta).
- **Feedback Constructivo:** Fomenta una cultura de feedback constructivo, donde se valoren las opiniones de todos los participantes, incluyendo a los stakeholders.
- **Identificación de Mejoras:** Aprovecha la revisión del Sprint para identificar áreas donde el equipo puede mejorar en futuros Sprints, manteniendo una mentalidad de mejora continua.

> **Caso práctico:** En una sesión de revisión del Sprint, los Developers se encuentran presentando el incremento de producto que lograron durante el Sprint. Laura, una de las Developers, toma la iniciativa de explicar cómo abordaron ciertos retos técnicos y lograron cumplir con los criterios de aceptación establecidos. A medida que se presentan los elementos, Juan, otro Developer, muestra ejemplos prácticos de las funcionalidades implementadas, permitiendo que las partes interesadas tengan una comprensión más clara del

> progreso. Juntos, los Developers colaboran para asegurar que todas las voces sean escuchadas y que el feedback recibido sea adecuadamente integrado en las consideraciones para el próximo Sprint.

Los Developers en la Retrospectiva del Sprint

- Participan en la inspección de cómo fue el último Sprint con respecto a los individuos, las interacciones, los procesos, las herramientas, y su Definición de terminado (DoD).
- Identifican las lecciones aprendidas y las suposiciones que los llevaron por el mal camino, lo que fue bien durante el Sprint, los problemas que encontraron, y cómo esos problemas fueron (o no fueron) resueltos
- Discuten y reflexionan sobre la velocidad del equipo en el Sprint

> **Aclaración:** Una retrospectiva no solo incluye la identificación de problemas, sino también la colaboración proactiva para diseñar soluciones viables y establecer metas claras para mejorar en áreas específicas.

□□ Tip para el Developer en la Retrospectiva

Cambia el Entorno y Añade un Toque Personal: No permitas que las retrospectivas se conviertan en una rutina aburrida. Sugiere realizar estas sesiones en una herramienta de colaboración gráfica, donde el equipo pueda visualizar de manera creativa los puntos de mejora, o incluso fuera de la oficina en un ambiente más relajado. Además, ¿por qué no agregar un elemento de diversión? Podría ser organizar una comida especial o un snack diferente para cada retrospectiva, fomentando un ambiente más ameno y relajado que puede ayudar a que fluyan mejor las ideas y la comunicación.

Consejos para Developers durante la Retrospectiva del Sprint

- **Valora la Retrospectiva:** La retrospectiva no es una mera formalidad, es una oportunidad crucial para reflexionar y crecer como equipo. Es importante que cada miembro del equipo entienda el valor de estas sesiones y participe activamente en ellas.
- **Abordar Temas Incomodos:** No evites los temas incomodos. Es vital abordar cualquier problema o desafío que el equipo pueda estar enfrentando para fomentar una mejora continua.
- **Fomenta la Comunicación Abierta:** Fomenta una atmósfera donde todos se sientan cómodos para compartir sus opiniones y feedback, esto podría llevar a descubrir áreas de mejora que no eran obvias en primera instancia.

> **Caso práctico:** En la retrospectiva, algunos miembros del equipo mencionan que sienten que no hay mucho que discutir o mejorar, mientras que otros sienten que hay problemas pero son reacios a hablar de ellos para evitar situaciones incómodas. **Acción:** Laura, una de las developers, decide tomar la iniciativa. Ella comparte honestamente algunos desafíos que ha notado durante

el Sprint, aunque son temas sensibles. Su apertura anima a otros a compartir sus propias observaciones y preocupaciones, lo que lleva a una discusión enriquecedora. **Resultado:** El equipo logra identificar varios aspectos cruciales para trabajar y mejorar en el próximo Sprint. Además, se dan cuenta de que, enfrentando los temas incomodos, pueden encontrar soluciones conjuntas y fortalecer la dinámica del equipo.

5.3 - Técnicas y herramientas útiles para los Developers

En el dinámico mundo de Scrum, las responsabilidades de los Developers trascienden las fronteras de las tareas convencionales. Se espera que estén equipados para abordar una amplia variedad de desafíos, adaptándose con agilidad a las demandas cambiantes del proyecto. En esta sección, exploraremos algunas técnicas y herramientas cruciales que pueden facilitar a los Developers esta navegación a través del vibrante entorno de Scrum. Dentro de ellas veremos:

- Técnicas para el desarrollo ágil
- Técnicas para estimar el trabajo
- Técnicas y herramientas de software para trabajar con Scrum

> **Consejo** Al seleccionar y emplear diversas técnicas y herramientas, es vital hacerlo con un enfoque centrado en el valor y la eficiencia. Considera la compatibilidad con las necesidades del organización, la facilidad de integración y cómo pueden potenciar una colaboración efectiva y una entrega ágil de incrementos de alta calidad. A menudo, una herramienta simple pero bien elegida o una técnica aplicada con pericia puede marcar una significativa diferencia, optimizando procesos y fomentando un ambiente de trabajo más productivo y armonioso.

5.3.1 - Para el desarrollo ágil

En esta sección, exploraremos técnicas y herramientas que pueden facilitar un proceso de desarrollo más fluido y colaborativo, promoviendo la calidad y eficiencia en cada incremento.

> **Nota:** Para aquellos interesados específicamente en el ámbito del desarrollo de software, les invitamos a consultar nuestra otra publicación enfocada especialmente en este sector, disponible a través de CertMind, donde profundizamos en técnicas y herramientas adaptadas a las particularidades del

> desarrollo de software.

Programación por parejas – Pair programming

La programación por parejas es una técnica de desarrollo ágil en la que dos desarrolladores colaboran en la misma tarea, trabajando en un único equipo o computadora. Este enfoque fomenta la colaboración, la comunicación directa y un intercambio fluido de ideas, lo que a menudo resulta en soluciones más innovadoras y eficientes.

Aunque puede parecer que esta técnica reduce la cantidad de funcionalidad producida, en realidad, tiende a mejorar notablemente la calidad del resultado final. Al tener dos pares de ojos revisando el trabajo en tiempo real, es más probable que se identifiquen y corrijan errores en una etapa temprana, lo que puede prevenir costos adicionales a largo plazo.

- En la programación por parejas, 2 Developers trabajan en un solo PC.
- La programación por parejas mejora notablemente la calidad del incremento, sin impactar el tiempo de entrega.
- Una sola persona sola agrega más funcionalidad, pero con menos calidad.
- La calidad alta puede ahorrar costes futuros en un proyecto.

Tip para el Pair Programming

La programación por parejas no es una sesión de enseñanza; de hecho, funciona mejor cuando ambos desarrolladores tienen un nivel similar de conocimiento y habilidades. Esto permite una colaboración más fluida y equitativa, donde ambos pueden contribuir significativamente al progreso de la tarea. Asegúrate de emparejar a desarrolladores con niveles de habilidad compatibles para aprovechar al máximo esta técnica.

El pair programming no solo se usa en programación de software

- **Diseño Gráfico:** Dos diseñadores podrían trabajar juntos en un diseño, con uno manejando las herramientas de diseño y el otro proporcionando retroalimentación en tiempo real.
- **Redacción y Edición:** Un escritor y un editor podrían colaborar en tiempo real en un documento. Uno escribe mientras el otro revisa la gramática, el estilo, etc.
- **Análisis de Datos:** En un proyecto de análisis de datos, un estadístico podría implementar modelos mientras otro experto en el dominio proporciona contexto y verifica la validez del análisis.
- **Educación:** En un entorno educativo, dos instructores podrían colaborar para enseñar una clase en línea o preparar material didáctico.
- **Investigación Científica:** Dos investigadores podrían colaborar en un experimento o en la redacción de un artículo científico.
- **Planeación Estratégica en Negocios:** En la gestión y planeación estratégica, un par podría trabajar en el análisis SWOT de un proyecto o en la elaboración de un plan de negocios.
- **Gestión de Proyectos:** Un administrador de proyecto y un stakeholder podrían revisar juntos el progreso de un proyecto, identificando riesgos y planificando mitigaciones en

tiempo real.

> **Caso práctico: El pair programming puede ser agotador** En una empresa de tecnología, se instauró la técnica de programación por parejas como una actividad regular pero no constante. Los equipos encontraron que trabajar en parejas durante periodos específicos (como por la mañana o en sesiones de no más de dos horas) era beneficioso sin ser agotador. Esta moderación permitió mantener altos niveles de energía y concentración, favoreciendo la calidad y eficiencia del trabajo realizado.

> **Caso práctico: Cambiando de pareja** En una startup, se estableció una rotación de parejas de programación con cada nuevo Sprint. Esta rotación no solo permitió que los Developers fortalecieran sus relaciones interpersonales, sino que también promovió una comprensión más profunda y diversificada del proyecto en su conjunto, ya que cada miembro del equipo tuvo la oportunidad de trabajar y aprender de diferentes perspectivas y enfoques a lo largo del tiempo.

Tip: el Pair Programming no se aplica a todas las tareas

El "Pair Programming" no es una regla fija que deba aplicarse a todas las tareas. Es una herramienta especialmente efectiva para abordar desafíos complejos, explorar nuevas tecnologías o desarrollar funcionalidades críticas. Considera implementarlo estratégicamente, especialmente en situaciones donde la colaboración directa puede potenciar la creatividad, la calidad y la eficiencia. Recuerda, la clave es usarlo donde agregue el mayor valor al proceso de desarrollo del producto.

La refactorización

La refactorización, en un contexto más amplio que el desarrollo de software, puede ser entendida como el proceso de revisar, ajustar y mejorar las estructuras existentes de un proyecto o sistema para hacerlo más eficiente, manejable o adaptable a nuevas necesidades, sin alterar su función fundamental. Esto puede aplicarse a una variedad de ámbitos, como procesos empresariales, estrategias de marketing, estructuras organizativas, etc.

> **Ejemplo: Antes de la Refactorización** Un equipo utiliza una plantilla de informe densa y complicada, que contiene una cantidad significativa de campos irrelevantes. Esto no solo consume tiempo sino que también dificulta la comprensión clara de los puntos clave. **Después de la Refactorización** - El equipo decide revisar y simplificar la plantilla, eliminando los campos innecesarios y dando prioridad a la información más crítica. Como resultado, los

> informes ahora son más concisos, centrados y fáciles de leer, facilitando la toma de decisiones y la comunicación efectiva dentro del equipo.

Tip: La refactorización debe ser vista como una actividad continua

La refactorización es un compromiso con la mejora constante. No es un "arreglo único", sino un proceso en el que se está siempre atento a las oportunidades de optimizar y ajustar para alcanzar una mayor eficiencia y efectividad. Además, es vital evitar la "refactorización por refactorización". Cada cambio debe tener un propósito claro y estar orientado a lograr mejoras tangibles, evitando alteraciones que no agregan valor significativo.

5.3.2 - Para estimar el trabajo

La estimación consiste en definir entre todos una suposición lo más precisa posible, de lo que se puede lograr y en cuanto tiempo.

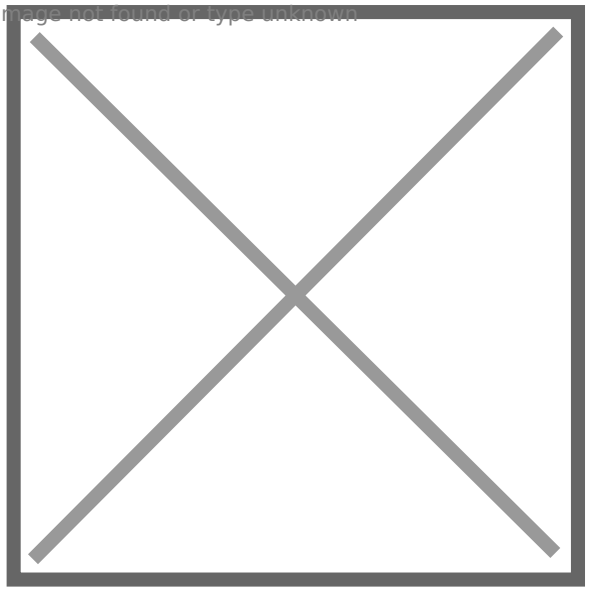
La estimación permite:

- Analizar dependencias
- Organizar las actividades prioritarias
- Definir las mejores técnicas de trabajo para ese Sprint
- Pronosticar diferentes escenarios (predecir tiempos y fechas de entrega)

Nota: El momento en que se realiza la estimación es durante la Planificación del Sprint.

Estimar con Planning Poker

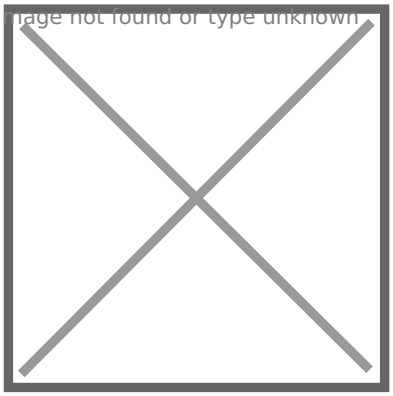
Image not found or type unknown



El póker de planificación consiste en un conjunto de tarjetas numéricas que sirven para realizar la estimación de tareas o historias de usuario.

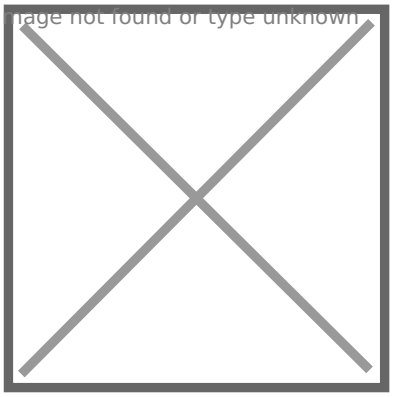
El póker puede ir numerado de 1 a 10 o utilizando la sucesión de Fibonacci

Estimar por afinidad



Consiste en encontrar las similitudes de las Historias de Usuario y tareas que se van a estimar. Lo que hace el equipo es agrupar los elementos que son similares. La mejor manera de “encontrar similitudes” es visualizar el panorama completo y luego agrupar elementos.

Estimar con Tallas de camiseta (t-shirt)



Se asigna una talla de camiseta a cada Historia de Usuario o tarea (desde XS hasta XXL) que representará el esfuerzo relativo de ese trabajo.

Con esta técnica se elimina la necesidad de la precisión de un modelo numérico, por lo que el equipo puede pensar de manera más abstracta en las estimaciones del trabajo a estimar.

¿Qué pasa cuando se Subestima/Sobreestima un Sprint?

Cuando los equipos Scrum son nuevos, o tienen poca experiencia es típico que sucedan las siguientes 2 situaciones:

Sobreestimación: Cuando los Developers proponen demasiado tiempo para el trabajo del Sprint y acaban mucho antes de la Revisión del Sprint. En este caso el Product Owner agregará más trabajo

al Sprint en curso (para que la Revisión no se cambie de fecha).

Subestimación: Cuando los Developers proponen muy poco tiempo para el trabajo del Sprint, lo que puede causar que el incremento no esté completamente terminado. En este caso la Revisión se hace sobre los elementos que se hayan completado y los demás quedan pendientes para ser terminados en el próximo Sprint (la revisión NO se aplaza).

Otras posibles técnicas que podrían ser útiles para mejorar el flujo de trabajo durante un Sprint

- Mesa de ayuda cruzada (Cross-Functional Help Desk): Un espacio donde los miembros del equipo pueden ofrecer o solicitar ayuda en diferentes áreas de especialización, fomentando la colaboración y el aprendizaje mutuo.
- Time-boxing: Establecer un tiempo límite para ciertas actividades puede ayudar a mantener el enfoque y evitar la procrastinación, así como fomentar la toma de decisiones eficiente.
- Revisión intermedia (Mid-Sprint Review): Una revisión a mitad de camino del Sprint puede ayudar a identificar cualquier obstáculo o cambio necesario a tiempo, permitiendo ajustes antes de la Revisión de Sprint.
- Kanban visual: Utilizar un tablero visual Kanban para monitorear el progreso de las tareas, lo que puede facilitar la identificación de cuellos de botella y ayudar a mantener un flujo de trabajo estable.
- Daily Stand-ups Mejoradas: Incorporar segmentos breves donde se compartan conocimientos o trucos útiles que hayan ayudado a los miembros del equipo a ser más eficientes.
- Refactorización de Procesos: Similar a la refactorización de código, esta técnica implica revisar y ajustar los procesos actuales para hacerlos más eficientes, eliminando pasos innecesarios o redundantes.
- Reuniones de Alineación de Equipos (Team Alignment Meetings): Organizar reuniones breves y focalizadas donde los equipos puedan alinear expectativas, definir objetivos claros y discutir estrategias para alcanzarlos con éxito.
- Workshops de Resolución de Problemas: Organizar talleres donde los equipos puedan colaborar para encontrar soluciones a problemas específicos, utilizando técnicas como el brainstorming o el pensamiento de diseño (Design Thinking).