**Tarea 3**

**Ciclo de vida del software (Relación 2)**

**1.- ¿Qué cuatro principios rigen el desarrollo ágil expresados en el Manifiesto Ágil?**

-Individuos e interacciones sobre procesos y herramientas: Este principio enfatiza la importancia de las personas en el desarrollo de software. Se valora la comunicación directa y la colaboración entre miembros del equipo por encima de seguir procesos y herramientas rígidos.

-Software funcionando sobre documentación extensiva: Aquí se hace hincapié en que el software en funcionamiento es más valioso que la documentación detallada. Aunque la documentación es importante, el enfoque principal debe estar en entregar software que funcione y que satisfaga las necesidades del cliente.

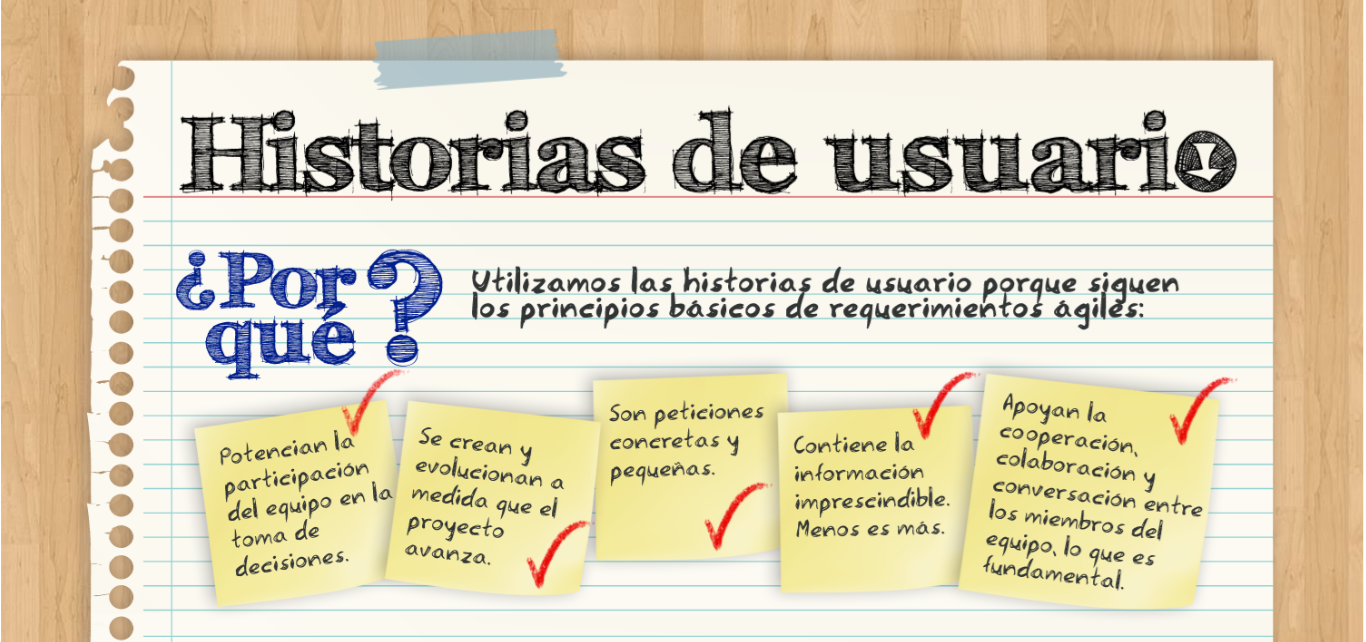
-Colaboración con el cliente sobre negociación de contratos: Se resalta la importancia de la colaboración cercana con el cliente en lugar de establecer contratos y acuerdos rígidos. El desarrollo ágil promueve la adaptación continua a medida que se comprenden mejor las necesidades del cliente.

-Responder al cambio sobre seguir un plan: Este principio reconoce que los requisitos y las circunstancias pueden cambiar a lo largo del proyecto. En lugar de aferrarse a un plan predefinido, el enfoque ágil se centra en ser ágil y receptivo a los cambios, lo que permite adaptarse y ajustar el proyecto según sea necesario.

**2.- ¿Qué es una historia de usuario?**

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos. Cada historia de usuario debe ser limitada, esta debería poderse escribir sobre una nota adhesiva pequeña. Dentro de la metodología XP las historias de usuario deben ser escritas por los usuarios.

Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. También permiten responder rápidamente a los requisitos cambiantes.



**3.- Haz un resumen sobre que se entiende por Lean software y qué principios lo rigen.**

Lean Software es un enfoque que aplica los principios del pensamiento Lean al desarrollo de software. Se centra en eliminar el desperdicio, entregar valor de manera continua, empoderar a los equipos, comprender las necesidades del cliente, buscar la mejora constante y construir calidad desde el principio. El objetivo es maximizar la eficiencia, la calidad y la satisfacción del cliente en el desarrollo de software.

Los principios que lo rigen son los siguientes:

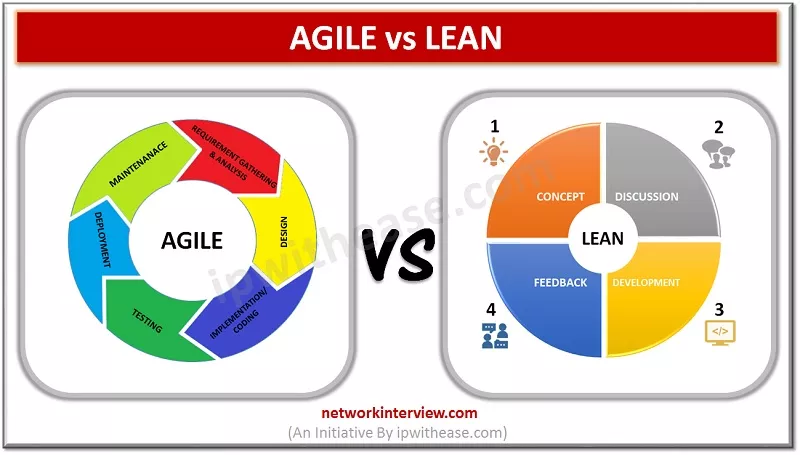
Eliminación del desperdicio: El principio central de Lean es eliminar cualquier actividad o proceso que no agrega valor al cliente. En el desarrollo de software, esto implica identificar y eliminar tareas innecesarias, características no utilizadas y cualquier otro tipo de desperdicio que pueda surgir en el proceso de desarrollo.

Entregar valor continuamente: Se busca entregar valor al cliente de manera constante y temprana. En lugar de esperar hasta el final del proyecto, se desarrollan funcionalidades que aportan valor y se entregan de forma iterativa, permitiendo una retroalimentación más rápida y la adaptación a las necesidades cambiantes del cliente.

Empoderamiento de los equipos: Se fomenta la autonomía y la toma de decisiones por parte de los equipos de desarrollo. Esto les permite adaptarse rápidamente a los cambios y tomar medidas para mejorar la eficiencia y la calidad del software.

Ampliación de la visión del cliente: Lean Software promueve una comprensión profunda de las necesidades del cliente y busca superar las expectativas mediante la entrega de soluciones que resuelvan problemas reales y agreguen valor.

Mejora continua: Se enfatiza la necesidad de evaluar y mejorar constantemente los procesos. Esto implica la retroalimentación regular, la reflexión sobre los resultados y la implementación de mejoras para optimizar el flujo de trabajo.

Construcción de calidad desde el principio: En lugar de centrarse en la corrección de errores después de la implementación, Lean Software aboga por la construcción de calidad desde el inicio del proceso de desarrollo, lo que reduce la necesidad de retrabajo y aumenta la eficiencia.

**4. KANBAN. Estudia las ventajas e inconvenientes de tener una pizarra web digital para la metodología Kanban.**

**Ventajas:**

-Accesibilidad global: Los equipos distribuidos geográficamente pueden acceder y colaborar en tiempo real en la pizarra web, lo que facilita la colaboración en proyectos remotos.

-Flexibilidad y escalabilidad: Puedes crear, modificar y ajustar tableros Kanban con facilidad para adaptarse a proyectos y equipos en constante evolución.

-Visibilidad y transparencia: Toda la información del proyecto es visible para todos los miembros del equipo, lo que promueve la transparencia y la comprensión de la situación actual.

-Actualización en tiempo real: Los cambios se reflejan instantáneamente en la pizarra, lo que permite un seguimiento en tiempo real del progreso del trabajo y la priorización de tareas.

-Notificaciones y recordatorios: Puedes configurar notificaciones y recordatorios para mantener a los miembros del equipo al tanto de las tareas y los plazos.

-Historial de cambios: La mayoría de las pizarras web mantienen un registro de todas las actividades pasadas, lo que facilita la revisión y el aprendizaje de proyectos anteriores.

**Inconvenientes:**

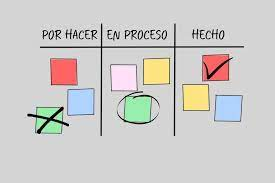
-Dependencia de la tecnología: Si la pizarra web experimenta problemas técnicos o interrupciones, podría afectar la productividad del equipo.

-Aprendizaje y adaptación: La implementación de una pizarra web Kanban puede requerir tiempo para que los miembros del equipo se adapten a la nueva herramienta y aprendan a utilizarla eficazmente.

-Costos asociados: Algunas pizarras web Kanban pueden tener costos asociados, lo que podría ser un inconveniente si el equipo tiene un presupuesto limitado.

-Posible falta de interacción personal: La colaboración en línea a través de una pizarra web puede llevar a una menor interacción cara a cara, lo que a veces es importante para la comunicación efectiva.

-Requisitos de conectividad y dispositivos: Los miembros del equipo necesitan acceso a Internet y dispositivos compatibles para utilizar una pizarra web, lo que puede limitar la accesibilidad en ciertas situaciones.



**5. KANBAN. Haz un resumen de la metodología Kanban e indica sus diferencias frente a SCRUM.**

Kanban es una metodología de gestión de proyectos y procesos que se centra en la visualización y el flujo del trabajo. Consiste en los siguiente:

-Visualización del trabajo: Kanban utiliza tableros visuales para representar el flujo de trabajo, donde las tareas se muestran como tarjetas que avanzan a través de columnas que representan estados o etapas del proceso.

-Sin iteraciones fijas: En Kanban, no hay iteraciones fijas de tiempo (como en Scrum). El trabajo se realiza de manera continua, y las tarjetas avanzan a medida que se completa el trabajo.

- Enfoque en el flujo: Kanban se enfoca en optimizar el flujo de trabajo y en la gestión de las limitaciones de capacidad (WIP, por sus siglas en inglés) para evitar la sobrecarga de tareas en proceso.

-Flexibilidad: Kanban es altamente adaptable y permite la incorporación de nuevas tareas o cambios de prioridades en cualquier momento, lo que lo hace ideal para entornos con requisitos cambiantes.

-No se requieren roles específicos: Kanban no especifica roles fijos como Scrum (Scrum Master, Product Owner). Los equipos de Kanban pueden variar en su estructura según las necesidades del proyecto.

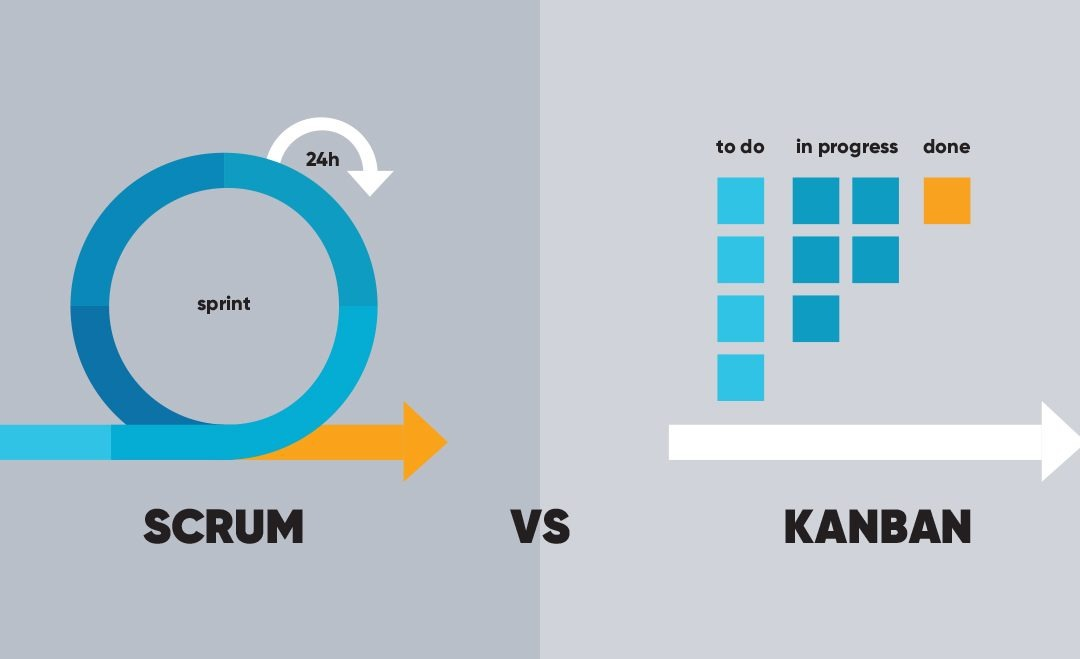
Algunas de las diferencias con Scrum son:

-Iteraciones vs. Flujo continuo: Scrum utiliza iteraciones (sprints) de tiempo fijo, mientras que Kanban se basa en un flujo de trabajo continuo sin iteraciones predefinidas.

-Roles definidos vs. Roles flexibles: Scrum define roles específicos, como Scrum Master y Product Owner, mientras que Kanban no establece roles fijos y permite una mayor flexibilidad en la organización del equipo.

-Planificación fija vs. Cambios en tiempo real: Scrum requiere una planificación fija al comienzo de cada sprint, mientras que Kanban permite cambios en tiempo real y la adición de nuevas tareas a medida que surgen.

-WIP limitado vs. Sin límite WIP: Kanban se centra en la gestión de la limitación de tareas en proceso (WIP), mientras que Scrum no tiene una limitación WIP explícita.

-Adaptabilidad vs. Previsibilidad: Kanban es altamente adaptable a cambios y es adecuado para proyectos con requisitos cambiantes. Scrum se enfoca en la previsibilidad y la planificación a corto plazo.

**6. SCRUM. Explica cómo funciona Scrum.**

Scrum es un marco de trabajo ágil para la gestión de proyectos que se centra en la colaboración, la adaptabilidad y la entrega incremental. Funciona de la siguiente manera:

Roles definidos: En Scrum, se asignan roles específicos, como el Scrum Master (responsable de garantizar que el proceso se siga correctamente), el Product Owner (responsable de definir y priorizar el backlog del producto) y el Equipo de Desarrollo (responsable de realizar el trabajo).

Backlog del producto: Se crea una lista de elementos de trabajo llamada "backlog del producto" que representa todas las tareas, características o funcionalidades necesarias para el proyecto.

Planificación de sprints: Se seleccionan elementos del backlog y se planifican en "sprints" (iteraciones de tiempo fijo, por lo general de 2 a 4 semanas).

Reuniones diarias: Cada día, el equipo de desarrollo se reúne en una "reunión diaria de Scrum" para revisar el progreso, identificar obstáculos y ajustar el plan.

Revisión del sprint: Al final de cada sprint, el equipo realiza una "revisión del sprint" para demostrar el trabajo completado y recopilar retroalimentación.

Retrospectiva del sprint: También al final de cada sprint, se lleva a cabo una "retrospectiva del sprint" para evaluar lo que funcionó bien y lo que se puede mejorar en el próximo sprint.

Entrega incremental: Al final de cada sprint, se entrega un producto funcional que agrega valor al cliente. Este proceso se repite en ciclos sucesivos.

**7. SCRUM. Define los siguientes términos:**

* El "Product Backlog" es una lista priorizada de todas las características, historias de usuario y tareas que deben abordarse en un proyecto ágil, como Scrum. Se utiliza para mantener un registro de los requisitos e ideas del producto, con los elementos más importantes en la parte superior. Esta lista es flexible y cambia con el tiempo a medida que se obtiene más información y se adapta a las necesidades del proyecto. El Product Owner es responsable de su gestión, pero es un esfuerzo colaborativo y se mantiene en constante revisión y actualización.
* El "Sprint Backlog" es una lista de tareas específicas y detalladas seleccionadas del "Product Backlog" para ser abordadas durante un sprint en el marco de trabajo Scrum. Es un conjunto concreto de trabajo que el equipo se compromete a completar en ese periodo. El Sprint Backlog es el plan para el sprint actual y se utiliza para guiar las actividades diarias del equipo durante el sprint.

**8. SCRUM. En la terminología Scrum qué términos se utilizan como sinónimo de:**

* Jefe de proyecto-Scrum Master.
* Cliente-Stakeholder.
* Equipo de desarrollo-Team

**9. SCRUM. Haz un resumen de los requisitos para poder utilizar Scrum.**

**Los requisitos para poder utilizar Scrum son:**

* Cultura de empresa basada en trabajo en equipo, delegación, creatividad y mejora continua.
* Compromiso del cliente en la dirección de los resultados del proyecto, gestión del ROI y disponibilidad para poder colaborar.
* Compromiso de la Dirección de la organización para resolver problemas endémicos y realizar cambios organizativos, formando equipos autogestionados y multidisciplinares y fomentando una cultura de gestión basada en la colaboración y en la facilitación llevada a cabo por líderes al servicio del equipo.
* Compromiso conjunto y colaboración de los miembros del equipo.
* Relación entre proveedor y cliente basada en ganar-ganar, colaboración y transparencia.
* Facilidad para realizar cambios en el proyecto.
* Tamaño de cada equipo entre 5 y 9 personas (con técnicas específicas de planificación y coordinación cuando varios equipos trabajan en el mismo proyecto).
* Equipo trabajando en un mismo espacio común para maximizar la comunicación.
* Dedicación del equipo a tiempo completo.
* Estabilidad de los miembros del equipo

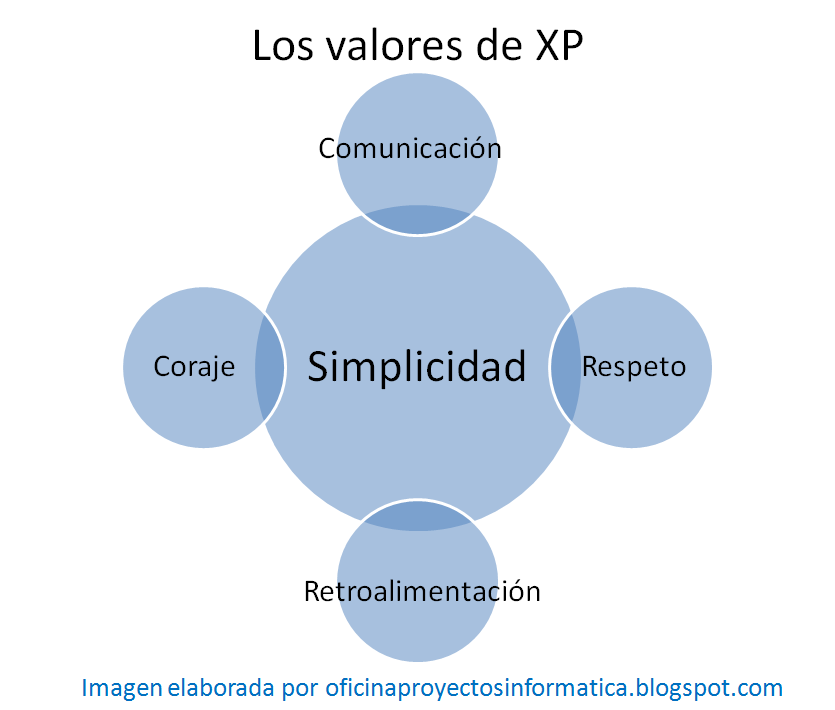
**10. XP. Explica los 5 valores de la Programación Extrema.**

Comunicación: La comunicación efectiva es esencial en XP. Los miembros del equipo, incluyendo a los desarrolladores, el cliente y otros stakeholders, deben mantener una comunicación constante y abierta. Se fomenta la conversación cara a cara, la retroalimentación constante y la colaboración activa para asegurarse de que todos tengan una comprensión clara de los requisitos y el progreso del proyecto.

Simplicidad: La simplicidad es un valor clave en XP. Se promueve la idea de hacer lo más simple posible, evitando la complejidad innecesaria en el diseño y la implementación del software. Esto incluye escribir código limpio y fácil de mantener, así como evitar la inclusión de características o funcionalidades no esenciales.

Retroalimentación: La retroalimentación constante es esencial para mejorar el proceso de desarrollo. En XP, se busca obtener retroalimentación temprana y frecuente de los usuarios y otros miembros del equipo. Esto permite realizar ajustes rápidos y mantener el producto en línea con las necesidades cambiantes.

Valentía: La valentía en XP se refiere a la disposición del equipo para tomar decisiones audaces y enfrentar desafíos. Los desarrolladores deben estar dispuestos a asumir riesgos y hacer cambios cuando sea necesario para mejorar el software y el proceso de desarrollo.

Respeto: El respeto es un valor central en XP. Los miembros del equipo deben respetarse mutuamente, así como a los clientes y stakeholders. Esto implica escuchar y valorar las opiniones y perspectivas de todos los involucrados y trabajar juntos de manera colaborativa y respetuosa.

**11. XP. ¿Cuáles son las características distintivas de XP frente a otras metodologías ágiles? Explícalas.**

La Programación Extrema (XP) se distingue de otras metodologías ágiles por sus características específicas y enfoques particulares. Algunas de las características distintivas de XP incluyen:

Pruebas Unitarias: XP enfatiza la escritura de pruebas unitarias antes de escribir el código de producción. Esto ayuda a garantizar que el código funcione correctamente y se pueda mantener a lo largo del tiempo.

Desarrollo Incremental: XP aboga por el desarrollo incremental, donde las características se implementan en pequeñas iteraciones y se entregan con frecuencia. Esto permite una rápida adaptación a las necesidades cambiantes del cliente.

Pair Programming: En XP, dos programadores trabajan juntos en una misma computadora, revisando y mejorando el código constantemente. Esta práctica fomenta la calidad del código y la transferencia de conocimiento.

Refactorización Continua: XP promueve la refactorización constante del código para mantenerlo limpio, eficiente y fácil de mantener.

Cliente en Sitio: Se espera que el cliente esté disponible en el sitio de desarrollo para brindar una retroalimentación constante y aclarar requisitos en tiempo real.

Iteraciones Cortas: XP utiliza iteraciones muy cortas, típicamente de una a tres semanas, para asegurar una entrega frecuente y adaptabilidad.

Planificación Semanal: Las reuniones de planificación se llevan a cabo semanalmente para ajustar y priorizar el trabajo según las necesidades cambiantes.

Historias de Usuario: XP utiliza historias de usuario para representar requisitos funcionales desde la perspectiva del cliente.

Simple Diseño: La simplicidad en el diseño es un valor fundamental de XP. Se enfoca en mantener el diseño lo más simple posible, evitando la complejidad innecesaria.

Valores Claves: XP se basa en cinco valores clave: comunicación, simplicidad, retroalimentación, valentía y respeto.