Tarea 3

Ciclo de vida del software (Relación 2)

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

1. **Las interacciones individuales son más importantes que los procesos y las herramientas**. Las personas impulsan el proceso de desarrollo y responden a las necesidades comerciales. Son la parte más importante del desarrollo y deben valorarse por encima de los procesos y herramientas. Si los procesos o las herramientas impulsan el desarrollo, será menos probable que el equipo responda y se adapte al cambio y, por lo tanto, será menos probable que satisfaga las necesidades del cliente.
2. **Enfoque en el software de trabajo en lugar de una documentación completa.** Antes de Agile, se dedicaba una gran cantidad de tiempo a documentar el producto durante todo el desarrollo para su entrega. La lista de requisitos documentados era extensa y causaría grandes retrasos en el proceso de desarrollo. Si bien Agile no elimina el uso de documentación, la simplifica de manera que proporciona al desarrollador solo la información necesaria para realizar el trabajo —como las historias de usuario. El Manifiesto Ágil sigue valorando el proceso de documentación, pero le da más valor al software que funciona.
3. **Colaboración en lugar de negociaciones contractuales.** Agile se enfoca en la colaboración entre el cliente y el gerente del proyecto, en lugar de negociaciones entre los dos, para resolver los detalles de la entrega. Colaborar con el cliente significa que se incluye a lo largo de todo el proceso de desarrollo, no solo al principio y al final, lo que facilita a los equipos la satisfacción de las necesidades de sus clientes. Por ejemplo, en el desarrollo de software ágil, el cliente puede ser incluido en diferentes intervalos para demostraciones del producto. Sin embargo, el cliente también podría estar presente e interactuando con los equipos a diario, asistiendo a todas las reuniones y asegurándose de que el producto satisfaga sus deseos.

4. **Enfoque en responder al cambio**. El desarrollo de software tradicional se utilizaba para evitar cambios porque se consideraba un gasto no deseado. Agile eliminó esta idea. Las breves iteraciones en el ciclo Agile permiten realizar cambios fácilmente, lo que ayuda al equipo a modificar el proceso para que se adapte mejor a sus necesidades y no al revés. En general, el desarrollo de software ágil cree que el cambio es siempre una forma de mejorar el proyecto y proporcionar valor adicional.

2.- ¿Qué es una historia de usuario? Consulta el siguiente enlace:
https://es.wikipedia.org/wiki/Historias_de_usuario

Una **historia de usuario** se define como una breve descripción de un requisito expresada en un lenguaje claro y accesible para cualquier usuario. En el contexto de las metodologías ágiles, estas historias desempeñan un papel fundamental en la definición de los requisitos, ya que involucran la participación activa de los usuarios y la realización de pruebas de validación.

Deben ser breves y concisas, del tamaño de una nota adhesiva, y en el marco de la metodología XP, son redactadas directamente por los propios usuarios.

En resumen, las historias de usuario simplifican la gestión de los requisitos de los usuarios al evitar la creación de documentos extensos y permiten una respuesta ágil a posibles cambios en los requisitos.

3.- Haz un resumen sobre qué se entiende por Lean software y qué principios lo rigen. Consulta el siguiente enlace:
https://es.wikipedia.org/wiki/Lean_software_development

El **Lean Software** es la aplicación de los principios eficientes de producción de Toyota al desarrollo de software. Esta metodología, respaldada por la comunidad Ágil, ofrece un marco teórico sólido basado en la experiencia para optimizar la gestión en el desarrollo de software.

El desarrollo Lean de software se rige por siete **principios:**

- Eliminar los desperdicios: Se busca reducir cualquier elemento que no aporte valor al cliente, como código innecesario, retrasos en el proceso, requisitos ambiguos, burocracia y comunicación lenta. Esto se logra identificando y eliminando estos elementos de manera iterativa.
- Amplificar el aprendizaje: Se fomenta el aprendizaje constante y rápido, realizando pruebas tempranas, recopilando retroalimentación de los clientes y adaptando el desarrollo en consecuencia.
- Decidir lo más tarde posible: Se postergan las decisiones críticas hasta que se basen en hechos concretos en lugar de suposiciones. Se promueve la flexibilidad y la adaptación a cambios.
- Entregar tan rápido como sea posible: La entrega temprana permite recibir retroalimentación más pronto, lo que mejora la comunicación y la calidad del producto. Las iteraciones cortas son clave.
- Capacitar al equipo: Se cambia el enfoque de gestión tradicional por uno más colaborativo, donde se escucha a los desarrolladores y se les brinda autonomía. Se valora el compromiso y la motivación del equipo.
- Construir integridad intrínseca: Se busca que el software sea integral, intuitivo y eficiente. La refactorización y pruebas automáticas son prácticas esenciales para mantener la calidad.
- Ver el conjunto: Se entiende que los sistemas de software son el resultado de interacciones entre componentes y equipos. Se enfatiza la importancia de definir y mantener relaciones claras entre proveedores para lograr una buena interacción entre las partes.
- En conjunto, la implementación de todos estos principios Lean en el desarrollo de software es fundamental para el éxito, siguiendo la filosofía de "Piensa en grande, actúa en pequeño, equivócate rápido; aprende con rapidez".

4. KANBAN. Estudia las ventajas e inconvenientes de tener una pizarra web digital para la metodología Kanban. Puedes consultar los siguientes enlaces:

- https://leankit.com/learn/kanban/kanban-board/
- https://trello.com/es
- https://taiga.io/
- https://kanbantool.com/es/

Ventajas:

- Acceso remoto: Puedes acceder a la pizarra digital desde cualquier lugar con conexión a Internet, lo que facilita el trabajo a distancia y la colaboración entre equipos dispersos geográficamente.
- Actualizaciones en tiempo real: Los cambios realizados en la pizarra son visibles al instante para todos los miembros del equipo, lo que mejora la transparencia y la comunicación.
- Automatización de tareas: Algunas herramientas de pizarra web digital pueden automatizar ciertas tareas, como el movimiento de tarjetas entre columnas, lo que ahorra tiempo y reduce errores humanos.
- Historial de actividades: La mayoría de las herramientas mantienen un registro de todas las acciones realizadas en la pizarra, lo que facilita la revisión y la resolución de problemas.
- Integración con otras herramientas: Muchas pizarras web digitales se integran con aplicaciones de gestión de proyectos, calendarios y otras herramientas, lo que facilita la administración y seguimiento de tareas.

Inconvenientes:

- Costo: Algunas pizarras web digitales de calidad pueden ser costosas, especialmente para equipos grandes o empresas.
- Aprendizaje y adaptación: La implementación de una nueva herramienta puede requerir tiempo y recursos para que los miembros del equipo se adapten y aprendan a utilizarla eficazmente.

- Dependencia de la tecnología: Si la herramienta experimenta problemas técnicos o caídas del servidor, podría afectar la productividad del equipo.
- Posible sobrecarga de información: La facilidad de agregar tarjetas y detalles en una pizarra digital puede llevar a la sobrecarga de información si no se gestiona adecuadamente.
- Falta de interacción física: La metodología Kanban tradicional a menudo involucra tarjetas físicas y un tablero físico, lo que permite una interacción táctil y visual que puede perderse en una pizarra web digital.

5. KANBAN. Haz un resumen de la metodología Kanban e indica sus diferencias frente a SCRUM. Puedes consultar el siguiente enlace: https://es.atlassian.com/agile/kanban

1. Cadencia: Mientras que Scrum opera en sprints de longitud fija y periódicos (por ejemplo, dos semanas), Kanban se basa en el flujo continuo, lo que significa que no tiene un ciclo fijo.
2. Metodología de publicación: En Scrum, la entrega de funcionalidades se produce al final de cada sprint, siempre que el propietario del producto lo apruebe. En Kanban, no hay un momento fijo de entrega, y los equipos pueden publicar funcionalidades en cualquier momento o según lo determine el equipo.
3. Funciones: Scrum tiene roles definidos, como el propietario del producto, el experto en Scrum y el equipo de desarrollo. En Kanban, no existen roles fijos, aunque algunos equipos pueden contar con la ayuda de un orientador ágil.
4. Métricas clave: En Scrum, una métrica clave es la "velocidad" del equipo, que mide la cantidad de trabajo completado en un sprint. En Kanban, la métrica clave es el "tiempo del ciclo," que mide cuánto tiempo lleva completar una tarea desde que se inicia hasta que se finaliza.
5. Filosofía de cambios: En Scrum, se enfatiza evitar cambios en la previsión durante el sprint, ya que esto podría afectar la estimación y el aprendizaje. En Kanban, los cambios pueden suceder en cualquier momento, lo que permite una mayor flexibilidad en la gestión del trabajo.

6. SCRUM. Explica como funciona Scrum. Consulta los siguientes enlaces:

- https://proyectosagiles.org/que-es-scrum/
- https://proyectosagiles.org/como-funciona-scrum/

Scrum es un proceso colaborativo que se basa en buenas prácticas para lograr los mejores resultados en proyectos. Se centra en entregas regulares y parciales del producto final, priorizadas según el beneficio que aportan al receptor del proyecto. Es adecuado para proyectos en entornos complejos, con requisitos cambiantes, y donde la innovación, la flexibilidad y la productividad son esenciales.

El proceso **Scrum** se ejecuta en ciclos temporales cortos y de duración fija (iteraciones). Cada iteración debe proporcionar un resultado completo y listo para entregar al cliente. Las actividades incluyen:

- Planificación de la iteración: En la reunión de planificación de la iteración, se seleccionan y estiman los requisitos prioritarios del producto que se completarán durante la iteración.
- Ejecución de la iteración: Durante la iteración, el equipo se sincroniza diariamente para inspeccionar el trabajo en curso, resolver obstáculos y adaptarse para cumplir con los objetivos de la iteración.
- Inspección y adaptación: Al final de la iteración, se realiza una revisión en la que se presentan los requisitos completados y una retrospectiva para analizar el proceso de trabajo y mejorar la productividad.

Scrum se enfoca en la transparencia, la inspección y la adaptación continua para entregar un producto de alta calidad y satisfacer las necesidades del cliente.

7. SCRUM. Define los siguientes términos:

- **Product backlog**: El Product Backlog es una lista priorizada de todas las características y requisitos que se desean para un producto. Es una herramienta clave en metodologías ágiles como Scrum para mantener un registro de lo que

se debe hacer y priorizar. Los elementos en la parte superior de la lista son los más importantes y se abordan primero. El backlog se actualiza constantemente.

- **Sprint backlog**: El Sprint Backlog es una lista de tareas y elementos específicos seleccionados del Product Backlog que un equipo de desarrollo se compromete a completar durante un sprint en la metodología Scrum. Estas tareas son detalladas y estimadas para asegurar que sean alcanzables dentro del tiempo del sprint, que generalmente dura de dos a cuatro semanas. El Sprint Backlog ayuda al equipo a enfocarse en las actividades necesarias para cumplir los objetivos del sprint y a medir su progreso durante ese período.

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

- **Jefe de proyecto**. Scrum Master, se centra en facilitar y apoyar al equipo de desarrollo en lugar de tener autoridad sobre ellos.
- **Cliente**. Product Owner, es la persona responsable de representar los intereses del cliente y definir los requisitos y prioridades del producto.
- **Equipo de desarrollo**. Development team, se refiere al grupo de personas que son responsables de diseñar, desarrollar, probar y entregar el producto.

9. SCRUM. Haz un resumen de los requisitos para poder utilizar Scrum. Consulta el siguiente enlace:

https://proyectosagiles.org/requisitos-de-scrum/

- **Cultura de empresa alineada con Scrum**: La empresa proveedora del proyecto debe promover la colaboración, el trabajo en equipo, la autogestión, la creatividad, la transparencia y la mejora continua. Los obstáculos existentes en la organización deben abordarse y solucionarse.
- **Compromiso del cliente**: El cliente debe estar altamente comprometido y disponible para dirigir los resultados del proyecto. Esto implica definir una lista de requisitos priorizada y planificar el proyecto en cada iteración para maximizar el Retorno de la Inversión (ROI).
- **Compromiso de la Dirección**: La dirección de la empresa debe respaldar el uso de Scrum. Debe estar preparada para abordar obstáculos organizativos,

técnicos y de relaciones interpersonales, y permitir la transición hacia equipos autogestionados y multidisciplinarios.

- **Compromiso del equipo**: Los miembros del equipo deben comprometerse y colaborar entre sí. La transparencia en el trabajo personal es esencial, y las personas deben ser capaces de adaptarse y salir de su zona de confort.
- **Relación entre proveedor y cliente**: La relación entre el cliente y el proveedor debe basarse en un enfoque de ganar-ganar, permitiendo cambios controlados en el proyecto.
- **Facilidad para realizar cambios en el proyecto**: El proyecto debe admitir la incorporación incremental de requisitos y cambios de manera controlada sin costos excesivos.
- **Tamaño del equipo**: El equipo Scrum ideal tiene entre 5 y 9 miembros para maximizar la comunicación. Sin embargo, Scrum puede adaptarse a equipos más pequeños o colaboración entre múltiples equipos.
- **Equipo trabajando en un mismo espacio común**: Todos los miembros del equipo deben trabajar en la misma ubicación física para facilitar la comunicación directa y minimizar canales de comunicación menos eficientes.
- **Dedicación del equipo a tiempo completo**: Los miembros del equipo deben estar dedicados al proyecto a tiempo completo para evitar la pérdida de productividad y simplificar la gestión de recursos humanos.
- **Estabilidad del equipo**: Se busca mantener la estabilidad del equipo, con cambios mínimos en la composición, para aprovechar las relaciones interpersonales y la organización del trabajo ya establecidas.

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

Los autores de XP han identificado cuatro valores esenciales que desempeñan un papel fundamental en la garantía del éxito de un proyecto:

- **Comunicación**: La comunicación ocupa un lugar muy importante. XP facilita la comunicación entre los diferentes miembros del equipo de trabajo, incluyendo jefes de proyecto, clientes y desarrolladores.

- **Sencillez**: Se enfatiza la importancia de mantener los programas tan simples como sea posible y de incluir solo la funcionalidad necesaria según los requisitos. No se debe agregar funcionalidad adicional que no sea requerida en el presente. Si surge la necesidad de añadir más funcionalidad en el futuro, se abordará en ese momento.
- **Retroalimentación**: La retroalimentación se considera crucial, ya que las pruebas aplicadas al software mantienen a todos informados sobre el grado de confiabilidad del sistema.
- **Valentía**: Se valora la disposición para asumir desafíos, enfrentar problemas y abordarlos. Esto incluye la voluntad de mejorar incluso cuando algo ya está funcionando. La presencia de pruebas unitarias reduce el riesgo de cometer errores graves.

Algunas voces también sugieren un quinto valor: **humildad**. Dado el énfasis en la compartición de código, la refactorización y el estrecho trabajo en equipo en XP, la humildad se aprecia siempre.

11. XP. ¿Cuáles son las características distintivas de XP frente a otras metodologías ágiles? Explícalas. Puedes consultar el siguiente enlace: http://www.davidvalverde.com/blog/introduccion-a-la-programacion-extrema-xp/

- **Énfasis en la Obtención de Resultados y Reducción de Burocracia**: XP pone en primer plano la consecución de resultados efectivos y minimiza la carga burocrática que suele existir en otras metodologías más complejas.
- **Capacidad de Adaptación al Cambio**: XP reconoce que en el ámbito del software, todo está en constante evolución, desde los requisitos y el diseño hasta el contexto del negocio, la tecnología y los equipos. La metodología abraza esta dinámica y se centra en ser altamente adaptable.
- **Prioridad en la Satisfacción del Cliente**: El propósito primordial de XP es asegurar la satisfacción del cliente, entregando rápidamente lo que este necesita y siendo receptivos a sus cambios, incluso si surgen en fases avanzadas del proyecto.

- **Trabajo Colaborativo**: XP fomenta la colaboración entre los líderes del proyecto, los clientes y los desarrolladores, considerándolos como una unidad de trabajo cohesionada.
- **Valores Clave**: XP se fundamenta en cuatro valores esenciales: comunicación efectiva, simplicidad, retroalimentación continua y valentía. Algunas interpretaciones también incorporan la humildad como un quinto valor.
- **Prácticas Clave de XP**: La metodología se apoya en 12 prácticas fundamentales que engloban la planificación dinámica, entregas en fases pequeñas, el uso de metáforas, enfoque en el diseño simple, pruebas exhaustivas, refactorización, programación en parejas, propiedad compartida del código, integración continua, una semana laboral de 40 horas, la presencia constante del cliente y la implementación de estándares de codificación.
- **Énfasis en Calidad y Comunicación**: XP promueve la calidad del código a través de prácticas como pruebas minuciosas y refactorización. Además, subraya la comunicación efectiva entre todos los miembros del equipo, incluyendo clientes y desarrolladores.
- **Enfoque en la Entrega Continua**: XP aboga por entregas continuas y frecuentes de software funcional en lugar de extensos ciclos de desarrollo.
- **Trabajo en Parejas**: La programación en parejas es una práctica característica de XP, en la que dos programadores colaboran en el mismo equipo, compartiendo un solo ordenador y contribuyendo juntos al desarrollo del software.
- **Participación Activa del Cliente**: Tener al cliente presente en las instalaciones de desarrollo es crucial para obtener retroalimentación inmediata y tomar decisiones basadas en las necesidades del cliente.
- **Énfasis en la simplicidad**: XP prioriza soluciones simples en lugar de complejas, lo que facilita la comprensión del código y su capacidad de adaptación a futuros cambios.