INTRODUCCIÓN:

**1.**

En cuanto a tipos de Software, diferenciaremos entre:

-**De sistema**, es el conjunto de instrucciones que permiten el manejo del ordenador, como el sistema operativo, los drivers…

**-Los de aplicación** son los programas diseñados para o por los usuarios para facilitar la realización de tareas específicas, como todos los relacionados con la ofimática, navegadores, editores de imagen/video…

**-De desarrollo** son generalmente un conjunto de herramientas de desarrollo de **software** que permite a un desarrollador de software crear una aplicación informática, que son los que nos pueden servir para programar como editores, compiladores, intérpretes etc…

**2.**

Disco duro- Almacena de forma permanente archivos y datos

Memoria RAM- Almacena de forma temporal el código binario de archivos y datos

CPU- Lee y ejecuta instrucciones almacenadas en la RAM

Dispositivos entrada/salida. Recoge nuevos datos desde la entrada, muestran resultados etc…

**3.**

**Código fuente:** Archivo de texto legible escrito en un lenguaje de programación.

**Código objeto**: Archivo binario no ejecutable.

**Código ejecutable:** Archivo binario ejecutable.

EJERCICIOS:

**1.**

Es cada una de las fases por las que un sofware pasa antes de ser creado, las cuales corresponden con: Análisis, Diseño, Codificación, Pruebas, Documentación y Mantenimiento.

**2.**

Análisis: Poner en común las ideas del cliente, hacer anotaciones sobre qué es lo que quiere, intentar explicar de forma clara y entendible etc…

Diseño: Se descompone y organiza el sistema en elementos componentes que puedan ser desarrollados por separado, se especifica la interrelación y funcionalidad de los elementos componentes.

Codificación: Se escribe el código fuente de cada componente, pueden utilizarse lenguajes de programación como C, Python… o de otro tipo como HTML, JSON…

Pruebas: El principal objetivo es resolver fallos, o resultados no deseados, debe realizarse el máximo número de pruebas posibles.

Documentación: Dejar todo bien explicado paso a paso tanto para el cliente, como para el propio programador, ya que puede llevar a confusiones en futuras revisiones de código.

Mantenimiento: Durante la explotación del software en ocasiones es necesario realizar cambios, para corregir defectos, mejorar funcionalidades, añadir nuevas funciones, adaptar a nuevos entornos etc…

**3.**

Identifica las fases principales del desarrollo del software y estas deben realizarse en orden, de modo que si fallamos en una de ellas, debemos repetir todas las fases desde esa misma.

**4.**

**VENTAJAS**- Identifica bien todas las fases del desarrollo software. Sigue un orden sencillo de entender

**DESVENTAJAS-** Si una fase falla, debemos volver a repetir desde allí. Se adapta mal al cambio contínuo de especificaciones

**5.**

Verificar supone que todo lo que realizamos funciona perfectamente

Validar supone que debemos comprobar paso por paso todo lo realizado.

**6.**

Se crea un prototipo durante la fase de análisis y es probado por el usuario/cliente para refinar los requisitos. Se repite el paso anterior las veces necesarias.

**8.**

Valorar más a los individuos y sus interacciones que a los procesos y las herramientas.

Valorar más el software funcionando que la documentación exhaustiva.

Valorar más la colaboración con el cliente que la negociación contractual.

Valorar más la respuesta ante el cambio que seguir un plan.

**9.**

Es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario.

**10.**

La metodología de desarrollo de software Lean es una traducción de los principios y las prácticas de la forma de producir lean, la cual se basa en minimizar las pérdidas de los sistemas de manufactura al mismo tiempo que maximiza la creación de valor para el cliente final.

Sus principios son los siguientes:

Eliminar los desperdicios

Amplificar el aprendizaje

Decidir lo más tarde posible

Entregar tan rápido como sea posible

Capacitar al equipo

Construir integridad intrínseca

Véase todo en conjunto

**11.**

**Ventajas:**

Medición del rendimiento de los equipos y trabajadores.

Organización del flujo de trabajo

Cumplir los tiempos de entrega

Distribución de tareas

Mejorar la calidad del producto

Evitar la acumulación de trabajo

**Inconvenientes:**

Coste elevado para unidades muy grandes

Es limitado

Kanban no es óptimo para todo tipo de proyectos

No permite anticiparse a grandes aumentos de la demanda.

**1.**

Kanban es un marco de trabajo muy popular a la hora de implementar un desarrollo software ágil. Los elementos de trabajo se representan visualmente en un tablero kanban, lo que permite a los miembros del equipo ver el estado de cada uno en cualquier momento.

Las diferencias frente a SCRUM son las siguientes:



**2.**

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (normalmente de 2 semanas). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo necesite.

**3.**

El Producto Backlog es un listado de todas las tareas que se pretenden hacer durante el desarrollo de un proyecto, todas las tareas deben listarse en el product backlog para que estén visibles ante todo el equipo y se pueda tener una visión panorámica de todo lo que se espera realizar.

El Sprint backlog es una planificación táctica del trabajo a realizar en la iteración actual. Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto.

**4.**

**Jefe de proyecto**= Scrum Master

**Cliente**= Product Owner

**Equipo de desarrollo**= Development Team

**5.**

[Cultura de empresa](https://proyectosagiles.org/requisitos-de-scrum#cultura-empresa) basada en trabajo en equipo, delegación, creatividad y mejora continua.

[Compromiso del cliente](https://proyectosagiles.org/requisitos-de-scrum#compromiso-cliente) en la dirección de los resultados del proyecto, gestión del ROI y disponibilidad para poder colaborar.

[Compromiso de la Dirección](https://proyectosagiles.org/requisitos-de-scrum#compromiso-direccion) de la organización para resolver problemas endémicos y realizar cambios organizativos, formando [equipos autogestionados y multidisciplinares](https://proyectosagiles.org/requisitos-de-scrum#equipos-autogestionados-multidisciplinares) y fomentando una cultura de gestión basada en la colaboración y en la [facilitación llevada a cabo por líderes al servicio del equipo](https://proyectosagiles.org/buen-gestor-equipo-agil-scrummaster).

[Compromiso conjunto y colaboración de los miembros del equipo](https://proyectosagiles.org/requisitos-de-scrum#compromiso-equipo).

[Relación entre proveedor y cliente](https://proyectosagiles.org/requisitos-de-scrum#relacion-proveedor-cliente) basada en ganar-ganar, colaboración y transparencia.

[Facilidad para realizar cambios](https://proyectosagiles.org/requisitos-de-scrum#facilidad-cambios) en el proyecto.

[Tamaño de cada equipo](https://proyectosagiles.org/requisitos-de-scrum#tamano-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](https://proyectosagiles.org/requisitos-de-scrum#espacio-comun) para maximizar la comunicación.

Dedicación del [equipo a tiempo completo](https://proyectosagiles.org/requisitos-de-scrum#equipo-tiempo-completo).

[Estabilidad de los miembros del equipo](https://proyectosagiles.org/requisitos-de-scrum#estabilidad-equipo).

**6.**

Simplicidad- Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento.

Comunicación- Si el código es complejo hay que esforzarse para hacerlo entendible, pero siempre intentamos hacerlo de la forma más simple posible.

Retroalimentación- Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto debe conocerse a tiempo real.

Valentía- Intentar siempre diseñar y programar para hoy, no para mañana.

Respeto- Los miembros del equipo se respetan los unos a los otros, porque los programadores no pueden realizar cambios que hacen que las pruebas existentes fallen o que demore el trabajo de sus compañeros.

**7.**

Juego de planificación- Permanente diálogo entre las partes empresarial y técnica.

Pequeñas entregas- Cada versión debe ser lo más pequeña posible.

Metáfora- Es una historia que todo el mundo puede contar acerca de cómo funciona el sistema.

Diseño sencillo- Siempre debemos buscar la manera más simple de realizarlo.

Pruebas- Todo debe ser probado.

Refactorización- Buscamos la manera de hacer nuestro programa más simple sin perder funcionalidad.

Programación por parejas

Integración contínua- El código se debe integrar como mínimo 1 vez al día.

40 horas semanales para estar frescos.

Cliente en casa- El cliente debe sentarse con el equipo de programadores, estar disponible para responder preguntas etc…

Estándares de codificación- se debe de realizar un estándar de codificación aceptado e implementado por todo el equipo.