***PRUEBA DE CONOCIMIENTO.***

***SANTIAGO SEPÚLVEDA BONILLA***

1. ***¿Qué es un Software?***

Un Software es el complemento lógico de un hardware, en él, se realizan todas las tareas referentes al sistema operativo en cuestión. Se especifica sistema operativo, debido a que no necesariamente se refiere a toda la parte “interna” de un PC, sino también de dispositivos móviles, que cuentan con su Software específico basado en su sistema operativo y características.

1. ***¿Qué es ingeniería del software?***

La ingeniería del software es un conjunto de procesos que garantizan el funcionamiento total del mismo, abarca diferentes etapas que, a priori son muy específicas, pero que desencadenan diferentes subprocesos que, al ser llevados a cabo de manera ordenada gracias al ciclo de vida del software (próximamente explicado), determinan y estructuran un proyecto.

1. ***¿Qué es el ciclo de vida del software y de qué etapas se compone?***

El ciclo de vida del software es un conjunto de pasos que se siguen al momento de realizar un proyecto, pasos que funcionan como una estructura firme y sólida para modelar un software, y por lo general, siempre siguen un patrón en espiral que se mantiene indefinidamente para garantizar funcionamiento a futuro, adaptándose a las nuevas tecnologías.

Se compone de 6 etapas:

-Requisitos

-Análisis

-Diseño

-Codificación

-Pruebas

-Despliegue

1. ***¿Qué son metodologías tradicionales y metodologías ágiles?***

Una metodología tradicional es aquella que se trabaja de manera lineal, es decir, etapa por etapa. Para trabajar esta metodología se requiere de una rigurosa planeación. Todo se plasma “sobre el papel” antes de comenzar con su desarrollo en código, y uno de los puntos importantes de esta, es que hasta que no se termine un proceso, no se comienza con el siguiente, lo cual puede ser negativo para la carga de ciertos grupos de trabajo. Un contra que puede tener, es que no se adapta muy bien a muchos proyectos y a sus constantes cambios, sin embargo, suele arrojar resultados positivos en la mayoría de los casos.

Por otro lado, las metodologías ágiles se adaptan al proyecto que se desea realizar. Se tienen unos grupos de trabajo definidos y se intenta sintetizar al máximo tanto el tiempo de desarrollo, como la información planteada dentro del software. Con estas metodologías, es necesario cuidar los detalles, ya que, a la larga, y sin una corrección a tiempo, pueden ser más perjudiciales que un error común.

1. ***Definir 2 metodologías ágiles***

SCRUM: La metodología ágil SCRUM es muy reconocida debido a su efectividad, agilidad, y funcionalidad. En esta, se realizan unos llamados Sprints (que suelen durar entre una y cuatro semanas), en los que equipos pequeños, por lo general entre 4 y 9 miembros.

SCRUM tiene como objetivo entregar resultados de valor en periodos realmente cortos de tiempo, se divide en:

-Sprint: Se divide el proyecto en tareas y se ponen objetivos

-Sprint Planning: todo el equipo define que tareas va a hacer y cuál es su objetivo

-Daily Scrum: Reunión diaria donde se ve que se hizo ayer y que se va a hacer hoy

-Sprint Review: Cuando se finaliza un sprint, se presenta lo desarrollado al cliente y se evalúan funcionalidades

-Sprint Retrospective: Se evalúa como se ha utilizado SCRUM en general a través del ultimo sprint

Su tiempo puede variar y tiene varios roles, el Product owner (Es aquel que tiene comunicación directa con el cliente), el scrum master (Se encarga de que se lleven a cabo todas las metodologías de scrum) y el equipo de desarrollo.

Desarrollo ligero / LEAN

Esta metodología ágil se enfoca en la pronta entrega del proyecto y en la economía. Se intenta reducir el tiempo de desarrollo haciendo pruebas continuas, tanto de ejecución, como de usuario, evitando así retrasos en la entrega, y creando un lazo de confianza con el cliente, debido a que él pasa a ser también, de cierto modo, parte del desarrollo.

Este se basa en ciertos principios, que buscan aligerar la carga del equipo, haciendo así el desarrollo lo más ameno y productivo posible:

-Supresión de desperdicios: Se elimina toda la información y funcionalidades que no sean útiles

-Ampliar aprendizaje: Esto con el objetivo de no acumular errores, y resolverlos rápidamente

-Decidir lo más tarde posible: Aunque no va de la mano con ser metodología ágil, se intenta llevar siempre la opinión del cliente de manera específica, sino, no se lleva a cabo, o al menos, no en profundidad.

-Entregar tan rápido como sea posible

-Potenciar el equipo: Utilizar las fortalezas de cada miembro del equipo es clave, así se integrarán más fácil al proyecto

-Crear integridad: Se hacen las constantes anteriormente mencionadas pruebas para crear integridad entre las diferentes partes del Software

-Visualización del proyecto: Esto con el fin de proyectar a futuro, probablemente inmediato, mejoras notables en cuanto a accesibilidad y funcionalidad (Recordemos que este análisis busca aplicar funcionalidades útiles, que, aunque suena redundante, es una de las bases de esta metodología).

1. ***¿Qué es un sistema operativo?***

Es el software básico de un equipo, y es aquel que permite la interacción entre el usuario y este. A través del sistema operativo, se pueden administrar los recursos a disposición.

Los sistemas operativos más utilizados y reconocidos son Windows (En 2020, se siguen usando versiones anteriores, siendo incluso más utilizada la versión 7 que la 8.1, pero prevaleciendo la 10 como la primera), Mac OS, Linux y Ubuntu, y en móviles, IOS y Android.

1. ***¿Qué es un patrón de arquitectura?***

Los patrones de arquitectura son esquemas utilizados para definir la estructura de un software, para así estandarizar cómo se va a llevar a cabo un proyecto, y sea más cómodo para todos los desarrolladores. En términos prácticos, se puede decir que un patrón es un idioma en el cual todos los desarrolladores van a hablar, de modo que siempre podrán ir de la mano sin discrepar en sus acciones e instrucciones. Estos patrones disminuyen notablemente los problemas en el desarrollo, y aunque no siempre se suelen seguir, actualmente son fundamentales para un diseño óptimo de software.

1. **Definir 2 patrones arquitectónicos**

Patrón Vista controlador:

Separa los datos de la aplicación, la lógica de control y la interfaz.

Es de los modelos más maduros actuales y de los que más se utiliza.

Se separa en:

-Vista: Es la interfaz del usuario, contiene los mecanismos de interacción con el usuario.

-Modelo: Lógica de negocio, contiene una representación de los datos del sistema y es el responsable de acceder a la capa de almacenamiento de datos.

-Controlador: Es el intermediario entre el modelo y la vista, gestiona el flujo de datos entre estos y transforma los datos para adaptarlos a las necesidades de cada uno.

Patrón filtro de tubería:

Este patrón se utiliza al momento de realizar un software en el cual se necesita llevar la información a través de varios filtros, esto con el fin de depurar toda la que no se necesite para la búsqueda final. Es un patrón que convierte la información de entrada en salida; básicamente, busca ofrecer una respuesta a una instrucción dada por el usuario.