Prueba técnica training calidad ***Sofka***

Cristian Camilo Arango Fernández
Contacto: 3164989470

1. **Calidad de software:** cuando hablamos de calidad, se hace referencia al cumplimiento de un conjunto de características asociadas a un objeto, cuyos enfoques pueden variar, teniendo como partida un objeto a comparar o criterios asociados a los requerimientos de dicho producto.
   En software, la calidad se puede visualizar desde diferentes enfoques. Teniendo una percepción desde el cliente, un software con calidad debe cumplir los requerimientos hechos por este, debe tener un soporte ante cualquier anomalía que se presente en el camino, además se deben tener en cuenta criterios como usabilidad, eficiencia, eficacia, entre otros que respalden la funcionalidad del software. Otro enfoque es el del desarrollador, que, si desea que su software sea de calidad, debe satisfacer todas las necesidades del cliente, cumplir las expectativas y comprometerse con su buen funcionamiento.
   En software, la calidad puede ser subjetiva, puesto que la visión de un cliente puede distanciarse de la visión del desarrollador, por eso es importante la constante comunicación y el respaldo, ya que la percepción que se deje al cliente, puede incidir en trabajos futuros con el mismo cliente, o con clientes cercanos al entorno del cliente anterior.
   A modo personal, siento que los aspectos más importantes en la calidad son satisfacer al cliente cumpliendo con todos sus requerimientos, y que el funcionamiento de dicho software funcione de manera eficiente proporcionando al cliente un respaldo o garantía ante cualquier problema que pueda surgir en el camino.

2. Control de versiones: Esto consiste en administrar los cambios y adecuaciones hechas sobre un repositorio en donde el sistema registra los archivos a lo largo del tiempo.
   a. Control de versiones centralizado: Basado en la relación cliente servidor, en donde todos los ajustes hechos a un repositorio, debe transitar por un servidor central.
   b. Control de versiones distribuido: Ajusta las modificaciones entre iguales, es decir que permite trabajar a muchos desarrolladores en un proyecto en común.

   Por lo tanto, la diferencia entre estos dos tipos es que, en un control de versiones distribuido, se pueden hacer operaciones comunes sin necesidad de comunicarse con un servidor central.

3. **Comando para clonar en git**: El comando para clonar un proyecto es *git clone*. Con este se obtiene una copia de un proyecto guardado en un repositorio existente en Git. La manera en cómo se puede clonar con dicho comando es:
   $ git clone [url del repositorio].

4. Comandos de git
   a. git add –miClase: Estte comando agrega un archivo nuevo o modificado al directorio de trabajo. En este caso, se adiciona o modifica un fichero llamado miClase sin eliminar los demás.
   b. git add –A: Agrega todos los archivos desde el repositorio local al directorio de trabajo.
   c. git status: Muestra el estado de diferencia entre los ficheros locales y los montados en el repositorio. Se pueden visualizar los cambios entre ambas etapas.
   d. git pull: Trae e integra al repositorio cambios que se hayan hecho desde otros colaboradores
   e. git push: Sube o actualiza en el repositorio cambios que se hayan hecho desde los ficheros locales.
   f. git commit –m “Hola”: Se incluye un mensaje al momento de actualizar o modificar el repositorio, En este caso, guarda el mensaje “Holas” como comentario.
   g. git log: Muestra los comentarios realizados durante las actualizaciones del repositorio.

5. Herencia y polimorfismo: Herencia es la práctica de ampliar una funcionalidad definiendo una nueva clase que hereda funcionalidad de una clase existente. Polimorfismo es la función que permite enviar mensajes a objetos de tipos distintos