Brian Torres

24042974

CUESTIONARIO

1. **Defina la diferencia entre un Sistema de Control de Versiones Centralizado y Distribuido.**

- No existe una copia de referencia del código base por defecto; solo copias de trabajo.
- Operaciones comunes (como commits, visualización del historial, y revertir cambios) son rápidas, porque no existe necesidad de comunicación con un servidor central.

2. **Explique en qué consisten los estados de Git, y de un ejemplo que ilustre los mismos.**

El directorio de Git es donde Git almacena los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando clonas un repositorio desde otro ordenador.

El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

El área de preparación es un sencillo archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice, pero se está convirtiendo en estándar el referirse a ella como el área de preparación.

El flujo de trabajo básico en Git es algo así:

1. Modificas una serie de archivos en tu directorio de trabajo.
2. Preparas los archivos, añadiendolos a tu área de preparación.
3. Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esas instantáneas de manera permanente en tu directorio de Git.

3. **Explique el término *debugging* y su importancia.**

Es el proceso de identificar y corregir errores de programación. En inglés se conoce como debugging, porque se asemeja a la eliminación de bichos (bugs), manera en que se conoce informalmente a los errores de programación.

El término bug proviene de la época de las computadoras de válvula termoiónica, en los cuales los problemas se generaban por los insectos que eran atraídos por las luces y estropeaban el equipo. Si bien existen técnicas para la revisión sistemática del código fuente y se cuenta con medios computacionales para la detección de errores (depuradores) y facilidades integradas en los sistemas lower CASE y en los ambientes de desarrollo integrado, sigue siendo en buena medida una actividad manual, que desafía la paciencia, la imaginación y la intuición de programadores.

Como el software y los sistemas electrónicos se vuelven generalmente más complejos, se han desarrollado varias técnicas comunes de depuración para detectar anomalías, corregir funcionalidades y optimizar código fuente.

4. **Defina el término *plug-in*.**

Es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz de programación de aplicaciones.

5. **Defina *JUnit 5* y su composición. Explique brevemente cada uno de sus componentes.**

Está compuesto por varios módulos diferentes de tres subproyectos diferentes.

JUnit 5 = Plataforma JUnit + JUnit Jupiter + JUnit Vintage

La plataforma JUnit sirve como base para lanzar marcos de prueba en la JVM. También define la API de TestEngine para desarrollar un marco de prueba que se ejecuta en la plataforma. Además, la plataforma proporciona un Iniciador de consola para iniciar la plataforma desde la línea de comandos y compilar complementos para Gradle y Maven, así como un Runner basado en JUnit 4 para ejecutar cualquier TestEngine en la plataforma.

JUnit Jupiter es la combinación del nuevo modelo de programación y extensión para escribir pruebas y extensiones en JUnit 5. El subproyecto Jupiter proporciona un TestEngine para ejecutar pruebas basadas en Jupiter en la plataforma.

JUnit Vintage proporciona un TestEngine para ejecutar pruebas basadas en JUnit 3 y JUnit 4 en la plataforma.