**ANÁLISIS DESARROLLO DE SOFTWARE**

**TEMA:**

Versionamiento

**INSTRUCTOR:**

Jabier Leonardo Pineda Uribe

**ELABORADO POR:**

Keiner Pedrozo Ospino

**FICHA: 3065835 – 2**

SERVICIO NACIONAL DE APRENDIZAJE SENA

BOGOTÁ D.C., OCTUBRE DE 2024

1. **¿Qué es un repositorio y mencione 2 ejemplos de los mismos (proveedor online)?**

Un repositorio es un lugar de almacenamiento donde se guarda y gestiona el código fuente de proyecto software, en un repositorio se puede organizar el código, realizar un seguimiento de los cambios mediante control de versiones, y colaborar con otros desarrolladores de manera eficiente. Los repositorios permiten tener un historial completo de versiones de proyectos facilitando la colaboración y la mejora continua.

Ejemplo:

* **GITHUB:** es una de las plataformas más populares para alojar repositorios de códigos, ofrec}
* e característica como control de versiones, gestión de proyectos, integración con CI/CD, y permite colabora fácilmente con otros desarrolladores, soportas GIT como sistema de control de versiones
* **GITLAB:** al igual que GITHUB, GITLAB ofrece almacenamiento de código control de versiones, pero también integra herramientas de DEVOPS completa para el ciclo de vida del desarrollo incluyendo el CI/CD gestión de seguridad y implementación. También utiliza GIT como sistema de control de versiones.

1. **¿Cuáles son los principales componentes de un versionamiento en la herramienta GIT?**

GIT, como sistema de control de versiones distribuidos, permite a los desarrolladores llevar un seguimiento detallados de los cambios en el código, los principales componentes que forman parte del versionamiento en GIT son:

* Repository (Respositorio)
* Commit
* Branch (Rama)
* Stanig Area (Área de Preparación)
* Head
* Merge (Fusión)
* Remote (Remoto)
* Tag (Etiqueta)

1. **¿Mencione con sus propias palabras las ventajas que tiene GIT frente a otros proveedores de repositorio?**

Git ofrece varias ventajas frente a otros sistemas de control de versiones, lo que lo puede o lo convierte en una de las herramientas más asequible entre los desarrolladores.

1. **Mencione por lo menos 5 ejemplos de los comandos básicos que se usan en GIT.**

A continuación daré mis 5 ejemplos

* **Git init:** Este comando se utiliza para crear un nuevo repositorio GIT vacío en un proyecto nuevo.

* **GIT clone <UERL>:** Este comando se utiliza para copiar un reposciroio remoto a una maquina local. Clona todo el historial del repositorio y crea una copia completa para trabajar
* **GIT ADD <Archivo>:** Con este comando, se agregan cambios específicos (o archivos) al área de preparación (Staging area) para ser incluido en el próximo compromiso. Se puede utilizar GIT ADD para agregar a todos los cambios de forma masiva.
* **GIT PUSH <REMOTO><RAMA>:** Este comando envía los cambios confirmados en el repositorio local al repositorio remoto (por ejemplo, ORIGIN) y la rama a la que desean enviar los cambios (por ejemplo, MAIN)
* **GIT COMMIT-M “MENSAJE”:** este comando se utiliza para guardar los cambios en el repositorio local. el argumento -M permite agregar un mensaje que describe lo que se ha modificado en este Commit.

1. **¿Que son y cuales son las funciones de los Branch?**

Las ramas (ramas) en Git son una característica fundamental que permite a los desarrolladores trabajar de manera aislada en diferentes versiones del código. Una rama es esencialmente una línea independiente de desarrollo que permite realizar cambios sin afectar el código en otras ramas, especialmente la rama principal (comúnmente llamada maino master).

**Funciones de las Ramas :**

* Aislamiento de cambios: Las ramas permiten a los desarrolladores trabajar en nuevas características, correcciones de errores o experimentos sin interferir con el código en la rama principal o en otras ramas. Esto minimiza el riesgo de introducir errores en el código estable.
* Desarrollo Paralelo: Varios desarrolladores pueden trabajar simultáneamente en diferentes ramas, lo que facilita la colaboración. Cada uno puede desarrollar su función o solución sin conflictos inmediatos, y luego integrar esos cambios cuando estén listos.
* Instalación para Pruebas y Experimentos: Las ramas permiten a los desarrolladores probar nuevas ideas o características sin comprometer el trabajo en la rama principal. Si algo no funciona como se esperaba, la rama puede eliminarse sin afectar el proyecto principal.
* Control de versiones: Usar ramas facilita el seguimiento de cambios específicos relacionados con una característica o corrección. Cada rama puede tener su propio historial de confirmaciones, lo que hace más fácil entender el desarrollo del proyecto.
* Integración y Fusión: Una vez que se haya completado el trabajo en una rama, se puede fusionar de nuevo en la rama principal o en otra rama. Git proporciona herramientas para resolver los conflictos que puedan surgir durante este proceso, garantizando que todos los cambios se integren de manera efectiva.

**Ejemplo de uso de Branches:**

Imagina que estás trabajando en una aplicación y deseas agregar una nueva función, como un sistema de comentarios. Crea una rama llamada feature/comentarios para desarrollar esta nueva funcionalidad. Una vez que haya terminado y probado el código, puede fusionar esta rama de nuevo mainpara que todos los usuarios puedan acceder a la nueva función.

Bueno las ramas son esenciales en Git para facilitar el trabajo colaborativo, permitir el desarrollo seguro y mantener un control claro sobre el progreso del proyecto.

1. **¿Cuál es el Nombre del branch principal?**

El nombre de la rama principal en Git suele ser main. Sin embargo, en algunos proyectos más antiguos o configuraciones por defecto, la rama principal puede llamarse master.

A partir de 2020, muchos nuevos repositorios y plataformas como GitHub y GitLab han comenzado a utilizar maincomo la rama principal por defecto para promover un lenguaje más inclusivo. Si trabajas en un proyecto nuevo, es probable que encuentres la rama principal con el nombre main.