**ESCUELA POLITECNICA NACIONAL**

**Parte 1: Investigación Teórica**

Nombre: José Morocho, Victor Narváez, John Cacuango Fecha: 04-08-2025 GRUPO 2

**1. ¿Qué es un sistema de control de versiones y por qué es útil en proyectos colaborativos?**

Un sistema de control de versiones es una herramienta que permite gestionar los cambios realizados sobre archivos, especialmente en proyectos de desarrollo de software. Su utilidad en proyectos colaborativos radica en que:

* Permite a varios usuarios trabajar en el mismo proyecto sin sobrescribir el trabajo de otros.
* Guarda un historial de cambios, facilitando volver a versiones anteriores.
* Ayuda a identificar errores introducidos en versiones específicas.
* Mejora la organización y la colaboración entre equipos.

**2. Explica los conceptos: repositorio, branch, merge y conflictos de código.**

* **Repositorio:** Es el lugar donde se almacena todo el proyecto y su historial de cambios. Puede estar en local o en plataformas en línea como GitHub.
* **Branch (rama):** Una copia paralela del código principal donde se pueden realizar modificaciones sin afectar la versión principal (main o master).
* **Merge (fusión):** Es el proceso de unir los cambios de una rama con otra, generalmente la rama principal.
* **Conflictos de código:** Ocurren cuando dos ramas modifican la misma línea de código y Git no puede decidir cuál conservar. Es necesario resolverlos manualmente.

**3. ¿Qué diferencia hay entre un sistema centralizado (SVN) y uno distribuido (Git)?**

* **SVN (centralizado):** Todos los usuarios dependen de un único servidor central. Si el servidor falla, se puede perder información o detener el desarrollo.
* **Git (distribuido):** Cada usuario tiene una copia completa del repositorio. Esto permite trabajar sin conexión y hace que el sistema sea más robusto ante fallos.

**4. Describe un flujo de trabajo típico con Git en un equipo (incluir creación de ramas, pull requests y fusiones).**

Un flujo de trabajo típico es:

1. Clonar el repositorio.
2. Crear una nueva rama para una tarea específica (git checkout -b nueva-funcionalidad).
3. Realizar cambios y confirmarlos con git commit.
4. Subir la rama al repositorio remoto (git push origin rama).
5. Crear un pull request para revisar los cambios.
6. Revisar y aprobar el pull request.
7. Fusionar la rama al main con git merge.
8. Resolver posibles conflictos si los hay.