UD1- Actividad 1

Se plantea una tarea de documentación sobre las plataformas de control de versiones con los siguientes puntos:

1. Propiedades y funcionalidades de las plataformas de control de versión.

**Registro de cambios:** Guardan el historial de modificaciones en un proyecto, permitiendo ver quién hizo qué y cuándo.

**Colaboración:** Varias personas pueden trabajar en el mismo proyecto sin sobrescribir el trabajo de otros.

**Ramas:** Permiten trabajar en diferentes partes del proyecto al mismo tiempo sin afectar la versión principal.

**Deshacer cambios:** Facilitan revertir errores al volver a versiones anteriores del proyecto.

**Sincronización:** Permiten tener copias locales y remotas del proyecto, lo que facilita el trabajo en equipo desde diferentes lugares.

2. Plataformas más importantes: características y diferencias.

**GitHub** - Características:

Alojamiento en la nube de repositorios Git.

Funciones de colaboración (pull requests, issues).

GitHub Actions para automatización (CI/CD).

Gratuito para proyectos públicos.

Diferencia: La plataforma más popular, especialmente para proyectos de código abierto.

**GitLab** - Características:

Alojamiento de repositorios Git.

Funciones de integración y entrega continua (CI/CD) muy avanzadas.

Puedes alojarlo en tu propio servidor.

Diferencia: Ideal para proyectos internos y empresas por sus opciones de seguridad y personalización.

**Bitbucket** - Características:

Soporte para Git y Mercurial (otro sistema de control de versiones).

Integración con Jira para gestión de proyectos.

Enfocado en equipos y empresas.

Diferencia: Bien integrado en el ecosistema de Atlassian (Jira, Trello).

3. Principales aplicaciones de cliente de control de versiones.

**Git (CLI)**

Es la herramienta principal para trabajar con Git desde la línea de comandos. Permite realizar todas las operaciones de control de versiones (commit, push, merge).

Desarrolladores que prefieren controlar versiones directamente desde la terminal.

**GitHub Desktop**

Aplicación de escritorio con una interfaz gráfica fácil de usar que permite interactuar con Git y GitHub sin usar la línea de comandos.

Ideal para principiantes o quienes prefieren una experiencia visual.

**GitKraken**

Un cliente de Git con interfaz gráfica avanzada, muy visual para gestionar ramas, commits y merges. Compatible con GitHub, GitLab, Bitbucket.

Desarrolladores que prefieren una interfaz visual rica en funcionalidades.

**SourceTree**

Cliente de Git y Mercurial de interfaz gráfica desarrollado por Atlassian. Facilita la visualización de cambios y manejo de repositorios.

Muy usado en equipos que también trabajan con Bitbucket y otras herramientas Atlassian.

**Tower**

Cliente de Git con interfaz limpia y profesional para Mac y Windows, que permite gestionar proyectos fácilmente.

Desarrolladores que buscan una herramienta profesional y bien diseñada.

4. Control de versiones integrados en los IDES (plugins)

**VSCode:**

Integrado: Soporte para Git básico (commits, pushes).

Plugins:

GitLens: Ver quién hizo cada cambio.

Git Graph: Visualiza el historial de ramas.

**IntelliJ IDEA:**

Integrado: Soporte para Git, GitHub, SVN, etc.

Plugins:

GitToolBox: Mejora el control de versiones con notificaciones.

**Eclipse:**

EGit: Plugin para trabajar con Git.

Subclipse: Para Subversion (SVN).

**NetBeans:**

Integrado: Soporte para Git, Mercurial y Subversion.

**Atom:**

Integrado: Soporte básico para Git.

Plugin GitHub: Gestión de repositorios de GitHub.