Sistemas de Control de Versiones

Programación Visual

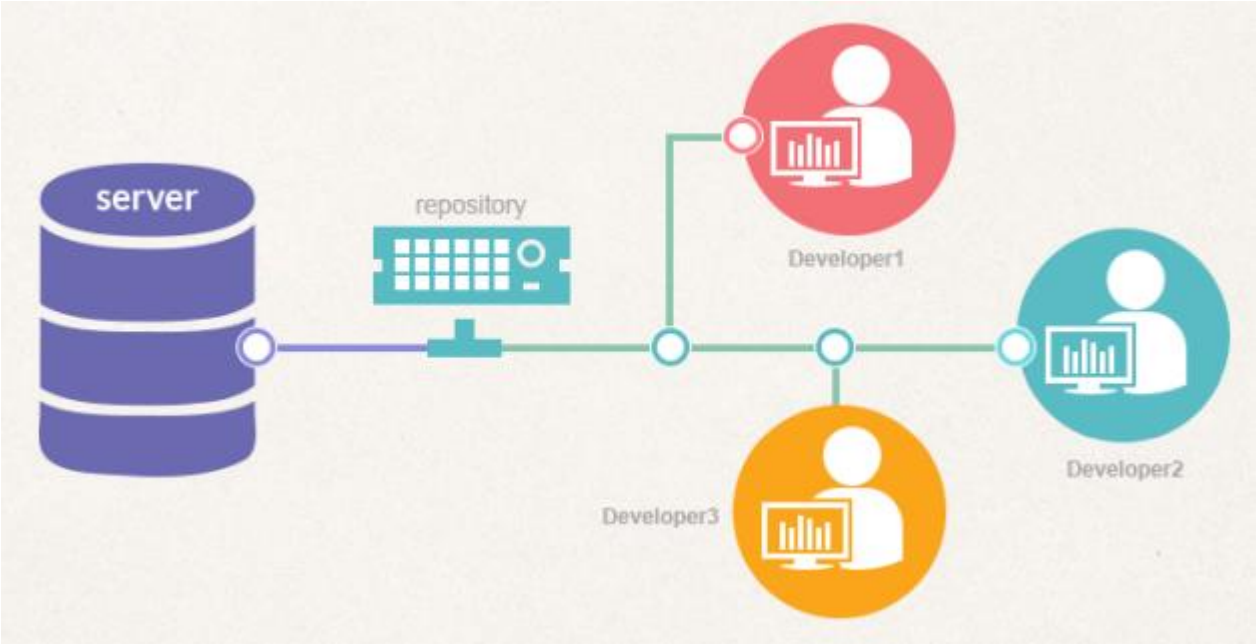
Extensión Áulica San Pedro

UNJu – Facultad de Ingeniería

¿Qué es un sistema de control de versiones?

- Permite implementar la codificación de un proyecto en forma colaborativa.
- Sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante.
- Permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, entre otras cosas
- Aplica para cualquier tipo de archivos, no necesariamente código fuente

Sistema de Control de Versiones



Repositorio y Copia de Trabajo

- Repositorio
  - Lugar en el que se encuentran almacenados los archivos con sus respectivas versiones
- Copia de Trabajo
  - Inicialmente es el directorio que se obtiene al hacer el **check-out** o **clonación** de un proyecto almacenado en un repositorio.
  - La copia de trabajo contiene información para saber qué ficheros están sincronizados y cuáles no.
  - También contiene los archivos que permiten mantener la sincronización con el repositorio

Git

- Nació en 2005
- Otros sistemas tienden a almacenar los datos como cambios de cada archivo respecto a una versión base.
- Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos.
- Cada vez que se confirma un cambio, o se guarda el estado del proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea

Implementaciones de Git

- GigHub
- GitLab
- Diferencias

GitHub	GitLab
Los elementos se pueden rastrear en varios repositorios	Los elementos no se pueden rastrear en varios repositorios
Los repositorios privados exigen la versión de pago	Los repositorios privados se permiten en la versión gratuita
No hay opción gratuita de hospedaje en servidor propio	Opción gratuita de hospedaje en servidor propio
Integración continua solo mediante herramientas de terceros como Travis CI, CircleCI etc.	Integración continua gratuita incluida
No cuenta con plataforma de implementación integrada	Implementación de software a través de Kubernetes
Rastreo completo de comentarios	Sin rastreo de comentarios
No hay opción de exportación de elementos como archivo CSV	Opción de exportación de elementos como archivo CSV por correo electrónico
Panel personal para rastrear elementos y solicitudes pull	Panel de análisis para planificar y supervisar proyectos

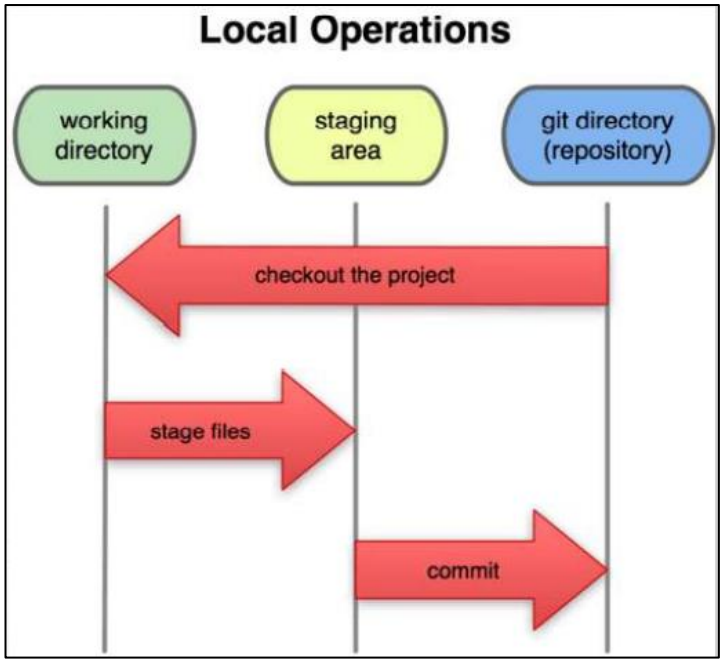
Estados

- Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (**committed**), modificado (**modified**), y preparado (**staged**).

  - **Confirmado:** significa que los datos están almacenados de manera segura en tu base de datos local.
  - **Modificado:** significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
  - **Preparado:** significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación

Secciones

- **El directorio de Git (Git directory):** es donde Git almacena los metadatos y la base de datos de objetos para el proyecto. Es la parte más importante de Git, y es lo que se copia cuando se clona un repositorio desde otro ordenador.
- **El directorio de trabajo (working directory):** 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 usar o modificar.
- **El área de preparación (staging area):** 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.



Comandos

- **git clone [url]:** obtiene copia del repositorio de la URL
- **git add [archivo]:** agrega archivo para ser controlado
- **git commit –m 'comentario':** confirma los cambios y quedan con el comentario ingresado en el historico
- **git status:** identifica qué archivos se encuentran en que estados y da información sobre el branch actual
- **git branch [nombre]:** crea una rama
- **git push [nombre-remoto][nombre-rama]:** sube cambios a repositorio remoto
- **git pull:** para recuperar y unir automáticamente la rama remota con tu rama actual.
- …otros

Ignorar archivos

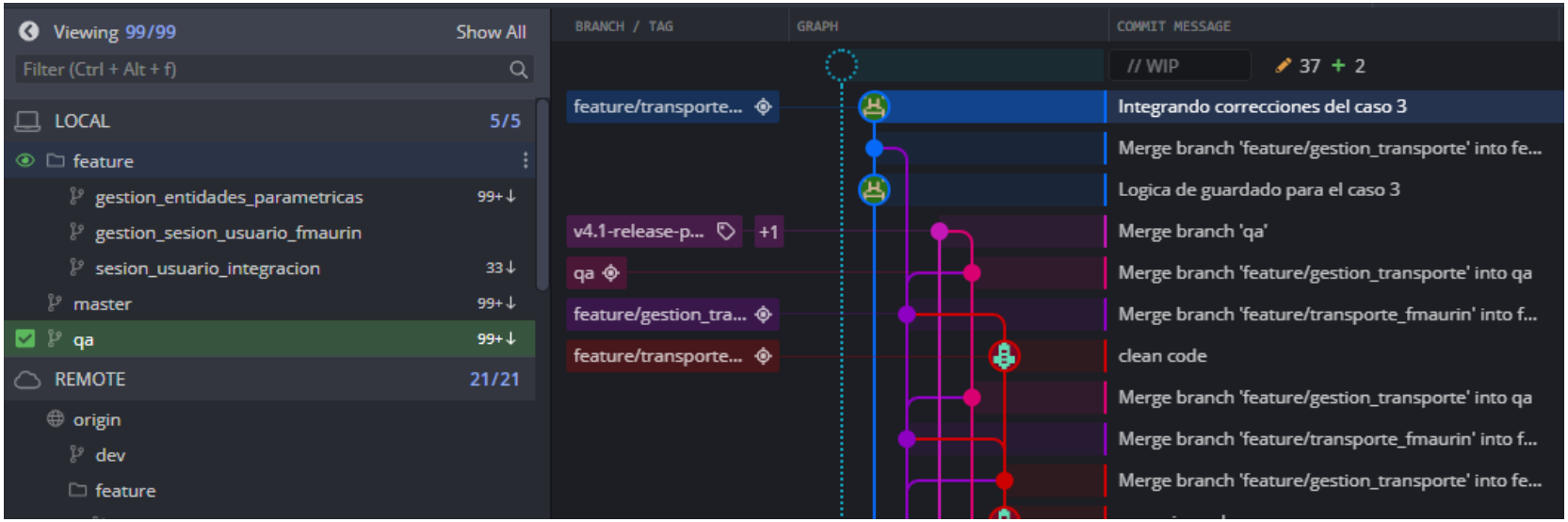
- Para estos casos se crea un archivo llamado **.gitignore**, en el que se listan los patrones de nombres que se desean ignorar.
- Por ejemplo:
  - *.[oa]
  - *~
  - miArchivo.extensión
- La primera excluye todos los archivos terminados en .a o .o, la segunda todos los que terminan en tilde y la tercera el archivo con nombre miArchivo.extensión

Ramas o Branches propuestos por Git Flow

- **Master:** se usa para tener la versión productiva
- **Develop:** se usa para integrar ramas y realizar regresiones
- **Hotfix:** estas ramas se usan para realizar ajustes a bugs que se encuentran sobre master. Se replican los cambios una vez son exitosos en master a develop.
- **Features:** branches usados para la creación de nuevas funcionalidades
- **Releases:** branches usados para paquetes que serán pasados a producción

Ramas o Branches propuestos por Git Flow

- Ejemplo



Herramientas

- Existen muchas herramientas para gestionar los comandos de Git mediante Interfaz de usuario
- Por ejemplo
  - GitKraken
  - GitAHead
  - SmartGit
  - Tortoise Git
  - IDE (Spring Tools)
  - …otros