Introducción a la Programación y Computación 1

Introducción al Manejo de Control de Versiones

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias Y Sistemas

Agenda

RECORDATORIOS

¿Qué es el control de versiones?

¿Por qué es importante el versionamiento?

Tipos de control de versiones

Ventajas del control de versiones

Herramientas populares de control de versiones

Introducción a Git

Instalación de Git

Crear un repositorio

Realizar un commit

Buenas prácticas en control de versiones

Competencias

Al finalizar esta unidad, el estudiante será capaz de:

- **Comprender** la importancia del control de versiones para la integridad y trazabilidad de los proyectos de software.
- **Aplicar** el flujo de trabajo básico en Git: inicializar repositorios, preparar archivos y confirmar cambios (commits).
- **Gestionar** ramas (branches) para desarrollar nuevas funcionalidades de manera aislada y segura.
- **Resolver** conflictos de fusión básicos durante la integración de código.
- **Utilizar** plataformas remotas como GitHub para el alojamiento y colaboración en la nube.

Valores y Actitudes

- **Orden y Disciplina:** Mantener un historial de cambios limpio, frecuente y con mensajes descriptivos para facilitar el seguimiento.
- **Colaboración Respetuosa:** Trabajar en equipo evitando sobrescribir el trabajo ajeno mediante el uso correcto de ramas y comunicación.
- **Responsabilidad:** Asegurar que el código funciona correctamente antes de fusionarlo a la rama principal o compartirlo en el repositorio remoto.

¿Qué es el control de versiones?

- El control de versiones es un sistema que registra los cambios realizados a un archivo o conjunto de archivos a lo largo del tiempo. Esto permite recuperar versiones anteriores, comparar cambios, y trabajar de forma ordenada y segura. Es una herramienta indispensable para cualquier desarrollador, ya que previene la pérdida de trabajo y facilita la colaboración.

¿Por qué es importante el versionamiento?

- Permite mantener un historial de modificaciones.
- Facilita el trabajo en equipo en proyectos grandes.
- Ayuda a detectar y corregir errores rápidamente.
- Permite revertir el código a un estado funcional anterior.

Ejemplo cotidiano: guardar copias de seguridad de un documento con nombres como "proyecto_final_v1", "proyecto_final_definitivo", etc. El control de versiones automatiza y mejora este proceso.

Tipos de control de versiones

1. Centralizado (CVCS): Un único servidor contiene todas las versiones. Ej: Subversion (SVN).
2. Distribuido (DVCS): Cada desarrollador tiene una copia completa del repositorio. Ej: Git.

Git ha ganado popularidad por su flexibilidad y capacidad de trabajo offline.

Ventajas del control de versiones

- Historial detallado de todos los cambios.
- Posibilidad de experimentar en ramas sin afectar el proyecto principal.
- Comparación entre versiones.
- Trabajo colaborativo sin sobrescribir el trabajo de otros.

Herramientas populares de control de versiones

- Git: El más usado actualmente.
- Mercurial: Similar a Git, pero con una curva de aprendizaje más suave.
- Subversion (SVN): Antiguo estándar, aún usado en algunos entornos corporativos.
- GitHub / GitLab / Bitbucket: Servicios para alojar repositorios Git y colaborar en línea.

Introducción a Git

Git es un sistema de control de versiones distribuido, creado por Linus Torvalds en 2005. Es rápido, eficiente y diseñado para coordinar el trabajo entre desarrolladores.

Git no se limita a programadores: escritores, diseñadores y científicos también lo usan para gestionar cambios en sus proyectos.

Instalación de Git

- Windows: Descargar desde git-scm.com.
- Linux: sudo apt install git
- Mac: brew install git
- Verificar instalación: git --version

¿Qué es un repositorio?

Un repositorio es un espacio donde se almacena toda la información del proyecto y su historial de cambios. Existen dos tipos:

- Repositorio local: Se encuentra en el equipo del desarrollador.
- Repositorio remoto: Está alojado en servicios en línea como GitHub, GitLab o Bitbucket.

La conexión entre ambos permite trabajar de forma flexible y colaborativa.

Ciclo de trabajo básico en Git

Git organiza el trabajo en varias etapas:

- Se crean o modifican archivos dentro del proyecto.
- Se seleccionan los archivos que se desean guardar mediante una acción llamada “preparación”.
- Finalmente, se registra el cambio en el historial con un mensaje descriptivo.

Este ciclo ayuda a mantener un control detallado sobre cada avance realizado.

Estados de los archivos

En Git, los archivos pueden tener diferentes estados:

- No rastreados: Archivos nuevos que aún no están siendo seguidos por el sistema.
- Preparados (staged): Listos para ser registrados en el historial.
- Confirmados (committed): Guardados de forma permanente en el historial del proyecto.

Comprender estos estados es esencial para manipular correctamente los cambios.

El archivo .gitignore

En un proyecto pueden generarse archivos que no necesitan ser almacenados ni compartidos, como archivos temporales o carpetas de dependencias. El archivo .gitignore permite especificar qué elementos deben ser ignorados por el control de versiones.
Esto evita subir información innecesaria o sensible al repositorio.

¿Qué es el historial de cambios?

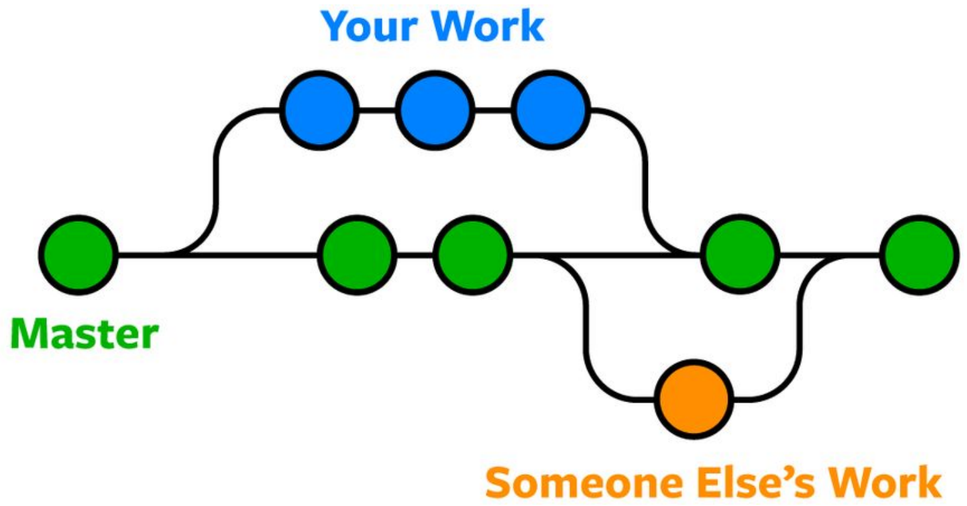
- Cada vez que se registra un cambio, Git guarda información como:
- El autor del cambio.
- La fecha y hora.
- Un mensaje que explica el motivo del cambio.
- Este historial es muy valioso, ya que permite revisar qué se hizo, cuándo y por qué. Es como un diario de desarrollo.

Revertir cambios

En cualquier momento, es posible volver atrás si algo sale mal. Git permite deshacer cambios recientes o incluso recuperar versiones anteriores del proyecto. Esto da mucha seguridad al programar, ya que se puede experimentar sin miedo a perder el progreso.

Uso de ramas (branches)

Una rama es como una línea paralela de trabajo donde se puede desarrollar una nueva funcionalidad sin afectar el resto del proyecto. Esto permite trabajar en diferentes ideas o soluciones de forma independiente. Luego, cuando se haya terminado, se puede unir con la rama principal.



Fusión de ramas

- Al completar una tarea en una rama, se puede integrar de nuevo con la rama principal del proyecto. Esta acción se conoce como "fusión". Git intentará combinar los cambios automáticamente, pero si hay conflictos, el desarrollador debe decidir qué cambios conservar. Este proceso es esencial en proyectos colaborativos.

¿Qué es un conflicto?

- Un conflicto ocurre cuando dos personas modifican la misma parte del código y Git no sabe qué versión conservar. Cuando esto sucede, se alerta al equipo para que revise el archivo y decida cómo combinar los cambios. Aunque puede parecer complicado al inicio, es parte natural del trabajo en equipo.

Introducción a GitHub

•GitHub es una plataforma en línea que aloja proyectos gestionados con Git. Además de almacenar el código, GitHub ofrece herramientas para la colaboración como:

•Seguimiento de tareas (issues)

•Revisión de cambios (pull requests)

•Automatización de procesos (actions)

•Es una herramienta esencial en el entorno profesional de desarrollo de software.

Clonar un proyecto existente

- Si ya existe un repositorio remoto, cualquier persona con acceso puede descargar una copia en su computadora para trabajar en él. A esto se le llama “clonar un repositorio”. Esta acción crea una versión exacta del proyecto, incluyendo todo su historial.
- Una vez se han hecho avances en el proyecto, es posible compartirlos con los demás integrantes subiendo las modificaciones al repositorio remoto. Este proceso permite que todos trabajen de forma sincronizada, evitando duplicaciones o sobrescrituras.

Colaboración con Fork y Pull Request

- En proyectos públicos o de código abierto, muchas veces no se tiene acceso directo para editar. En estos casos, se puede hacer una copia del proyecto (fork), realizar cambios en la copia, y luego enviar una solicitud al autor original (pull request) para que evalúe y acepte las mejoras.
- Este modelo de colaboración es ampliamente usado en comunidades como GitHub.

Conclusiones

Referencias

- **Sitio Oficial de Git:** Documentación y descargas. Disponible en: *git-scm.com*.
- **Chacon, S., & Straub, B.** *Pro Git*. Libro oficial y guía completa. Disponible gratuitamente en: *git-scm.com/book/es/v2*.
- **GitHub Docs:** Guías de colaboración y gestión de repositorios remotos. Disponible en: *docs.github.com*.
- **Atlassian Git Tutorial:** Conceptos de versionamiento distribuido y centralizado.

¡GRACIAS POR SU ATENCIÓN!

DUDAS

RECUERDA QUE TENEMOS NUESTRO FORO SEMANAL DONDE PUEDES CONSULTAR CUALQUIER DUDA QUE TE SURJA EN LA SEMANA