Año 2025

Proyecto de Software
Actividad 1
Versionado de código

Objetivo

La finalidad de esta actividad es fijar algunos de los conceptos básicos para configurar y comenzar a utilizar **Git** correctamente.

Introducción

Las herramientas de versionado de código fuente son fundamentales a la hora de llevar adelante un desarrollo de software. Resulta prácticamente imposible que en algún momento de su trayectoria laboral no tengan que hacer uso de una herramienta de este tipo para alguna de sus actividades, y por este motivo es muy importante que aprendan a usarlas correctamente. Estas herramientas permiten desarrollar software entre un gran número de personas, un claro ejemplo de esto es la comunidad del Kernel de Linux, en la que participan casi 17.000 desarrolladores/as.

Herramientas a utilizar

Deben utilizar la herramienta Git en sus entornos locales y el servidor Gitlab de la Cátedra para realizar la actividad.

Tareas

1. Instalar Git en sus máquinas de trabajo.
2. Configure su identidad globalmente utilizando el comando **git config**. Principalmente deberá configurar el user.name y el user.email, **que son fundamentales para que los commits de cada integrante se registren correctamente**.
3. Para poder trabajar de forma cómoda y segura, es necesario crear una clave SSH. De esta manera podrán conectarse al servidor remoto sin suministrar el nombre de usuario y contraseña cada vez que sea necesario subir los cambios al repositorio. Cada miembro del equipo deberá generar su clave SSH siguiendo los pasos que aquí se detallan, a su vez, deberán agregar a Gitlab la clave generada, siguiendo estos pasos.
4. Uno de los miembros deberá crear el archivo **README.md** con el contenido: `# GrupoXX`. Donde XX es el número de grupo. Una vez realizado el cambio debe subirse al repositorio remoto en el branch main que es la rama principal por defecto.
5. En Git existe el concepto de **Branching** (ver fig. 1), teniendo en cuenta esto deberán

configurar sus ramas principales para trabajar correctamente en equipo. En este sentido, deberán generar una nueva rama llamada **development,** partiendo de la rama principal **main.** Una vez creada la rama **development**, deberá subirse al repositorio remoto.

6. Una buena práctica de git consiste en tratar de mantener la rama **main** lo más estable posible, por lo tanto debe asegurarse que el código subido a esta rama haya sido probado suficientemente. Otra buena práctica para el desarrollo de cada nueva funcionalidad, consiste en crear una rama por cada funcionalidad que necesitemos desarrollar partiendo de la rama **development.** Por ejemplo, si la nueva funcionalidad que vamos a agregar a nuestra aplicación es poder dar de alta usuarios al sistema el nombre del *feature branch* en este caso podría ser **feature/alta-usuarios**. La idea de esta actividad es aplicar esta técnica en un desarrollo simple. La idea es desarrollar la calculadora explicada en la explicación práctica de la semana pasada de manera colaborativa. Primero uno de los miembros del grupo deberá comenzar el repositorio de código implementando el archivo principal. Luego, *cada miembro del equipo,* deberá generar una nueva rama nombrada con el prefijo **feature/** seguido de funcionalidad que va a agregar a la calculadora.
   Un ejemplo de ramas que podrían crear para este caso podrían ser:
   a. **feature/programa-inicial**: rama inicial donde se crea el programa principal
   b. **feature/suma**: rama donde se agrega la operación de suma a la calculadora
   c. **feature/resta**: rama donde se agrega la resta
   d. **feature/multiplicacion**: rama donde se agrega la multiplicación
   e. **feature/division**: rama donde se agrega la funcionalidad de división

   Es necesario que los cambios y las nuevas ramas generadas anteriormente sean subidas al repositorio remoto (ver fig. 1).
7. El paso siguiente consiste en unificar el código generado en las distintas ramas. De a uno por vez, todos los miembros del equipo deberán hacer un merge de la rama con su nombre a la rama **development**.
8. Ahora se deberá crear dentro de **Gitlab** un *merge request* de la rama **development** hacia **main**, donde cada miembro del equipo dejará un comentario y finalmente alguno confirmará el merge logrando que todos los cambios que estaban en **development** pasen a **main**. En esta guía se describen los pasos para realizar un merge request.
9. Cómo último paso, y teniendo nuestro código en la rama **main,** debemos generar un **git tag** desde **main**. Esto nos permite congelar una versión de nuestro código. El tag se deberá nombrar como **v0.1.0** para luego subir al repositorio remoto. Para certificar si generamos el tag correctamente, pueden ingresar al repositorio remoto y buscar en nuestros branches el tag generado.

Ejemplo gráfico

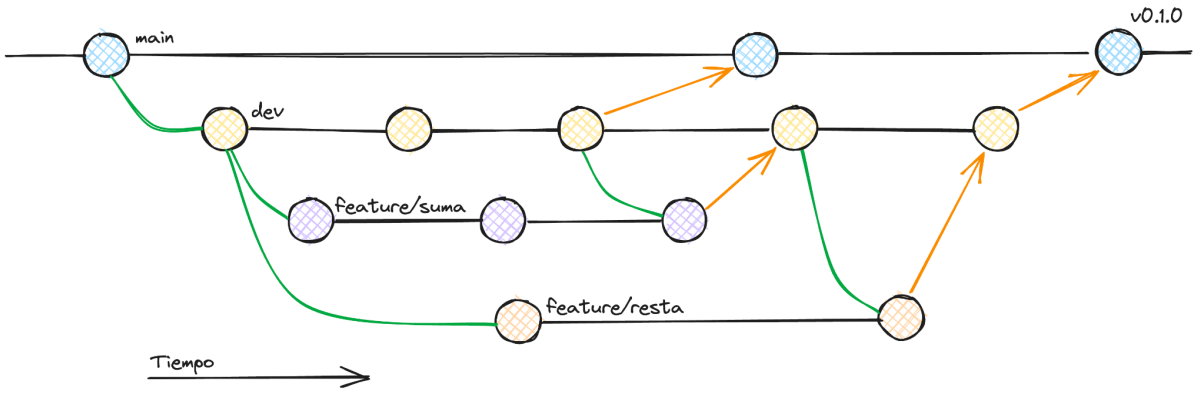

Fig. 1 - Gráfico de ramas ejemplo.

Guías

Pueden consultar la documentación y el libro del sitio oficial: Documentación base

Adicionalmente pueden completar el siguiente tutorial https://learngitbranching.js.org/ que los/as guiará en el uso de los comandos básicos de Git.

Algunas guías adicionales:

- Git básico + Merge Conflicts + Branching y Git básico + html + Css
- Principios de Git Flow
- Github flow
- Gitlab flow
- Git - Flujos de Trabajo Ramificados
- Git Feature Branch

Pautas

Modalidad de entrega

La modalidad de entrega es automática ya que todas las acciones que realicen se visualizarán en el repositorio remoto. Tener en cuenta de realizar todos los pasos antes de la fecha límite. Desde la interfaz de Gitlab deben visualizarse todas las ramas creadas para la actividad y el tag final.

Se evaluará de forma individual aunque trabajarán sobre el mismo repositorio. Por lo tanto, deberá quedar demostrada la participación de cada miembro del grupo con el aporte realizado en los commits.

Fecha límite de entrega

Viernes 05/09/2025 23:59hs.