Carrera: **TÉCNICO SUPERIOR EN ANÁLISIS FUNCIONAL DE SISTEMAS INFORMÁTICOS**

Cátedra: **PRÁCTICAS PROFESIONALES II**

Curso: **3° Año**

Profesor: **Verónica M. Piccini**  f t3 c D I i **@veropiccini**

**A13 GIT - GITHUB**

# **GIT: CONTROL DE VERSIONES – DESARROLLO COLABORATIVO**

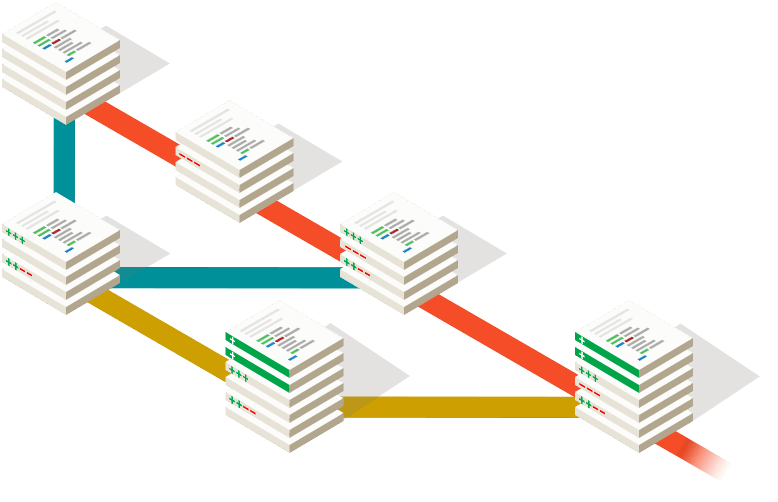
**GIT permite guardar ´puntos en el tiempo´o snapshot o capturas de pantalla del proyecto en distintos momentos.**



**GIT** es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

El kernel Linux fue nombrado en honor a su creador, Linus Torvalds. Cuando escribió git, quiso dar con una palabra corta, fácil de recordar y de tres letras. La razón por la que eligió "git” podría haber sido dado por Linus Torvalds cuando escribió la primera versión. Describió la herramienta como "el estúpido rastreador de contenido" y el nombre como (dependiendo de tu estado de ánimo):

- una combinación aleatoria de tres letras que es pronunciable, y que en realidad no es utilizada por ningún comando UNIX común. El hecho de que sea una mala pronunciación de "get" puede o no ser relevante.  
- **"rastreador de información global"**: usted' está de buen humor, y realmente le funciona.



**EL CONTROL DE VERSIONES**

Imaginemos que nos despertamos un día y tenemos que comenzar con un proyecto nuevo. Este nuevo proyecto consiste en sólo un archivo fuente. Todos los cambios que le podamoshacer a ese archivo durante el día, lo más probable es que puedan ser deshechos. Al día siguiente continuamos con el desarrollo de nuestro proyecto y mejoramos nuestro archivo pero en el intento nuestro programa empieza a presentar errores. Necesitamos volver nuestro proyecto a una versión anterior, donde no había errores pero ahora estamos restringidos a solo poder deshacer los cambios hechos durante el día.

**GIT** entonces nos ayuda con este aspecto del desarrollo. Nos ofrece herramientas para poder gestionar cada una de las etapas y versiones por las que va transitando un proyecto de desarrollo. Por otro lado puede realizar, pero sin estar restringido un **Desarrollo Colaborativo** .

**QUÉ ES EL DESARROLLO COLABORATIVO?**

El **Desarrollo colaborativo de software** es un modelo de desarrollo de software cuyas bases son la disponibilidad pública del código y la comunicación vía Internet. Este modelo se hizo popular a raíz de su uso para el desarrollo de Linux en 1991.

Teniendo como contexto a **GIT** , podríamos decir que el **Desarrollo Colaborativo** nos proporciona herramientas para poder desarrollar entre un gran número de individuos de una manera más fácil , menos propensa a errores y rápida de implementar. De esta manera siempre tenemos la opción de , por medio de algún cliente , publicar nuestro código junto con todas las etapas y versiones que nos llevó el proyecto para que otras personas puedan sumar y aportar nuevas ideas a nuestro repositorio.

# **GITHUB PLATAFORMA DE DESARROLLO COLABORATIVO**

GITHUB es la herramienta de versionado más utilizada del mercado, a través del uso de repositorios de datos y asegurando el código fuente, con un desarrollo colaborativo

Link Oficial: <https://github.com/>



**GitHub** es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública. El 4 de junio de 2018, **Microsoft compró GitHub** por la cantidad de 7500 millones de dólares, GitHub continúa siendo la plataforma más importante de **colaboración para proyectos de código abierto.**



La consola se utiliza de forma intensiva para la ejecución conjunta de casos y simulaciones, realiza versionados locales y remotos, ejecuta workflows y migraciones de un repositorio SVN a GIT, evalúa situaciones tales como el uso standard de un desarrollador, situaciones críticas de emergencia y riesgo, y atención urgente a problemas productivos, simula pérdidas de código, errores de manipulación, reestructuraciones masivas en el repositorio.

Algunas de sus funcionalidades son:

* Versionado de archivos desde cero
* Utilizar registro de cambios
* Trabajar con Ramas. Flujos de trabajo con Ramas
* Estrategia de unificación de trabajo
* Depuración de trabajo
* Manejo avanzado de GIT
* Repositorios centralizados y distribuidos
* Migraciones hacia repositorios GIT
* Detección temprana de problemas
* Reestructuraciones complejas en repositorio

# **VERSIONADO DEL SOFTWARE**

El **versionado de software es una opción de gran utilidad para mantener organizadas las labores** y establecer comunicación entre la organización y sus usuarios.

El proceso de versionado de software se refiere a una opción que permite identificar de forma**unívoca las nuevas features (características)  o funciones, actualizaciones y/o los fixes (correcciones)  corregidos.** De modo que los cambios incluidos en el versionado de un software deben contar con la característica de **aumentar las funcionalidades del sistema y solucionar sus errores**, sin modificar la aplicación o producto a nivel esencial.

Cabe destacar que **estas actualizaciones del sistema forman parte de la vida útil de un software,** por lo que se requiere mantener un control de estas nuevas versiones lanzadas.

## Características del versionado de software

1. P**ermite la identificación de bugs (errores) o problemas de funcionamiento** a partir de una versión determinada.
2. Proporciona la hoja de ruta de la aplicación o programa, lo que **permite observar el avance y se actualice de acuerdo a lo estipulado**.
3. **Fija siempre las versiones de las librerías, así como de los componentes** que se usan. Esto ayudará a evitar los errores de utilización frente a la actualización de estos por parte del sistema o aplicación
4. **Incluye cambios en el sistema, en su estructura de database*,* en los protocolos**implementados y demás utilidades.

# **PRIMEROS PASOS CON GIT – DESCARGA E INSTALACIÓN DEL SOFTWARE**

Link Oficial de descarga:

<https://gitforwindows.org/> O <https://git-scm.com/>

Manual y Ayuda de Comandos GIT

<https://git-scm.com/docs>

Seguir los pasos de Instakación desde:

**GUIA DE ACTIVIDADES A13**

**A13.1) Instalación**

**A13.2) Crear un repositorio para guardar el archivo de documento del INFORME FINAL DE PRÁCTICA PROFESIONALIZANTE**

**A13.3) Crear un repositorio a utilizar con los archivos del código fuente del software a presentar como Proyecto Final de la carrera TÉCNICO SUPERIOR EN ANÀLISIS FUNCIONAL DE SISTEMAS INFORMÀTICOS.**

**FUENTES DE INFORMACIÓN**

<https://www.youtube.com/watch?v=NwlZZfcv400> Introducción al Versionado con Git

<https://keepcoding.io/blog/que-es-un-commit-en-git/> comando COMMIT

[Licencia Creative Commons](http://creativecommons.org/licenses/by-nc-sa/3.0/)  
GIT Software Colaborativo - Versionado por  [Veronica Piccini](http://creativecommons.org/choose/www.tecnologiasvp.com.ar) se encuentra bajo una Licencia [Creative Commons Atribución-NoComercial-CompartirDerivadasIgual 3.0 Unported](http://creativecommons.org/licenses/by-nc-sa/3.0/).