

---

# Git 101

## 1. Objetivo

Describir el uso de GitHub como repositorio para almacenar, versionar, y publicar los programas que escribimos en el curso.

- Coautores:
  - Sr. Flavio Cangini
  - Sr. Axel Navarro
  - Ing. Natalia Pérez
  - Srta. Elizabeth Sosa
- Coordinador y coautor:
  - Prof. Ing. Esp. José María Sola

20150325 contenido, 20160326 formato

## 2. Introducción

### 2.1. ¿Qué es un Sistema de Control de Versiones?

Se llama **control de versiones** a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. Aunque un sistema de control de versiones VCS (del inglés Version Control System) puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión.

Un sistema de control de versiones es una herramienta fundamental en el desarrollo de software. Nos permite mantener todo el código de un proyecto en un solo lugar permitiendo que todos los interesados tengan acceso a él en tiempo y forma. Mediante un VCS es posible tener un historial de todos los cambios (commits) realizados en el código de programa. Cada commit consta de un autor, una fecha y los cambios realizados en cada archivo del proyecto. Cabe aclarar que los archivos de código son archivos de texto, por lo que se puede identificar qué líneas se agregaron, que líneas se borraron y cuales fueron modificadas. Este historial de cambios es muy similar al que podemos ver en servicios como Dropbox, incluso este tuvo

como objetivo llevar el control de versiones de una forma más simplificada a usuarios que no eran desarrolladores de software.

Hay distintas corrientes pero las principales son Git, Mercurial y Subversion, cada uno con sus pros y contras.

## 2.2. ¿Qué es Git?

Git es un **sistema de control de versiones distribuido**, fue desarrollado en 2005 por Linus Torvalds para reemplazar BitKeeper, el sistema de versionado que utilizaban en ese momento para el proyecto del kernel de Linux.

¿Por qué Git es **Distribuido**? A diferencia de otros sistemas de versionado, cada persona que ha clonado el repositorio tiene todo el historial de cambios del proyecto, en otros sistemas sólo se baja la última versión del código alojado en el servidor.

## 2.3. ¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. Existen otros servicios que implementan Git e.g., Bitbucket, GitLab, Gitorious, CloudForge.

## 2.4. ¿Por qué Usamos GitHub en el Curso?

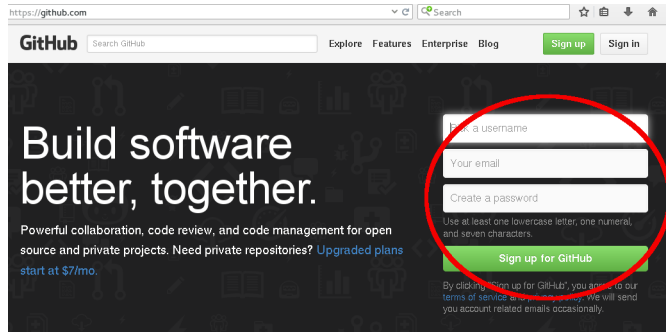
- Provee interfaces web, de escritorio, y móvil para la **administración de los repositorios**.
- Permite el **trabajo colaborativo** mediante funcionalidades de redes sociales.
- Provee **repositorios públicos** y, de forma gratuita para instituciones educativas, **repositorios privados**.
- Provee **respaldo e historial** del trabajo de los estudiantes.
- Muestra el **avance** de cada estudiante.
- Permite **interactuar** con la cátedra y otros estudiantes con funcionalidades similares a redes sociales pero orientadas a la programación.
- Permite el seguimiento de **correcciones y mejoras**.
- Con una PC con conexión a internet disponible, es **más eficiente que trabajos impresos**.
- Es un primer contacto con los **sistemas de control de versiones distribuidos** y con los lenguajes de marca livianos como **markdown**.

Durante las primeras clases, cada estudiante trabaja en su propio repositorio, el trabajo colaborativo comienza realmente cuando la cátedra asigna un repositorio privado a cada equipo.

## 3. Primeros Pasos en GitHub

### 3.1. Creación del Usuario

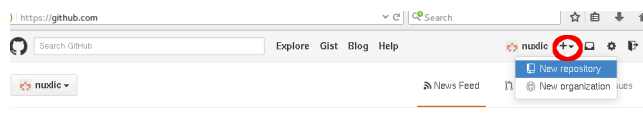
Para crear el usuario debemos dirigirnos utilizando un navegador a la pagina <https://github.com/>



En la imagen se puede observar un círculo rojo que remarca el formulario de registro. En el primer casillero se debe colocar un nombre de usuario a elección aunque se recomienda poner la inicial del nombre seguido del apellido, por ej si me llamo Juan Perez mi usuario podría ser jperez. En el siguiente casillero hay que poner un email válido dado que recibirás un mail para validar tu cuenta, luego colocamos una contraseña y apretamos el botón “Sign up for github”.

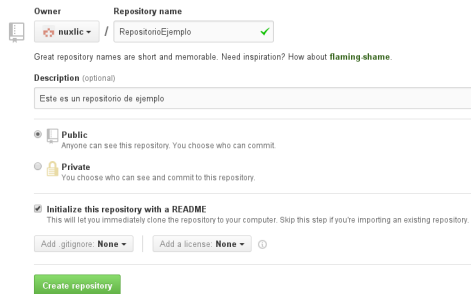
### 3.2. Creación de repositorio

Una vez que la cuenta fue validada ingresando a la casilla de mail, se puede proceder a crear un repositorio. Para ello lo primero que hay que hacer es autenticarse con la cuenta de github apretando en el botón de “Sign in” que aparece en la página. Luego aparecerá la pantalla “Home” del usuario github la cual posee un panel de herramientas y opciones tal como muestra la imagen.



Haciendo click sobre el icono “+” que está al lado del nombre de usuario accederemos a 2 opciones de la cual sólo interesa “New repository”. Esta opción permite crear un nuevo repositorio, darle nombre y una descripción tal como se muestra en la siguiente imagen. En

nuestro caso el nombre del repositorio será el **nombre del TP** y en la descripción deberá figurar el **nombre completo del alumno** y el **curso** al que pertenece.



The screenshot shows the GitHub repository creation interface. At the top, there are two dropdown menus: 'Owner' set to 'nuxlic' and 'Repository name' set to 'RepositorioEjemplo'. Below these is a note: 'Great repository names are short and memorable. Need inspiration? How about flaming shame.' The 'Description (optional)' field contains the text 'Este es un repositorio de ejemplo'. There are three radio button options: 'Public' (selected), 'Private', and 'Initialize this repository with a README' (checked). At the bottom, there are two dropdown menus for 'Add a gitignore' and 'Add a license', both set to 'None'. A green 'Create repository' button is at the bottom.

Es importante seleccionar la opción “Initialize this repository with a README” dado que de esta manera luego se podrá hacer un clone de este repositorio sin muchos inconvenientes. Video instructivo: <http://youtu.be/WnCpQggK4CU>

## 4. "Hello, World!" de GitHub

Para conocer un lenguaje de programación, es tradición crear un programa simple que escriba el mensaje "Hello World" (Hola Mundo). Aunque GitHub no es un lenguaje de programación, existe una guía para realizar los primeros pasos <https://guides.github.com/activities/hello-world/> De esta guía, por ahora, podemos obviar los **branches** y **pull requests**, los vamos a usar más adelante.

## 5. Una Guía Sencilla de Git por Línea de Comando

Al poseer una interfaz visual, GitHub no necesita el uso de la línea de comando, pero al usarlo más frecuentemente, vamos a notar que algunas tareas son más eficientes usando la línea de comandos sobre una copia local del repositorio.

**Roger Dudler** escribió una guía directa y sencilla, la versión en castellano está en: <http://rogerdudler.github.io/git-guide/index.es.html>. Para esta primera parte, solo necesitamos leer hasta la sección "pushing changes" (envío de cambios) inclusive; También esta a disposición del alumno un video ilustrativo aplicando los conceptos expuestos por Roger Dudler en: <http://youtu.be/RO98a40Fx3A>

## 6. Resumen

Durante el curso escribimos programas; para respaldarlos, controlar sus versiones, y publicarlos, usamos **GitHub**, que es una implementación del sistema de control de versiones

distribuido **Git**. El repositorio GitHub se puede administrar desde la **línea de comando**, desde una **aplicación cliente**, o desde <https://github.com/>.

Un repositorio Git es la carpeta de nuestro proyecto, se puede crear uno con **git init** o copiar uno existente con **git clone <repositorio>**. Para conectarse a un repositorio remoto en un server usamos **git remote add origin <servidor>**.

Un commit es una revisión a nuestro trabajo que los cambia de alguna forma. El trabajo que realizamos se agrega al **index** o **staging area** con **git add <archivos>**, luego se agregan al commit con **git commit -m "<descripción del cambio>"**. Publicamos el commit al repositorio remoto con **git push origin master**.

El repositorio puede ser administrado mediante línea de comando, interfaz web, de escritorio o móvil.

En Git202 presentamos los conceptos y comandos para la construcción de software de forma colaborativa, que es el objetivo principal de Git.

## 7. Bibliografía

- <http://rogerdudler.github.io>
- <http://en.wikipedia.org/>
- ProGit by Scott Chacon and Ben Straub
- <https://guides.github.com/activities/hello-world>

