**GIT**

Es el sistema de control de versiones más moderno y popular del mundo. Fue creado por [**Linus Torvalds**](https://github.com/torvalds) para garantizar la eficiencia y confiabilidad del mantenimiento íntegro de versiones.

Facilita el desarrollo del código entre un equipo de desarrolladores de manera rápida e inteligente. Además, podrás comparar los cambios realizados a lo largo del tiempo, ver quién modificó algo en el código del software y en qué momento se generó un cambio o error.

**Control de versiones**

Cambios realizados sobre los elementos o la configuración de un trabajo, guardando información de qué incluyen los cambios y cuándo se hicieron. Este control comienza con una versión básica del documento y luego va guardando los cambios que se realizan.

El control de versiones es muy útil ya que, si una versión nueva no funciona por algún tipo de error, tienes acceso a versiones anteriores.

**GitHub**

Plataforma que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones (GIT).

Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real. Es decir, nos permite **centralizar el contenido del repositorio** para poder colaborar con los otros miembros de nuestra organización.

**Ventajas de GitHub**

* Alojar proyectos en repositorios de forma gratuita
* Personalización del perfil
* Permite alojar tus proyectos de forma pública y privada
* Puedes crear y compartir [páginas web estáticas con GitHub Pages](https://platzi.com/blog/github-pages/)
* Facilita compartir tus proyectos de una forma mucho más fácil y crear un portafolio
* Colaborar y mejorar los proyectos de otros, al igual que ser ayudado
* Reduce los errores humanos y escribe tu código más rápido con [GitHub Copilot](https://platzi.com/blog/github-copilot/)

**Funcionamiento**

GitHub te permite subir tus repositorios de código para que sean almacenados en la nube a través del sistema de control de versiones de Git y participar también en el desarrollo de proyectos de terceros, lo que significa que cualquiera en el mundo que use GitHub puede encontrar tu código, aprender de él y, por qué no, mejorarlo. Al igual que tú puedes acceder a los repositorios de código de otras personas.

**¿Cómo usar GitHub?**

**1. Crear un Repositorio de GitHub**

Créate una cuenta gratuitamente. Luego, sigue estos pasos:

* En la esquina superior derecha, encontrarás un **+** que sirve para realizar las acciones de la página. Das clic en el símbolo y creas un **nuevo repositorio** (new repository).
* Dale un nombre, que de preferencia debe ser claro, definir si será público o privado y colocar una pequeña descripción sobre tu repositorio. Es opcional, pero para organizarte mejor y que los demás usuarios sepan sobre lo que trata el repositorio.
* Activa el checkbox que dice iniciar tu repositorio con un **README**, este será tu primer archivo, la presentación de tu proyecto.
* Presiona el botón de “crear repositorio” y listo. Ya tienes tu primer repositorio creado.

**2. Crear ramas (branches) en GitHub**

Cuando empezamos un proyecto con GitHub, automáticamente nos crea una rama llamada **master**, a partir de la cual comenzaremos a crear nuestras propias ramas.

* Entra en tu repositorio.
* En la parte superior haz clic en **Rama actual**. En la lista de rama selecciona la rama llamada master, que será nuestra base para la que estamos creando.
* Luego presiona **New Branch** y añade nombre
* Selecciona la rama actual (master) en la que se basará la nueva rama y presiona **Create**

**3. Entender los commits de GitHub**

*Commit* es la denominación que se le da a los cambios guardados en Github.

Para realizar commits:

1.- Se debe verificar el estado de nuestro repositorio ejecutando el siguiente comando:

git **status**

Una vez realizado el comando anterior, te aparecerá una lista con los archivos que fueron modificados y con los que están agregados al índice, listos para subir.

2.- Si aún existen archivos sin agregar al índice, debes ejecutar el siguiente comando:

git **add**

De esta forma se añaden todos los cambios pendientes.

3.- Ahora vamos a generar el **commit** ejecutando el siguiente comando:

**git commit -m "Un comentario de los cambios realizados"**

Es importante que en este paso agregues una descripción clara, esta se guardará en el historial y podremos entender mejor los cambios más adelante. Cuando no se describen bien los cambios que se realizaron, volver a reparar un bug (error) puede ser una pesadilla.

Y eso es todo. Son pasos bastante sencillos, pero nunca está demás revisar un par de veces lo que vamos a subir a nuestro repositorio remoto y seguir practicando.

**4. Crear solicitudes de extracción (pull requests) en GitHub**

Las solicitudes de extracción o **pull requests** son el formato para contribuir con los cambios que realizaste a un código base **para que sean fusionados**.

Los pasos que debes seguir son los siguientes:

1. Clic en el botón bifurcación (**Fork**) en la página de GitHub que contiene la base de código original al que deseas contribuir, para crear una copia en tu cuenta. Bifurcar un repositorio te permite realizar cambios y experimentos sin afectar el original.
2. Obtén la URL de la bifurcación que acabas de crear
3. Usa el comando **git clone** para clonar la base de código de Fork en la página de tu Github en tu computadora local
4. Cambia lo que consideres necesario en tu repositorio local. Agrega y modifica archivos.
5. Ahora inserta el repositorio de código local en el repositorio de código de Fork en Github, con los siguientes comandos:

git **add ,** git **commit,** git **push**

1. Vuelve a la página de tu fork en Github
2. Presiona el botón Solicitud de Extracción (pull request)
3. Dale un nombre a tu solicitud de extracción, colocando los detalles de los cambios que realizaste y finalmente presiona el botón **enviar**. Ya has colaborado oficialmente con un proyecto y tu solicitud será agregada si el administrador la considera adecuada.