



No olvides poner a grabar la clase

# Diplomatura

FullStack



Centro Universitario Chivilcoy

- HERMINIA BRUMANA -

**UNTREF**

UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO

# Terminal

# ¿Que es la terminal?

Una terminal en las computadoras es una herramienta que permite a los usuarios interactuar con el sistema operativo a través de una interfaz de línea de comandos en lugar de una interfaz gráfica de usuario. Es decir, es un programa que permite a los usuarios ingresar comandos de texto para ejecutar tareas en la computadora, como crear, mover o eliminar archivos y directorios, instalar software, configurar la red y mucho más.

# ¿Que es la terminal?

En la mayoría de los sistemas operativos, la terminal es una herramienta muy poderosa y flexible que permite a los usuarios realizar una amplia variedad de tareas de forma rápida y eficiente. Además, la terminal a menudo proporciona acceso a herramientas y utilidades que no están disponibles en la interfaz gráfica de usuario.

La terminal también es muy útil en la programación, ya que muchos lenguajes de programación proporcionan una interfaz de línea de comandos para compilar, ejecutar y depurar programas. Los programadores también pueden utilizar la terminal para automatizar tareas repetitivas o para realizar tareas avanzadas de administración del sistema.

# ¿Que es la terminal?

Existen muchas aplicaciones de terminal disponibles para diferentes sistemas operativos, como la Terminal en macOS, el Símbolo del sistema en Windows y la Terminal en Linux. Además, también existen programas de terminal de terceros que proporcionan funcionalidades adicionales, como pestañas y atajos de teclado personalizados.

# Comandos básicos de la terminal

1. **ls:** Este comando se utiliza para listar los archivos y directorios en el directorio actual.
2. **cd:** Este comando se utiliza para cambiar de directorio. Por ejemplo, "cd ~/Documents" cambiará el directorio actual a "Documents" en el directorio de inicio del usuario.
3. **mkdir:** Este comando se utiliza para crear un nuevo directorio. Por ejemplo, "mkdir my\_new\_directory" creará un nuevo directorio llamado "my\_new\_directory" en el directorio actual.
4. **rm:** Este comando se utiliza para eliminar archivos o directorios. Por ejemplo, "rm my\_file.txt" eliminará el archivo llamado "my\_file.txt" en el directorio actual.
5. **cp:** Este comando se utiliza para copiar archivos. Por ejemplo, "cp file1.txt file2.txt" copiará el archivo "file1.txt" en uno nuevo llamado "file2.txt".

# Comandos básicos de la terminal

6. **mv:** Este comando se utiliza para mover archivos o cambiarles el nombre. Por ejemplo, "mv file1.txt new\_folder/" moverá el archivo "file1.txt" al directorio llamado "new\_folder".
7. **cat:** Este comando se utiliza para mostrar el contenido de un archivo en la terminal. Por ejemplo, "cat my\_file.txt" mostrará el contenido del archivo "my\_file.txt".
8. **grep:** Este comando se utiliza para buscar patrones de texto en un archivo o en la salida de otro comando. Por ejemplo, "grep 'error' log.txt" buscará todas las líneas en el archivo "log.txt" que contengan la palabra "error".
9. **pwd:** Este comando se utiliza para mostrar la ruta del directorio actual.
- 10 **touch:** este comando sirve para generar archivos nuevos, dentro de un directorio.



# Git y Github

# Git y Github

Es hora de integrar **Git** a nuestro entorno de desarrollo.

El paso primordial para poder utilizarlo con un versionado en la nube de **Github**, es contar con una cuenta previamente gestionada en esta plataforma: [Github](https://github.com).

Si aún no lo hiciste, procede a gestionar una cuenta para poder continuar.



# Git y Github

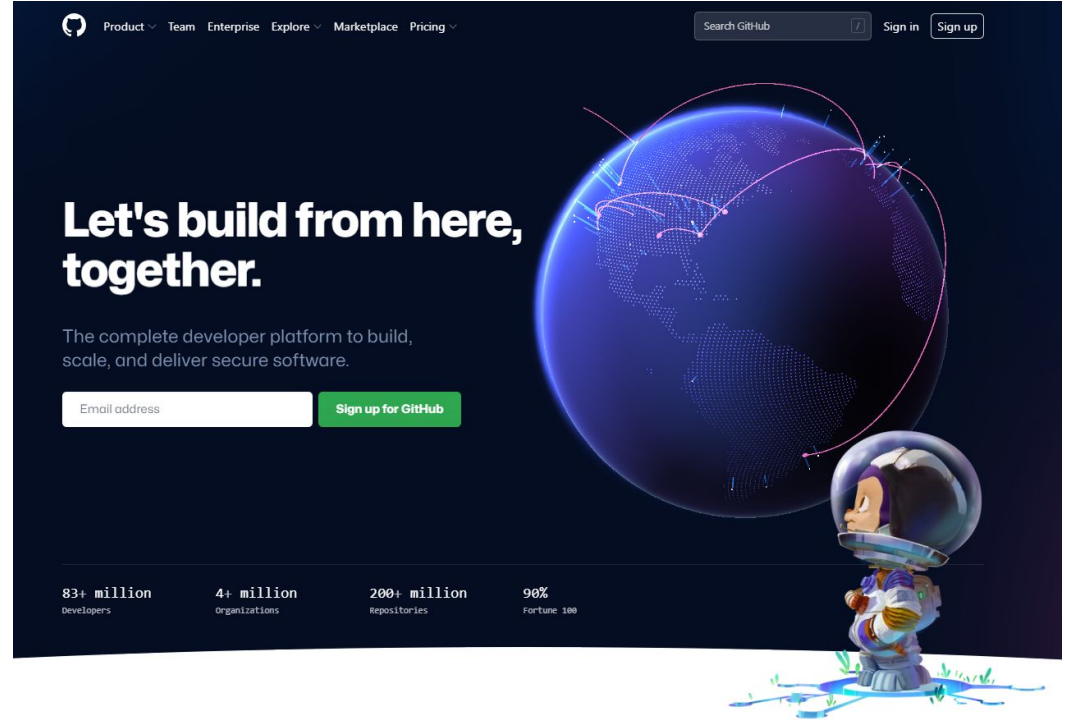
Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue adquirido por Microsoft en julio del 2018.

La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas y que, como usuario, no sólo puedas descargar la aplicación sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.



# Git y Github

Para crear nuestra cuenta,  
hacemos clic en “**Sign up**”



**UNTREF**

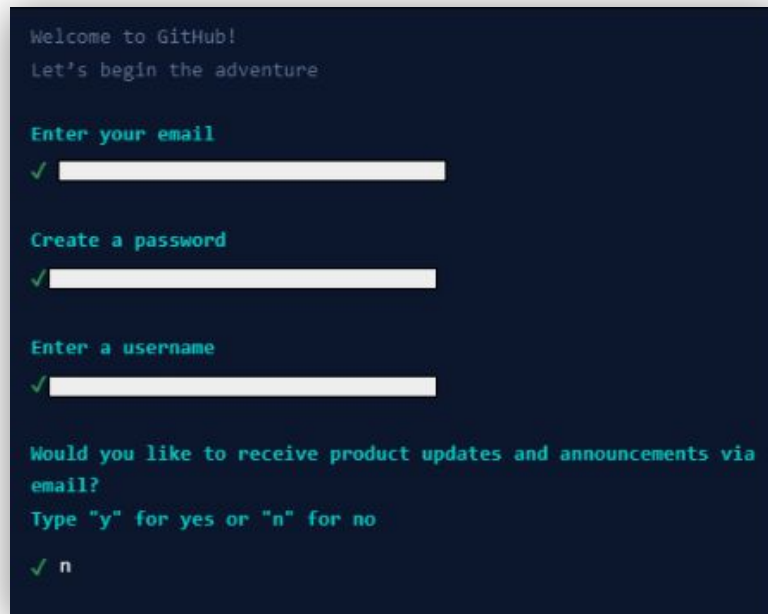
UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO

# Git y Github

## Llena el formulario:

- Nombre de usuario.
- Dirección de e-mail (*recomendamos usar el mismo email de tu perfil de Git*)
- Contraseña.

Luego **“Next: select a plan”**, para seleccionar el plan que queremos tener.



Welcome to GitHub!  
Let's begin the adventure

Enter your email  
✓

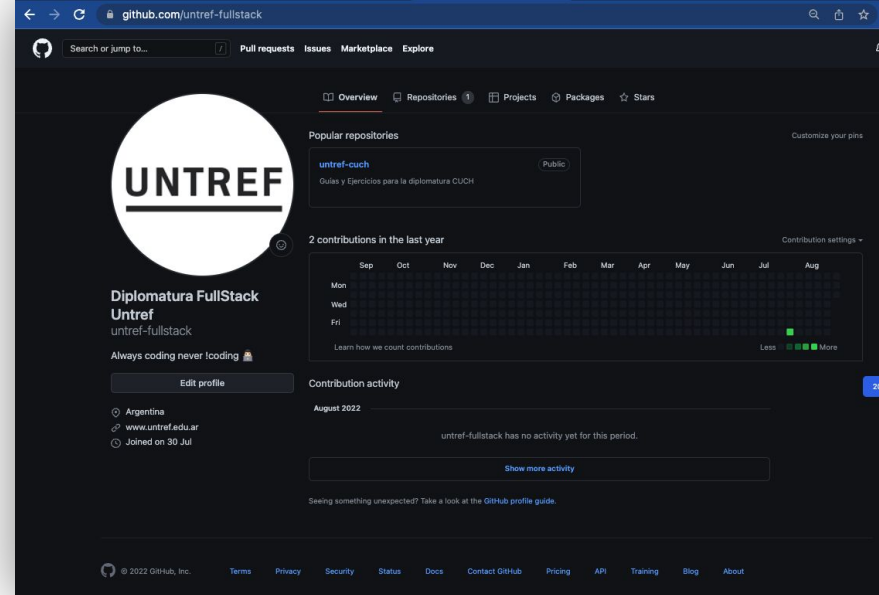
Create a password  
✓

Enter a username  
✓

Would you like to receive product updates and announcements via email?  
Type "y" for yes or "n" for no  
✓ n

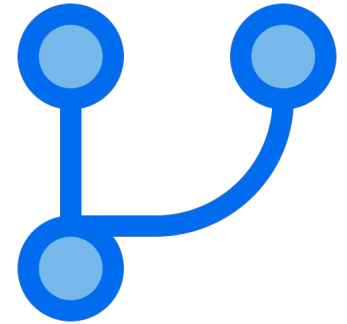
# Git y Github

Gestionada tu cuenta en **github.com**,  
ya estamos en condiciones de proceder  
con la descarga de Git para nuestro  
entorno de trabajo.



# Git y Github

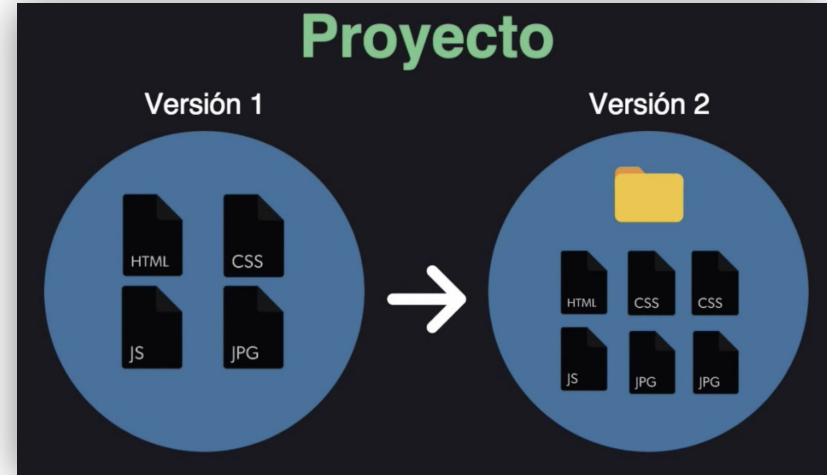
Como bien hablamos anteriormente, Git es un sistema de control de versiones distribuido, gratuito, y de código abierto. Está diseñado para manejar todo tipo de proyectos de software, hasta incluso la realización de backups y control de versiones de otros tipos de archivos.



Es rápido y aporta un rendimiento sumamente versátil a la hora de utilizarlo.

# Git y Github

Una de las ventajas de Git, es la de poder volver a versiones anteriores de tu código. Es muy útil volver sobre tus pasos, para evitar errores y para tener más organizado el proyecto.

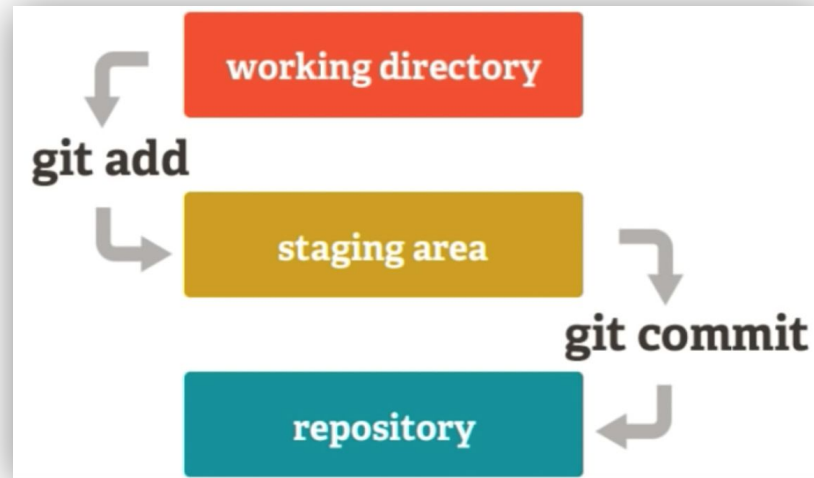




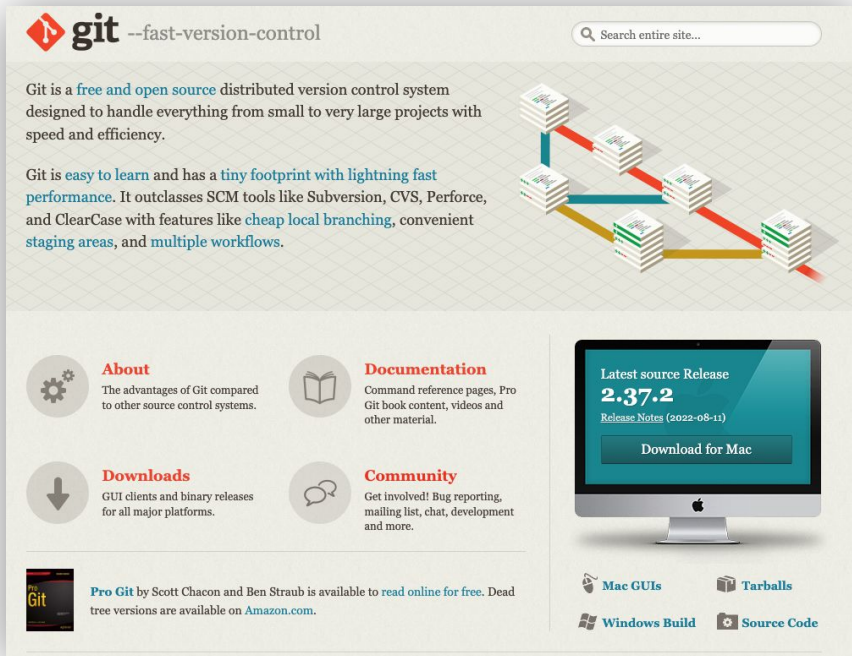
# Git y Github

Cuando hablamos de Git, debemos tener en mente sus tres principales Estados:

- El primero es el **working directory**  
(donde estamos trabajando)
- El segundo es el **staging area**  
(donde preparamos los archivos)
- El tercero es el **repository**  
(donde ya están registrado los archivos)



# Git y Github



The screenshot shows the Git website homepage. At the top left is the Git logo and the tagline "--fast-version-control". A search bar is at the top right. The main text describes Git as a "free and open source distributed version control system" and lists its features: "easy to learn", "tiny footprint with lightning fast performance", "cheap local branching", and "multiple workflows". A diagram illustrates a branching model with stacks of code. Below this are four sections: "About" (advantages), "Documentation" (reference pages), "Downloads" (GUI clients), and "Community" (bug reporting). A central monitor displays the "Latest source Release 2.37.2" and a "Download for Mac" button. At the bottom, there are links for "Mac GUIs", "Tarballs", "Windows Build", and "Source Code". A small box on the bottom left mentions "Pro Git" by Scott Chacon and Ben Straub.

Debemos descargar el instalador de acuerdo a la versión de nuestro sistema operativo.

La URL de descarga del mismo es:

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

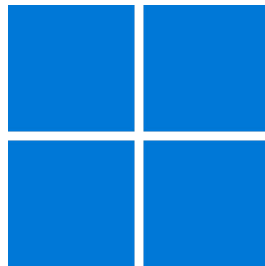
Descargado el instalador, ejecútalo.

# Git y Github

Elige la opción que mejor se ajuste a tu sistema operativo:



<https://git-scm.com/download/mac>



<https://git-scm.com/download/win>



<https://git-scm.com/download/linux>

**UNTREF**

UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO

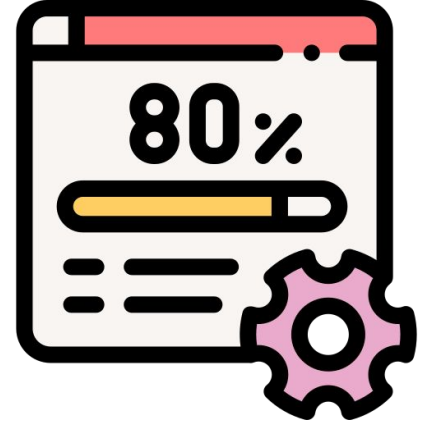
# Git y Github

Si tienes un asistente de instalación que te guía durante el proceso, ten en cuenta la siguiente configuración:

- Deja la ubicación por defecto de instalación
- Permite que actualice de forma automática ante nuevas versiones
- Permite que el instalador se conecte con Visual Studio Code

*(para trabajar de forma colaborativa desde el IDE)*

- Selecciona que utilice los comandos de UNIX desde el Prompt
- Permite que utilice OpenSSL library
- Permite que Git gestione las credenciales (*Credential Manager*)

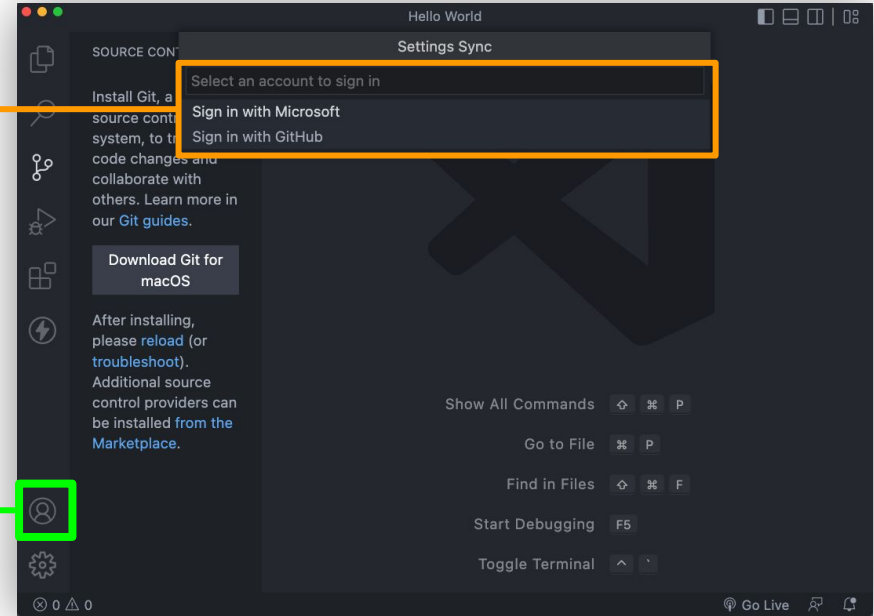


# Conectar Github con VS Code

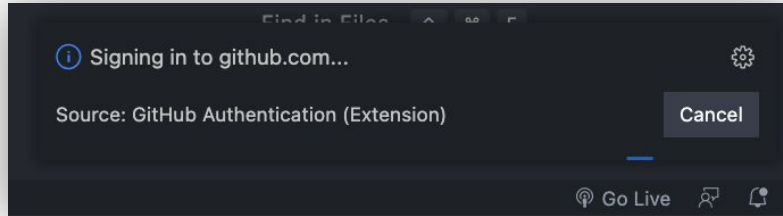
# Conectar Github con VS Code

(2) Selecciona la forma de identificarte en Github  
(*Microsoft Account* o *Github Account*)

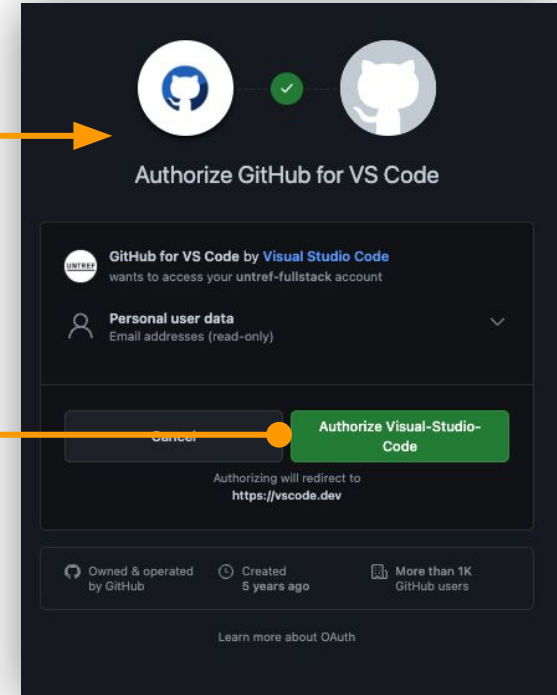
(1) Conecta tu cuenta de Github con **VS Code**,  
seleccionando **Turn on settings sync**.



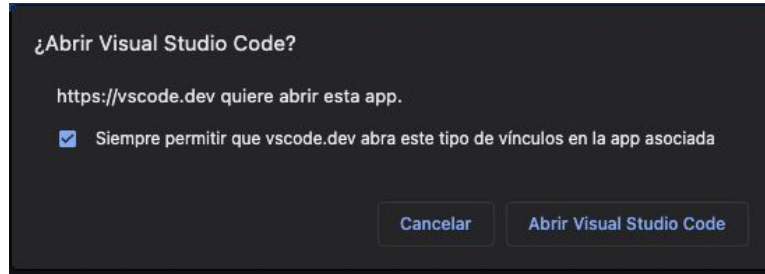
# Conectar Github con VS Code



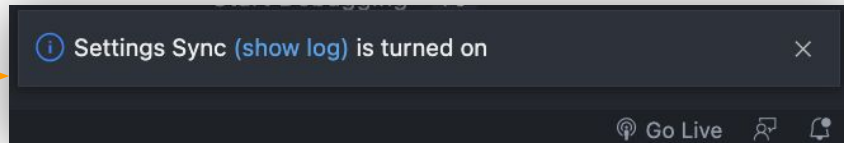
(1) VS Code abrirá una ventana del navegador web para esperar que autorices a Github trabajar con el editor de código.



(2) El navegador web te pedirá permiso para abrir VS Code.



(3) Finalmente, VS Code te confirma cuando todo esté sincronizado.



# Manejo de Git mediante línea de comandos



# Setup (Configuración)

# Configurar información de usuario

Comando	Descripción
<code>git config --global user.name "[firstname lastname]"</code>	Configura el nombre de usuario que sea identificable para los créditos cuando revisiones código o el historial de versionado.
<code>git config --global user.email "[valid-email]"</code>	Configura un Email el cual será asociado con cada historial de versionado.
<code>git config --global color.ui auto</code>	Configura el formato de color de línea para Git.

# Inicializar repositorios

# Inicializar repositorios

Comando	Descripción
<code>git init</code>	Inicializa un directorio existente como un repositorio de Git.
<code>git clone https://github.com/untref-fullstack/untref-cuch.git</code>	Recupera un repositorio entero desde una ubicación hosteada mediante una URL.

# Stage e Instantánea

# Configurar información de usuario entre repositorios

Comando	Descripción
<code>git status</code>	Muestra los archivos modificados en el directorio de trabajo, que están preparados para tu próximo <b>commit</b> .
<code>git add [file]</code>	Agrega un archivo nuevo al área de Stage para el próximo <b>commit</b> .
<code>git reset [file]</code>	Desafecta un archivo mientras retiene los cambios en el directorio de trabajo.
<code>git diff</code>	Muestra las diferencias de qué ha cambiado en un archivo sin haberlo agregado al área de Stage.
<code>git diff -- staged</code>	Muestra las diferencias de qué ha cambiado en un archivo que aún no ha sido confirmado [ <b>committed</b> ].
<code>git commit -m "[mensaje que describa el commit]"</code>	Realiza un commit al área de stage como una nueva instantánea.

# Compartir y Actualizar

# Compartir y Actualizar

Comando	Descripción
<code>git remote add [alias] [url]</code>	Aplica un alias a la URL de un proyecto de Git.
<code>git fetch [alias]</code>	Baja todas las ramas de la URL remota. ( <i>en este caso, indicada mediante un alias</i> )
<code>git merge [alias]/[branch]</code>	Fusiona una rama remota dentro de la rama actual para actualizarla.
<code>git push [alias] [branch]</code>	Envía los commit de la rama (branch) local hacia la rama del repositorio remoto.
<code>git pull</code>	Baja y fusiona todos los commit desde la rama remota ( <i>remote branch</i> ).

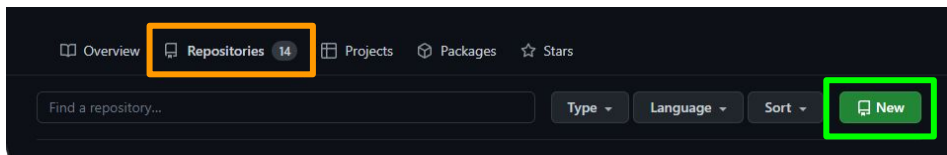


# Crear nuestro primer repositorio

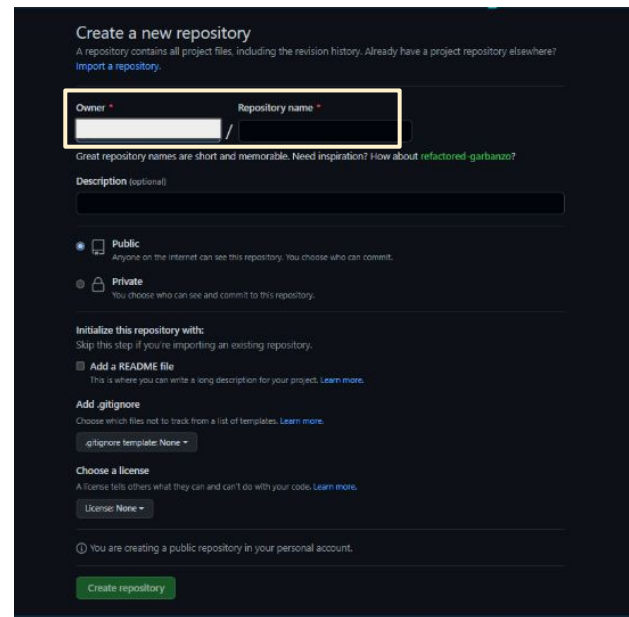
# Git y Github

Una de las formas para generar un repositorio es a través del perfil de inicio.

En el panel central, hacemos clic en **Repositories** y, luego, pulsamos el botón **New**.



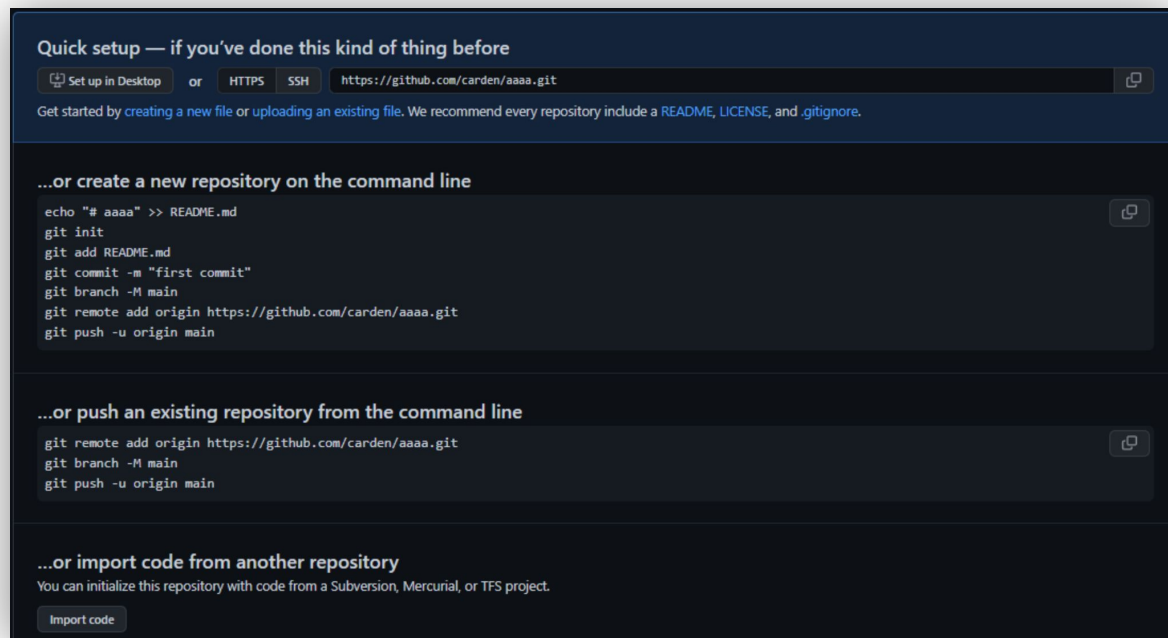
Definimos el propietario (*owner*) y nombre del repositorio (*repository name*) en las cajas de texto superiores. Luego la visibilidad del proyecto (*Public / Private*), y finalmente pulsamos el botón **Create Repository**. 🖱️



# Git y Github

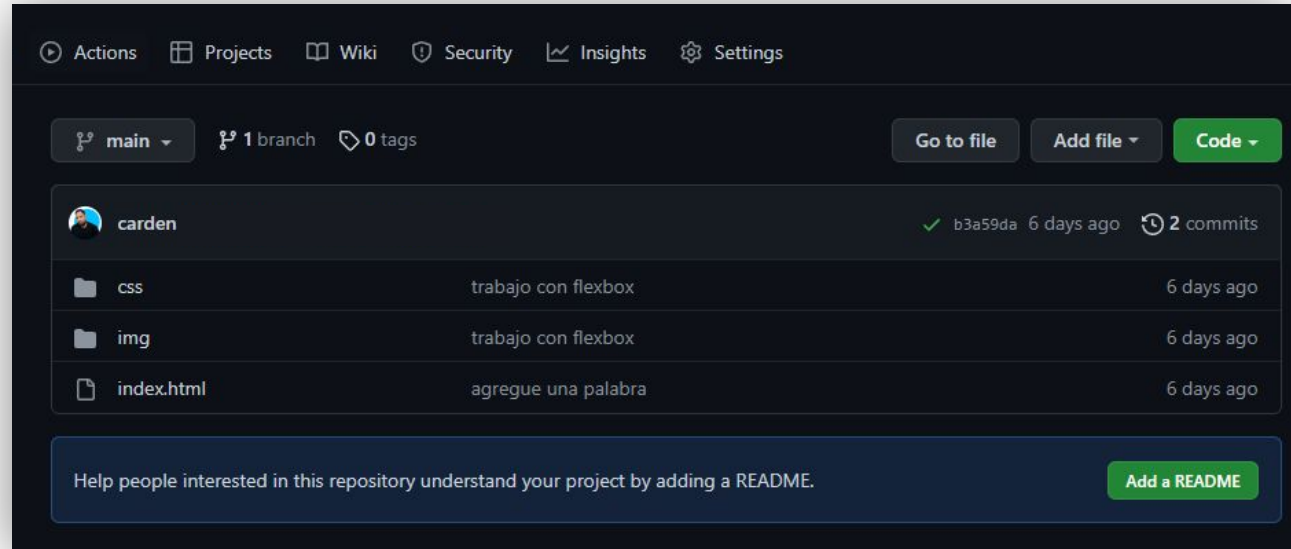
Una vez que realizamos el repaso de los comandos de Git más importantes, lo que veamos en Github, no nos va a ser tan ajeno.

Lo que refleja aquí es el *Paso a Paso* que necesitamos para enlazar Git con Github.



# Git y Github

Una vez enlazado y subido nuestro proyecto al repositorio, veremos una estructura similar a la siguiente: 📁



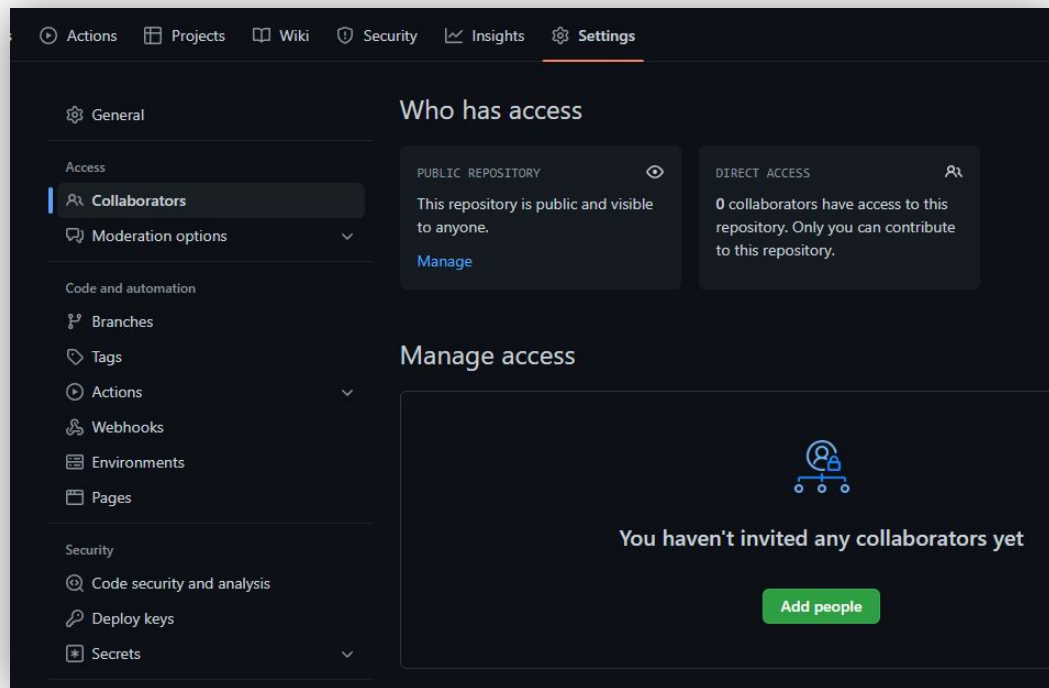
# Git y Github

La versatilidad de Github lo convierte en una de las plataformas más utilizadas por los equipos de trabajo.

Dentro del menú **Settings** >

**Collaborators**, podemos sumar colaboradores, quienes pueden trabajar en conjunto con nosotros.

Entre sus funciones permite restringir y asignar partes en las cuales los colaboradores pueden moverse.



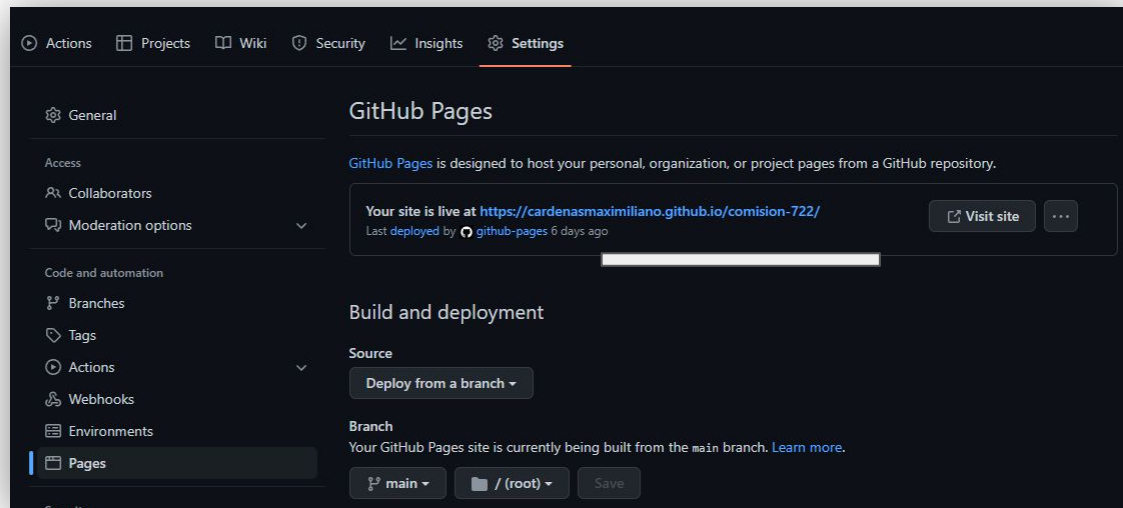
# Git y Github

Github también nos permite emular una subida de nuestro proyecto a un servidor, a través de **Github Pages**.

Para activarlo, debemos ir a:

## Settings > pages

1. Buscamos la sección **branch**
2. Elegimos a continuación la rama a mostrar
3. Finalmente pulsar el botón **Save**



# README.md

# Readme.md

Un archivo README.md es un archivo de texto que se encuentra comúnmente en los proyectos de software y que se utiliza para proporcionar información sobre el proyecto. La extensión ".md" significa "Markdown", que es un lenguaje de marcado ligero que se utiliza para dar formato al texto en el archivo.

El archivo README.md suele estar ubicado en el directorio principal del proyecto y puede contener información como:

- Una descripción general del proyecto.
- Instrucciones para instalar y utilizar el software.
- Una lista de dependencias del proyecto.
- Una guía de contribución para desarrolladores interesados en contribuir al proyecto.
- Una sección de preguntas frecuentes (FAQ) con respuestas a las preguntas comunes.
- Información de contacto para el equipo de desarrollo o para obtener soporte técnico.



# Readme.md

El archivo README.md es una parte importante de la documentación de un proyecto de software y puede ayudar a los usuarios y desarrolladores a comprender rápidamente cómo utilizar el proyecto. Además, los servicios de alojamiento de repositorios de código fuente, como GitHub, GitLab y Bitbucket, suelen mostrar el contenido del archivo README.md en la página de inicio del proyecto, lo que lo convierte en una herramienta útil para promocionar y presentar el proyecto a otros usuarios y desarrolladores.

# .gitignore

# Readme.md

Un archivo .gitignore es un archivo de texto que se coloca en el directorio raíz del proyecto y que contiene una lista de patrones de nombres de archivo y directorio que Git debe ignorar al hacer seguimiento de los cambios en el proyecto. Cualquier archivo o directorio que coincida con un patrón en el archivo .gitignore no se incluirá en el control de versiones de Git.

Por ejemplo, si un proyecto utiliza un sistema de construcción que genera automáticamente archivos binarios, puede incluir una regla en el archivo .gitignore para evitar que Git haga seguimiento a estos archivos. De esta manera, el sistema de construcción puede generar los archivos binarios sin preocuparse de que se incluyan en el control de versiones.



# Readme.md

En resumen, el archivo .gitignore es una herramienta útil para evitar que archivos y directorios no deseados se incluyan en el control de versiones de Git. Es importante tener en cuenta que el archivo .gitignore solo evita que Git haga seguimiento a los archivos y directorios especificados; no los elimina del sistema de archivos.

Algunos ejemplos de editores

<https://readme.so/es/editor>

<https://profilinator.rishav.dev/>

# ACTIVIDAD

Luego de lo visto en clase crearemos nuestro primer repositorio en github.

Para ello algunas recomendaciones.

- Recordar siempre estar bien posicionado en la consola.
- Revisar bien los nombres de los archivos que no posean espacios ni caracteres especiales.
- Revisar bien las rutas de tu proyecto.

# ¿Dudas o preguntas?

Muchas gracias



**UNTREF**

UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO