

«Talento Tech»

Git y GitHub

Introducción a Git

1



Sesión 1: Introducción a Git

Índice

Sesión 1: Introducción a Git

Introducción al control de versiones

¿Cómo funciona a nivel conceptual?

Ventajas del control de versiones

Git

¿Qué es Git?

Instalación y configuración inicial de Git

Git por la terminal

Creación de un repositorio local

Registro de cambios: Flujo básico add + commit

Estados de los archivos

Flujo básico

Commit

⚙️ Mini-práctica:

Pasos

Ver historial simple

Integración con VS Code

⚙️ Mini-práctica:

Pasos

Cierre del encuentro

⚙️ Desafío

Objetivos de la clase

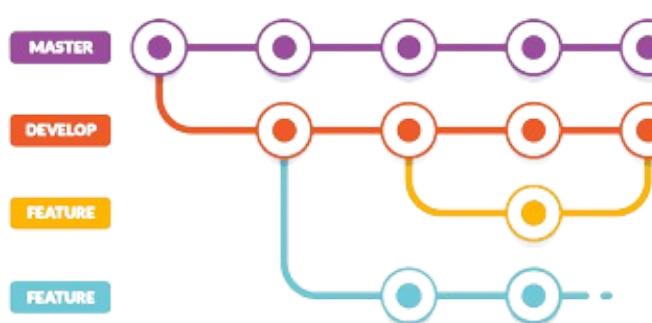
En esta primera sesión trabajaremos con **Git**, una herramienta fundamental para el desarrollo de software en general. El objetivo principal es comprender cómo funciona Git, qué es el control de versiones y cómo podemos registrar los cambios de un proyecto local.

En esta primera parte trabajaremos:

- Explicación teórica inicial
- Demostración del uso de los comandos
- Práctica individual guiada

Este contenido sentará las bases para el segundo encuentro, en el que sumaremos otros conceptos y funciones: repositorios remotos, ramas y colaboración.

Introducción al control de versiones



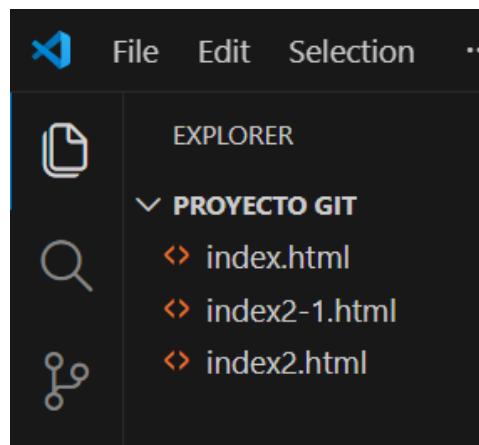
El control de versiones es una herramienta que nos permite registrar cambios, organizar el trabajo, comprender la evolución de un proyecto y facilitar la colaboración entre múltiples personas. En proyectos profesionales es impensado trabajar sin un sistema como Git, ya que garantiza orden, trazabilidad y seguridad en todas las etapas del desarrollo.

¿Cómo funciona a nivel conceptual?

Podemos imaginar el control de versiones como una "línea del tiempo" del proyecto. Cada vez que guardamos un avance, estamos sacando una fotografía del estado actual de los archivos. Git almacena estas fotografías de manera eficiente, reconociendo qué cambió realmente y guardando sólo las diferencias necesarias.

¿Por qué es importante?

Sin control de versiones, es común terminar con archivos repetidos como por ejemplo:



Esto genera confusión, pérdida de información y dificulta la colaboración porque podríamos tener varias versiones de un mismo archivo repetidas, sin utilidad y sin sentido alguno.

Ventajas del control de versiones

- **Historial completo** de todos los cambios
- **Recuperación de versiones anteriores**
- **Trabajo colaborativo seguro**
- **Experimentación** sin riesgo de romper el proyecto principal

Git



¿Qué es Git?

Es el **sistema de control de versiones mas utilizado y reconocido**, que nos ayuda a guardar diferentes versiones de nuestro proyecto. Si tenemos algún error, podemos acceder a una de las versiones anteriores para volver a empezar. Esto se realiza a través del historial de versiones que se genera cada vez que se guarda una nueva versión.

Este flujo permite un control detallado. No estás obligado a guardar *TODO* de una vez, sino que podés preparar commits limpios y organizados.

Su creador Linus Torvalds, lo ideó con tres pilares fundamentales:

- **Velocidad:** Operaciones rápidas incluso con miles de archivos
- **Integridad:** Cada commit tiene un identificador único (hash) basado en su contenido
- **Flexibilidad:** Permite crear múltiples líneas de trabajo, fusionarlas y experimentar sin romper nada

Instalación y configuración inicial de Git

Instalación

Para poder utilizar esta herramienta en nuestro dispositivo, será necesario su instalación:

Si tenés Mac podés escribir en la terminal el comando *git* y seguir el paso a paso porque te ofrece una instalación automática. Si tenés Linux suele ver preinstalado y podés consultarla ejecutando el comando *git -version* desde la terminal. En Windows será necesario realizar su instalación.

1. Para ello vamos a es necesario entrar al siguiente enlace: <https://git-scm.com/install/>.
2. Elegir tu sistema operativo
3. Clickear sobre “Click here tu download”. Esto reconocerá la arquitectura del sistema (32 bit o 64 bit) y te descargará un ejecutable.
4. Una vez descargado el ejecutable, seguís el paso a paso hasta confirmar la instalación.

Verificar instalación

Para poder verificar si tu computadora cuenta con Git instalado, solamente basta con escribir el siguiente comando en el CMD: **Command Prompt** o símbolo del sistema. También conocida como la terminal.

Comando: `git -version`

Ejemplo de respuesta: `git version 2.51.0.windows.1`

The screenshot shows a Windows Command Prompt window titled "Símbolo del sistema". The window displays the following text:
Microsoft Windows [Versión 10.0.19045.4046]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Talento-tech>git --version
git version 2.51.0.windows.1

De lo contrario si no obtenemos la respuesta deseada y nos indica que no es conocido git, debemos volver a ejecutar el instalador o descargarlo.

Configurar identidad

La configuración de la identidad en Git es fundamental para identificar quién hizo los cambios en un repositorio, lo cual es crucial para la colaboración y el seguimiento del historial del proyecto. Al configurar un nombre de usuario y una dirección de correo electrónico, Git los registra automáticamente en cada "commit" que realizas, creando un registro de autoría claro para ti y otros colaboradores. Esto es vital en proyectos de equipo para saber quién contribuyó a cada cambio, y también es útil en proyectos personales para un seguimiento organizado.

```
git config --global user.name "Tu Usuario"
git config --global user.email "tu@gmail.com"
```

Ejemplo:

The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git config --global user.name "talentotech-lab"
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git config --global user.email "alumnotalentotech@gmail.com"
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git config --global user.name
talentotech-lab
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git config --global user.email
alumnotalentotech@gmail.com
```

Es recomendable usar el mismo correo y user que usaremos en GitHub.

Verificar configuración

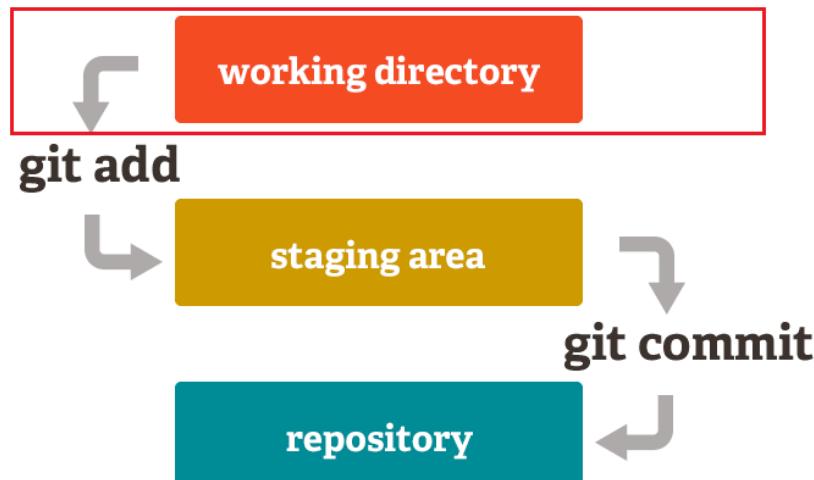
Para verificar que los datos se hayan ingresado correctamente y para contar con que efectivamente todos los cambios sean registrados correctamente ejecutamos:

```
git config --list
```

Para comenzar a usar Git debemos entender su funcionamiento. Sabemos que es el encargado de controlar las versiones de las modificaciones que se van realizando dentro de un archivo o proyecto, pero ¿cómo funciona? Para eso, es necesario comprender los siguientes conceptos:

Conceptos claves sobre su funcionamiento:

- **Repositorio local:** Toda la historia y configuración del proyecto guardadas en tu PC de manera local.
- **Working directory:** El espacio donde trabajás y editas los archivos.
- **Staging área:** Una zona intermedia donde seleccionas qué cambios vas a guardar.
- **Repositorio (committed):** El lugar donde los commits quedan almacenados de forma permanente.



git add y **git commit** son palabras con las que nos empezaremos a familiarizar mientras avanzamos en el conocimiento de GIT.

Instalación de Visual Studio Code

Para comenzar a utilizar git, utilizaremos una interfaz que nos facilite la interacción entre los archivos que estemos trabajando y la terminal. Utilizaremos la terminal de Visual Studio Code, el editor de código mas utilizado actualmente.

Para eso vamos a repasar su instalación:

Windows

1. Haz clic en "Download for Windows".
2. Ejecuta el archivo instalador (.exe).
3. Acepta el acuerdo y da clic en "Siguiente" hasta llegar a la pantalla de "Selección de tareas adicionales".
4. **Importante:** Marca todas las casillas (especialmente "Aregar 'Abrir con Code'..." y "Aregar a PATH").
5. Haz clic en Instalar y luego en finalizar.

macOS

6. Haz clic en "Download Mac Universal".
7. Abre el archivo .zip descargado (si no se descomprime solo).
8. Crucial: Arrastra el ícono de Visual Studio Code a tu carpeta de Aplicaciones.
9. Ejecútalo desde "Aplicaciones" o Spotlight (Cmd + Espacio).

Tip extra: Ponerlo en español

Por defecto VS Code viene en inglés. Para cambiarlo:

1. Abre VS Code.
2. Presiona **F1** (o **Ctrl+Shift+P**).
3. Escribe "**Configure display language**".
4. Selecciona "**Install additional languages...**" y elige **Español**.
5. Reinicia el programa.

Iniciemos nuestro repositorio

Opción 1: Usando una consola (terminal)

- Abrí una terminal (PowerShell, Git Bash o similar).
- Ubicate en la carpeta donde querés tu proyecto:

None

```
cd nombre-de-la-carpeta
```

- Inicializá el repositorio:

None

```
git init
```

- Verificá el estado del repositorio:

None

```
git status
```

Esto crea un repositorio Git local en esa carpeta.

Opción 2: Usando Visual Studio Code (recomendado)

- Abrí **Visual Studio Code**.
- Abrí la carpeta del proyecto ([Archivo → Abrir carpeta](#)).
- Abrí la terminal integrada ([Terminal → Nueva terminal](#)).
- Ejecutá:

None

```
git init
```

- Verificá el estado con:

None

```
git status
```

→ El resultado es el mismo, pero todo sucede dentro del mismo entorno.

En este curso trabajaremos principalmente con la **terminal integrada de Visual Studio Code**, para mantener el proyecto y el versionado en un único entorno.

Registro de cambios: Flujo básico add + commit

Estados de los archivos

Estos estados describen en qué situación se encuentra cada archivo dentro del proyecto:

- **Untracked (Sin seguimiento)**: Son archivos que existen en tu directorio de trabajo pero Git aún no los está controlando. Como archivos nuevos que todavía no los agregaste al repositorio; Git los detecta, pero no forman parte del historial de versiones.
- **Staged (Preparado)**: Es el estado de los cambios que marcaste para incluir en tu próximo commit. Funciona como una etapa intermedia donde seleccionas exactamente qué modificaciones querés guardar, permitiéndote revisar y elegir qué partes de tu trabajo pasarán al historial.
- **Committed (Confirmado)**: Indica que los cambios que estaban en la zona de preparación ya fueron guardados en el historial del repositorio. En este estado, las modificaciones quedan registradas de manera permanente en la base de datos interna de Git.

Flujo básico

1. Crear o modificar un archivo (ej. `index.html`).
2. Ver estado: `git status`
3. Agregar cambios al área de preparación: `git add archivo` o `git add .`
4. Crear un commit: `git commit -m "Mensaje descriptivo"`
Los mensajes deben ser claros y específicos.

Commit

Un commit es una acción para guardar los avances en el repositorio local, cada vez que se realice un commit se guarda la información junto al mensaje descriptivo, un identificador único (hash) y la referencia al commit anterior, formando así el historial del proyecto.

Mini-práctica 1

Nuestro primer repositorio y primer commit

Objetivo: Crear un repositorio local, configurar Git y registrar el primer cambio.

Pasos

1. Crear carpeta [Proyecto Git](#).
2. Abrir terminal dentro de la carpeta. O hacer uso de Visual Studio Code
3. Ejecutar: [git init](#)
4. Utilizar el archivo “certificado” de la siguiente carpeta ([Carpeta de código](#)) o crear un [index.html](#) con tu proyecto.
5. Ver estado: [git status](#)
6. Agregar archivo: [git add .](#)
7. Primer commit: [git commit -m "Inicializamos el proyecto y agregamos descripción"](#)

Histórico de versiones y exploración

Ver histórico simple

`git log`

Comando de Git que te permitirá ver el histórico completo de los cambios (commits) que se realizaron, muestra una lista cronológica inversa, es decir del más reciente al más antiguo.

Cada commit muestra: Hash (recordemos era el código de identificación del commit), el autor indicando usuario e email, fecha incluido el horario y el mensaje que se agregó en el commit

Ejemplo:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Talento-tech\Desktop\Proyecto Git> git log
commit c4fff43c9886a6262ed20748f1a312f7d3135625 (HEAD -> master)
Author: talentotech-lab <alumnotalentotech@gmail.com>
Date:   Wed Dec 10 16:45:08 2025 -0300

    Primer commit
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

Ver los cambios de un commit

`git show`

Muestra información detallada, también se puede ver el histórico desde VS Code.

Ejemplo:

```
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git show
commit d2dbcb3671d90e294a22afee88ceb691bb0a8942 (HEAD -> master)
Author: talentotech-lab <alumnotalentotech@gmail.com>
Date:   Wed Dec 10 16:59:24 2025 -0300

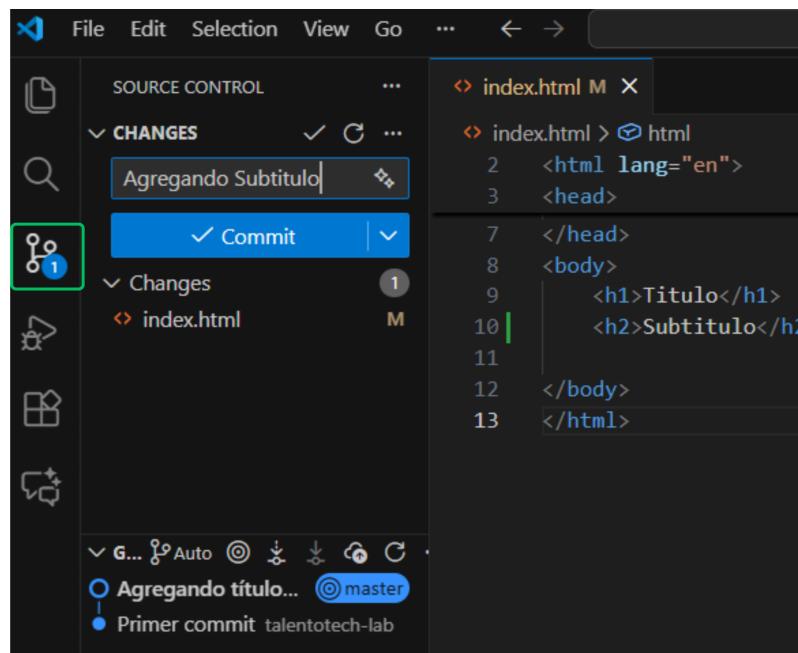
    Agregando título

diff --git a/index.html b/index.html
index d01f779..7fb289d 100644
--- a/index.html
+++ b/index.html
@@ -6,6 +6,7 @@
<title>Document</title>
</head>
<body>
+    <h1>Titulo</h1>

</body>
</html>
\ No newline at end of file
```

Integración con VS Code

- Abrir carpeta del repositorio en VS Code.
- Usar el panel **Source Control**.
- Permite ver cambios, seleccionar archivos y hacer commits desde una interfaz visual.



Mini-práctica 2

Modificamos el proyecto y analizamos el historial

Objetivo: Practicar múltiples commits y entender el historial.

Pasos

1. Modificar archivo principal y ver su estado.
2. Agregar cambios: `git add .`
4. Crear commit. Por ejemplo: `git commit -m "Agregamos nuestro nombre a la página"`.
5. Ver historial: `git log`
6. (Opcional) Ver detalles del último commit: `git show`

Cierre del encuentro

Al finalizar este encuentro comprendimos:

- Qué es Git y para qué sirve.
- Cómo funciona el flujo local de trabajo.
- Cómo registrar cambios con commits.

En el próximo encuentro sumaremos GitHub, repositorios remotos y trabajo colaborativo.

Desafío

Versionamos nuestro primer mini–proyecto web.

Objetivo: Aplicar el flujo completo sobre el proyecto HTML/CSS, personal o de referencia.

Si es tu proyecto personal, podés crear una página simple utilizando HTML y CSS, con un título, párrafo y secciones. Si estás usando nuestro código de referencia, agregá tu nombre al HTML y modificá algunos estilos.

Realizar al menos **tres commits significativos**:

- **Commit 1:** Estructura base.
- **Commit 2:** Estilos básicos.
- **Commit 3:** Ajustes de contenido.

Verificar historial, logs y mensajes claros para cualquier persona que lea el proyecto.

Resultado esperado: Al finalizar este desafío, deberás contar con un **repositorio Git local funcionando**, que incluya un **proyecto HTML/CSS básico** y un **historial de commits claro y ordenado**, aplicando correctamente el flujo **add + commit**.

 **Este desafío es clave para la Sesión 2**, ya que el repositorio local que construyas será la base para trabajar con **GitHub, repositorios remotos y colaboración**, sin necesidad de rehacer el proyecto.



Buenos Aires
aprende:

Agencia de Habilidades para el Futuro

