Hoja informativa: Git

Git es un **sistema de control de versiones**. Realiza un seguimiento de todos los cambios introducidos en el proyecto (y sus autores y autoras) y puede revertir los archivos a las versiones anteriores en caso de errores.

El archivo `.gitignore`

Hay varios tipos de archivos que no se deben subir a los repositorios. Ya sea porque no es necesario o por seguridad. Incluyen archivos generados automáticamente, tales como los registros. Para que Git los descarte, debes agregar un archivo especial `.gitignore` a tu proyecto.

El archivo `README.md`

Es una descripción de tu proyecto que ayudará a tus colaboradores a entender cómo usarlo.

README es un archivo de texto markdown con formato `.md`. Markdown es similar a HTML: todos los encabezados, párrafos e hipervínculos tienen una forma de indicarse, solo que con símbolos especiales en lugar de etiquetas.

Repositorio local

Un repositorio es un lugar donde se almacenan los datos del proyecto, el historial de cambios, entre otras cosas. Básicamente, aquí es donde el sistema recopila todas las versiones guardadas de tu proyecto.

Normalmente se crea un repositorio individual para cada proyecto.

Cómo crear un repositorio local

Accede a la carpeta del proyecto con el comando `cd` y especifica la ruta completa al proyecto entre comillas:

```
cd "ruta completa al proyecto"
```

Luego, debes informar a Git que se deben rastrear los archivos en esta carpeta. Esto se llama **inicialización**. A este fin se utiliza el comando `git init`.

```
git init
```

Estados de los archivos en Git

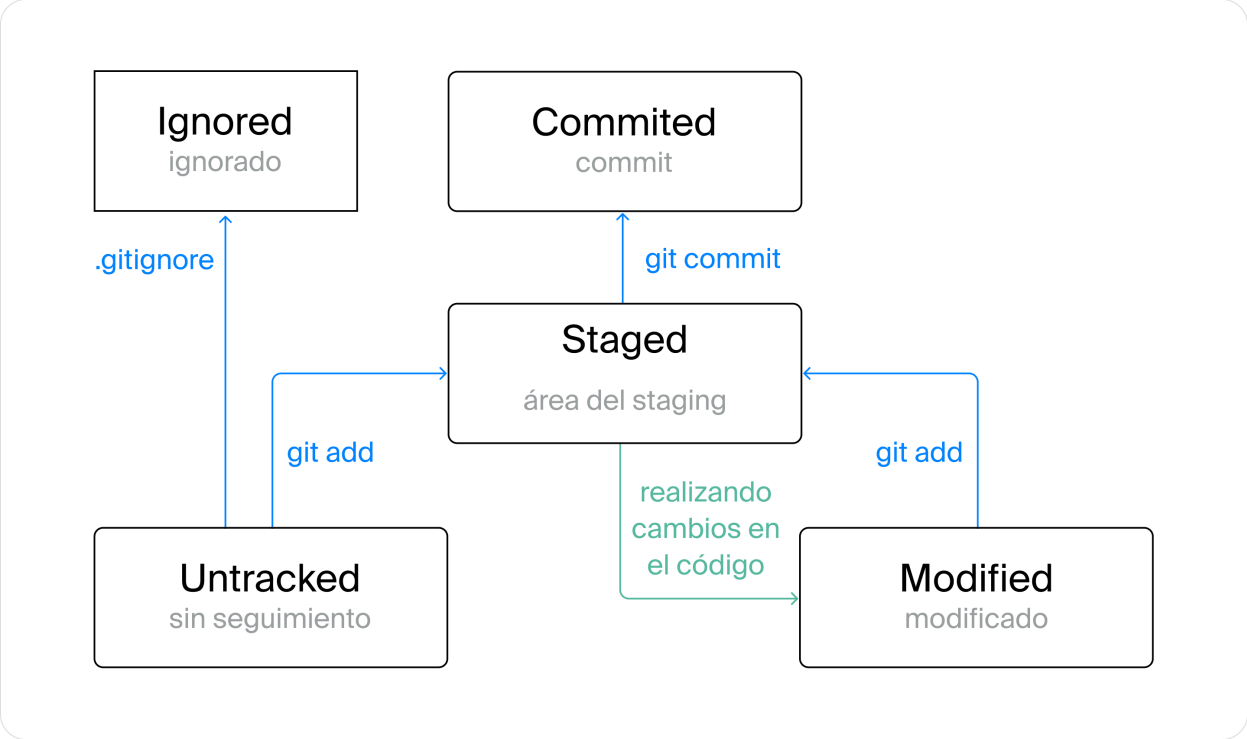
El control de versiones Git implica que cada archivo en un repositorio existe en uno de cuatro estados:

1. **Untracked** (Sin seguimiento).
2. **Staged** (Añadido al área de preparación).
3. **Modified** (Modificado).
4. **Committed** (Confirmado).

Aquí tienes una explicación:

- Cuando un archivo aparece en un directorio inicializado, está sin seguimiento ("untracked"). Puedes hacer lo que quieras en este archivo y Git no rastreará los cambios.
- Para que Git comience a rastrear este archivo, debes prepararlo colocándolo en el área de preparación, o **staging**, donde se almacena temporalmente. El estado del archivo cambiará a "staged". Para hacerlo, necesitarás el comando `git add`.
- Si realizas algún cambio en el archivo que está en el área de staging, se modifica ("modified").
- Cuando hayas terminado de editar el archivo, debes confirmarlo, lo que significa que se le informa a Git que se debe guardar la versión actual del archivo. Esto te permite volver a esta versión particular del archivo en caso de que algo salga mal más adelante.

Este proceso se llama crear un **commit** (confirmación).



💡 Git realiza un seguimiento de todos los cambios en los archivos en el área de staging, pero no los registra. Solo registra que el archivo ha sido modificado, haciendo una especie de borrador de cada nuevo estado. Pero este borrador es solo una plantilla para el futuro commit.

Cómo mostrar el estado del archivo

Para mostrar el estado de los archivos en un repositorio, debes ejecutar el comando `git status`.

```
git status
```

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
```

```
        Readme.md
        configuration.py
        create_user_test.py
        data.py
        main.py
        sender_stand_request.py

nothing added to commit but untracked files present (use "git add" to track)
```

Git puede ver los archivos, pero aún no han sido preparados. Los archivos **sin seguimiento** están resaltados en rojo. Git no se ha inicializado para rastrear sus estados.

Cómo agregar archivos al área de staging

El comando `git add` agrega archivos al **área de staging**.

Este comando va seguido del nombre del archivo que se rastreará: `git add file_name`.

```
git add data.py
# Agregar data.py al área de staging
```

Puedes añadir todos los archivos a la vez con el comando `--all`. Todos los archivos de todos los subdirectorios también se agregarán: `git add --all`.

Puedes usar un punto en lugar de `-all`: `git add .`.

```
git add --all
# Agrega todos los archivos al área de staging

git add .
# Agrega todos los archivos al área de staging
```

Cómo hacer un commit

```
git commit -m "Mi primer commit"
# Primer commit creado
# El comentario es "Mi primer commit"
```

Después de ejecutar este comando, verás el siguiente mensaje:

```
[master (root-commit) b40b200] Mi primer commit
 7 files changed, 216 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Readme.md
 create mode 100644 configuration.py
 create mode 100644 create_user_test.py
 create mode 100644 data.py
 create mode 100644 main.py
 create mode 100644 sender_stand_request.py
```

La primera línea muestra tu comentario: `Mi primer commit`. Antes de eso, entre corchetes, tenemos información sobre la confirmación:

- **main**: el nombre de la rama donde se hizo el commit.
- **root-commit** significa que este es el primer commit. Solo aparece una vez, durante ese primer commit.
- **Checksum**: los primeros siete caracteres del hash único asignado al commit. En nuestro caso, es `b40b200`.

A continuación, encontrarás información sobre los archivos confirmados:

- El número de archivos actualizados: `7 files changed`.
- El número de líneas agregadas y eliminadas (en este caso, solo agregadas): `216 insertions(+)`.
- Los archivos y la acción realizada: `create mode 100644 data.py`.

Comentar sobre el commit

En el comentario de tu commit, debes dejar claro qué se cambió y porqué. Crea el hábito de escribir estos comentarios, ya que te ayudarán a ordenar las versiones registradas más adelante.

Si intentas hacer un commit sin un comentario, verás el siguiente mensaje: "Please enter the commit message for your changes" (Ingresa el mensaje de commit para tus cambios).

Repositorio remoto en GitHub

Para vincular tu repositorio en línea a tu repositorio local, debes ir a la carpeta del repositorio local en la consola y usar el comando `git remote add`. Pasa dos parámetros:

el nombre del repositorio remoto y su URL.

```
git remote add origin remote_repository_URL.git
```

`origin` es un alias estándar que se puede utilizar para referirse a cualquier repositorio remoto.

Cómo subir cambios a GitHub

Para subir, o empujar, cambios a un repositorio remoto, usa el comando `git push`.

El repositorio local se copió del servidor, por lo que para subir cambios por primera vez, debes usar `git push` con un indicador `u` y dos argumentos:

- El primer argumento es el nombre del servidor del que se copió el repositorio `origin`.
- El segundo argumento es el nombre de la rama `main`.

El indicador `-u` vincula la rama local con la remota. Necesitas usar este flag cuando publiques nuevas ramas.

```
git push -u origin main  # Si este comando provoca un error,
                         # intenta reemplazar main por master
```