Guia Git
Guia SQLite

MediCare Desk

Presentado por:

Yamid Alfonso Gonzalez Torres

Jenny Catherine Herrera Garzon

Edwin Andres Marin Vanegas

Diego Steven Pinzon Yossa

Profesor:

Oscar Eduardo Alvarez Rodriguez

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de software

2025

Antes: verificar una versión de python 3.7 o superior

1. Descargar Git e instalar

Ve al sitio oficial y descarga el instalador para Windows:

👉 https://git-scm.com/downloads

Haz clic en Windows y se descargará el instalador (.exe).

con git --version en el cmd pueden comprobar si se instalo correctamente (incluso se puede validar antes de instalar en caso de que ya esté instalado)

2. Configurar tu nombre y correo

git config --global user.name "Catherine Herrera" → Define tu nombre para firmar los commits.

git config --global user.email "CatherineHerrera96@outlook.com" → Define tu correo, también para firmar los commits.

El nombre no necesariamente debe ser el usuario, por lo que puede contener espacios

con git config --list pueden comprobar si se guardó la configuración

NOTA: Pueden averiguar cómo guardar sus credenciales por su cuenta en este punto , al parecer es con

con git config --global credential.helper store. →(Le dice a Git que **guarde tus credenciales (usuario/token)** la primera vez que las uses, para no pedirlas de nuevo.)

Yo no lo hice en este punto y cuando hice el push se abrió una pestaña donde me tuve que loguear, pero es mejor si se hace en este punto. si lo logran hacer por favor actualicen este documento

3. Clonar el repositorio MediCareDesk

Abre una terminal en la carpeta (**cmd** en dicha carpeta) donde quieras trabajar y ejecuta:

git clone https://github.com/PanDebuggers/Proyecto.git

Esto descargará todo el repositorio.

Luego navega a la carpeta del proyecto:

cd Proyecto/Proyecto/MediCareDesk

aca estariamos dentro de la carpeta que contiene el código

4. Agregar archivos

En mi caso habia faltado una carpeta dentro del repositorio (resources), porque estaba vacia y git no permite guardar carpetas vacias, entonces en caso de que quieran agregar una carpeta pueden agregar un archivo vacio.txt para que si se guarde.

Entonces procedi a crearla de manera manual en mi pc, dentro de la carpeta en la que estamos trabajando. Con git status podremos ver si se ha realizado algun cambio dentro de la carpeta raiz y se vera algo asi:

```
C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>
```

ahora procedemos a añadir la carpeta al control de versiones con **git add, git commit** para guardar los cambios (se puede añadir un comentario con los cambios realizados) y **git push** para subirlo:

git add resources/

git commit -m "Agregar estructura de carpeta resources con iconos y sonidos"

git push origin main

```
C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>git add resources/

C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>git commit -m "Agregar carpeta resources con subd
irectorios de iconos y sonidos"
[main c3f4d37] Agregar carpeta resources con subdirectorios de iconos y sonidos
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Proyecto/MediCareDesk/resources/iconos/vacio.txt
 create mode 100644 Proyecto/MediCareDesk/resources/sonidos/vacio.txt

C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>git push origin main
info: please complete authentication in your browser...
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 543 bytes | 135.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/PanDebuggers/Proyecto.git
   bfbbd0b..c3f4d37  main -> main

C:\Users\Usuario\Documents\Proyecto uwu\Proyecto\Proyecto\MediCareDesk>
```

NOTA: Al hacer git push se abrió una ventana para loguearme, cosa que no había realizado. (eliminar este comentario si ya actualizaron el logueo arriba)

4. **Sincroniza antes de trabajar o subir tus propios cambios:** Hagan **pushs** frecuentes pero con **pull** previo siempre.

¿Qué es un git pull?

Es un **comando de Git** que sirve para:

> Traer los últimos cambios del repositorio remoto (en la nube, por ejemplo GitHub) hacia tu repositorio local (tu PC).

Es como decirle a Git:

> "Tráeme lo que otros han subido para que yo también lo tenga antes de seguir trabajando".

git pull origin main ← comando para la terminal

🌿 ¿Qué es una rama (branch) en Git?

Una **rama** es una "línea de trabajo" separada dentro del proyecto.
Te permite **hacer cambios, probar cosas o desarrollar funcionalidades sin afectar directamente el proyecto principal**.

La rama principal suele llamarse main.

✅ ¿Debo trabajar directamente sobre main?

Si estás trabajando **sola o hay buena coordinación** (como en MediCareDesk), **sí puedes trabajar sobre main**, pero **crear ramas personales o por módulo** te da más control y seguridad.

🛠️ ¿Cómo es el proceso típico usando ramas?

🧪 Ejemplo: quieres trabajar en el módulo de login

1. Crea una nueva rama con tu nombre o tarea:
   git checkout -b login-catherine
2. ✍️ Trabajas normalmente: editas archivos, haces commits
3. Subes la rama al repositorio remoto:
   git push origin login-catherine
4. Cuando termines, te cambias a **main**:

   git checkout main

   git pull origin main # Te aseguras de tener lo último

   git merge login-catherine # Traes tu rama a main

5. Y subes los cambios finales:

   git push origin main

Resumen:

 # CONFIGURACIÓN INICIAL (una sola vez)

git config --global user.name "Tu Nombre" # Establece tu nombre para los commits

git config --global user.email "tucorreo@ejemplo.com" # Establece tu correo para los commits

git config --global credential.helper store # Guarda tus credenciales (token) para no ingresarlos cada vez

 # CLONAR UN REPOSITORIO

git clone https://github.com/usuario/repositorio.git # Descarga el proyecto en tu PC

 # NAVEGAR AL PROYECTO

cd nombre-del-proyecto # Entra a la carpeta del proyecto

 # VERIFICAR EL ESTADO

git status # Muestra los archivos modificados o sin guardar

 # VER CAMBIOS REALIZADOS

git diff # Muestra qué cambió línea por línea desde el último commit

 # AGREGAR ARCHIVOS PARA GUARDAR

git add archivo.py # Prepara un archivo para el commit

git add . # Prepara todos los archivos modificados

 # GUARDAR CAMBIOS CON UN MENSAJE

git commit -m "Mensaje claro del cambio" # Guarda un punto de control en tu historial

 # SUBIR CAMBIOS AL REPOSITORIO REMOTO

git push origin main # Sube tus commits a la rama 'main' en GitHub

 # TRAER CAMBIOS DEL REMOTO

git pull origin main # Actualiza tu proyecto con los últimos cambios del equipo

 # TRABAJO CON RAMAS

git branch # Lista todas las ramas existentes

git checkout -b mi-rama # Crea y cambia a una nueva rama

git checkout main # Cambia de nuevo a la rama principal

git merge mi-rama # Une los cambios de 'mi-rama' a 'main'

git push origin mi-rama # Sube tu nueva rama a GitHub

 # DESCARTAR CAMBIOS ANTES DEL COMMIT

git restore archivo.py # Revierte el archivo a su última versión guardada

 # VER HISTORIAL DE COMMITS

git log --oneline # Muestra los últimos commits resumidos

 # OPCIONAL: ARCHIVO .gitignore (no es comando, pero útil)

Contenido sugerido de .gitignore:

*.pyc

__pycache__/

*.db

data/*.db

venv/

.env

Flujo para unir ramas uwu

```python
# 1. Ve a la rama principal y actualízala
git checkout main
git pull origin main

# 2. Cambia de rama (si vas a crear una nueva, usa: git checkout -b RamaCathe)
git checkout RamaCathe

# 3. Haz tus cambios, agrégalos y haz commit
git add .
git commit -m "Describe brevemente los cambios realizados"

# 4. Sube tu rama al repositorio remoto
git push origin RamaCathe

# 5. Haz un pull preventivo para asegurarte de que main está actualizado
git checkout main    # (Es obligatorio cambiarse a main, de lo contrario el pull mezcla main en tu rama)
git pull origin main
# (Si hay conflictos, resuélvelos y haz git add <archivo> y git commit)

# 6. Haz el merge (debes estar en main, es la rama que recibe los cambios)
git merge RamaCathe
# (Si hay conflictos, resuélvelos, haz git add <archivo> y git commit)

# 7. Sube los cambios fusionados al remoto
git push origin main

# 8. (Opcional) Elimina la rama si ya no la necesitas
git branch -d RamaCathe               # Elimina la rama local
git push origin --delete RamaCathe    # Elimina la rama remota
```

SQLite

¿Debes instalar DB Browser for SQLite?

No es obligatorio, pero sí recomendable.

¿Para qué sirve?

- Ver tablas y sus datos gráficamente.
- Hacer pruebas de consultas SQL.
- Insertar o editar datos manualmente (por ejemplo, probar pacientes).
- Ver si tus cambios realmente se guardaron.

¿Es necesario para que el proyecto funcione?

No. El proyecto usa `sqlite3` desde Python directamente, **no depende de DB Browser para funcionar**.

🤝 ¿Tus compañeros deberían instalarlo?

Recomendado, pero opcional.

- Les facilitará **ver qué hay en la base de datos** mientras programan.

- Podrán entender más fácilmente cómo se guardan tomas, medicamentos, etc.
- Pero si no quieren, pueden trabajar solo desde el código.

Descarga: https://sqlitebrowser.org/dl/


UNIVERSIDAD NACIONAL DE COLOMBIA
1867