1. Instalación de Git en Linux

El primer paso es instalar Git, para esto utilizamos el comando ***sudo apt install git***

```
gabrielpm@vbox:~$ sudo apt update
sudo apt install git
[sudo] password for gabrielpm:
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://security.debian.org/debian-security bookworm-security InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Ign:4 https://releases.warp.dev/linux/deb stable InRelease
Hit:5 https://releases.warp.dev/linux/deb stable Release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
110 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl patch
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
```

2. Obtener Git a través del propio Git

Ahora para configurar Git de forma que se actualice automáticamente desde su propio repositorio debemos realizar varios cambios:

Lo primero es irnos a nuestra carpeta src con ***cd /usr/local/src*** una vez aquí clonamos el repositorio oficial de Git con ***sudo git clone https://github.com/git/git.git*** Finalmente compilamos el código del repositorio en nuestra carpeta y lo instalamos con: ***cd git sudo make prefix=/usr/local all sudo make prefix=/usr/local install***

Para actualizar Git de esta forma solamente haría falta hacer un pull del repositorio con ***git pull origin master*** y cada vez que se haga repetir el proceso de compilado e instalación con ***sudo make prefix=/usr/local all sudo make prefix=/usr/local install***

Otra opción podría ser la de clonar el repositorio cada ve que una nueva versión salga con: ***git clone https://github.com/git/git***

3. Comprobar la versión instalada de Git

Para comprobar la versión de Git simplemente usamos el comando porpio e Git seguido de –-version. En mi caso es la versión 2.39.5

```
gabrielpm@vbox:~$ git --version
git version 2.39.5
```

4. Configurar el nombre de usuario y correo electrónico en Git

Para configurar el nombre de usuario y correo con git solamente debemos utilizar git config. Por ejemplo en mi caso:

git config --global user.name "Gbapmm"

git config --global user.email "gvpaz@icloud.com"

Para comprobar que se haya aplicado nuestra configuración podemos usar ***git config --global –list***

```
gabrielpm@vbox:~$ git config --global user.email "gvpaz@icloud.com"
gabrielpm@vbox:~$ git config --global --list
user.email=gvpaz@icloud.com
user.name=Gbapmm
```

5. Cambiar el editor de texto predeterminado a Emacs

Para cambiar el editor d etexto predeterminado usaermos el mismo comando que en el paso anterior, git config – global pero esta vez utilizando también core.editor:

git config --global core.editor Emacs

Igual que en el paso anterior podemos verificar si el cambio ha surgido efecto:

```
gabrielpm@vbox:~$ git config --global core.editor emacs
gabrielpm@vbox:~$ git config --global core.editor
emacs
```

6. Crear un nuevo repositorio en /var/cache/git/

Para crear un nuevo repositorio simplemente creamos las carpetas donde se almacenará y lo inicializamos:

sudo mkdir -p /var/cache/git/repoNuevo

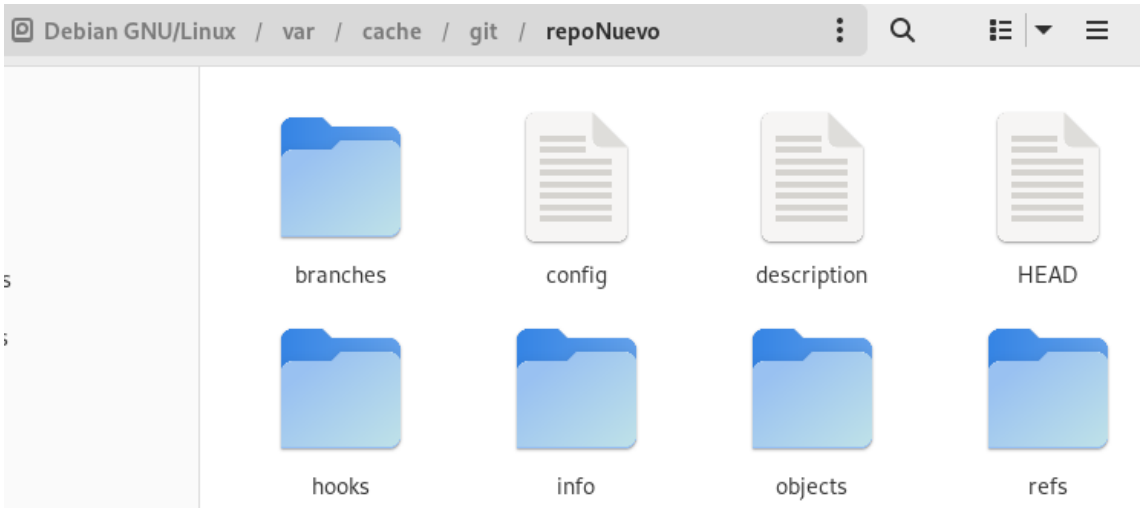
cd /var/cache/git/tarea_ repoNuevo

sudo git init --bare

```
gabrielpm@vbox:~$ sudo mkdir -p /var/cache/git/repoNuevo
```

```
gabrielpm@vbox:~$ cd /var/cache/git/repoNuevo
gabrielpm@vbox:/var/cache/git/repoNuevo$ sudo git init --bare
hint: Using 'master' as the name for the initial branch. This default br
e
hint: is subject to change. To configure the initial branch name to use
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this com
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /var/cache/git/repoNuevo/
```

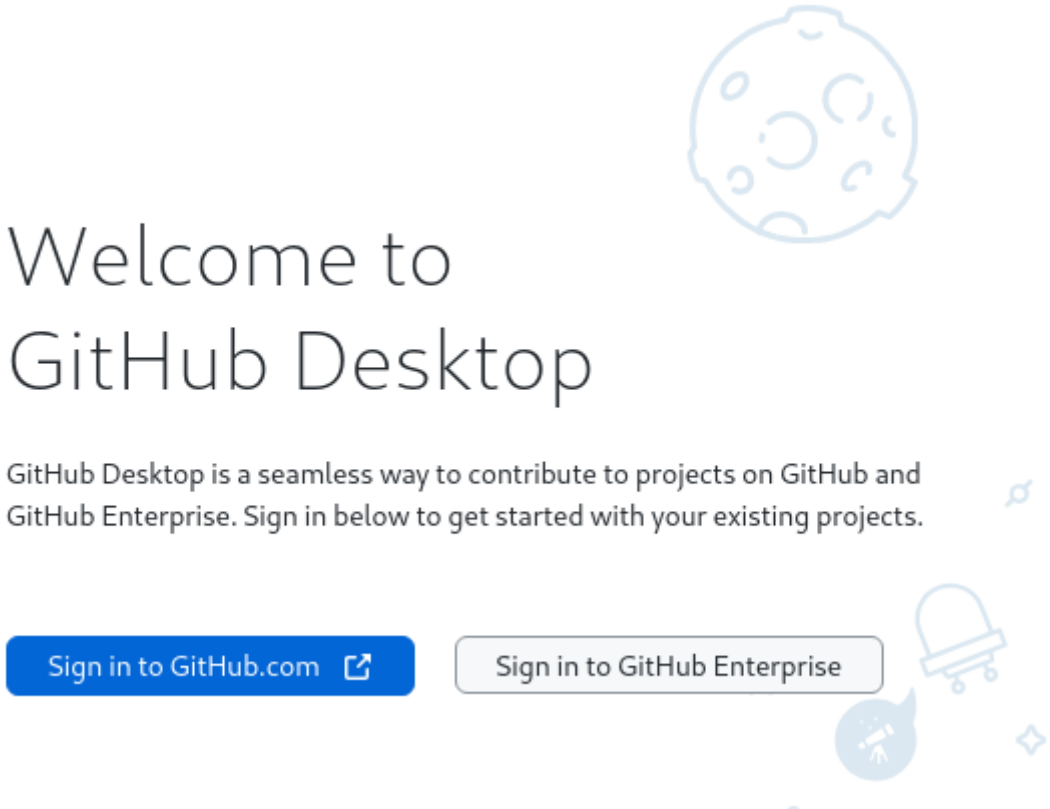
Navegamos al directorio que hemos creado y podemos ver el repositorio, que en este caso al crearlo como -- bare está “vacio” es decir, no tiene un workspace asociado. Para crearlo con workspace deberíamos eliminar el – bare.



Github Desktop no tienen una versión oficial para Linux, sin emabrgo hay numerosas versiones mantenidas por la comunidad, una de ellas se encuentra en este repositorio. La instalación es bastante simple si se siguen las indicaciones del readme.md del repositorio.

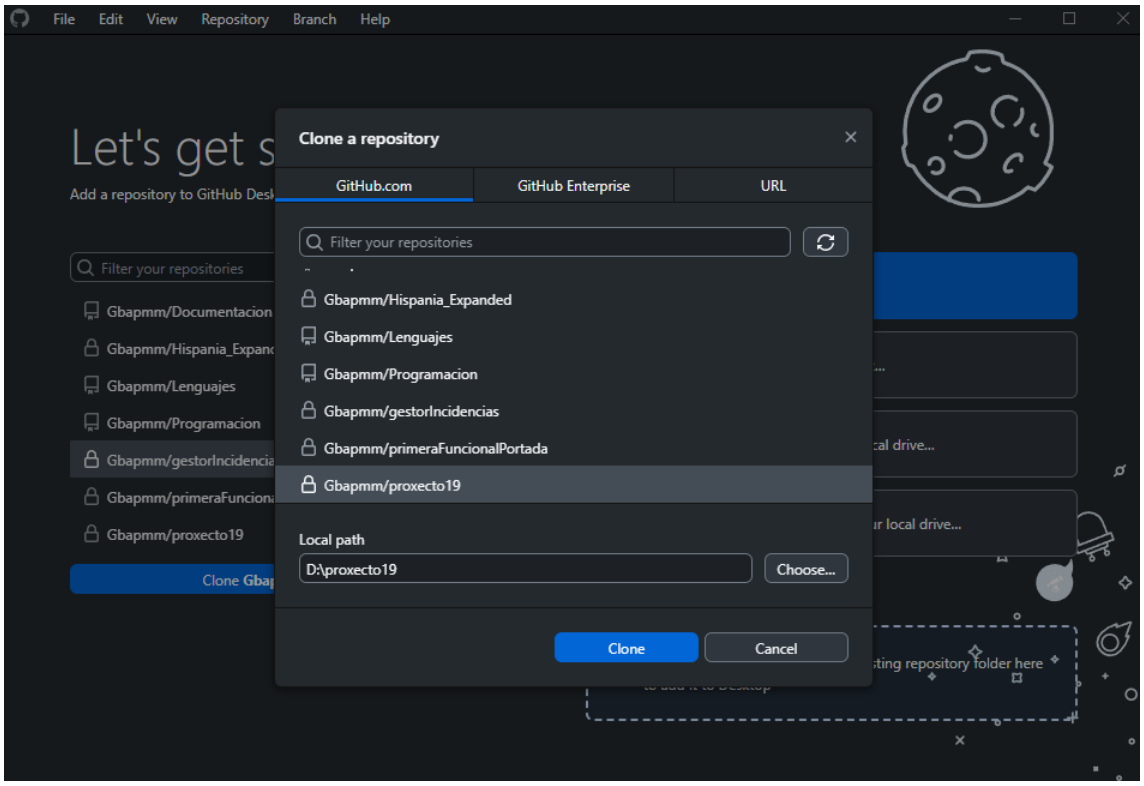
```
gabrielpm@vbox:~$ wget -qO - https://apt.packages.shiftkey.dev/gpg.key | gpg -
earmor | sudo tee /usr/share/keyrings/shiftkey-packages.gpg > /dev/null
sudo sh -c 'echo "deb [arch=amd64 signed-by=/usr/share/keyrings/shiftkey-packa
s.gpg] https://apt.packages.shiftkey.dev/ubuntu/ any main" > /etc/apt/sources.
st.d/shiftkey-packages.list'
gabrielpm@vbox:~$ sudo apt update && sudo apt install github-desktop
Hit:1 http://security.debian.org/debian-security bookworm-security InRelease
Hit:2 http://deb.debian.org/debian bookworm InRelease
```

Tras instalarlo nos pedirá iniciar sesión:

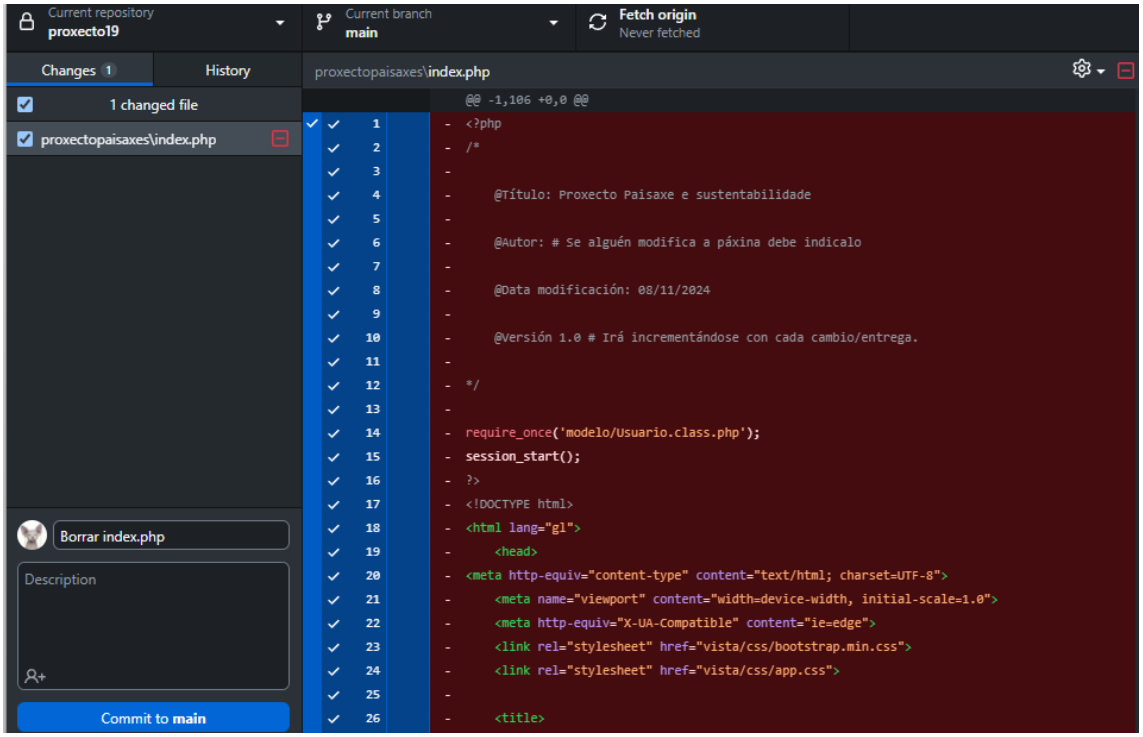


Tras iniciar sesión con nuestra cuenta desde github.com podremos trabajar con git. Por ejemplo podemos clonar, añadir o eliminar nuestros repositorios sin tener que usar líneas de comando (Lo haré fuera de la máquina virtual para agilizar el mostrar la funcionalidad):

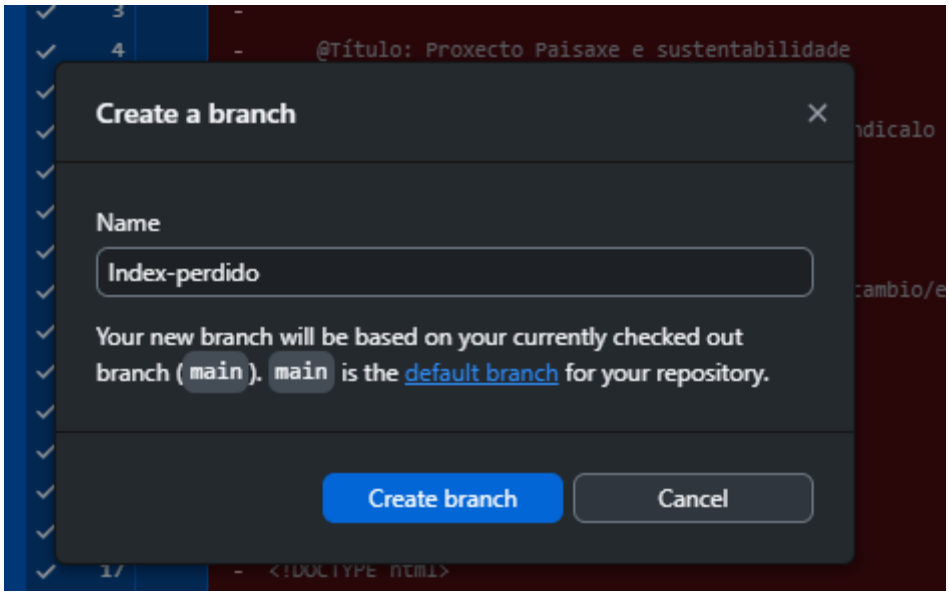
Clonaré mi repositorio del proyecto de Desarrollo en contorno servidor ya que tengo que terminarlo antes de la fecha de entrega, para esto, selecciono el repositorio y configuro su ruta:



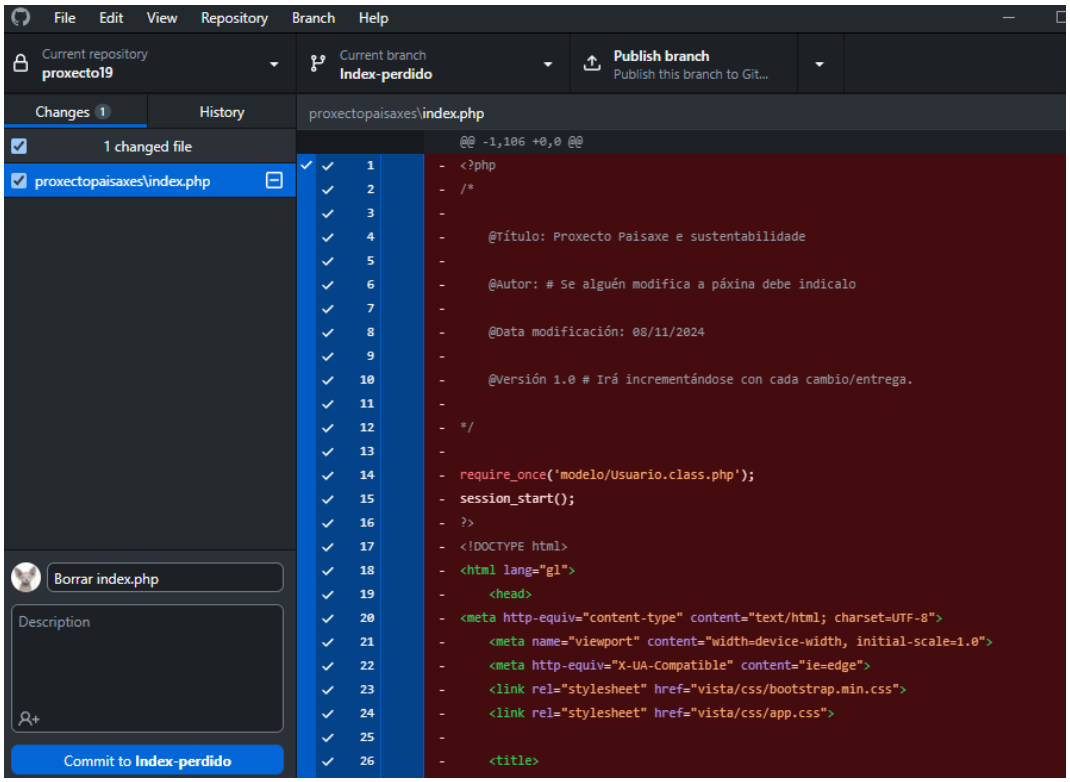
Tras esto si yo eliminase en mi copia local del proyecto por ejemplo el archivo index.php podría hacer un commit desde la interfaz gráfica:



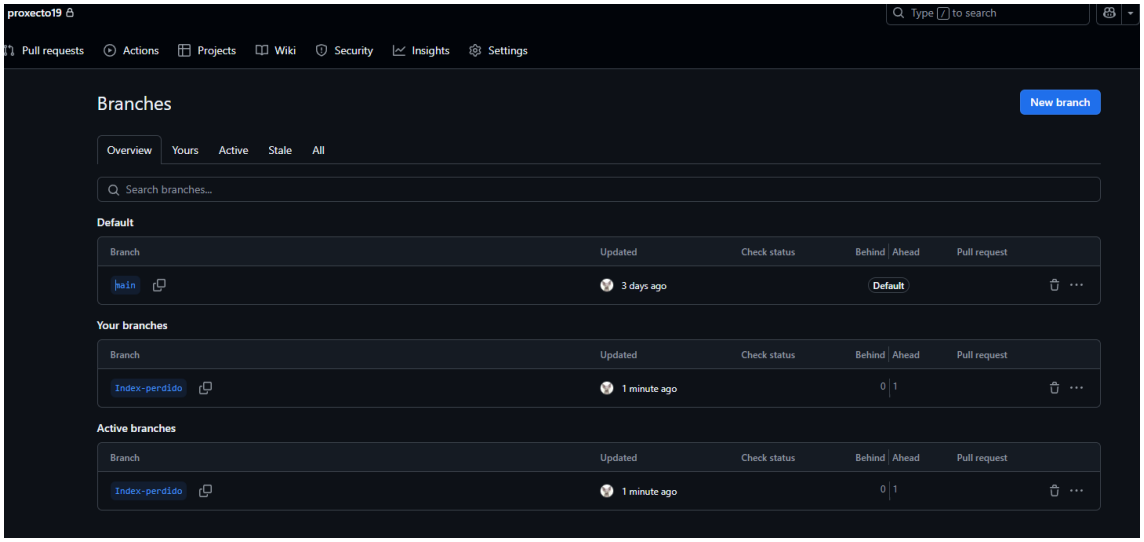
Sin embargo no quiero borrar este archivo index.php de mi repositorio así que en vez de hacer el *commit* a mi rama principal voy a crear una nueva rama en la que este archivo no esté:



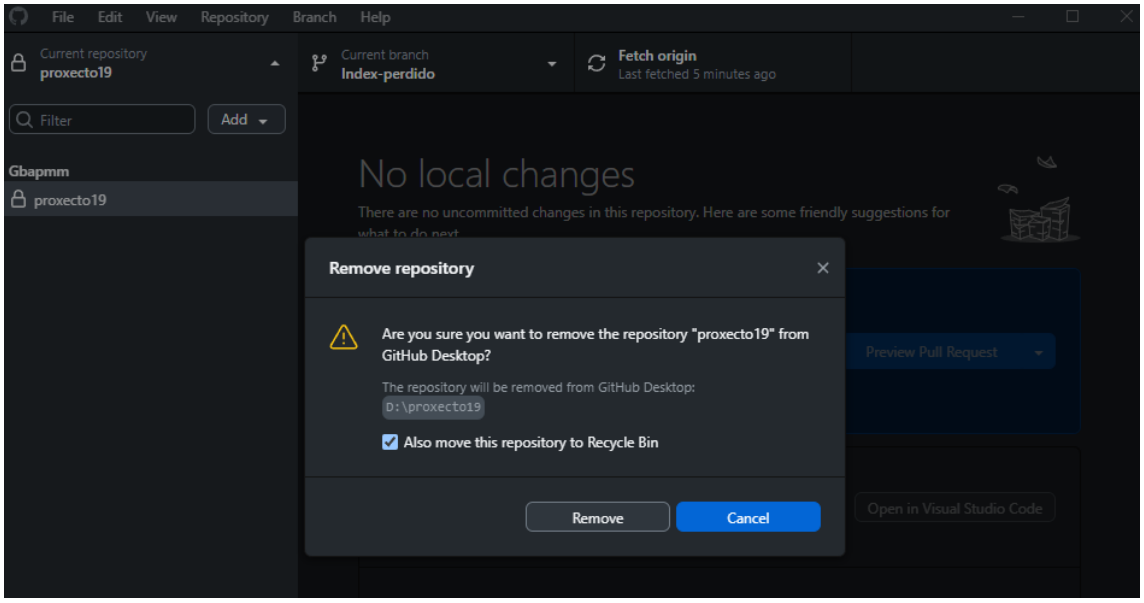
Ahora mi cambio se efectuará en la nueva rama, manteniendo la rama principal sin modificar:



Tras hacer el commit y publicar la rama si voy a mi repositorio en github.com podré ver como ahora tengo dos ramas, la rama principal o main y la activa, Index-perdido:

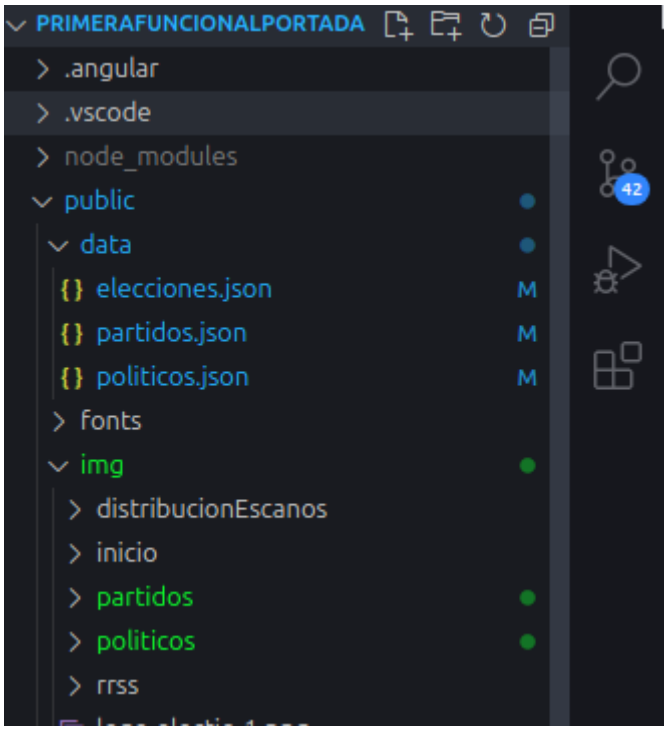


Desde Github Desktop también podemos eliminar fácilmente nuestros repositorios locales haceindo click derecho y seleccionando remove, esto puede ser útil para desechar proyectos viejos rápidamente. También se puede borrar el repositorio directamente de Github.

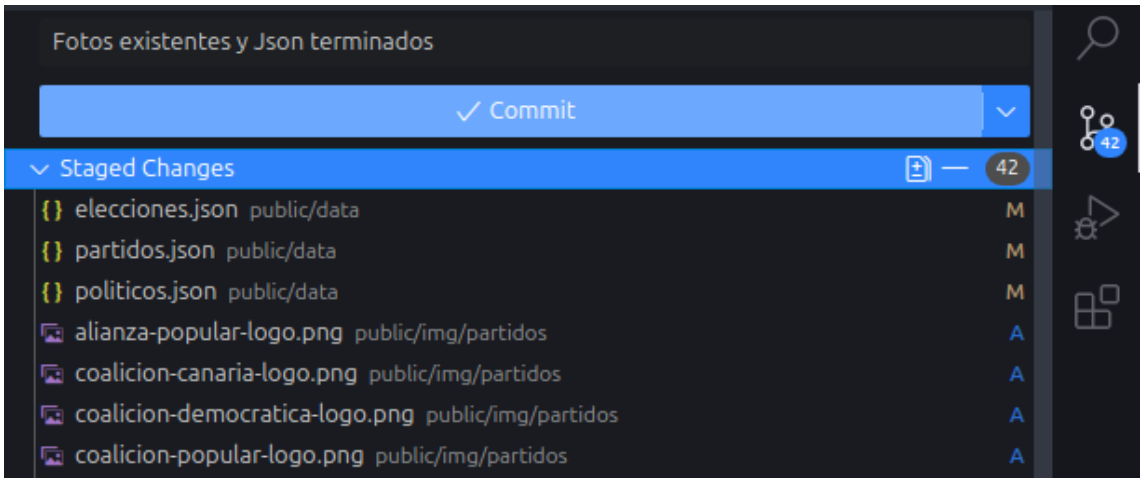


Para integrar Git en un IDE como el Visual Studio Code simplemente debemos abrir nuestro repositorio, en mi caso tengo un repositorio con mi proyecto de desarrollo en contorno en cliente en el que he hecho numerosos cambios:

Como se puede ver en la estructura de mis archivos los puntos en verde indican nuevas creaciones de archivo mientras que los pntos azules y las M modificaciones, para añadir a mi repositorio los cambios debo seleccionar el icono debajo del buscador:



Ahora se puede ver un historial con todas las nuevas modificaciones o creaciones (así como eliminaciones, pero en mi caso no hay). Para realizar un commit es obligatorio un mensaje (cosa que con Github Desktop no siempre es necesario) y seleccionar los cambios a los que hacer commit, en mi caso serán todos:



Tras esto, se hará el commit y se actualizará nuestro mapa de la rama principal, solamente hará falta hacer pull al repositorio en línea dándole a sincronizar cambios: