PRÁCTICA 1 TEMA 5

15 DE MAYO DE 2025

CURSO DE ESPECIALIZACIÓN EN CIBERSEGURIDAD

Puesta en Producción Segura

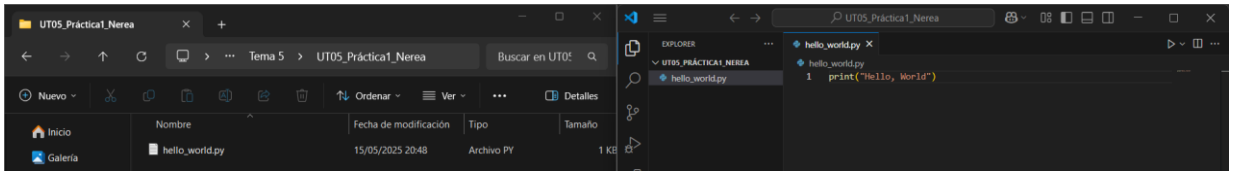
TAREA 1

Con lo visto en la Actividad 1 y en la actual práctica se pide realizar las siguientes acciones relacionadas con Git y GitHub. Para cada acción se puede emplear un programa de ejemplo con Python el cuál puede ser generado con inteligencia artificial, y que será de libre desarrollo por parte del alumnado.

1. Crear un repositorio local.

Para hacer esto, debemos seguir los pasos establecidos en el documento que contiene la guía de la actividad 1.

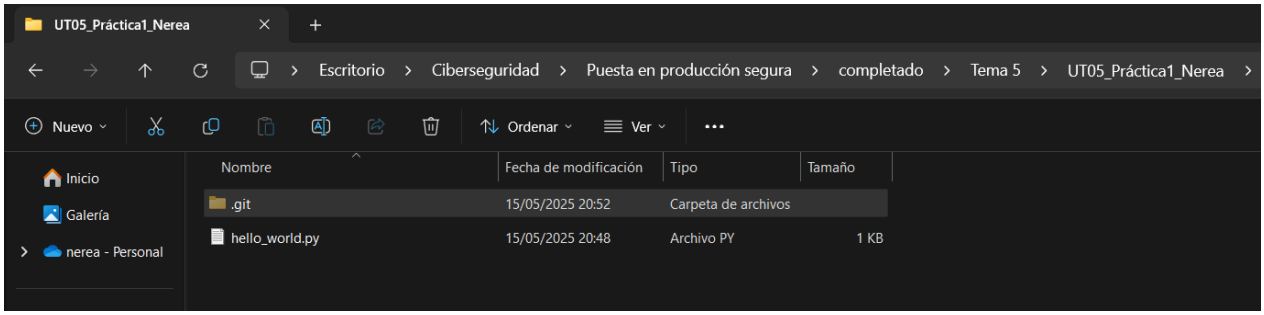
En primer lugar, crearemos una carpeta donde queramos, mediante la interfaz gráfica de nuestro ordenador. Luego, la abriremos en Visual Studio Code y crearemos un nuevo fichero Python cualquiera:



Una vez hecho esto, nos vamos al directorio que contiene el fichero Python en el programa Git Bash con el comando ***cd \ruta\al\directorio*** y, cuando ya estemos dentro, ejecutamos ***git init*** para trabajar con Git en esta carpeta.

```
nerxt@PC-NEREA MINGW64 ~/Desktop
$ cd Ciberseguridad/Puesta\ en\ producción\ segura/completado/Tema\ 5/UT05_Práctica1_Nerea/

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea
$ git init
Initialized empty Git repository in C:/Users/nerxt/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea/.git/

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (master)
$
```

Una vez hecho esto, se creará una carpeta oculta en el directorio llamada .git:



2. Cambiar el nombre de la rama principal a “main” y hacer un commit.

Al crear el repositorio local, la rama se llama por defecto “master”, para cambiar el nombre ejecutamos el comando ***git branch -m “main”***:

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (master)
$ git branch -m "main"
```

Como se puede ver, ya hemos cambiado el nombre, por lo que ahora debemos hacer el commit pero, si lo ejecutamos con la carpeta vacía, saldrá un error:

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git commit
On branch main

Initial commit

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

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

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$
```

Es por ello que a continuación, añadiremos el fichero de Python ***git add hello_world.py***, listaremos para comprobar ***ls*** y luego ejecutaremos el commit ***git commit -m “apartado 2”:***

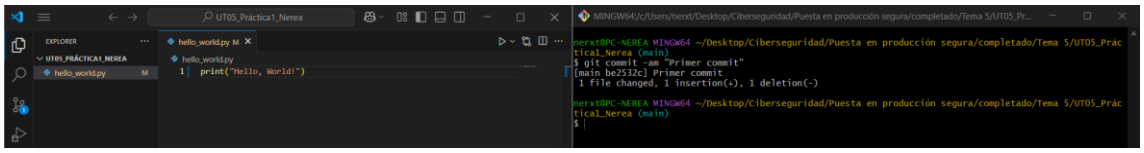
```
MINGW64:/c/Users/nerxt/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Pr...

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git add hello_world.py

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ ls
hello_world.py

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git commit -m "apartado 2"
[main (root-commit) 6867569] apartado 2
 1 file changed, 1 insertion(+)
 create mode 100644 hello_world.py

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$
```

Una vez hecho el commit, hacemos un ***git status*** para comprobar que se haya completado el commit y, para ver que se ha realizado correctamente, ejecutamos ***git log***:

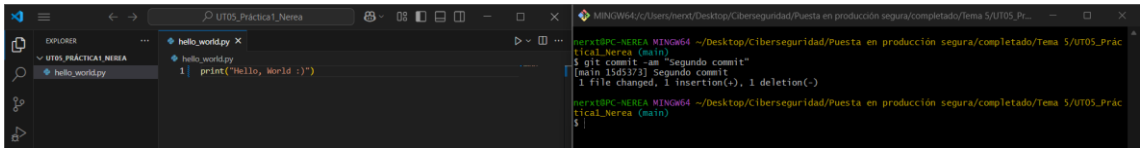
```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git status
On branch main
nothing to commit, working tree clean

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git log
commit 68675698304a2012f1a79b6c01c024235b328e00 (HEAD -> main)
Author: Nerea Carrasco <nerxtkd@gmail.com>
Date:   Thu May 15 21:17:34 2025 +0200

    apartado 2

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$
```

3. Hacer 3 commits y volver al primero.

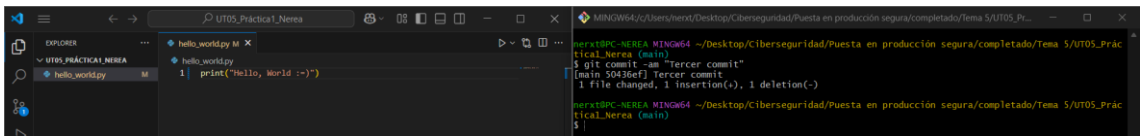
Primero, haremos los 3 commits, cambiando el contenido del fichero de cualquier forma y usando comentarios para indicar en cada uno de ellos el orden, ***git commit -am "comentario***. Usamos el parámetro *-am* ya que, solo *-m* me estaba dando fallo ya que el fichero no estaba siendo añadido cada vez, este nuevo parámetro, usa la letra *a* para añadir el fichero y la *m* para añadir un mensaje o comentario. El primer cambio ha sido:



El segundo es:



Y, el tercer y último cambio va a ser:



Luego de hacer los cambios, ejecutamos el comando ***git status*** para comprobar que se haya creado todo:

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git status
On branch main
nothing to commit, working tree clean
```

Luego, para obtener los IDs, ejecutamos ***git log --oneline:***

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git log --oneline
50436ef (HEAD -> main) Tercer commit
15d5373 Segundo commit
be2532c Primer commit
6867569 apartado 2
```

Ahora mismo, el fichero saca el texto ***Hello, World :=)*** que es el tercer commit. Como queremos que saque el texto del primer commit, ***Hello, World!***, usamos ***git checkout <hash_del_primer_commit>***:

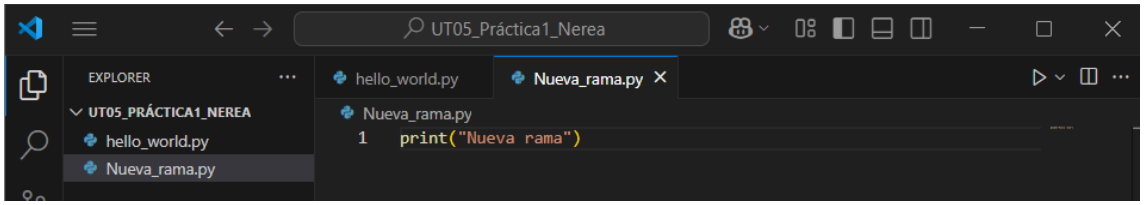
```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git checkout be2532c
Note: switching to 'be2532c'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at be2532c Primer commit
```

Por último, para comprobar que hayamos vuelto al primer commit, ejecutamos un ***git log*** de nuevo y, como vemos, se han eliminado los dos últimos commits:

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea ((be2532c...))
$ git log
commit be2532c750b8c0f2bb8af0d911d341714163a8ea (HEAD)
Author: Nerea Carrasco <nerxtkd@gmail.com>
Date:   Thu May 15 21:28:35 2025 +0200

    Primer commit

commit 68675698304a2012f1a79b6c01c024235b328e00
Author: Nerea Carrasco <nerxtkd@gmail.com>
Date:   Thu May 15 21:17:34 2025 +0200

    apartado 2

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea ((be2532c...))
$
```

4. Crear otra rama y hacer un commit sobre ella.

Primero, creamos otro fichero de Python igual que hemos creado el primero:



Después, dentro de la rama *main*, usamos el comando ***git branch <nombre_rama>*** para añadir la nueva rama y, posteriormente, nos cambiaremos a dentro de esa rama para seguir, ***git checkout <nombre_rama>*** o ***git switch <nombre_rama>*** :

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git branch nueva_rama2

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git checkout nueva_rama2
Switched to branch 'nueva_rama2'

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (nueva_rama2)
$
```

Una vez creada la rama, añadimos el fichero de Python:

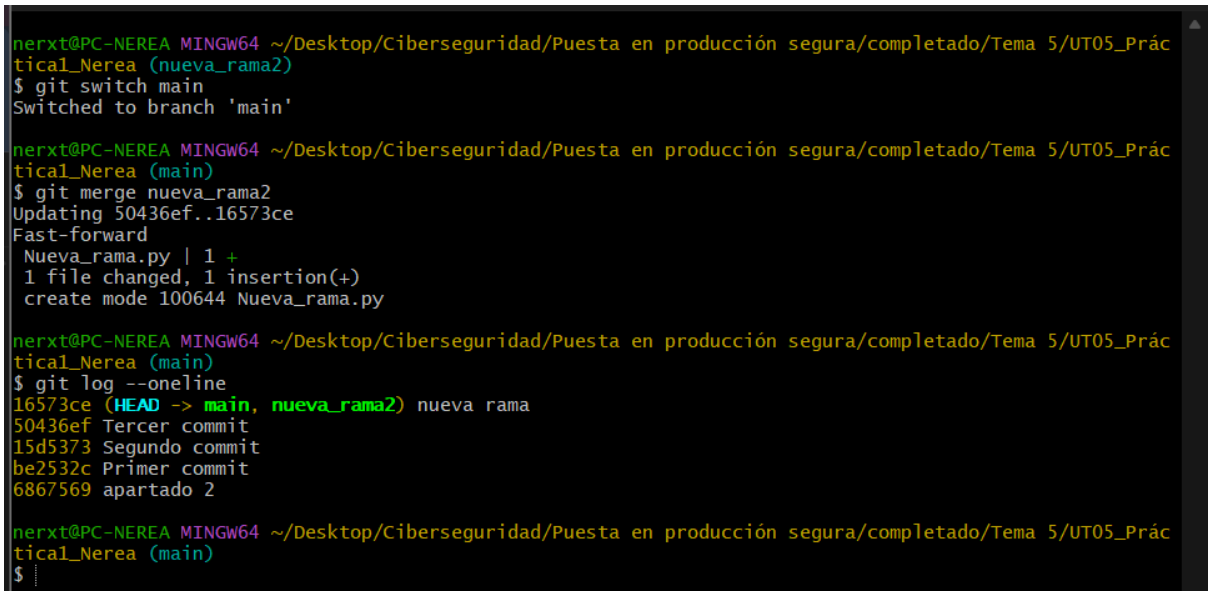
```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (nueva_rama2)
$ git add Nueva_rama.py
```

Cuando se añada, ejecutamos el commit con un comentario y, después, comprobamos que se haya completado. Como se observa, salen todos los anteriores pero se indica que son de la rama *main.*

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (nueva_rama2)
$ git commit -m "nueva rama"
[nueva_rama2 16573ce] nueva rama
 1 file changed, 1 insertion(+)
 create mode 100644 Nueva_rama.py

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (nueva_rama2)
$ git log --oneline
16573ce (HEAD -> nueva_rama2) nueva rama
50436ef (main) Tercer commit
15d5373 Segundo commit
be2532c Primer commit
6867569 apartado 2
```

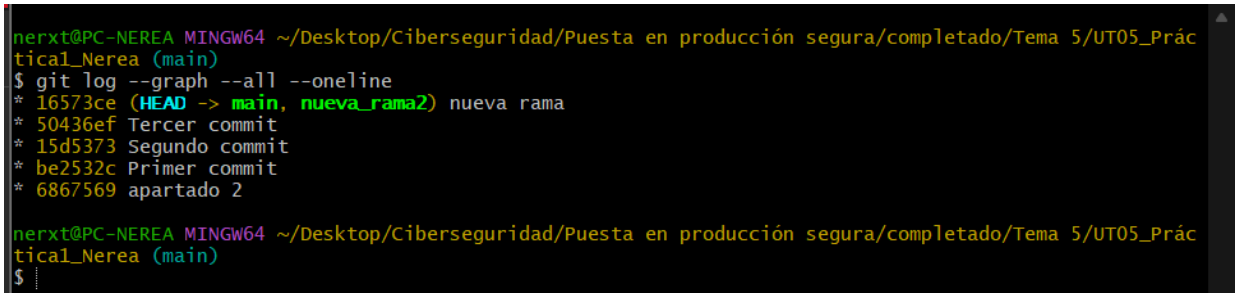
5. Actualizar la rama “main” con los cambios realizados en la otra rama, realizando un merge.

Para fusionar el trabajo de las dos ramas, usaremos el comando ***git switch main***, para movernos a la rama principal y luego ***git merge nueva_rama*** para actualizarla. Si hacemos un ***git log --oneline*** al finalizar, podremos comprobar que efectivamente se ha fusionado el trabajo:



6. Mostrar el árbol de commits.

En mi caso, como solo lo voy a usar ahora, no le voy a asociar un alias, por lo que el comando a usar será ***git log --graph --all --oneline*** (el orden de los parámetros da igual):



7. Crear un repositorio en GitHub.

A partir de este ejercicio, dejamos atrás lo visto en la actividad 1 y comenzamos con lo relacionado con GitHub.

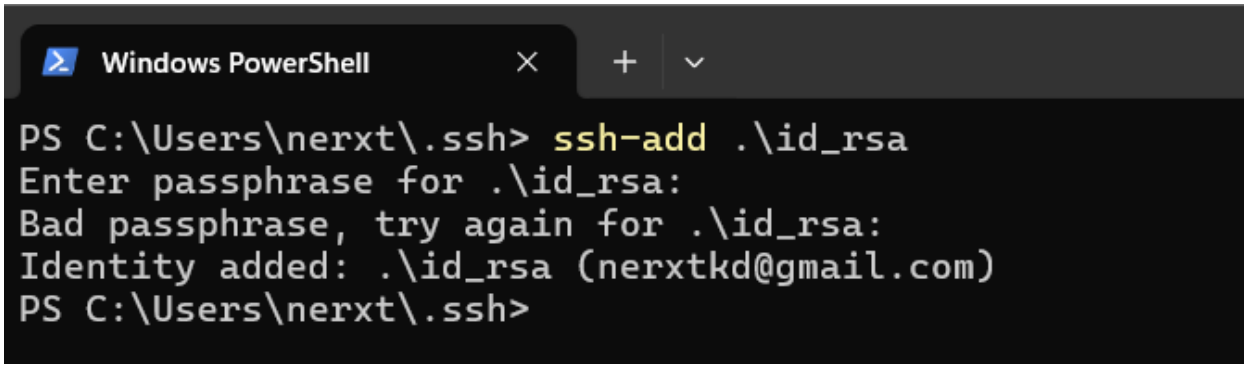
El primer paso será crearnos una cuenta en este foro y conectarnos mediante SSH como se ha indicado en la práctica. Para ello, generamos una clave a ssh en el directorio que queramos con clave ***ssh-keygen -t ed25519 -C "correo@gmail.com" :***

```
nerxt@PC-NEREA MINGW64 ~
$ ssh-keygen -t ed25519 -C "correo@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/nerxt/.ssh/id_ed25519): /c/Users/nerxt/.ssh/id_rsa
Created directory '/c/Users/nerxt/.ssh'.
Enter passphrase for "/c/Users/nerxt/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/nerxt/.ssh/id_rsa
Your public key has been saved in /c/Users/nerxt/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:o2IJdXztb2aoqM7m99lMUpBz/6mfW72x+v5ZD1er3LA correo@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|     .   o       |
|    . o = o      |
|   . . . = .     |
|  .     S o .    |
|   . . . o o . .+|
|    + . . o =.+o=|
|   o.... B +..==B|
|   +=o..+ o .EBO+|
+----[SHA256]-----+

nerxt@PC-NEREA MINGW64 ~
$
```

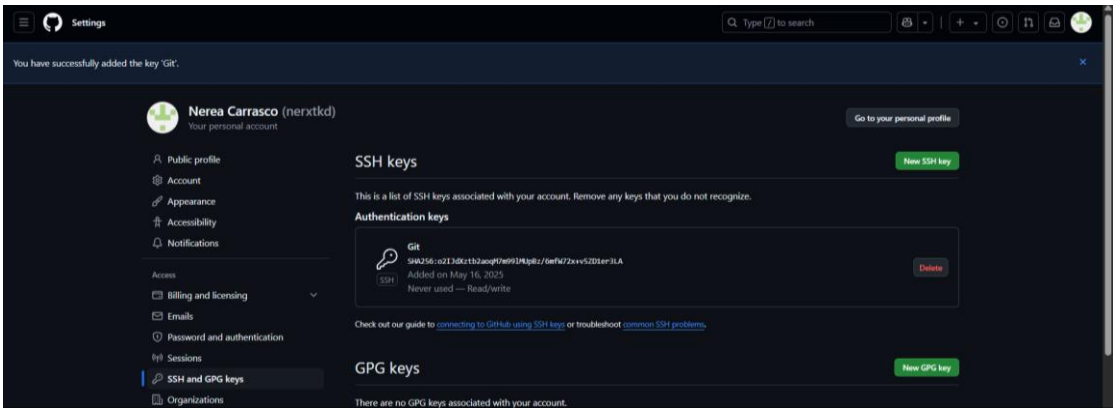
Luego, desde PowerShell con permisos de administrador, iniciamos el servicio. En mi caso, lo hice en 2 comandos porque solo en 1 me daba error.:

```
PS C:\WINDOWS\system32> Get-Service -Name ssh-agent | Set-Service -StartupType Manual
PS C:\WINDOWS\system32> Start-Service ssh-agent
PS C:\WINDOWS\system32>
```

Agregamos la clave privada en otra ventana de PowerShell sin permisos de administrador:



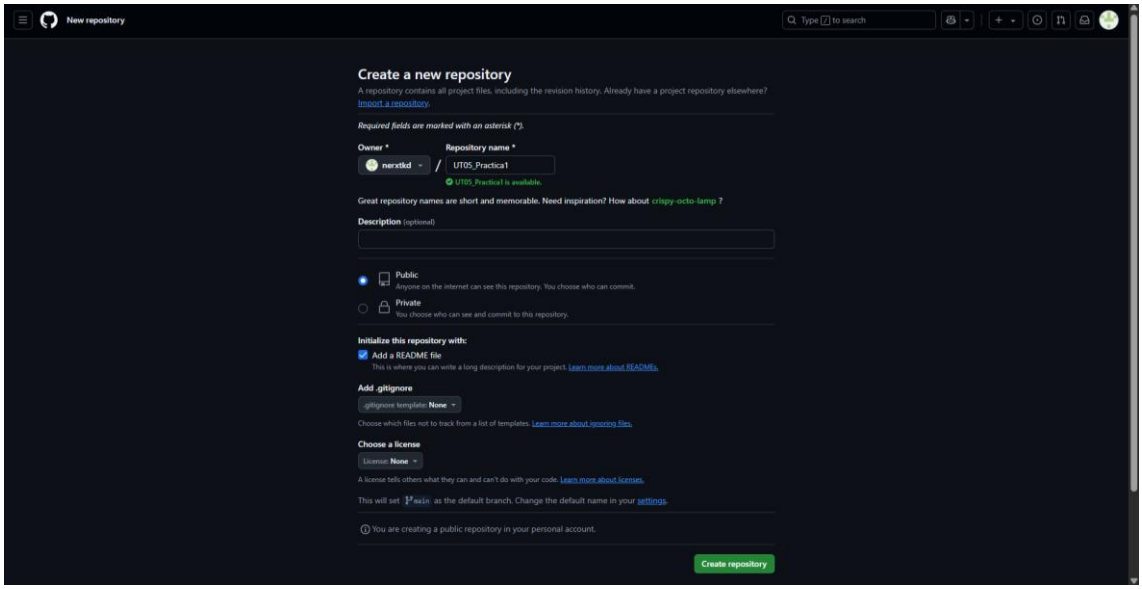
Añadimos la clave pública a nuestro perfil de GitHub, como se indica en el enunciado de la práctica y comprobamos que se haya guardado esta nueva configuración:



Para probar la conexión, desde Git escribimos ***ssh -T git@github*** dos veces, una para terminar de establecer la autenticación y conexión y otra para comprobarlo.

```
nerxt@PC-NEREA MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/c/Users/nerxt/.ssh/id_rsa':
Hi nerxtkd! You've successfully authenticated, but GitHub does not provide shell access.

nerxt@PC-NEREA MINGW64 ~
$ ssh -T git@github.com
Enter passphrase for key '/c/Users/nerxt/.ssh/id_rsa':
Hi nerxtkd! You've successfully authenticated, but GitHub does not provide shell access.

nerxt@PC-NEREA MINGW64 ~
$
```

Para crear el nuevo repositorio, desde *Home*, clicamos en *New Repository* y completamos los campos:



8. Actualizar el repositorio remoto de GitHub con lo que se tenga en el repositorio local.

Una vez tengamos creado el repositorio, ya solo quedará actualizarlo con lo que hemos creado en local en los anteriores ejercicios. Para ello, desde el repositorio local de Git, ejecutamos ***git remote add origin https://github.com/nerxtkd/UT05_Practica1.git***:

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git remote add origin https://github.com/nerxtkd/UT05_Practica1.git

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$
```

Como no ha salido ningún error, podemos seguir. Para subir todo de local al repositorio remoto, ejecutamos ***git push -u origin main***

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git push -u origin main
To https://github.com/nerxtkd/UT05_Practica1.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/nerxtkd/UT05_Practica1.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

En mi caso, me da error ya que en el repositorio remoto he añadido el contenido *README.md* y, en local, se creó todo vacío. Si, como en mi caso, existe un README.md en el repositorio remoto y no en local, se recomienda hacer un pull con rebase antes de hacer push. Para arreglarlo, ejecutamos el comando **git pull --rebase origin main** y luego, repetir el *push:*

```
nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git pull --rebase origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 863 bytes | 123.00 KiB/s, done.
From https://github.com/nerxtkd/UT05_Practica1
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
Successfully rebased and updated refs/heads/main.

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$ git push -u origin main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 32 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.43 KiB | 244.00 KiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nerxtkd/UT05_Practica1.git
   d2dc75d..20fb548  main -> main
branch 'main' set up to track 'origin/main'.

nerxt@PC-NEREA MINGW64 ~/Desktop/Ciberseguridad/Puesta en producción segura/completado/Tema 5/UT05_Práctica1_Nerea (main)
$
```

9. Realizar algún cambio en el repositorio remoto desde GitHub, y tras ello actualizar el repositorio local.

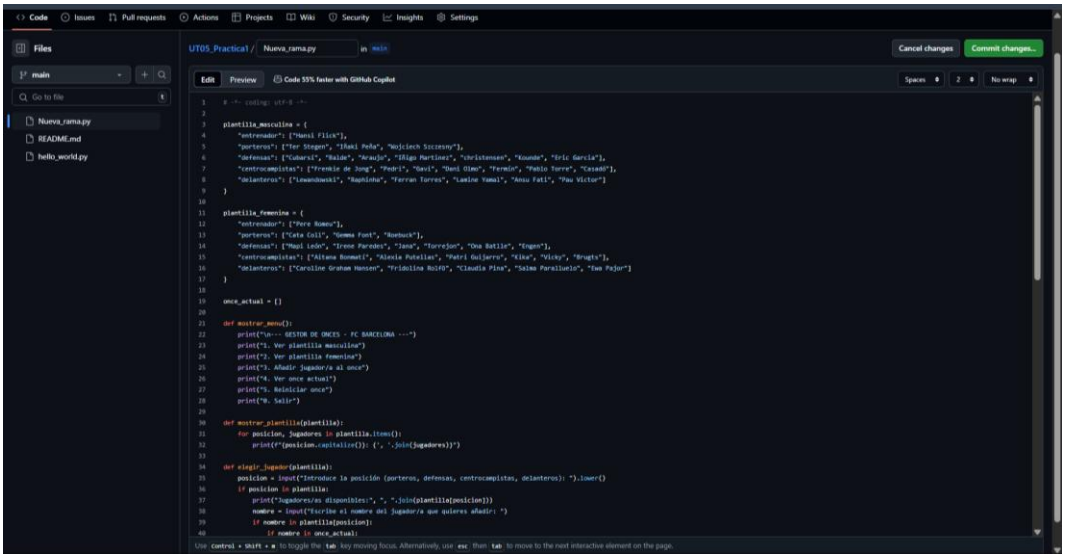
En mi caso, para las primeras tareas he seguido la documentación de la Actividad 2. Como para la realización de esta práctica se pedía un programa y no solo un *print* en Python, voy a aprovechar este ejercicio para eliminar el *print* del documento y meter el código.

Hay que aclarar que en este caso, según se cita en el enunciado de la práctica, *"Para cada acción se puede emplear un programa de ejemplo con Python el cuál puede ser generado con inteligencia artificial"*, este código fue generado por la IA ChatGPT y, posteriormente testeado y corregido por mí hasta obtener el resultado que quería.

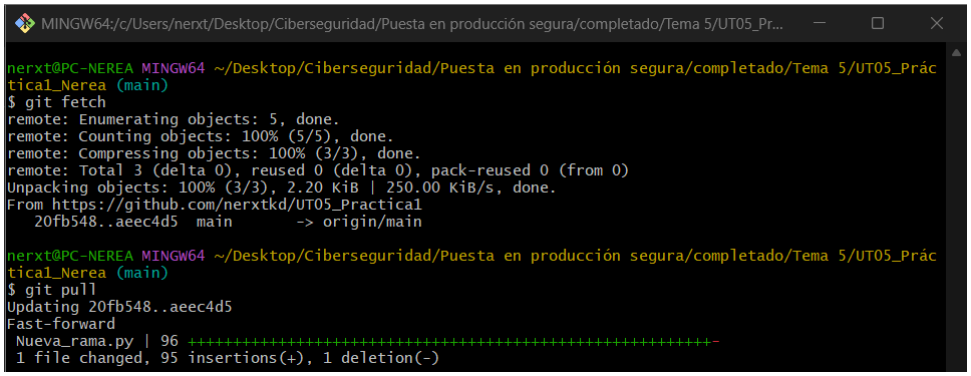
El programa va a ser una especie de juego del FC Barcelona, en el cual, puedes ver tanto la plantilla masculina como la femenina y crear tu once ideal. Además, va a basarse en un menú de 6 opciones:

- --- GESTOR DE ONCES - FC BARCELONA ---
  1. Ver plantilla masculina
  2. Ver plantilla femenina
  3. Añadir jugador/a al once
  4. Ver once actual
  5. Reiniciar once
  6. Salir

Una vez creado el código, debemos hacer los cambios. Para ello, vamos a GitHub y editamos el fichero que hemos creado en la segunda rama, en mi caso, llamado *nueva_rama2*:



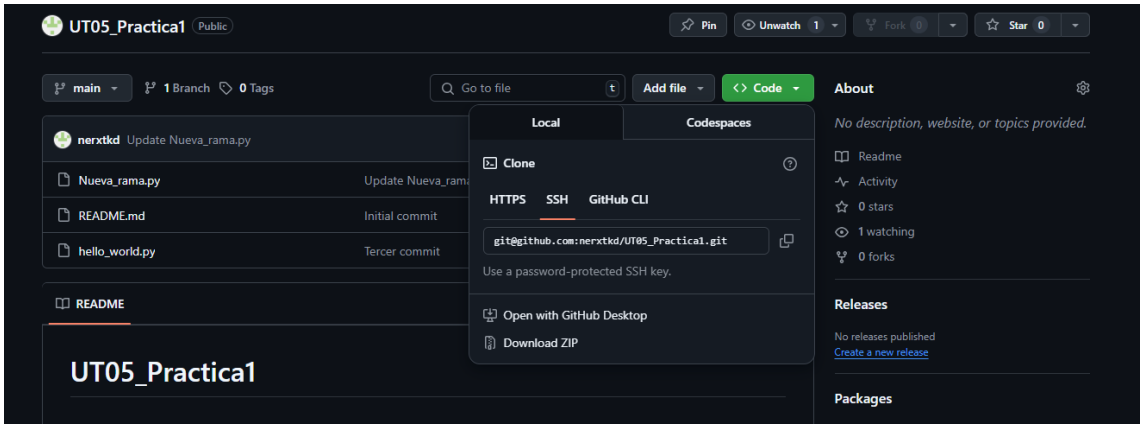
Guardamos los cambios y volvemos al Git Bash para actualizar el repositorio local con ***git fetch*** y ***git pull***:



10. Realizar un clonado del repositorio remoto.

Para acabar con la práctica, haremos un clonado. Para ello, necesitamos una nueva carpeta que crearé, en mi caso, en el mismo directorio que la otra en la que hemos trabajado durante toda la práctica.

Para poder clonar el repositorio, debemos obtener la dirección de la misma, la mía es *git@github.com:nerxtkd/UT05_Practica1.git* :



Una vez la tengamos, vamos al Git Bash de nuevo, nos movemos a la carpeta en la que vamos a clonar el repositorio y ejecutamos el comando que lo va a clonar, ***git clone git@github.com:nerxtkd/UT05_Practica1.git***