

Trabajo de GitHub



Francisco José Barrera Román

1º DAW

Índice

¿Qué es GIT?

Características de GIT

Diagrama de las áreas de Git

¿Qué es GitHub?

Cómo instalar git y creación de GitHub

- Instalación de GIT

- Creación de GitHub

Un ejemplo de creación y administración de un repositorio local

Uso de repositorio remoto

¿Qué es GIT?

GIT es un sistema de control de versiones. Permite a los desarrolladores gestionar, rastrear y controlar los cambios en el código fuente de un proyecto de software. Se utiliza mucho en el desarrollo de software colaborativo porque facilita el trabajo en equipo al permitir que varias personas trabajen simultáneamente en el proyecto sin que se sobrescriban los cambios realizados.

Características de GIT

Las características de GIT lo convierten en una herramienta fundamental en el desarrollo de software colaborativo.

1. **Distribución completa:** es un sistema de control de versiones distribuidos ya que cada usuario tiene una copia completa del historial de los cambios que sus compañeros del proyecto en el que estén trabajando. Permitiendo el trabajo offline sin que haya un servidor.
2. **Alto rendimiento:** está diseñado para ser rápido y eficiente optimizando el manejo de grandes proyectos y facilitando operaciones de control de versiones como commits, comparaciones de cambio y fusiones de rama.
3. **Integridad de los datos:** usa un sistema hashing (SHA-1) para identificar cada archivo y cambio dentro del proyecto. Asegura que la integridad de los datos se mantenga y no haya ninguna posibilidad de que la información se corrompa sin detectar el problema.
4. **Ramas y fusiones:** facilita la creación de ramas permitiendo a los desarrolladores trabajar en diferentes partes del proyecto de manera independiente. También facilita la manera de fusionar distintos cambios de diferentes ramas sin afectar a la rama principal.
5. **Control de versiones efectivo:** permite rastrear y deshacer cualquier cambio hecho en el proyecto, lo que significa que cualquier cambio que se haga queda registrado en el historial y se puede deshacer con cualquier versión previa.
6. **Soporte para trabajo colaborativo:** permite que múltiples personas trabajen en el mismo proyecto a la vez. Incluye soporte para tareas de colaboración como por ejemplo, pull request (solicitudes de cambios), revisiones de código y ramas específicas para cada desarrollador o tarea.
7. **Flexibilidad:** permite variedad de configuraciones que van desde el flujo de trabajo local hasta el uso de repositorios (GitHub, GitLab o Bitbucket). Lo que significa que se adapta perfectamente a proyectos pequeños, grandes, de código abierto o cerrados.
8. **Compresión de archivos:** utiliza algoritmos de compresión lo cual hace que se minimice el uso del espacio de almacenamiento al evitar duplicados de datos.

Diagrama de las áreas de Git

Lo primero será explicar las 3 áreas que existen.

1. Working Directory

- Es el área de trabajo donde se encuentran los archivos en los que el desarrollador está trabajando.
- Se pueden ver y editar los archivos de manera directa.
- Los archivos pueden tener tres estados diferentes los cuales son: sin seguimiento, modificados o sin cambios.

2. Staging Area

- Conocida como índice o caché, aquí es donde se colocan los cambios que están listos para el commit.
- Se usa el comando "git add".
- Es donde se "preparan" los archivos lo que permite seleccionar qué cambios específicos se desean para el commit y cuáles se dejan para más adelante.

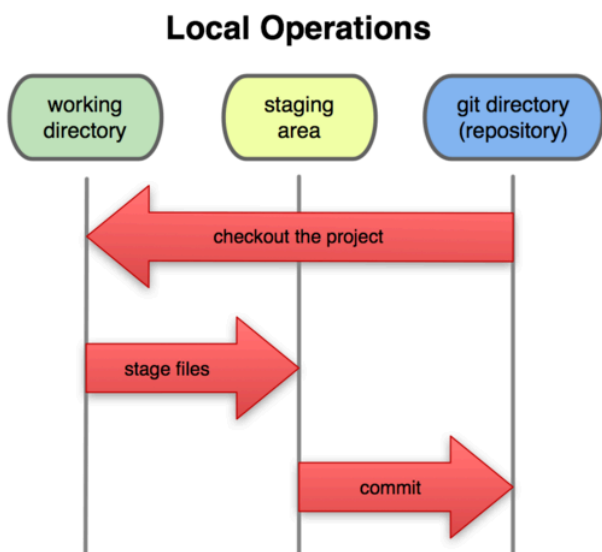
3. Repository

- Es la base de datos interna donde GIT almacena para siempre todos los cambios confirmados en el historial del proyecto.
- Para mover los archivos desde el área de preparación hasta el repositorio se usa el comando "git commit".
- En este área cada cambio está identificado por un hash único y se todo se guarda en el historial de versiones del proyecto.

Aquí tenemos el diagrama resumido:

Working Directory → `git add` → **Staging Area** → `git commit` → **Repositorio Local**

Además aquí tenemos una foto:



¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código fuente que utiliza GIT como un sistema de control de versiones y ofrece un espacio en el que múltiples desarrolladores pueden colaborar a la vez y también ofrece equipos de software. Es un espacio en la nube en la que los usuarios pueden almacenar, gestionar y compartir proyectos de software, también pueden colaborar en su desarrollo desde cualquier parte del mundo.

Sus principales funciones son:

- Repositorios remotos.
- Control de versiones.
- Colaboración y revisión de código.
- Documentación y wiki.
- Integración con CI/CD.
- Issues y tableros de proyecto.
- GitHub pages.

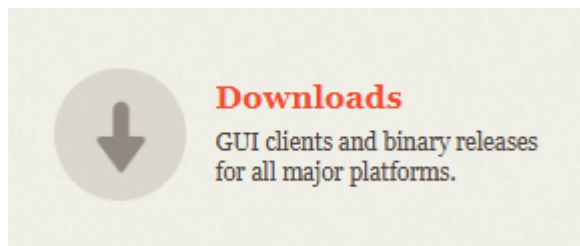
Cómo instalar git y creación de GitHub

Instalación de GIT

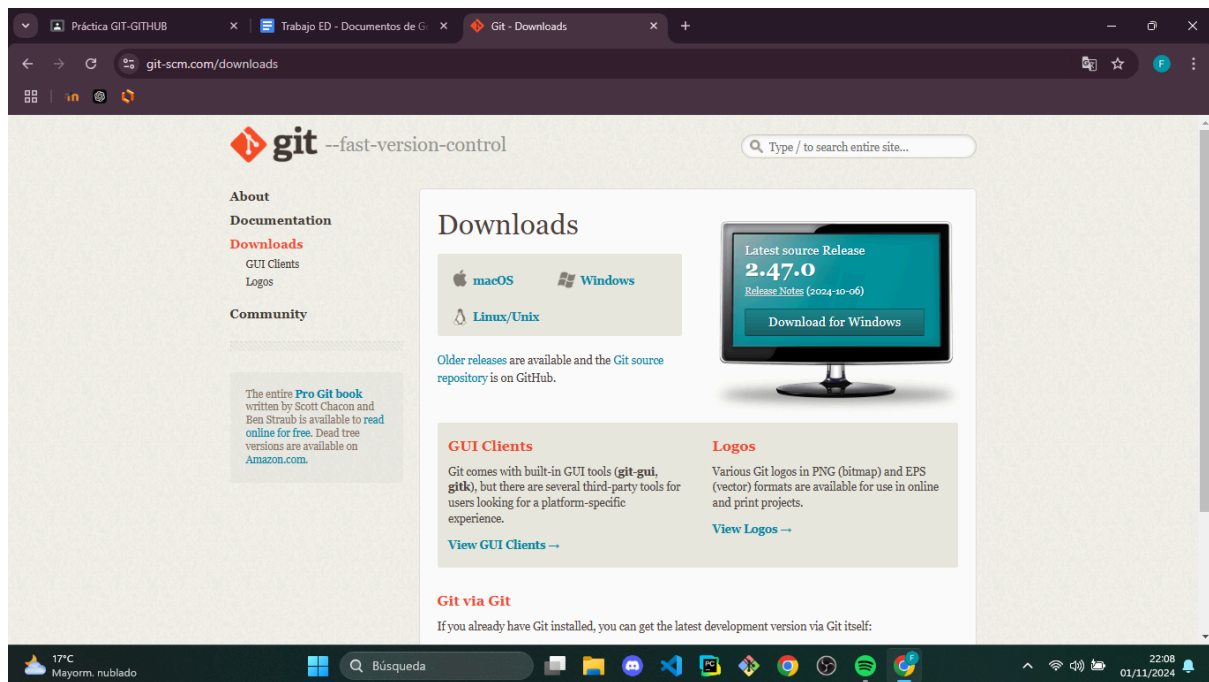
Para instalar GIT lo primero que tenemos que hacer es dirigirnos a la página oficial, la cual es <https://git-scm.com/> , nos saldrá este menú.



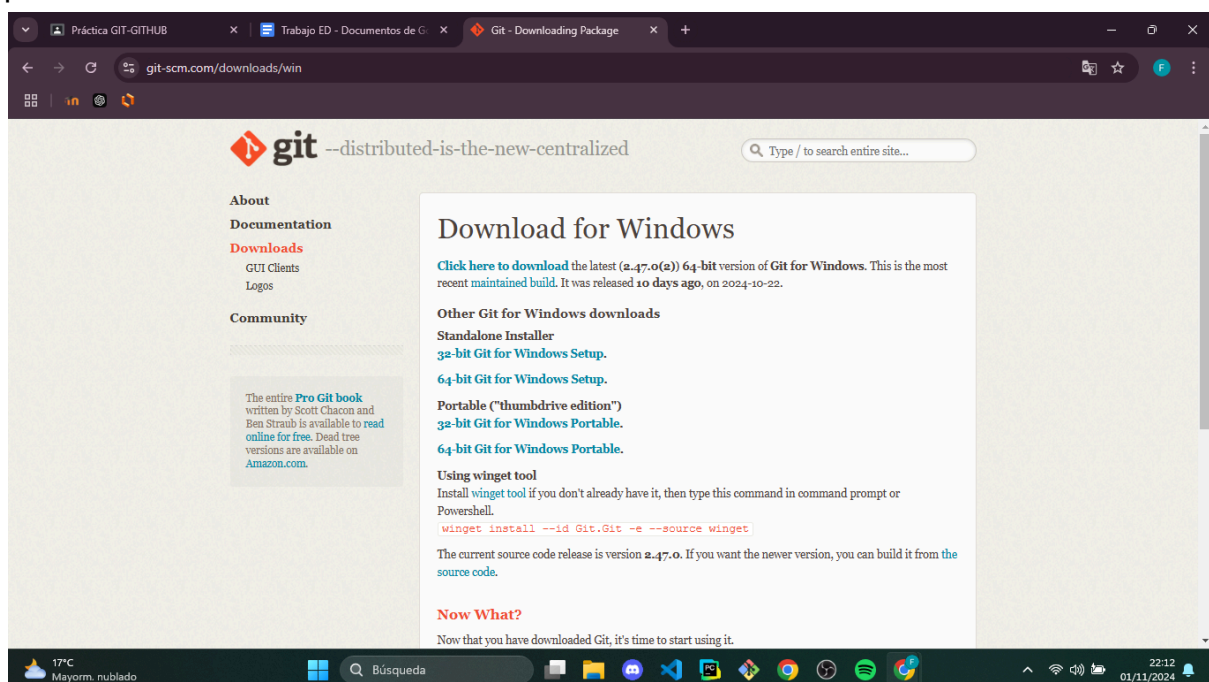
Luego tendremos que pinchar en Downloads.



Una vez hayamos pinchado nos saldrá esta página.



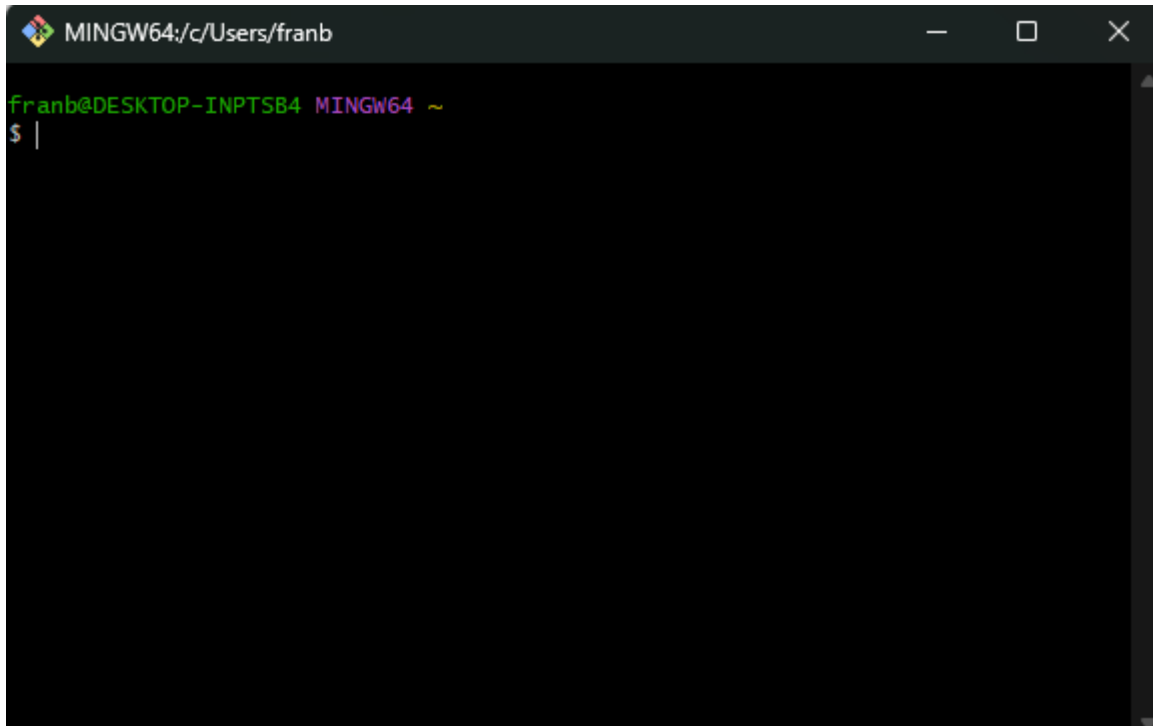
Luego pinchamos en el sistema operativo que tengamos en nuestro equipo para instalarlo, en este caso yo le he dado a Windows ya que es mi SO. Nos saldrá esta pantalla.



Lo siguiente que tendremos que hacer será pinchar en los bits que tenga nuestro SO para que la instalación sea correcta.

Ahora lo que tendremos que hacer será seguir las instrucciones en pantalla y utilizar las opciones predeterminadas, a menos que tengas alguna preferencia específica.

Luego abriremos Git Bash. Nos tiene que salir algo así.



```
MINGW64:/c/Users/franb
franb@DESKTOP-INPTS84 MINGW64 ~
$ |
```

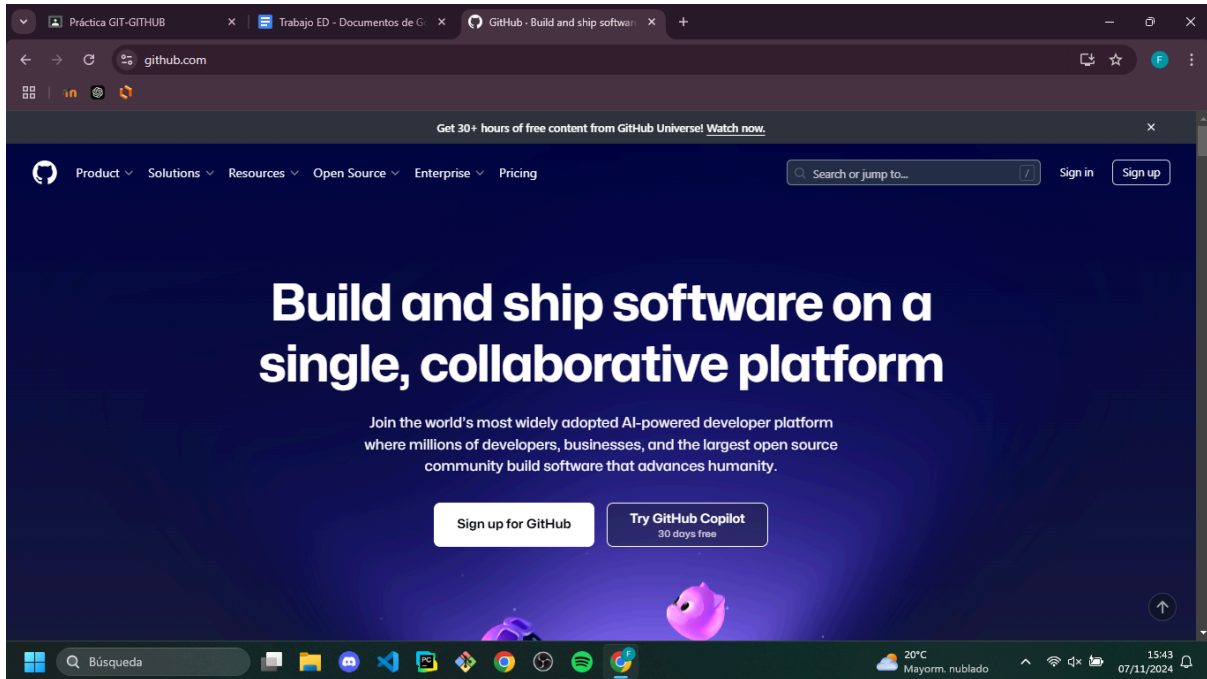
Y por último escribimos el comando “git --version” para verificar que todo se ha instalado correctamente y también lo pudimos utilizar para ver la versión que se ha instalado.



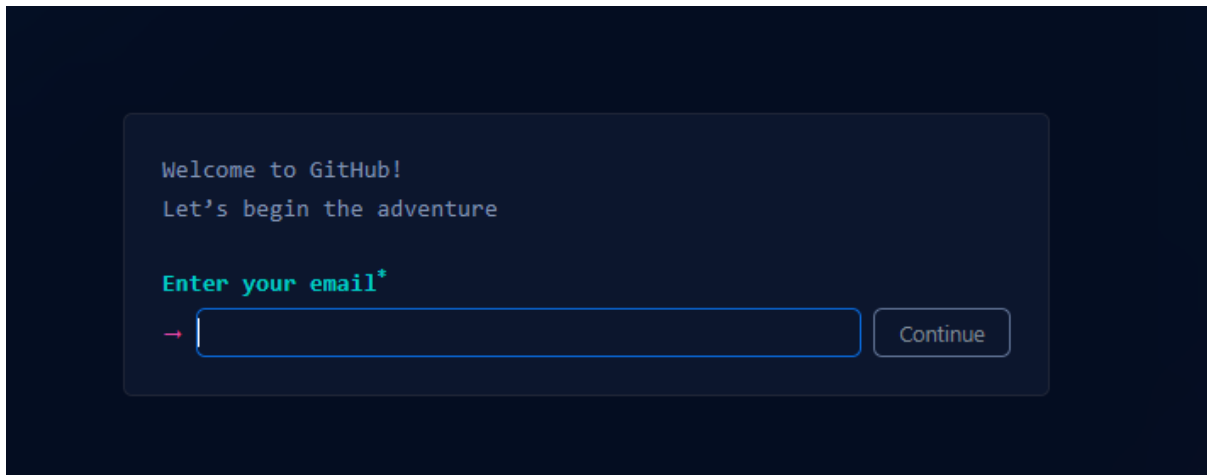
```
MINGW64:/c/Users/franb
franb@DESKTOP-INPTS84 MINGW64 ~
$ git --version
git version 2.45.1.windows.1
```

Creación de GitHub

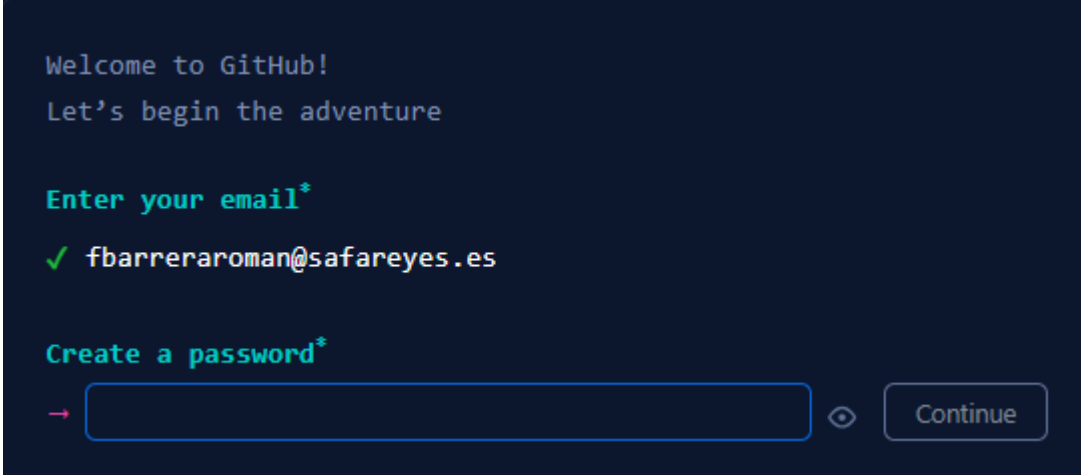
Lo primero que vamos a hacer es crearnos una cuenta en GitHub, para eso tenemos que entrar en la página web de GitHub que es este enlace de aquí “<https://github.com/>”. Nos saldrá esta página:



Ahora tendremos que pinchar en sign up que está situado arriba a la derecha de la página. Una vez pinchemos se vería así:



Ahora pondremos nuestro email y le damos a continue para avanzar. Una vez le damos a continue nos saldría esto:




Welcome to GitHub!
Let's begin the adventure

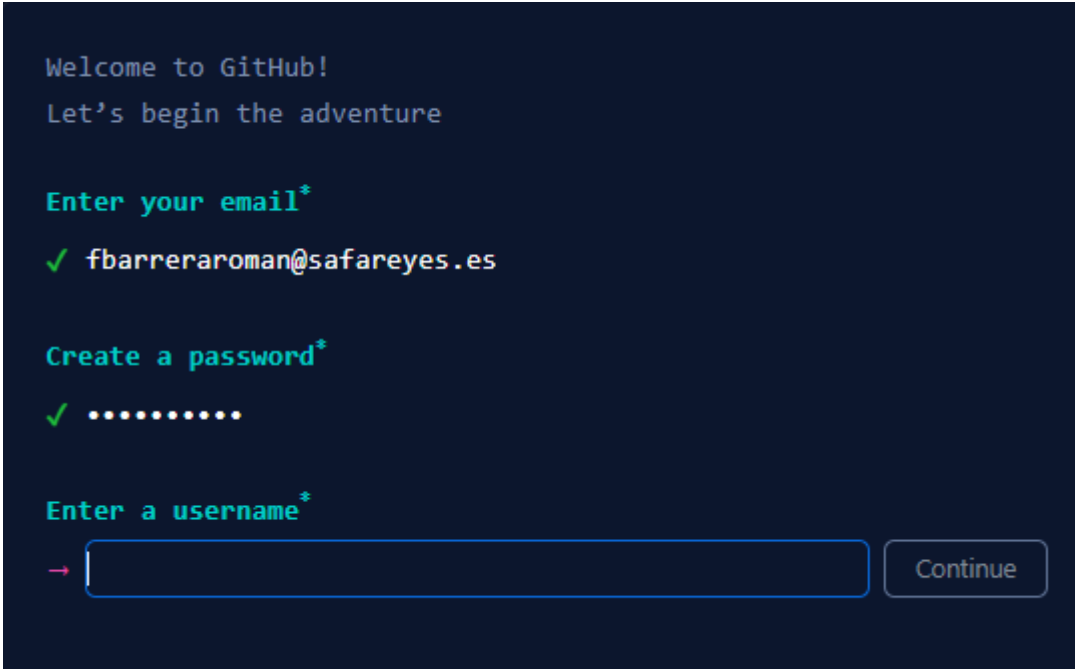
Enter your email*

✓ fbarreraroman@safareyes.es

Create a password*

→ 

Ahora tendremos que crear una contraseña y le volvemos a dar a continue para avanzar. Una vez le damos a continue saldría esto:



Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ fbarreraroman@safareyes.es

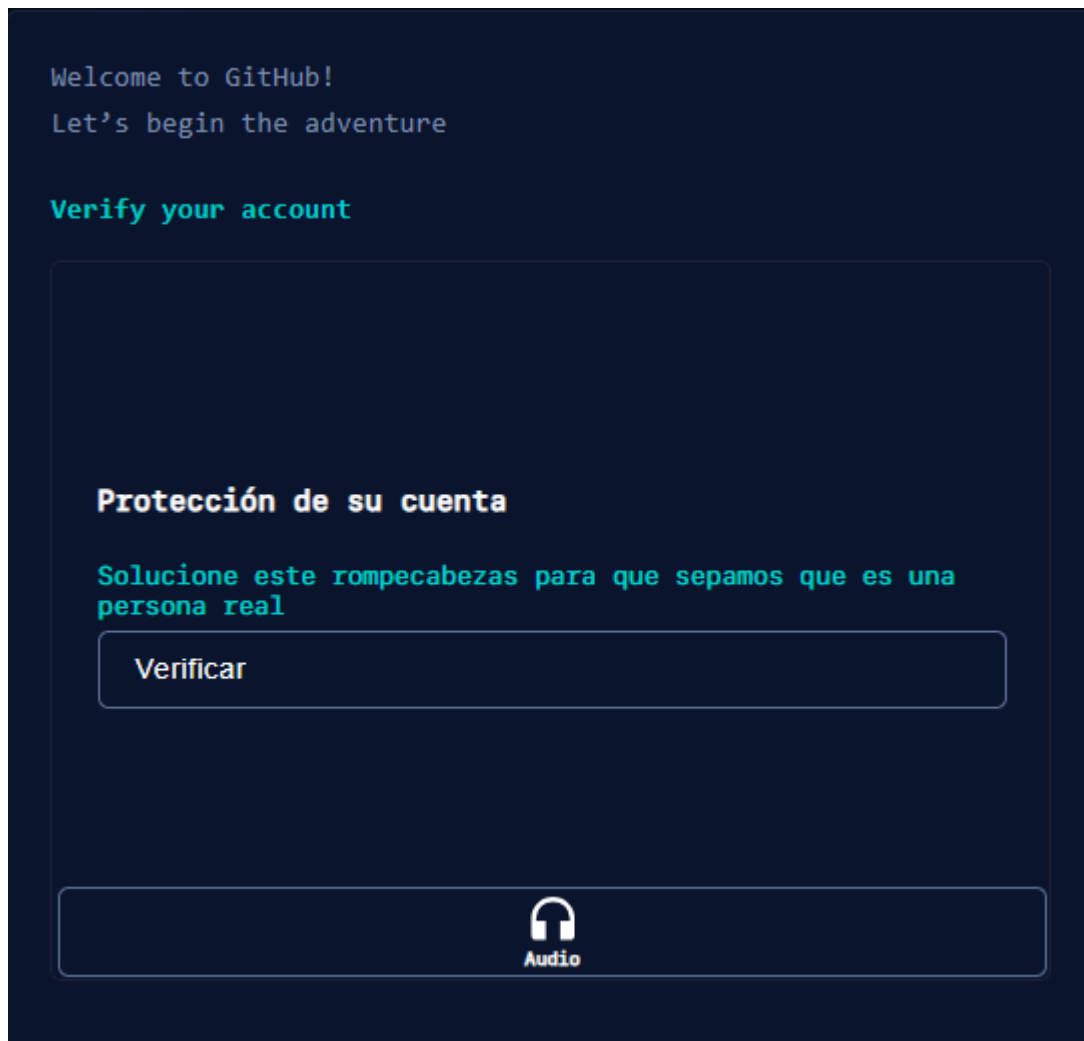
Create a password*

✓ ••••••••

Enter a username*

→

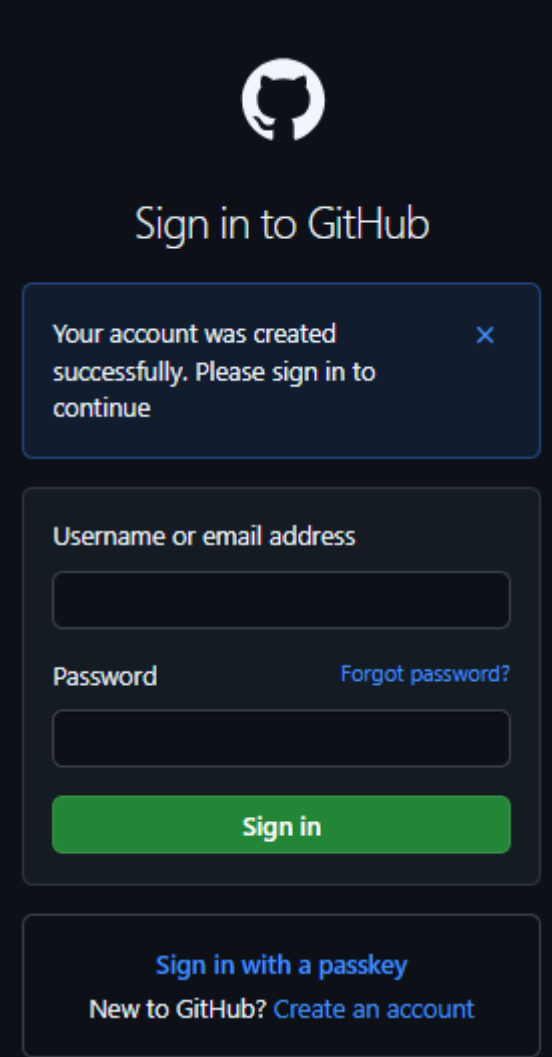
Ahora tendremos que ponernos un nombre de usuario, una vez lo pongamos le volvemos a dar a continue para seguir avanzando en la creación de la cuenta. Una vez le demos saldría esto:



Ahora tendremos que darle a verificar para verificar que somos una persona real. Cuando lo resolvamos nos saldrá esto:



Ahora nos habrán enviado un correo al correo que hemos puesto previamente en la creación de la cuenta, una vez lo pongamos nos saldrá esto:



The image shows the GitHub sign-in interface on a dark background. At the top is the GitHub Octocat logo. Below it is the heading "Sign in to GitHub". A success message box states: "Your account was created successfully. Please sign in to continue", with a close button (X) on the right. The main form contains a label "Username or email address" above a text input field. Below that is the "Password" label with a "Forgot password?" link to its right, followed by another text input field. A green "Sign in" button is positioned below the password field. At the bottom, there is a section with the link "Sign in with a passkey" and the text "New to GitHub? Create an account".

Sign in to GitHub

Your account was created successfully. Please sign in to continue

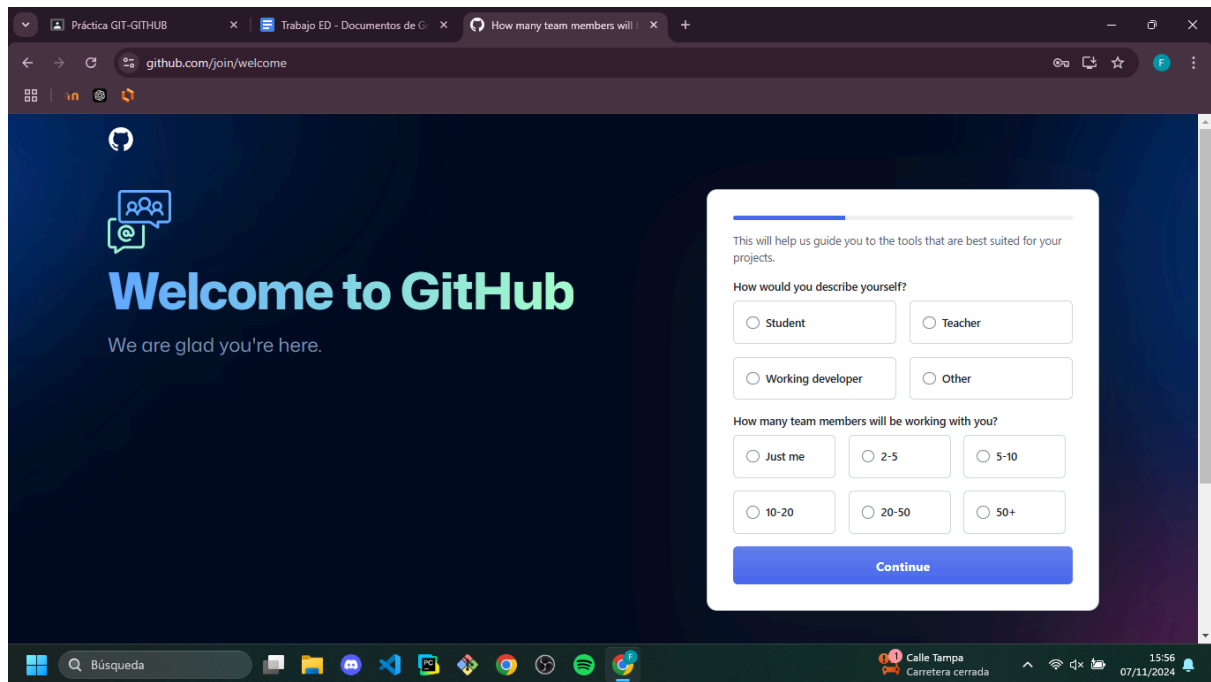
Username or email address

Password [Forgot password?](#)

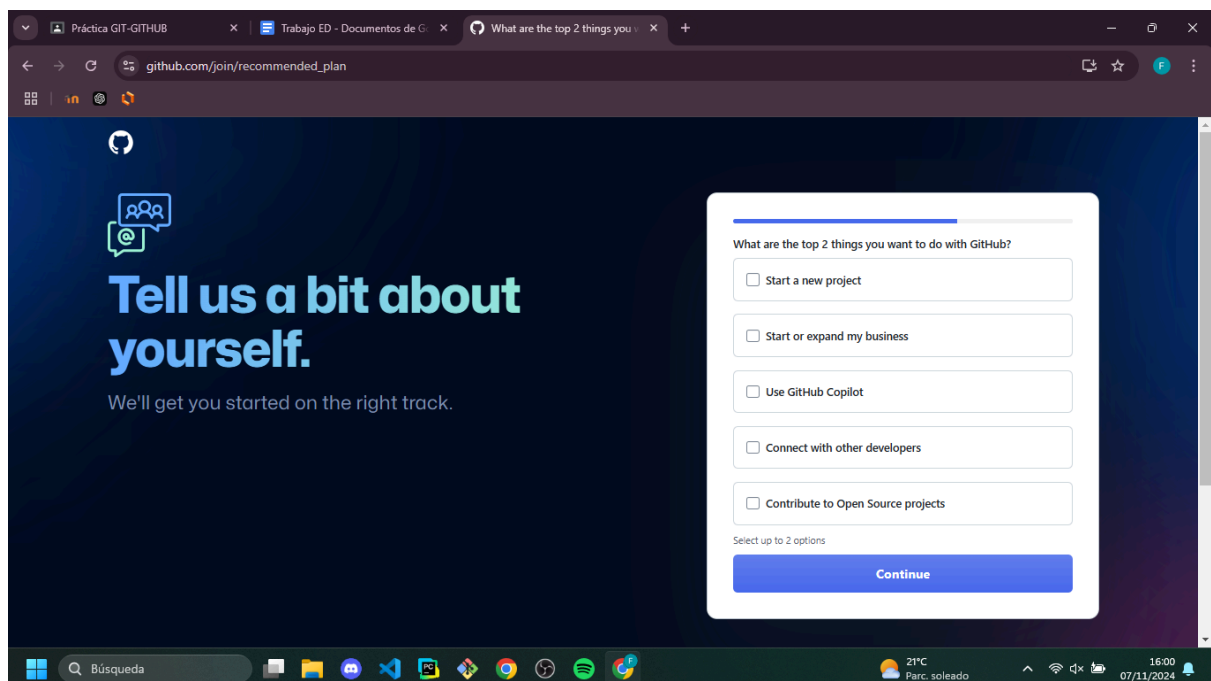
Sign in

[Sign in with a passkey](#)
New to GitHub? [Create an account](#)

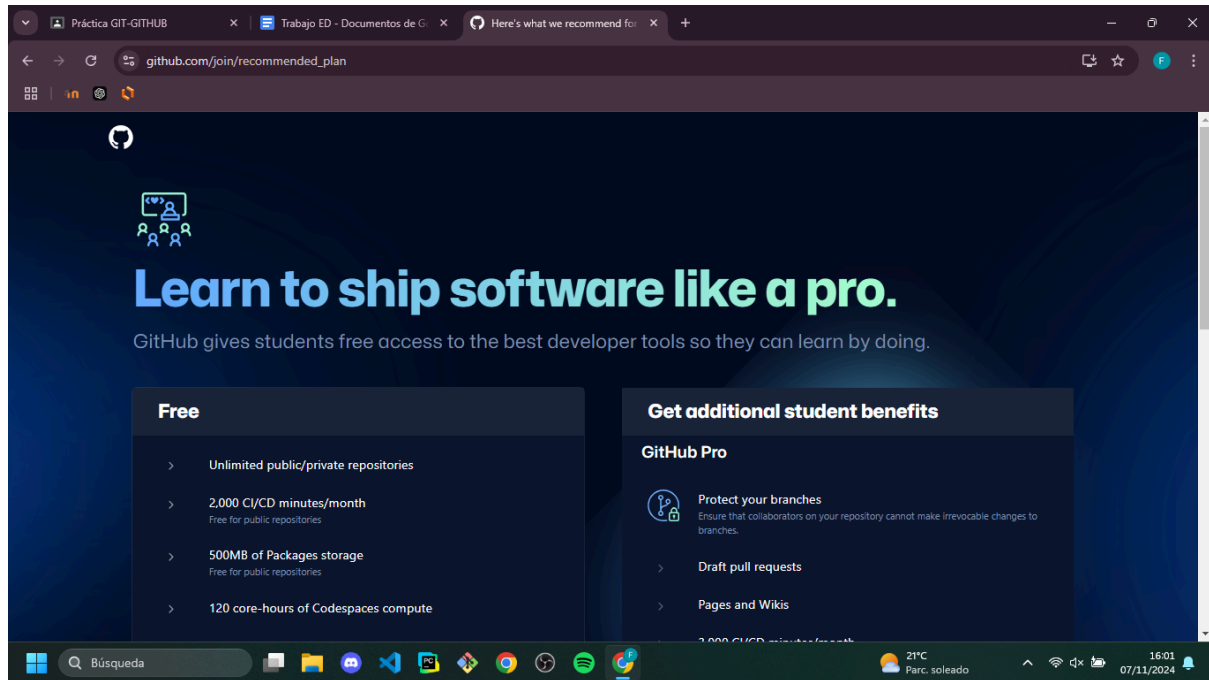
Ahora tendremos que poner primero o el nombre de usuario que hemos puesto en la creación de la cuenta o el correo que también hemos puesto en la creación de la cuenta y después pondremos la contraseña y le daremos a sign in. Una vez dentro nos saldrá esto:



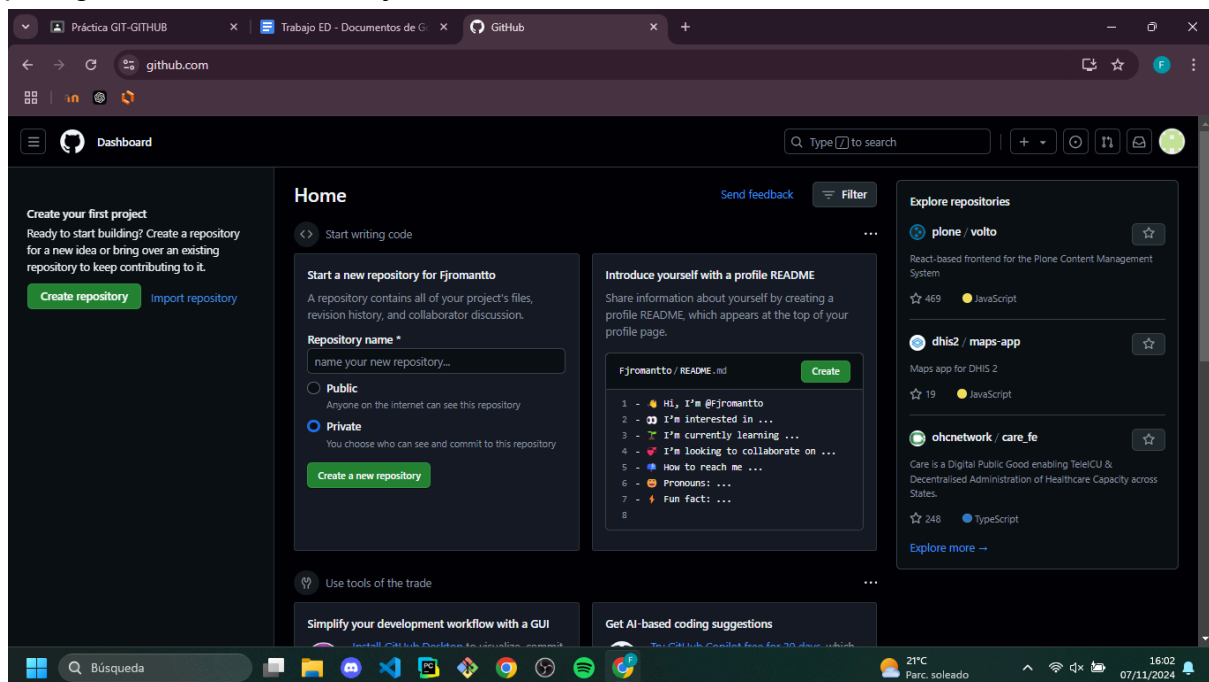
Ahora tenemos que contestar las diferentes preguntas que nos hace y luego darle a continue para seguir avanzando, una vez que respondamos las preguntas nos saldrá esto:



Ahora tenemos que responder esta pregunta que nos hace y una vez que la respondamos le volvemos a dar a continue, una vez le demos nos saldrá esto:



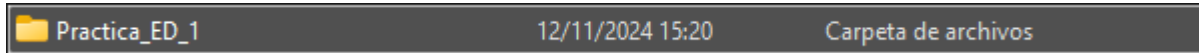
Ahora tenemos que elegir el plan que queremos, yo en mi caso, voy a seguir con el plan gratuito, una vez lo elijamos nos saldrá esto:



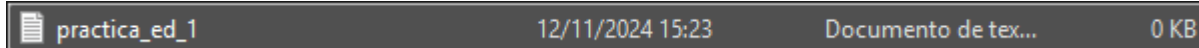
Y ya estaría, ya tendríamos la cuenta de GitHub creada.

Un ejemplo de creación y administración de un repositorio local

Lo primero que tendremos que hacer será crear una carpeta en el disco raíz a la que llamaremos como queramos.



Ahora dentro de la carpeta crearemos un archivo txt para empezar a hacer el repositorio.



Ahora abrimos git bash.



Y una vez estemos dentro del git bash lo primero que tenemos que poner es “git init” para que se inicie el repositorio.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1
$ git init
Initialized empty Git repository in C:/Practica_ED_1/.git/
```

Lo siguiente que tenemos que poner será esto “git config --global user.name "NOMBRE” para crear un nombre de usuario a nuestro repositorio.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git config --global user.name "fran"
```

Y lo siguiente que tenemos que hacer será poner esto “git config --global user.email "CORREO” para crear una cuenta de correo a nuestro repositorio.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git config --global user.email "fbarreraroman@safareyes.es"
```

Ahora lo que tendremos que hacer será ver el estado de nuestro proyecto para eso utilizaremos el comando “git status”.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    practica_ed_1.txt

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

Ahora tenemos que agregar el archivo que se ve ahí en rojo que es el archivo que hemos creado previamente al principio para eso utilizaremos "git add --all".

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git add --all
```

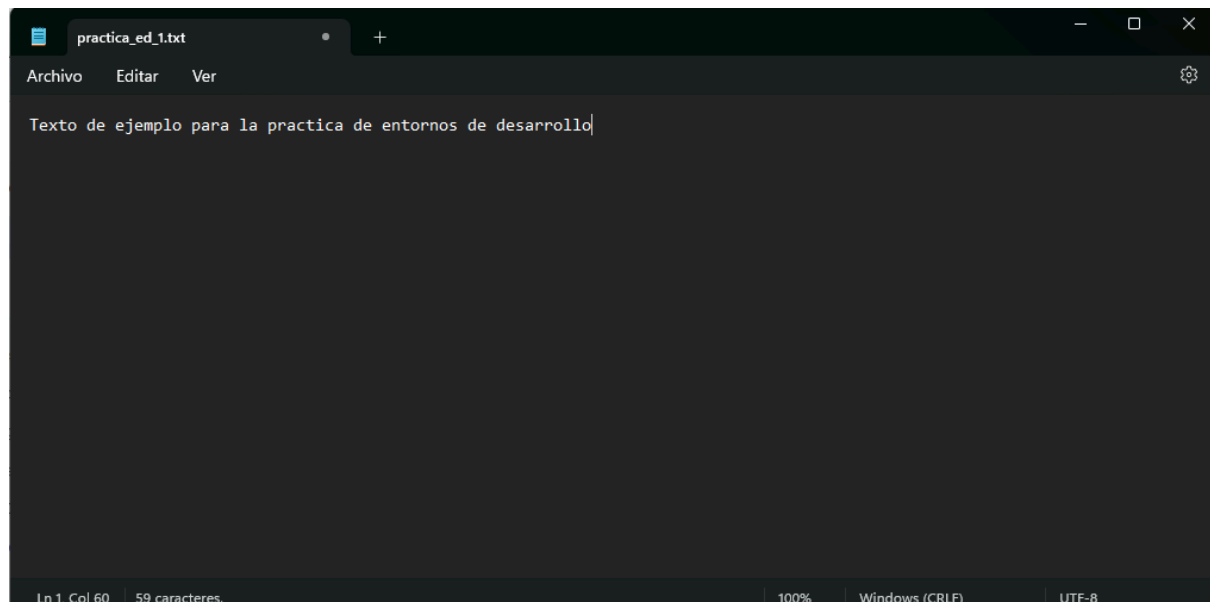
Ahora si volvemos a hacer un git status nos saldrá que se ha añadido correctamente el archivo de texto.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   practica_ed_1.txt
```

Ahora vamos a abrir el documento txt que hemos creado y le vamos a escribir algo.



Ahora nos vamos al git bash otra vez y vamos a hacerle un commit para que se guarde dentro de nuestro repositorio.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Practica_ED_1 (master)
$ git commit -m "practica_ed_1.txt"
[master (root-commit) 5794e5d] practica_ed_1.txt
1 file changed, 1 insertion(+)
create mode 100644 practica_ed_1.txt
```

Ahora vamos a poner el comando “git log” que sirve para ver el rastro de commits que hemos ido haciendo a lo largo del trabajo.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (master)
$ git log
commit 5794e5d8ac512a214c6a1759de12392e20625665 (HEAD -> master)
Author: fran <fbarreraroman@safareyes.es>
Date: Tue Nov 12 15:54:35 2024 +0100

    practica_ed_1.txt
```

Ahora lo que vamos a hacer será crear una rama en el repositorio para eso utilizaremos el comando “git branch nombreRama” que para lo que sirve es como una especie de punto de guardado.

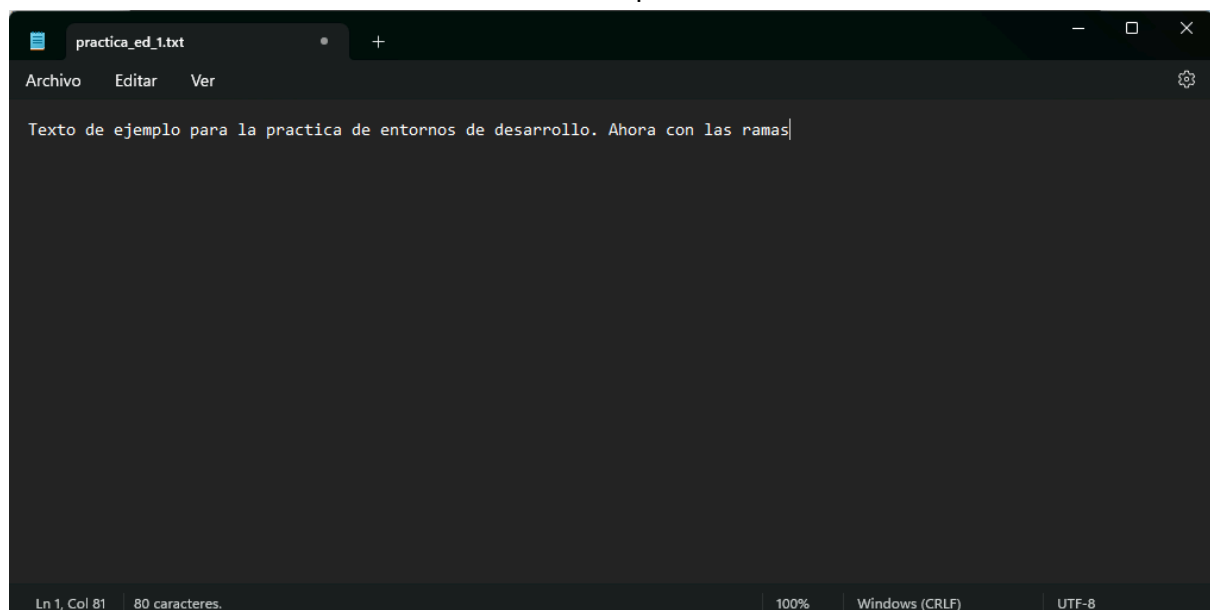
```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (master)
$ git branch rama_edicion_ED
```

Ahora vamos a entrar dentro de la rama para eso tendremos que utilizar el comando “git checkout”.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (master)
$ git checkout rama_edicion_ED
Switched to branch 'rama_edicion_ED'

franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (rama_edicion_ED)
$
```

Ahora volvemos a editar el documento de texto que hemos creado al inicio.



Ahora le hacemos un nuevo commit al archivo de texto.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (rama_edicion_ED)
$ git commit -m "practica_ed_1.txt"
[rama_edicion_ED 18e2d3d] practica_ed_1.txt
1 file changed, 1 insertion(+), 1 deletion(-)
```

Y volvemos a la rama master que es la rama principal para posteriormente unir esas dos ramas.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (rama_edicion_ED)
$ git checkout master
Switched to branch 'master'

franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (master)
$
```

Y ahora vamos a unir esas dos ramas para eso utilizaremos el comando “git merge”.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Practica_ED_1 (master)
$ git merge rama_edicion_ED
Updating 5794e5d..18e2d3d
Fast-forward
 practica_ed_1.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

Uso de repositorio remoto

Lo primero que tendremos que hacer será hacer un git clone y la url del repositorio y automáticamente nos lo descargará en el disco local.

Se nos creará esta carpeta.

 Bomba-nuclear-DAW1	13/11/2024 15:06	Carpeta de archivos
--	------------------	---------------------

Lo siguiente que vamos a hacer será darle click derecho sobre la carpeta y pulsar en “open git bash here” y crearemos una rama nueva.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Bomba-nuclear-DAW1 (main)
$ git branch rama_fran
```

Luego nos cambiaremos a esa rama.

```
franb@DESKTOP-INPTSB4 MINGW64 /c/Bomba-nuclear-DAW1 (main)
$ git checkout rama_fran
Switched to branch 'rama_fran'
```

Ahora añadiremos a la rama todos los archivos que hayan dentro de la carpeta.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Bomba-nuclear-DAW1 (rama_fran)
$ git add .
```

Haremos un git status para comprobar que todo está dentro de la rama.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Bomba-nuclear-DAW1 (rama_fran)
$ git status
On branch rama_fran
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Lista_Barbacoa.py
```

Ahora abriremos el archivo a modificar y modificaremos algo en mi caso ha sido la última línea en la que pone en la clave del diccionario "ordenador".

```
lista_barbacoa = {
    "comida":["salchicha", "chuletas","salchichón"],
    "bebida":["cocacola", "fanta naranja","Nestea"],
    "cubata":["whisky_solo", "Ron"],
    "carbon":["encendedor","gas"],
    "herramientas":["cuchillo"]
    "Juegos":["Uno"],
    "Tonto el que lo lea": ['bomba']
    |"ordenador": ["ram", "procesador"]
}
```

Una vez modificado haremos un commit para guardar los cambios que hemos hecho.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Bomba-nuclear-DAW1 (rama_fran)
$ git commit -m "archivo editado para el trabajo de ED por Fran"
[rama_fran e04b164] archivo editado para el trabajo de ED por Fran
1 file changed, 1 insertion(+)
```

Ahora lo que vamos a hacer es subir la rama con los que hemos editado y todo para eso utilizaremos git push.

```
franb@DESKTOP-INPTS84 MINGW64 /c/Bomba-nuclear-DAW1 (rama_fran)
$ git push origin rama_fran
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 375 bytes | 375.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'rama_fran' on GitHub by visiting:
remote:   https://github.com/Arthen-code0/Bomba-nuclear-DAW1/pull/new/rama_fr
an
remote:
To https://github.com/Arthen-code0/Bomba-nuclear-DAW1
 * [new branch]      rama_fran -> rama_fran
```