

# Repositorios

## Git

## GitHub

Sprint 1 Lab 2



Felipe Izquierdo Romero

# Índice

<u>1</u>	<u>Descripción del problema</u>
<u>2</u>	<u>Desarrollo</u>
2.1	Creación de una cuenta en GitHub
2.2	Iniciar un repositorio remoto en GitHub
2.3	Iniciar repositorio de manera local con Git y clonar repositorio remoto
2.4	Crear una rama independiente al tronco
2.5	Realizar primer commit
2.6	Eliminar commit
2.7	Eliminar la rama en local y en el repositorio remoto
2.8	Evitar la subida de algunos ficheros al repositorio remoto
2.9	Instalar GitKraken y asociarlo al repositorio
<u>3</u>	<u>Link del repositorio remoto</u>

# 1 Descripción del problema

El siguiente laboratorio de repositorios propone una serie de ítems a realizar para crear un repositorio y hacer algunas acciones a través de la consola de comandos de Git.

Los ítems son los siguientes:

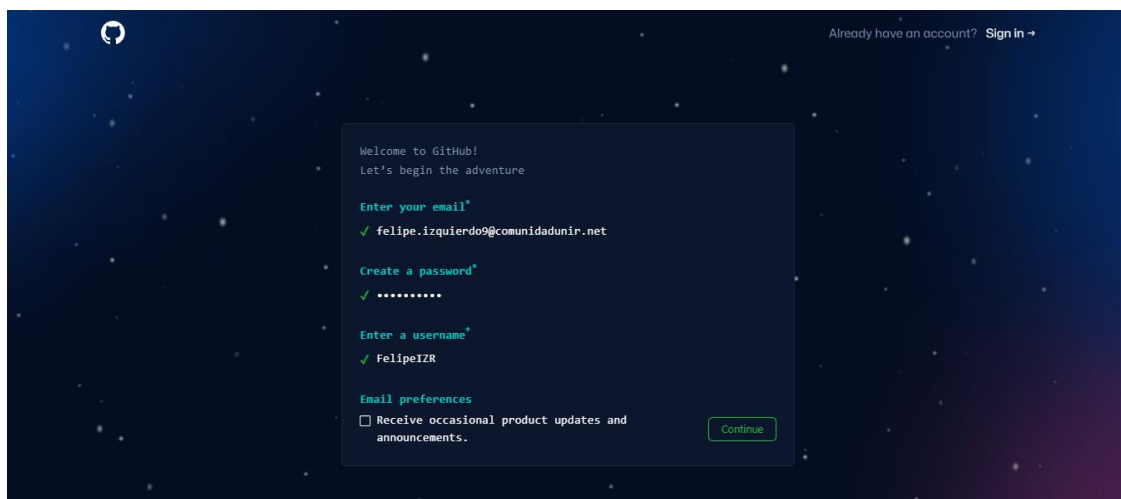
- Creación de una cuenta en GitHub
- Iniciar un repositorio remoto en GitHub
- Iniciar repositorio de manera local con Git y clonar repositorio remoto
- Crear una rama independiente al tronco
- Realizar primer commit
- Eliminar commit
- Eliminar la rama en local y en el repositorio remoto
- Evitar la subida de algunos ficheros al repositorio remoto
- Instalar GitKraken y asociarlo al repositorio

## 2 Desarrollo

### 2.1 Creación de una cuenta en GitHub

Para comenzar se creará la cuenta en el siguiente link:

[https://github.com/signup?ref\\_cta=Sign+up&ref\\_loc=header+logged+out&ref\\_page=%2F&source=header-home](https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home)



*Ilustración 1 Registro de GitHub*

Tras realizar el registro y pasar la verificación de correo electrónico ya estaría la cuenta creada con éxito.

### 2.2 Iniciar un repositorio remoto en GitHub

Para crear el primer repositorio es fácil simplemente poniendo el nombre y eligiendo la privacidad del repositorio se creará en segundos.

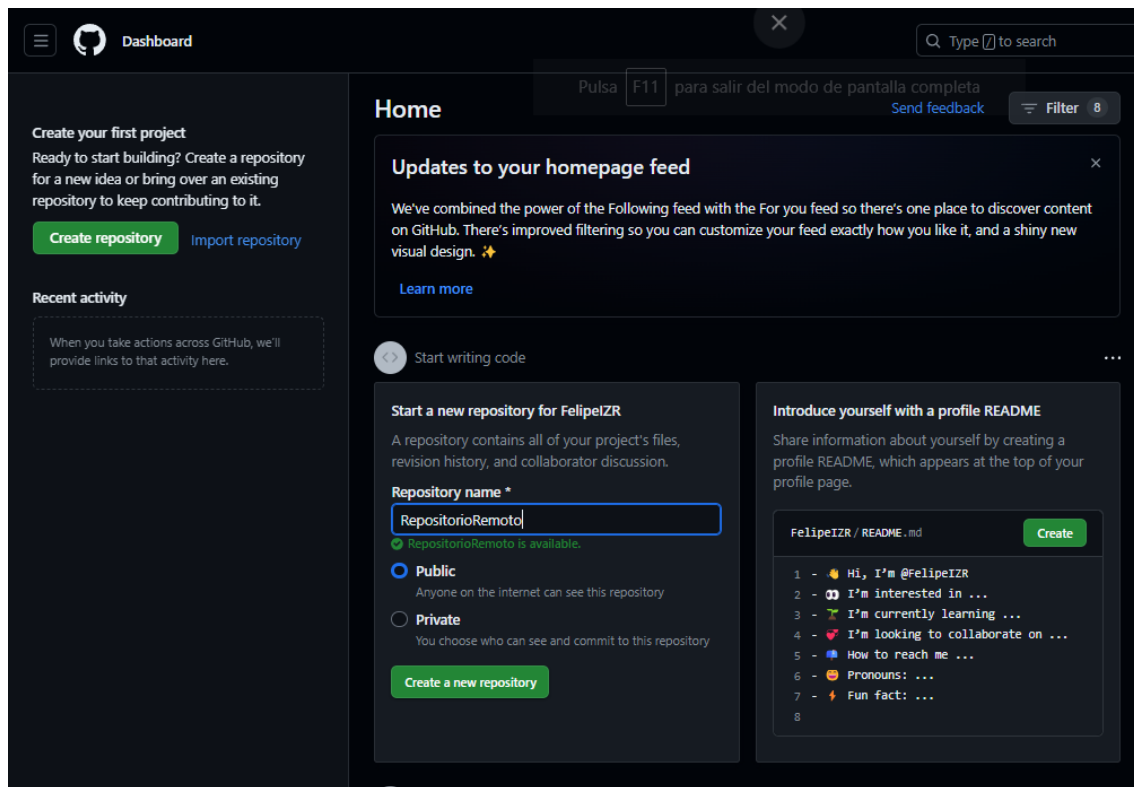


Ilustración 2 Creación de repositorio remoto

Este sería nuestro repositorio remoto vacío como se observa en la Ilustración 4

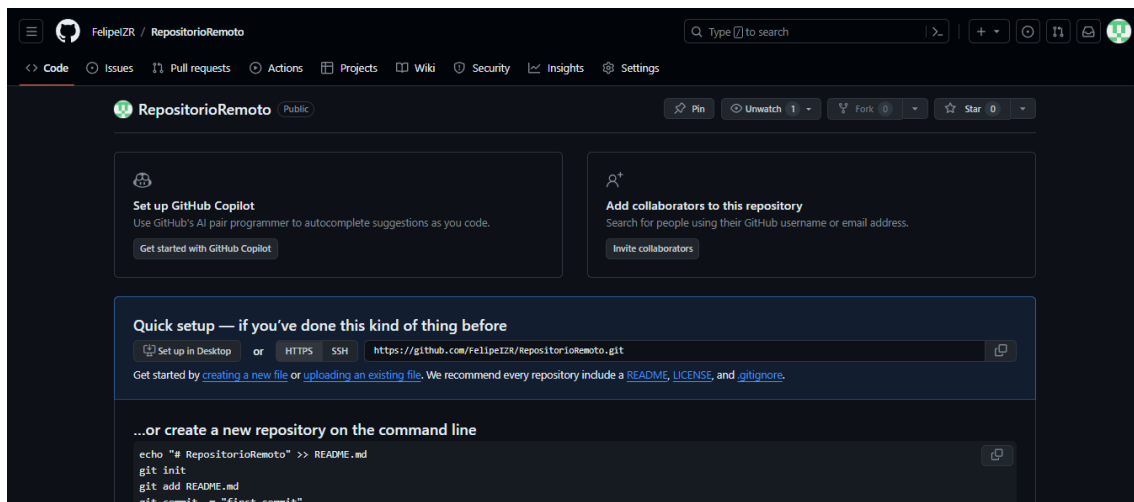


Ilustración 3 Repositorio remoto

## 2.3 Iniciar repositorio de manera local con Git y clonar repositorio

A continuación, se habrá que iniciar el programa Git y elegir el lugar donde crear el repositorio con el comando `cd <ruta_del_directorio>` e iniciar el repositorio con el comando `git init`.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~
$ cd "C:\Users\felix\SprintsBootcamp\Sprint_1\Lab_2\RepositorioEjercicio"

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio
$ git init
Initialized empty Git repository in C:/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio/.git/

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (master)
$
```

Ilustración 4 Creación de repositorio local

Para clonar el repositorio remoto se deberá usar `git clone <link_del_repositorio_remoto>` en el directorio donde se quiera clonar dicho repositorio.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (RamaIndependiente)
$ cd "C:\Users\felix\SprintsBootcamp\Sprint_1\Lab_2\RepositorioEjercicioRemoto"

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicioRemoto
$ git clone https://github.com/FelipeIZR/RepositorioRemoto
Cloning into 'RepositorioRemoto'...
warning: You appear to have cloned an empty repository.
```

Ilustración 5 Clonación de repositorio remoto

## 2.4 Crear una rama independiente al tronco

Para crear la rama con el comando `git checkout -b <nombre_nueva_rama>`

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (master)
$ git checkout -b RamaIndependiente
Switched to a new branch 'RamaIndependiente'

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (RamaIndependiente)
$
```

Ilustración 6 Creación de la rama independiente

## 2.5 Realizar primer commit

Para realizar un commit primero se creará un archivo de prueba con el comando `echo <texto> > <nombre_del_fichero.txt>` se añadirá con el comando `git add <archivo_que_añadir>` y finalmente `git commit -m <"comentario_del_commit">`.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ echo Archivo de Prueba > ArchivoPrueba.txt

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ ls -lra
total 9
-rw-r--r-- 1 felix 197609 18 Mar 20 18:48 ArchivoPrueba.txt
drwxr-xr-x 1 felix 197609  0 Mar 20 18:47 .git/
drwxr-xr-x 1 felix 197609  0 Mar 20 18:21 ../
drwxr-xr-x 1 felix 197609  0 Mar 20 18:46 ./

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git add ArchivoPrueba.txt
warning: in the working copy of 'ArchivoPrueba.txt', LF will be replaced by CRLF
the next time Git touches it

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git commit -m "Comentario de Prueba primer commit"
[RamaIndependiente (root-commit) d3fb1c1] Comentario de Prueba primer commit
1 file changed, 1 insertion(+)
create mode 100644 ArchivoPrueba.txt
```

Ilustración 7 Primer commit

El comando `ls -lra` solamente se uso como verificación de que se creó el fichero de prueba.

## 2.6 Eliminar commit

Antes de eliminar un commit se realizarán 2 commits más para tener un total de 3 y se eliminara el ultimo commit de dos maneras “soft” y “hard” la diferencia es que “hard” no conservara los cambios y eliminará todos los cambios añadidos en el directorio y “soft” conservara los cambios pero se eliminara el commit del historial.

Al escribir `git log` se podrá ver el historial de commits para verificar que hay 3 commits.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git log
commit 181f9673dfda44869e5726a34a37b3b6aa3a6da9 (HEAD -> RamaIndependiente)
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Thu Mar 21 12:03:13 2024 +0100

    Tercer commit

commit 7adbfa401ca7be9e73fbd91e4f6337524a56609
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Thu Mar 21 12:02:43 2024 +0100

    Segundo commit

commit d3fb1c148e06cc4748e644e480127393b679995d
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Wed Mar 20 18:49:49 2024 +0100

    Comentario de Prueba primer commit

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$
```

Ilustración 8 Historial de commits

A continuación el comando `git reset --hard HEAD~1` y `git reset --soft HEAD~1` para realizar la eliminación de un commit y HEAD~1 para eliminar el ultimo.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ ls -lra
total 13
-rw-r--r-- 1 felix 197609  0 Mar 21 12:19 ArchivoPrueba3.txt
-rw-r--r-- 1 felix 197609  0 Mar 21 11:57 ArchivoPrueba2.txt
-rw-r--r-- 1 felix 197609 18 Mar 20 18:48 ArchivoPrueba.txt
drwxr-xr-x 1 felix 197609  0 Mar 21 12:20 .git/
drwxr-xr-x 1 felix 197609  0 Mar 21 11:55 ../
drwxr-xr-x 1 felix 197609  0 Mar 21 12:19 ./

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git reset --hard HEAD~1
HEAD is now at 7adbfa4 Segundo commit

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ ls -lra
total 13
-rw-r--r-- 1 felix 197609  0 Mar 21 11:57 ArchivoPrueba2.txt
-rw-r--r-- 1 felix 197609 18 Mar 20 18:48 ArchivoPrueba.txt
drwxr-xr-x 1 felix 197609  0 Mar 21 12:23 .git/
drwxr-xr-x 1 felix 197609  0 Mar 21 11:55 ../
drwxr-xr-x 1 felix 197609  0 Mar 21 12:23 ./
```

Ilustración 9 Hard reset

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ ls -lra
total 13
-rw-r--r-- 1 felix 197609  0 Mar 21 12:24 ArchivoPrueba3.txt
-rw-r--r-- 1 felix 197609  0 Mar 21 11:57 ArchivoPrueba2.txt
-rw-r--r-- 1 felix 197609 18 Mar 20 18:48 ArchivoPrueba.txt
drwxr-xr-x 1 felix 197609  0 Mar 21 12:25 .git/
drwxr-xr-x 1 felix 197609  0 Mar 21 11:55 ../
drwxr-xr-x 1 felix 197609  0 Mar 21 12:24 ./

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git reset --soft HEAD~1

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git log
commit 7adbfa401ca7be9e73fbd91e4f6337524a56609 (HEAD -> RamaIndependiente)
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Thu Mar 21 12:02:43 2024 +0100

    Segundo commit

commit d3fb1c148e06cc4748e644e480127393b679995d
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Wed Mar 20 18:49:49 2024 +0100

    Comentario de Prueba primer commit

felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ ls -lra
total 13
-rw-r--r-- 1 felix 197609  0 Mar 21 12:24 ArchivoPrueba3.txt
-rw-r--r-- 1 felix 197609  0 Mar 21 11:57 ArchivoPrueba2.txt
-rw-r--r-- 1 felix 197609 18 Mar 20 18:48 ArchivoPrueba.txt
drwxr-xr-x 1 felix 197609  0 Mar 21 12:25 .git/
drwxr-xr-x 1 felix 197609  0 Mar 21 11:55 ../
drwxr-xr-x 1 felix 197609  0 Mar 21 12:24 ./
```

Ilustración 10 Soft reset

Ahora eliminaremos un el commit dos dejando solo el commit 1 y 3. Con el comando `git rebase -i HEAD~2` escogerá los dos últimos commits realizados y abrirá un fichero de donde vienen los commits precedidos de la palabra “pick”. En el mismo fichero puedes observar unos comentarios donde explica las palabras claves que puedes utilizar pero para borrar usaremos la palabra “drop” y al pulsar la tecla “esc” y escribir “:wq” saldremos del fichero y se guardaran los cambios habiendo borrado el commit numero 2.

Antes de todo configurar el editor de texto de git `git config --global core.<editor_de_texto>`. Yo usare vim como editor de texto.

```
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git config --global core.editor Vim
```

Ilustración 11 Configurar editor de texto



Tras el comando rebase este será el fichero que hay que configurar.

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...  
drop| 7adbfa4 Segundo commit  
pick 1f078e2 Tercer commit  
  
# Rebase d3fb1c1..1f078e2 onto d3fb1c1 (2 commands)  
#  
# Commands:  
# p, pick <commit> = use commit  
# r, reword <commit> = use commit, but edit the commit message  
# e, edit <commit> = use commit, but stop for amending  
# s, squash <commit> = use commit, but meld into previous commit  
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous  
#               commit's log message, unless -C is used, in which case  
#               keep only this commit's message; -c is same as -C but  
#               opens the editor  
# x, exec <command> = run command (the rest of the line) using shell  
# b, break = stop here (continue rebase later with 'git rebase --continue')  
# d, drop <commit> = remove commit  
# l, label <label> = label current HEAD with a name  
# t, reset <label> = reset HEAD to a label  
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]  
#       create a merge commit using the original merge commit's  
#       message (or the oneline, if no original merge commit was  
#       specified); use -c <commit> to reword the commit message  
# u, update-ref <ref> = track a placeholder for the <ref> to be updated  
#                   to this position in the new commits. The <ref> is  
#                   updated at the end of the rebase  
#  
# These lines can be re-ordered; they are executed from top to bottom.  
#  
# If you remove a line here THAT COMMIT WILL BE LOST.  
#  
# However, if you remove everything, the rebase will be aborted.  
#  
  
.git/rebase-merge/git-rebase-todo[+] [unix] (12:49 21/03/2024) 1,5 All  
-- INSERT --
```

*Ilustración 12 Fichero donde borrar el commit*

```
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
RamaIndependiente)
$ git log
commit dc1eaa388e8d2b2747dac68641aae16ba9d18836 (HEAD -> RamaIndependiente)
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Thu Mar 21 12:34:19 2024 +0100

    Tercer commit

commit d3fb1c148e06cc4748e644e480127393b679995d
Author: Felipe <felipe.iz.ro.98@gmail.com>
Date: Wed Mar 20 18:49:49 2024 +0100

    Comentario de Prueba primer commit
```

Ilustración 13 Muestra de eliminación del commit 2

## 2.7 Eliminar la rama en local y en el repositorio remoto

Para eliminar una rama en local sería suficiente con dejarla libre, que nadie este trabajando con ella y escribir el comando `git branch -d <nombre_de_la_rama>`

```
MINGW64:/c/Users/felix/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjerci...
felix@FelipeIzRo MINGW64 ~/SprintsBootcamp/Sprint_1/Lab_2/RepositorioEjercicio (
main)
$ git branch -d RamaIndependiente
Deleted branch RamaIndependiente (was dc1eaa3).
```

Ilustración 14 Borrar rama en local

Para hacerlo en el repositorio remoto de Git Hub habría que ir dentro del apartado Code abrir el desplegable de las ramas y seleccionar “View all branches”

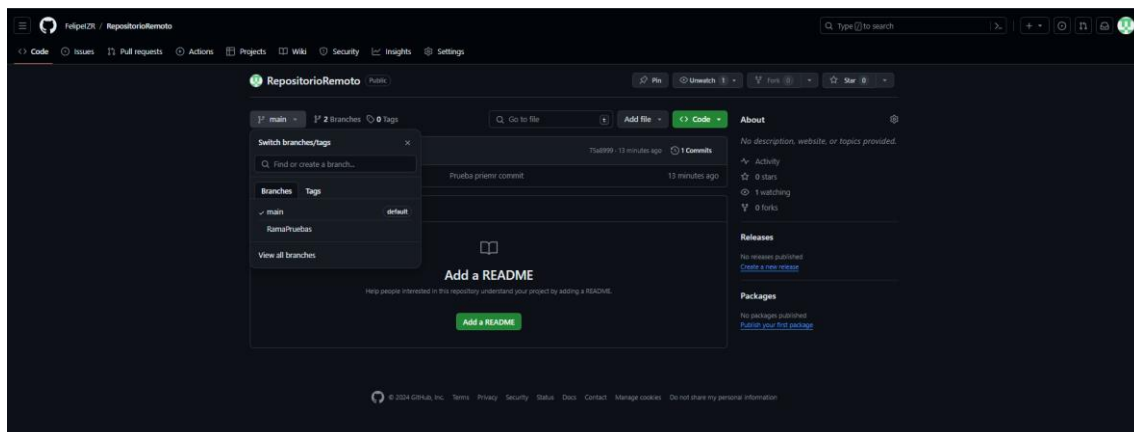


Ilustración 15 Apartado Code repositorio remoto

Escoger la rama que quieres eliminar y darle al botón con el icono de una papelera

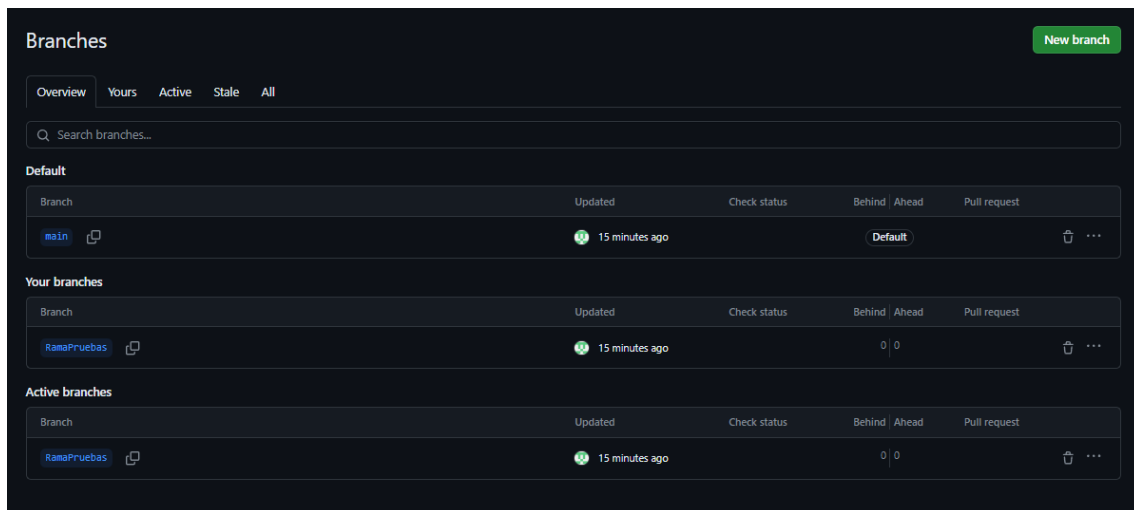


Ilustración 16 Rama borrada

Al recargar la página se observa como dejo de existir la rama borrada

## 2.8 Evitar la subida de algunos ficheros al repositorio remoto

Para lograr el siguiente punto en la raíz del repositorio hay que crear un fichero “.gitignore” con la consola de comandos o el explorador de archivos. En su interior escribir los directorios o los archivos que ignorar de la siguiente manera:

- Si es un directorio con la barra al final “/”.
- Si es un archivo su <nombre\_archivo>.<extension>.  
Ejemplo: (“Archivo.txt”).
- Con el asterisco “\*” se podrá usar como comodín.  
Ejemplo: Si quiero ignorar todos ficheros con extensión “.docx” escribiría “\*.docx”.

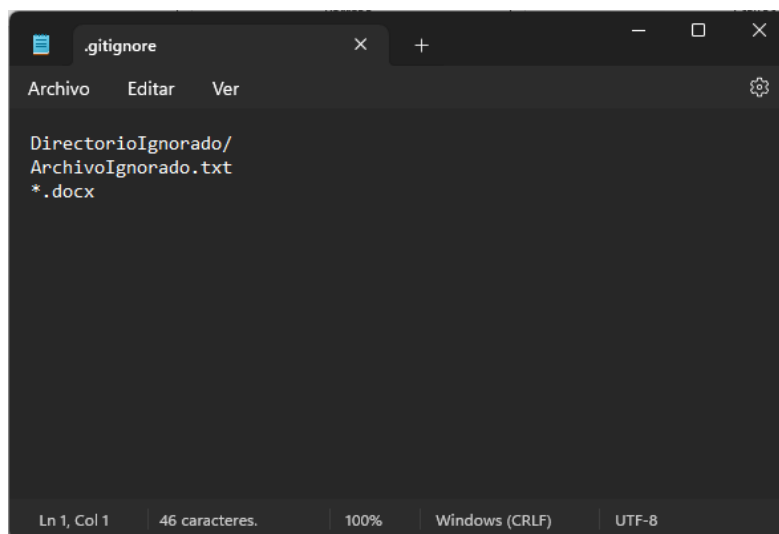


Ilustración 17 Archivo .gitignore

## 2.9 Instalar GitKraken y asociarlo al repositorio

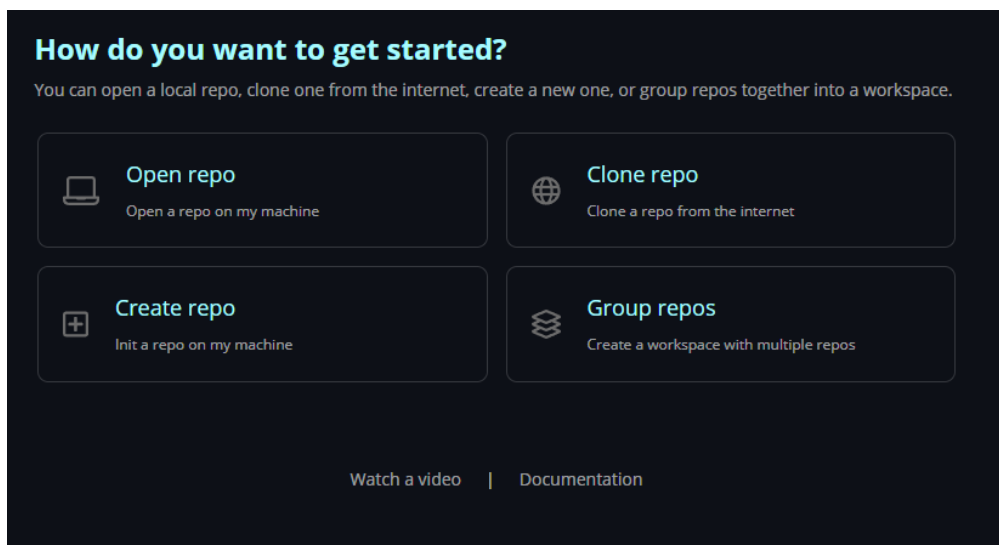
Para descargar GitKraken primero hay que ir a la pagina web.

<https://www.gitkraken.com/download>

Se descargará un archivo “.exe” que es el instalador. Se ejecuta y se instalará en unos segundos y a continuación tendrás dos opciones. Abrir un repositorio o iniciar sesión.

En este punto se elegirá iniciar sesión con GitHub y se abrirá en tu navegador y automáticamente se iniciará GitKraken.

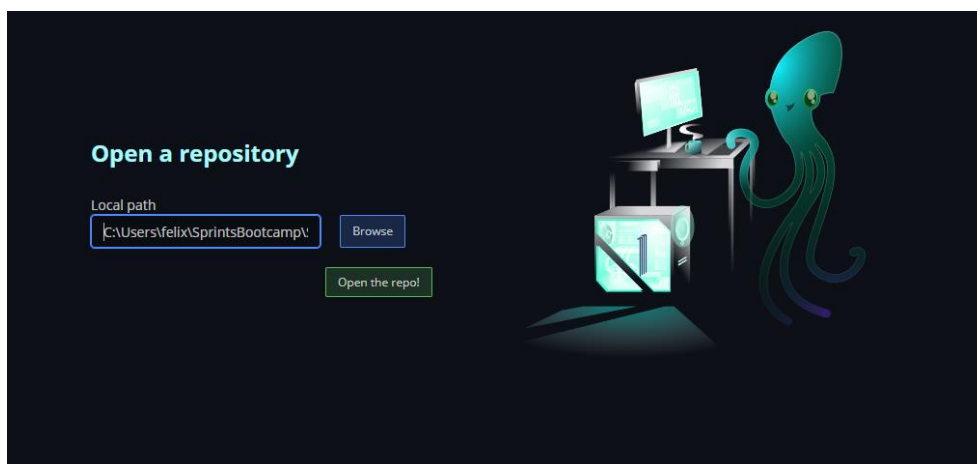
En la siguiente ventana abriremos un Repositorio en la opción “Open repo”.



*Ilustración 18 Inicio de GitKraken*

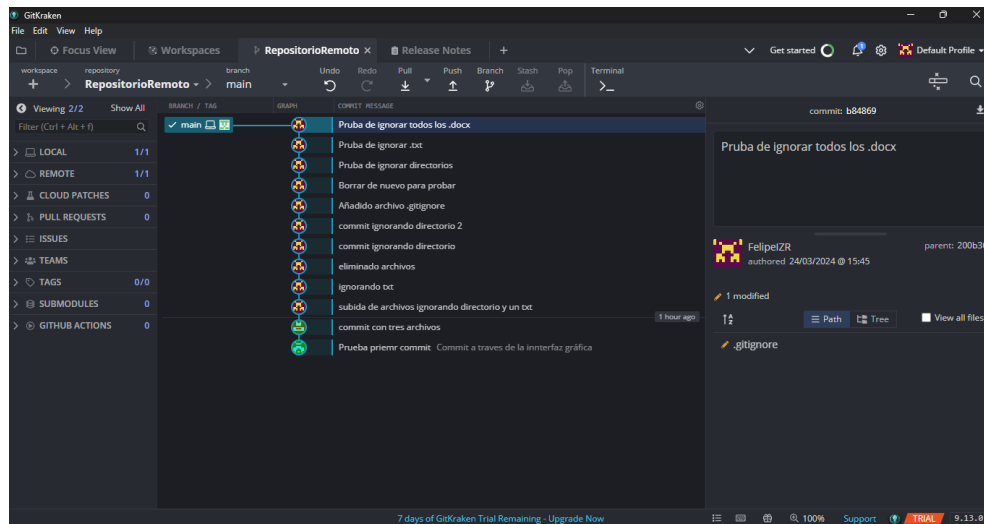
En el siguiente apartado pondremos la ruta absoluta del repositorio que esta en local en nuestro ordenador. En mi caso

“C:\Users\felix\SprintsBootcamp\Sprint\_1\Lab\_2\RepositorioEjercicioRemoto\RepositorioRemoto”.



*Ilustración 19 Introducir repositorio local*

Tras pulsar el botón verde “Open the repo!” y aceptar los términos y condiciones se abrirá la siguiente ventana con tu repositorio vinculado.



*Ilustración 20 Repositorio local vinculado*

### 3 Link del repositorio remoto

<https://github.com/FelipeZR/RepositorioRemoto>