Introducción a Git y GitHub

En esta práctica vamos a utilizar de forma básica el Sistema de Control de Versiones Git y la plataforma web que lo utiliza, GitHub.

Para esta práctica, podemos trabajar directamente sobre nuestro equipo Windows.

Parte 1 - Git en el equipo local

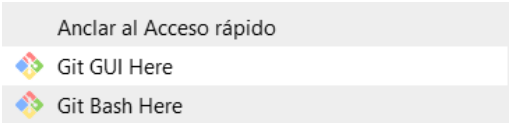
1. Instalar Git en el equipo

Para empezar, vamos a instalar Git en el equipo. Para esta práctica, lo haremos en un **entorno Windows**. Podemos descargarlo desde la página https://git-scm.com/download.

Una vez descargado, ejecuta el instalador y sigue el proceso de instalación. **Es recomendable modificar el** programa utilizado como **editor por defecto** (de primeras este será Vim, es más cómo cambiarlo por Notepad++ o Visual Studio Code).

Junto con Git se instalará **Git Bash** y **Git GUI**, ambas herramientas para facilitar el uso de Git: Bash desde consola y GUI con interfaz. En la práctica probaremos Git Bash.

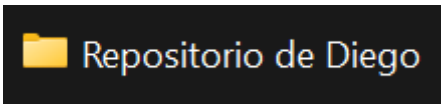
Si habilitamos “Windows Explorer Integration”, se añadirán al explorador de Windows las opciones de abrir directorios con Git GUI o Git Bash.



Adjunta un par de capturas del proceso de instalación, mostrando el editor por defecto elegido y el resultado exitoso de la instalación.

2. Lanzar Git desde consola

Primero, vamos a probar a utilizar Git con la consola de comandos, aunque después utilizaremos aplicaciones con interfaz. Para empezar, crea una carpeta “Repositorio de TuNombre”, que será en la que trabajaremos en esta parte.



Para abrir la carpeta en la consola Git Bash, hacemos **Click derecho -> Git Bash here.** También podemos abrir Git Bash y movernos a la ruta de la carpeta.

Muestra que has abierto la carpeta con Git Bash

2.1. Crear un repositorio local

Ahora ya tenemos la carpeta abierta en Git Bash, pero esta aún no es un repositorio (una carpeta configurada para utilizar Git como control de versiones). Si hacemos **git status**, veremos que no encuentra ningún repositorio.

Ejecuta git status y muestra la salida

Para configurar la carpeta como un repositorio, utilizamos el comando **git init**. Este comando creará una carpeta oculta llamada **.git**, en la que se guardan los metadatos de la aplicación.

Ejecuta git init y muestra la salida. Muestra la carpeta .git que se ha creado.

Antes de seguir, también nos interesa registrar un nombre y correo. Para esto hacemos:

git config.user.name “Tu nombre”

git config.user.email “Tu correo”

Ejecuta los dos comandos config y muestra la salida.

2.2. Registrar cambios

Con esto ya tenemos un repositorio, pero este aún está vacío. Si utilizamos el comando **git status** podemos ver el estado del repositorio, este nos dirá que no hay cambios.

Ejecuta git status y muestra la salida.

Para poder guardar diferentes versiones de la carpeta, tenemos que crear archivos dentro y añadirlos como cambios del repositorio.

Vamos a empezar creando un archivo “hola.html” con un header:

```
<h1> Hola, soy TuNombre <h1>
```

Si guardamos el archivo y volvemos a hacer **git status**, veremos que detecta el cambio.

Vuelve a ejecutar git status y muestra que ya hay cambios en la salida.

El programa nos está avisando de que hay cambios nuevos sin registrar. Para que Git guarde los cambios y podamos añadirlos a una versión/commit, es necesario añadirlos.

Para esto, utilizamos el comando **git add nombre_de_archivo**. Para hacerlo más cómodo, si hacemos **git add .** (con un punto) se añaden todos los cambios nuevos.

Si volvemos a hacer git status, veremos que el cambio ya está añadido y listo para añadirlo a un *commit* o versión nueva.

Ejecuta git status y muestra que el cambio ya está registrado

2.3. Crear versiones o *commits*

Una vez guardados los cambios, podemos crear una versión o *commit*. Los commits son cada versión de este repositorio y nos permiten volver a puntos anteriores de este.

Los podemos ver como “puntos de guardado” de un estado concreto de nuestra carpeta.

Para crear la primera versión, utilizamos el comando **git commit -m “Primer commit del proyecto”** El parámetro -m sirve para ponerle un nombre al proyecto.

Ejecuta git commit y luego haz un git status. Muestra la salida de los dos.

Con esto ya tenemos la versión registrada con un nombre. Vamos a probar a añadir otra más, en la que insertamos una imagen en el .html.

Añade una imagen a la carpeta del proyecto e inclúyela en el archivo .html con una etiqueta <img src=”nombre_imagen”>. Muéstralo en una captura.

Después, añadimos estos cambios con **git add .** y creamos un commit nuevo con el nombre “Añadida imagen a hola.html”.

Añade los cambios con git add, crea un commit con ellos y muestra la salida.

Para poder consultar los diferentes commit de un proyecto, podemos utilizar el comando **git log** , a este se le pueden añadir muchos parámetros para modificar su aspecto, p.ej. **git log --pretty=oneline**

Ejecuta git log y muestra la salida.

2.4. Deshacer cambios

Para acabar esta introducción a la consola de Git, vamos a ver cómo deshacer un cambio.

Vamos a simular que hemos borrado los archivos sin querer, para esto, borramos hola.html y el archivo de imagen.

Si hacemos **git status**, el borrado se detecta como un cambio, lo añadimos con **git add .**

Después de borrar el archivo .html, haz git add . y git status

Cuando queremos deshacer un cambio, podemos utilizar el comando **git reset**

git reset simplemente quita el cambio del registro de los que se añadirán al commit

git reset --hard elimina todos los cambios y vuelve al estado guardado en el commit

Recupera el archivo con el comando git reset --hard y muéstralo.

Git tiene muchos más comandos, pero para esta introducción, nos quedamos aquí.

Con esto, hemos visto el ciclo de uso básico de un control de versiones:

- Creamos un repositorio

- Creamos y registramos cambios

- Los añadimos a versiones (commits) del repositorio

Al tener un registro de cada versión del proyecto, se pueden hacer muchas operaciones como volver a versiones anteriores, eliminarlas, crear diferentes ramas paralelas del proyecto…

Parte 2 - GitHub: Plataforma web

1. Crear una cuenta en GitHub

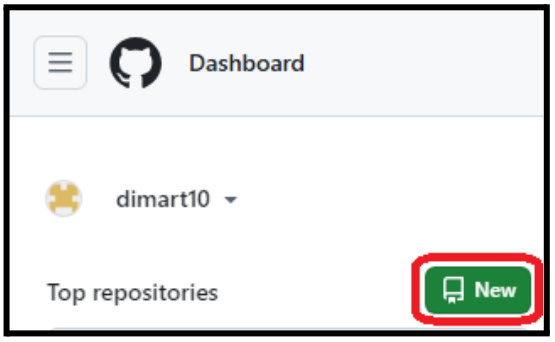
Para empezar a trabajar con GitHub, necesitamos crear una cuenta, lo podemos hacer desde la página https://github.com/

Por motivos de seguridad, la aplicación pide una autenticación de dos pasos, la más sencilla de configurar es un número de teléfono.

Una vez tengamos la cuenta creada, podremos ver la página de inicio, desde donde vamos a empezar creando un nuevo repositorio.

2. Crear un repositorio en la web

Podemos crear repositorios con el botón **New** en esta página.



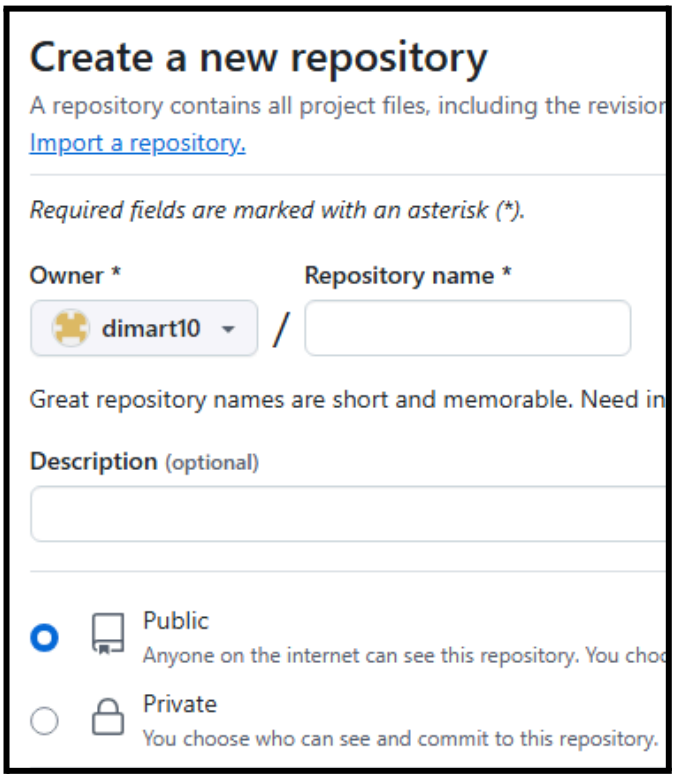
GitHub no tiene un límite estricto sobre el número de repositorios que podemos crear, pero sí en el espacio que estos ocupan. Del mismo modo, podemos borrar repositorios ya creados.

Al crear un repositorio, le ponemos un nombre. Este debe ser único para nuestra cuenta, no podemos tener dos con el mismo nombre.

Otro ajuste importante es su privacidad:

- **Público**: Cualquier persona puede ver y clonar el repositorio, pero sólo los colaboradores pueden subir cambios.

- **Privado**: Sólo los colaboradores pueden ver el repositorio y subir cambios.



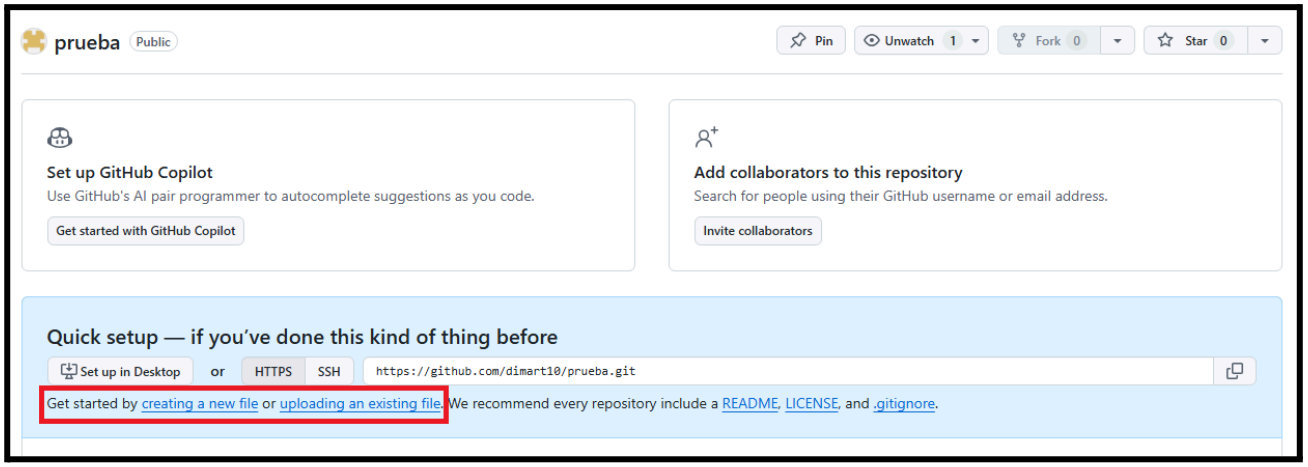
Muestra la pestaña de creación del repositorio y este una vez esté creado.

3. Subir cambios desde GitHub.com

De momento, vamos a subir cambios al repositorio desde la aplicación web, como lo haríamos en otros servicios como Google Drive.

Nada más crear el proyecto, se nos muestra una ventana de inicio con mucha información útil sobre cómo hacer su configuración inicial.

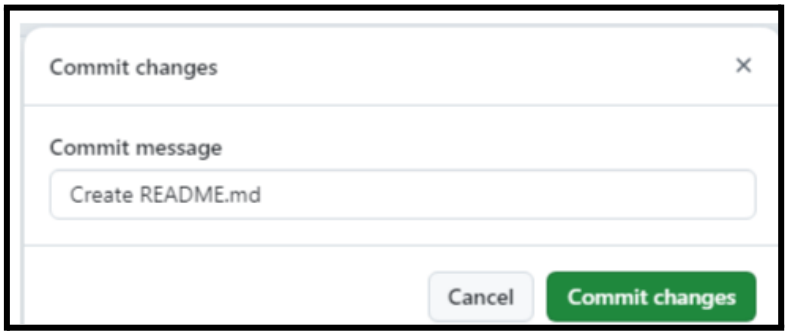
Lo más habitual, es abrirlo en un entorno de escritorio como GitHub Desktop. De momento, seleccionamos la opción de crear un nuevo archivo.



Creamos un archivo llamado README.md con un contenido parecido al de la imagen que tenga nuestro nombre.

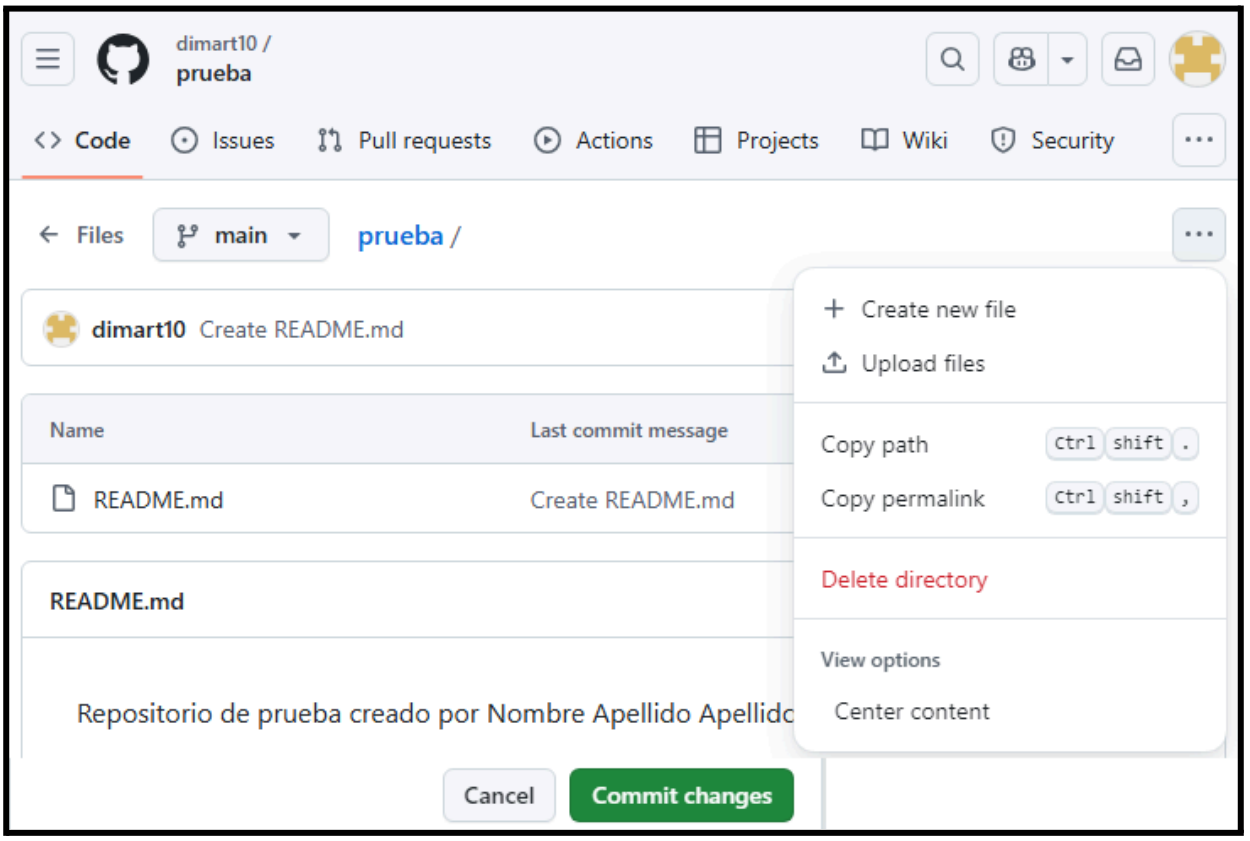
Muestra la ventana de creación del nuevo archivo.

Una vez listo el cambio, tendremos que añadirlo a un commit, para esto pulsamos en **Commit changes** y le ponemos un nombre a esta versión del proyecto.



Una vez creado el commit, este nos aparecerá en la pantalla de nuestro repositorio. Para ver los detalles sobre estos, podemos pulsar en la sección **<> Code**.

Como ya tenemos la configuración inicial hecha, podemos añadir nuevos cambios directamente desde esta pantalla, con las opciones **Create new file o Upload files.**



Antes de seguir, sube un archivo al proyecto desde la opción **Upload files**, esta es muy sencilla de utilizar, con un sistema drag and drop.

Muestra como subes un nuevo archivo al repositorio desde la web.

Muestra el commit desde la vista de commits en <> Code

Aunque, como hemos visto, podemos utilizar el repositorio desde la página web, lo más normal es sólo entrar en esta para gestionar el proyecto.

Para subir cambios y trabajar en el proyecto, lo más común es tener una copia en el escritorio. Para acabar, vamos a utilizar el programa GitHub Desktop.

Parte 3 - GitHub Desktop; Programa de escritorio

1. Instalar GitHub Desktop

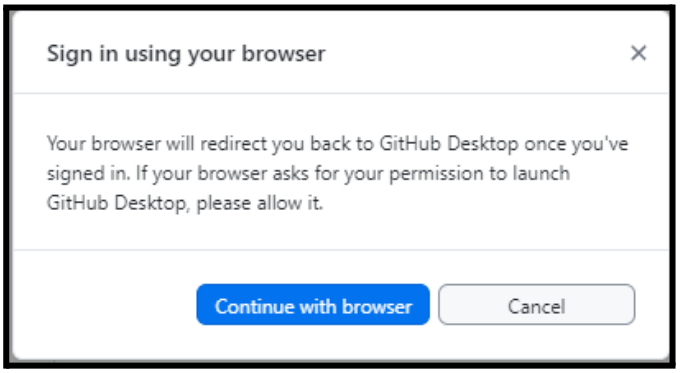
Podemos descargar el programa desde: https://desktop.github.com/download/

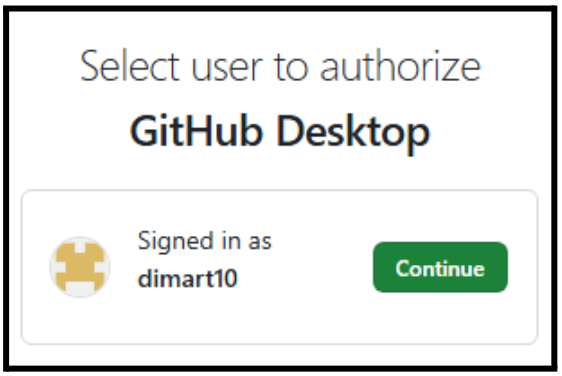
Muestra el programa una vez esté instalado.

2. Iniciar sesión en GitHub Desktop

Para empezar a trabajar, es necesario iniciar sesión con nuestra cuenta de GitHub.

La forma más sencilla de iniciar sesión es desde el navegador, si tenemos la cuenta ya abierta, podremos autorizar a GitHub el acceso a la cuenta directamente con un botón.



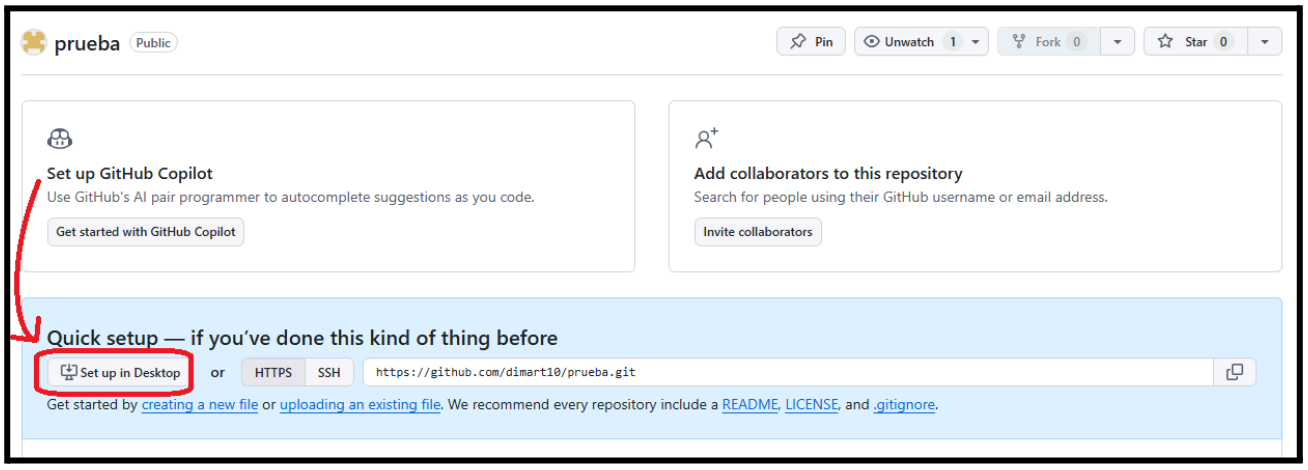


Muestra que has iniciado sesión en GitHub Desktop desde File -> Options

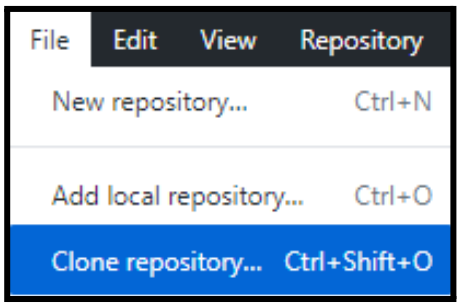
3. Descargar el repositorio de GitHub

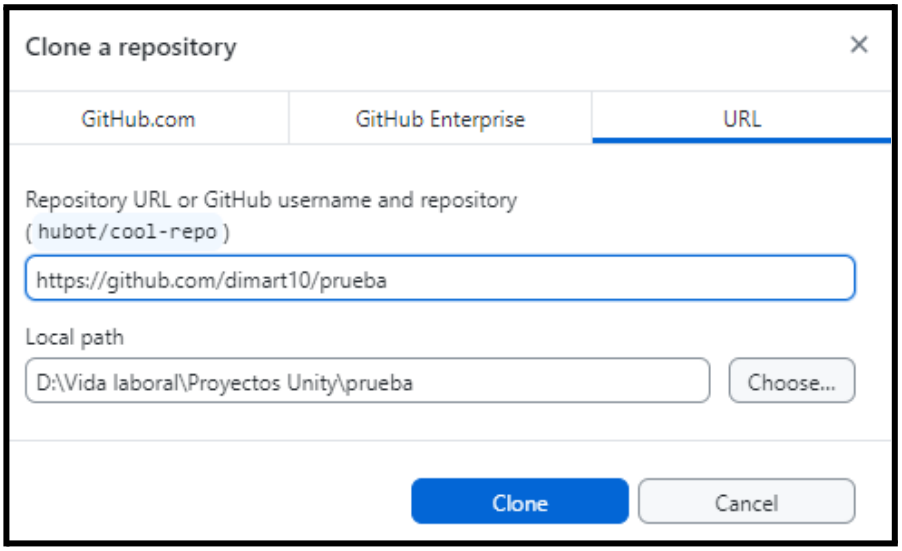
Una vez tenemos la sesión iniciada, tenemos varias formas de clonar el repositorio desde la página web a nuestro ordenador local:

- Al crear un repositorio, podemos hacerlo con la opción **Set up in Desktop.**

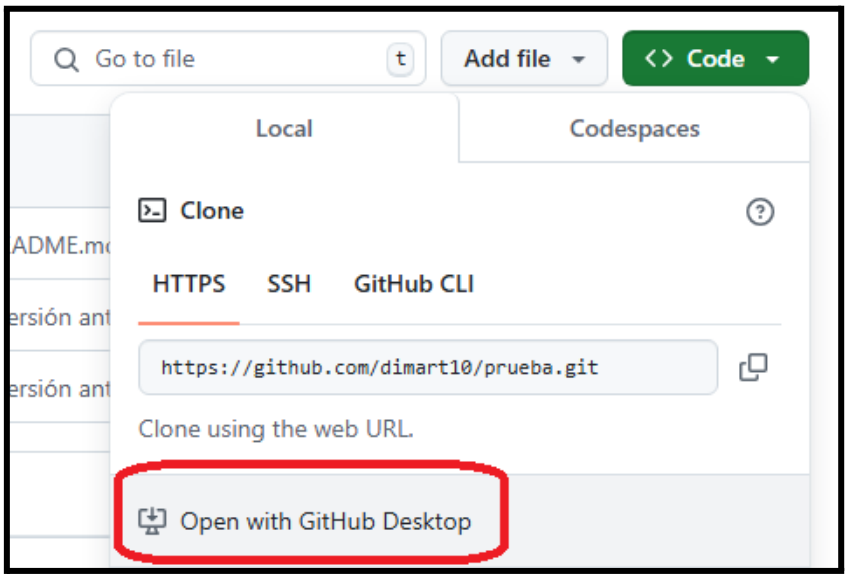


- Podemos clonar los repositorios a los que tengamos acceso con nuestra cuenta desde GitHub Desktop, con la opción **File -> Clone repository…**



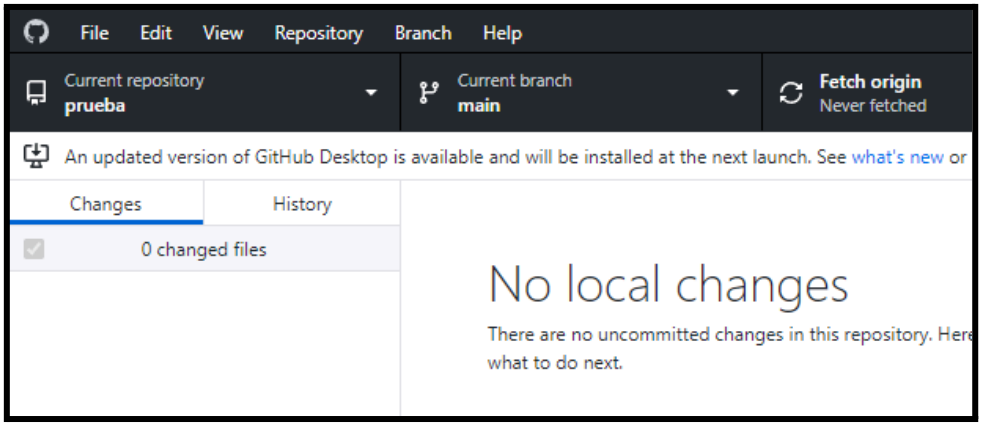


- Desde GitHub.com, podemos clonar repositorios desde su página, en el menú **<> Code** con la opción **Open with GitHub Desktop.**

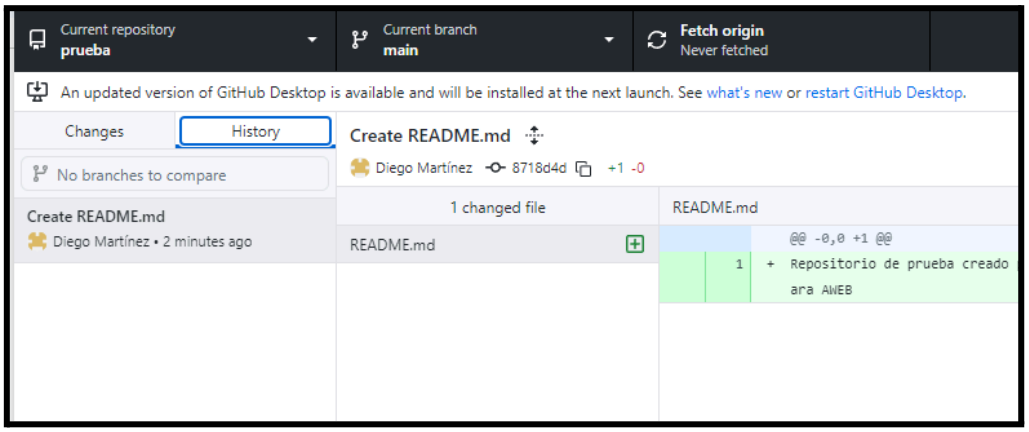


Clona el repositorio a tu ordenador de una de las formas explicadas. Comenta cuál has utilizado y muéstralo con capturas de pantalla.

Una vez tenemos el proyecto clonado, en **Changes** podemos ver los cambios que haya en el repositorio, y seleccionarlos para añadirlos o no a un commit.



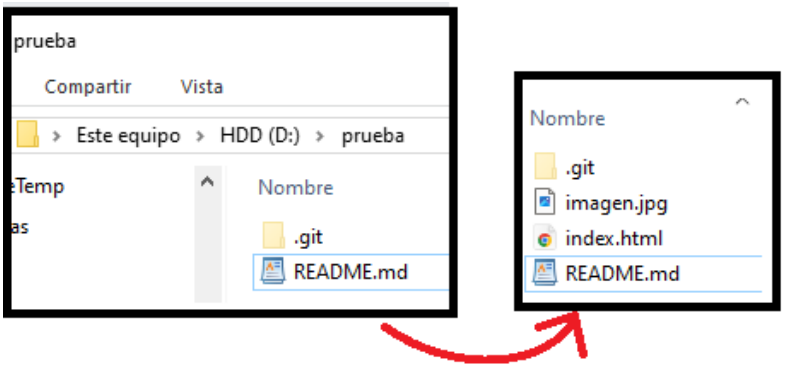
Desde el menú **History**, podemos ver todos los commits que se han hecho en el repositorio, y examinar los cambios que se han añadido en cada uno de ellos.



4. Subir cambios desde GitHub Desktop

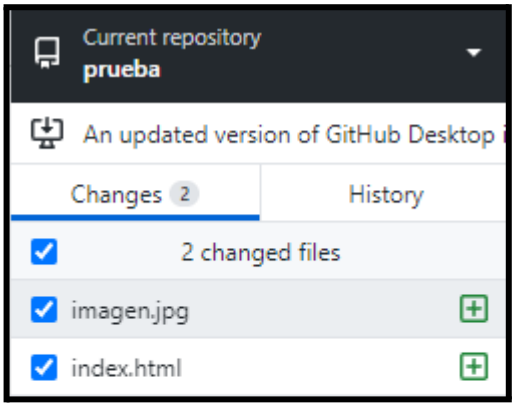
Para ver cómo subir cambios desde la aplicación de escritorio, primero vamos a modificar nuestra copia local del repositorio. Este podemos encontrarlo en la ruta que elegimos al clonarlo, o acceder desde la opción **Repository -> Show in explorer**

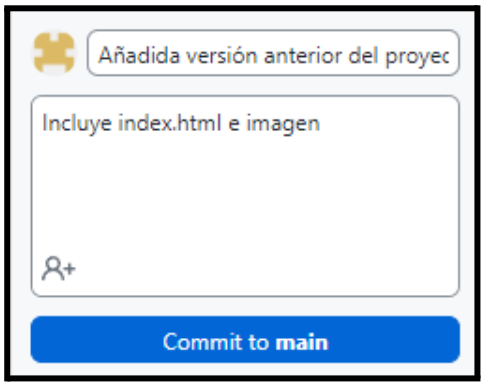
Una vez en la carpeta, añadimos el archivo .html y la imagen creada en la parte anterior.



Muestra que añades los archivos a la carpeta del repositorio.

Como podemos ver, los cambios que hemos hechos se reflejan en la aplicación GitHub Desktop. Desde este menú, podemos seleccionar si añadir estos cambios a la versión, si ignorarlos o si deshacerlos.



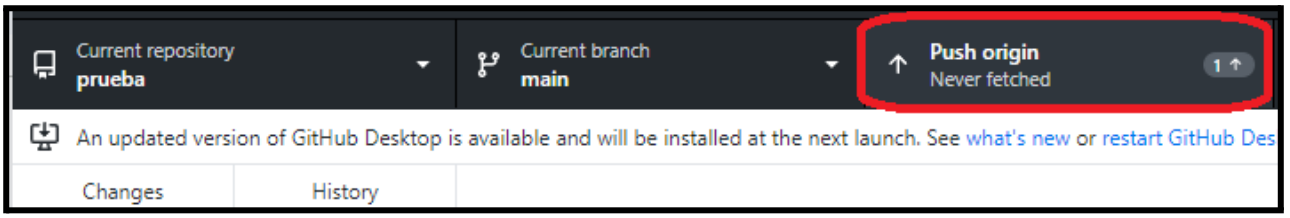


Después, podemos poner un nombre al commit y pulsar en **Commit to main**, lo que registra esta versión a nuestra versión local del repositorio.

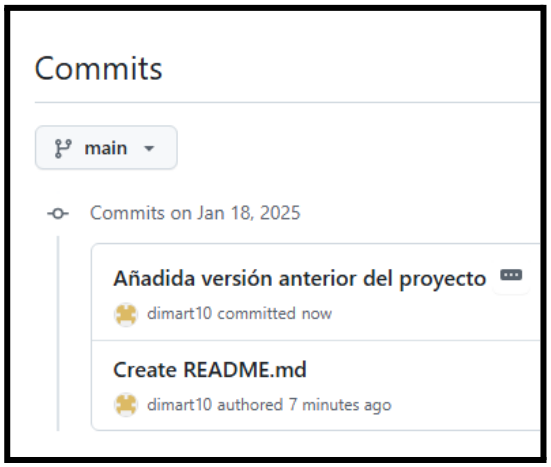
Muestra los cambios en GitHub Desktop y el proceso en el que añades el commit al repositorio.

Una particularidad de utilizar GitHub es que aunque creemos una versión del proyecto en nuestra copia local, este commit aún no aparecerá en la página web.

Para subir cambios al repositorio en la web, tenemos que hacer la operación **push** (en consola, sería **git push**). En GitHub Desktop, tenemos un botón para hacerlo.



Este botón también sirve para descargar cambios desde el repositorio de la web, que podríamos haber creado nosotros u otro colaborador del proyecto. Con la opción **Fetch**, comprobamos si hay cambios y con **Pull** los descargamos si los hay.



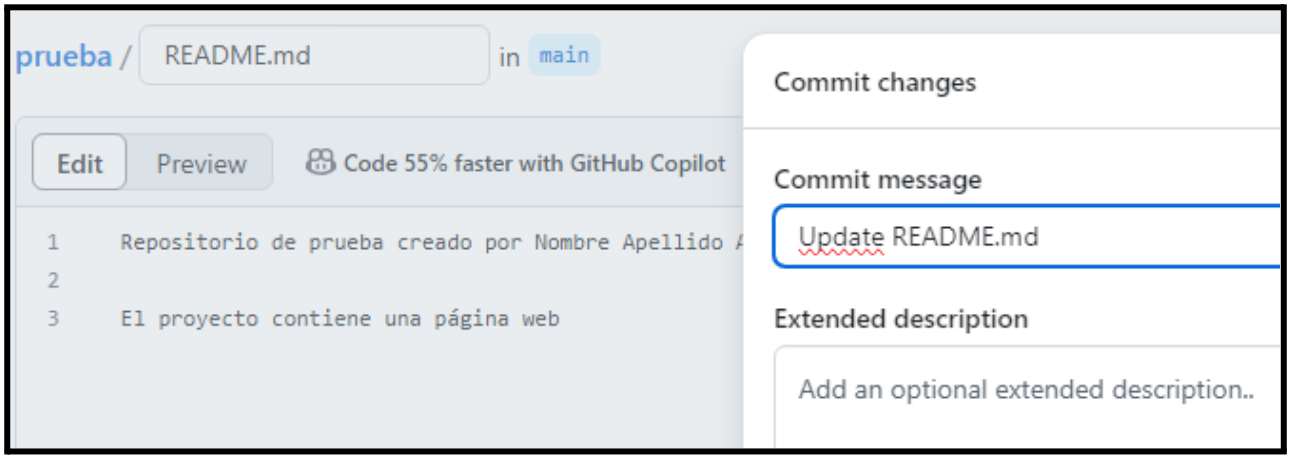
Para comprobar que el commit se ha subido, podemos acceder a la lista de commits en la aplicación web.

Muestra que el nuevo commit aparece en GitHub.com

4. Descargar cambios desde GitHub Desktop

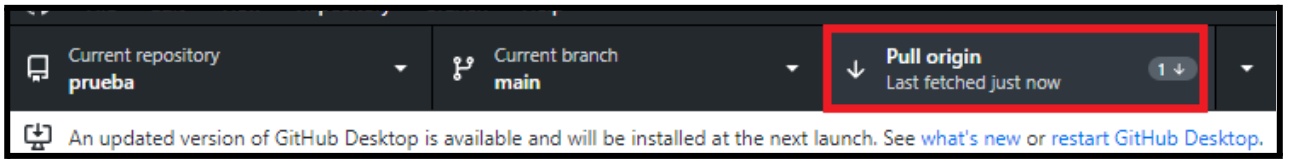
Para probar la función **Pull** de GitHub Desktop, vamos a añadir un cambio al proyecto desde GitHub.com y a luego descargarlo.

Para esto, podemos editar el archivo README o subir un nuevo archivo desde la web.



Muestra como haces un nuevo commit al repositorio desde la página web.

Como dijimos, podemos descargar los cambios en GitHub.com con **Pull origin**



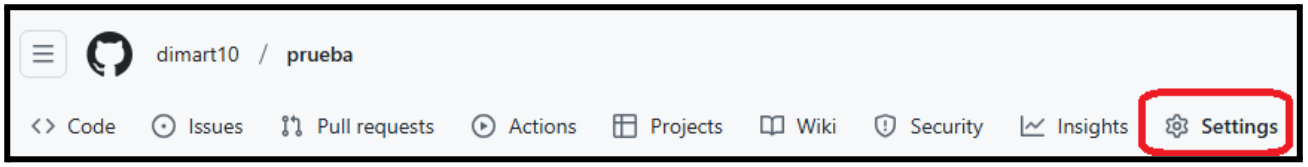
Una vez descargados los cambios, estos se aplicarán a la carpeta de la copia local de nuestro repositorio.

Muestra que los cambios hechos aparecen en la carpeta del repositorio y que el nuevo commit aparece en la pestaña History de GitHub Desktop.

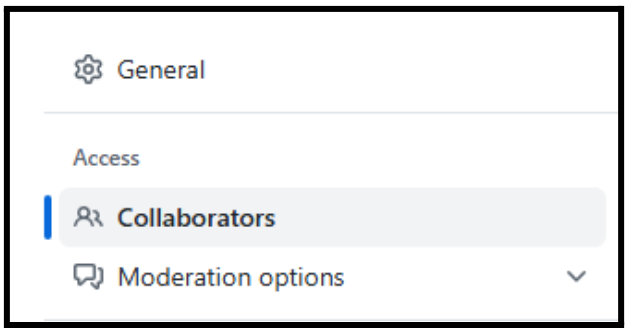
5. Invitar a otras personas a colaborar

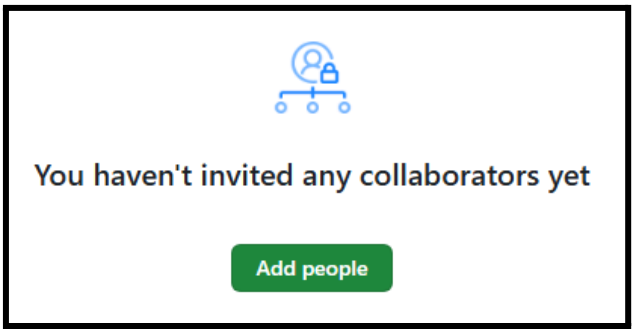
Como ya comentamos, una de las principales ventajas de los Sistemas de Control de Versiones es la capacidad para colaborar con otras personas y facilitar la gestión de proyectos complejos con muchos ficheros, especialmente en programación.

Por eso, vamos a ver cómo invitar a otras personas como colaboradores del repositorio.



Desde **Settings,** accedemos a la sección **Collaborators** y pulsamos en **Add people**. Aquí, podemos invitar a otros usuarios con su nombre de usuario en GitHub.





Para probar esto, intercambia invitaciones a vuestros repositorios con un compañero de clase. Una vez hecha la invitación, basta con aceptarla a través del correo electrónico.

Muestra que su repositorio aparece en tu cuenta de GitHub.

Una vez en el repositorio, vamos a probar a crear un commit en el repositorio del compañero. Para esto puedes clonar el repositorio en tu equipo y hacerlo con GitHub Desktop o hacerlo desde la página web.

Muestra que tu commit aparece en el repositorio de un compañero

Conclusión

En esta práctica hemos visto el uso básico de un VCS. Hay muchas opciones y funcionalidades tanto en la consola, como en las aplicaciones web y de escritorio que no hemos tratado.

Cabe mencionar, que todas las configuraciones que hacemos desde Desktop, se pueden hacer también con comandos en Git Bash, aunque resulte más complicado.

Lo importante de momento es que entendamos la utilidad de estos sistemas y las bases sobre cómo funcionan y cómo utilizarlos; ya que son muy habituales en el mundo del desarrollo software, tanto a nivel doméstico como profesional.

Instrucciones de entrega

Entrega un archivo en formato .pdf en la entrega del Aula Virtual para esta tarea. En este archivo, adjunta capturas de pantalla con la información pedida en el texto en rojo. Organiza el documento según los apartados del enunciado.

Puedes utilizar el documento plantilla que hay en la entrega del Aula Virtual.

Además, escribe la URL de tu repositorio en el campo de texto de la entrega. Recuerda tener el proyecto configurado como público.