**C.F.G.S DAM Y DAW .ENTORNOS DE DESARROLLO**

**TEMA 4. Control de versiones**

**PRÁCTICA DE CONTROL DE VERSIONES. (SCV)**

|  |
| --- |
| RA4. Optimiza código empleando las herramientas disponibles en el entorno de desarrollo.    f) Se ha realizado el control de versiones integrado en el entorno de desarrollo. |

En esta práctica aprenderás el concepto y el uso de un sistema de control de versiones. Debes realizar la práctica a partir de las explicaciones iniciales del profesor.

Debes entregar la práctica en formato pdf. El trabajo preferiblemente se hará en parejas pero podrá hacerse individualmente.

Esta práctica tendrá 2 puntos para el examen. Para superar la práctica debes tener una nota mínima de 1 punto. Superar la práctica exime de hacer las preguntas de esta parte en el examen final.

# PARTE 1

En tu trabajo debes entregar:

* Portada con el nombre del trabajo y autores.
* Índice con los apartados del trabajo y las páginas.
* ¿Qué es un control de versiones y para qué sirve?

Un programa para el control de versiones es una aplicación ideada para gestionar ágilmente los cambios en el código fuente de los programas y poder revertirlos. Registrando los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo de tal manera que sea posible recuperar versiones especificas más adelante.

* ¿Qué es git y Github?

Git

Git es un sistema de control de versiones distribuido. Git fue impulsado por Linus Torvalds y el equipo de desarrollo de del Kernel de Linux. Debido a que el software utilizado anteriormente para tal tarea se convirtió en software propietario, el equipo de Linux creo su propio software también distribuido que aportase lo mejor de los sistemas existentes hasta el momento. Así nació Git, un sistema de control de versiones de código abierto y multiplataforma. Existen varias interfaces gráficas, pero se recomienda usarlo con líneas de comandos.

Github

Es un servicio para el alojamiento de repositorios de software gestionados por el sistema de control de versiones de Git. A diferencia de Git, Github es algo mas particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software.

* ¿Qué otros sistemas de control de versiones se suelen actualizar en la actualidad?
* ¿Qué son los sistemas centralizados y distribuidos? Diferencias
* Define los siguientes conceptos en el SCV (Sistema de control de versiones) git: o Repositorio

o Que incluye el fichero .git o Explicar los siguientes comandos:  Comando Add.  Comando Commit  Comando Pull.

* + Comando Push
  + Comando Status

Una rama Git es un apuntador móvil apuntando a una de las confirmaciones (commit). La rama por defecto es la rama master, con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama ira avanzando automáticamente. Y la rama master apuntara a la última confirmación realizada. En resumen, una rama es una modificación de los datos que difiere de la rama principal o rama master.

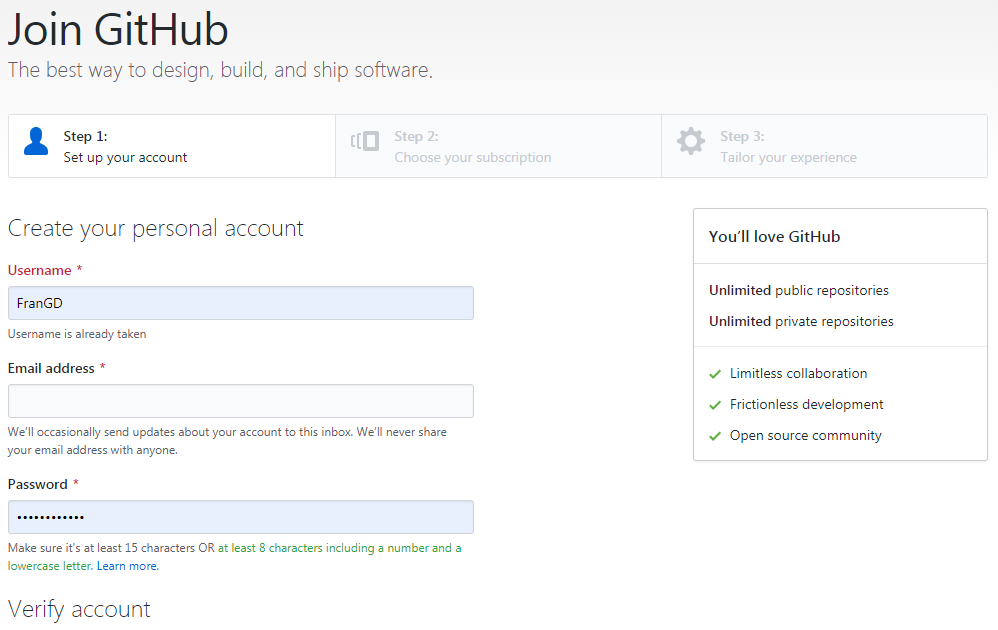
**C.F.G.S DAM Y DAW .ENTORNOS DE DESARROLLO**

**TEMA 4. Control de versiones**

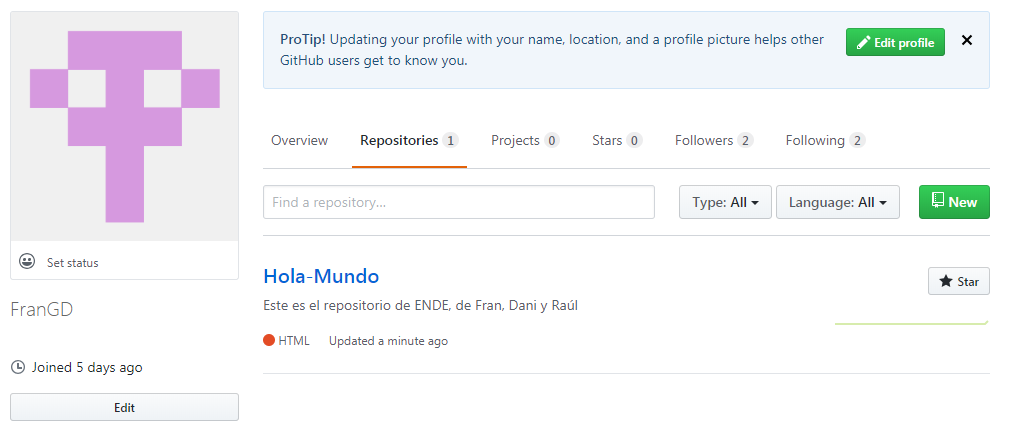
## **PARTE 2** (Para realizar esta segunda parte debes entender bien los conceptos de la primera parte)

Para que puedas tener en tu ordenador una copia de un sistema de control de versiones y puedas trabajar en paralelo con un compañero, haremos lo siguiente:

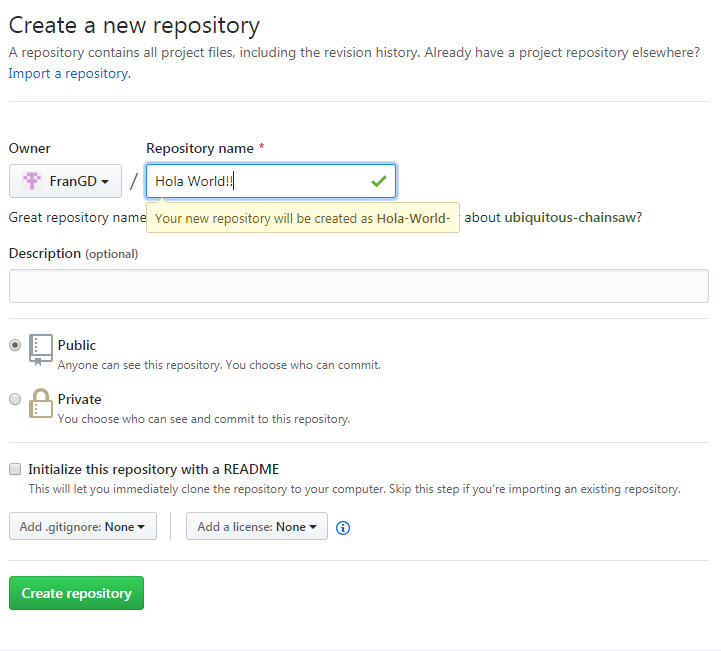
Lo primero que necesitaremos será crear una cuenta de Github donde crearemos los repositorios para ello accedemos a github.com y pulsamos en la opción de sign up e introducimos nuestros datos.

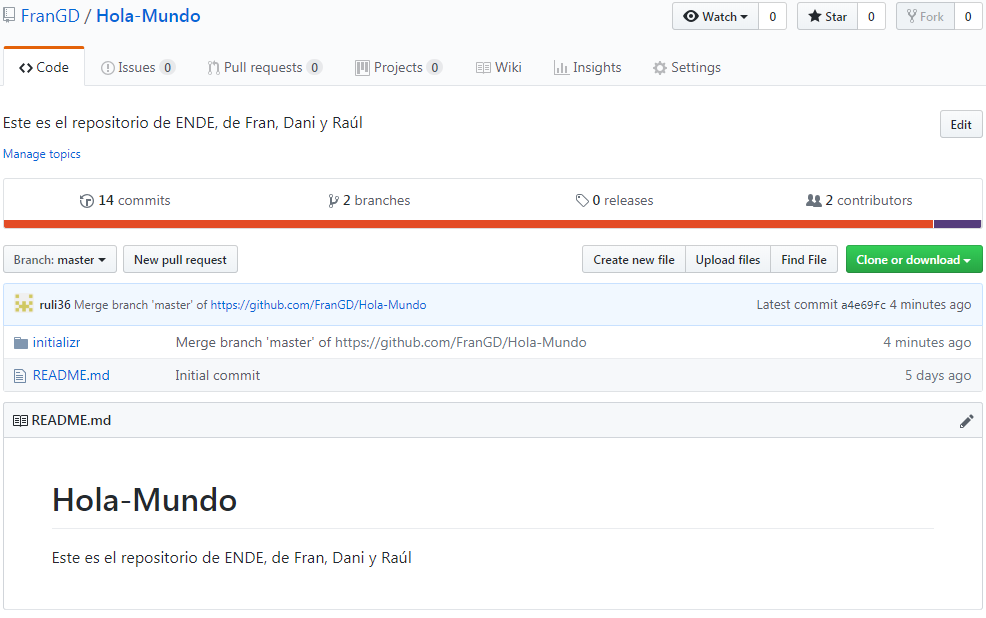


Una vez creada la cuenta crearemos nuestro primer repositorio para ello en página del perfil creado y accedemos a la pestaña Repositories. En la pestaña pulsamos en new.

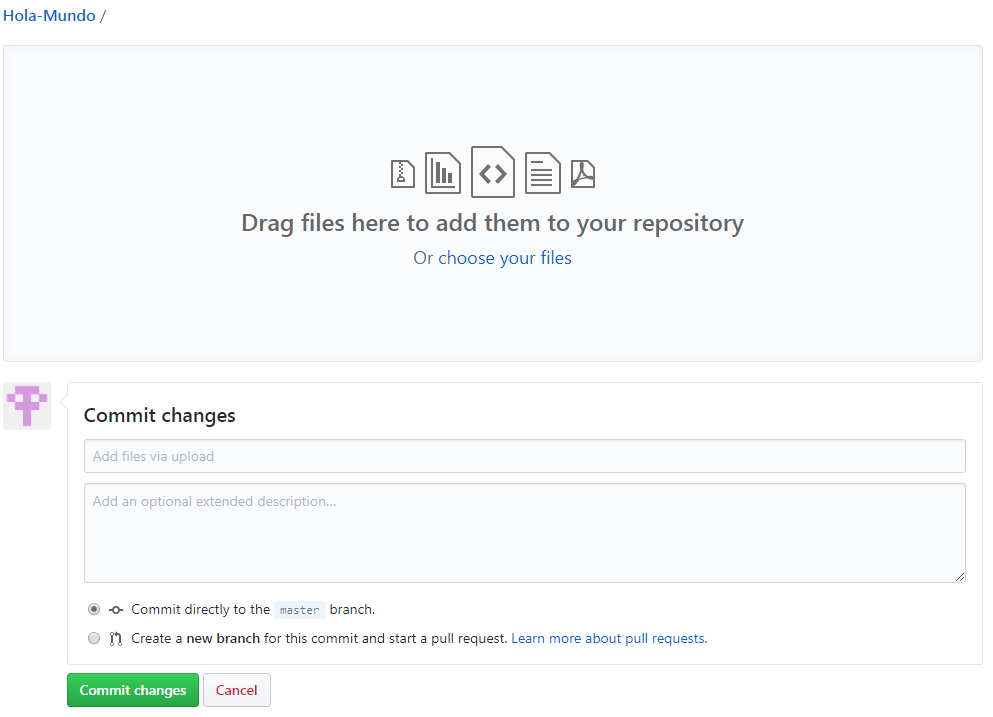


La nueva ventana nos pedirá el nombre del repositorio, una descripción opcional y si queremos que este sea publico o privado.

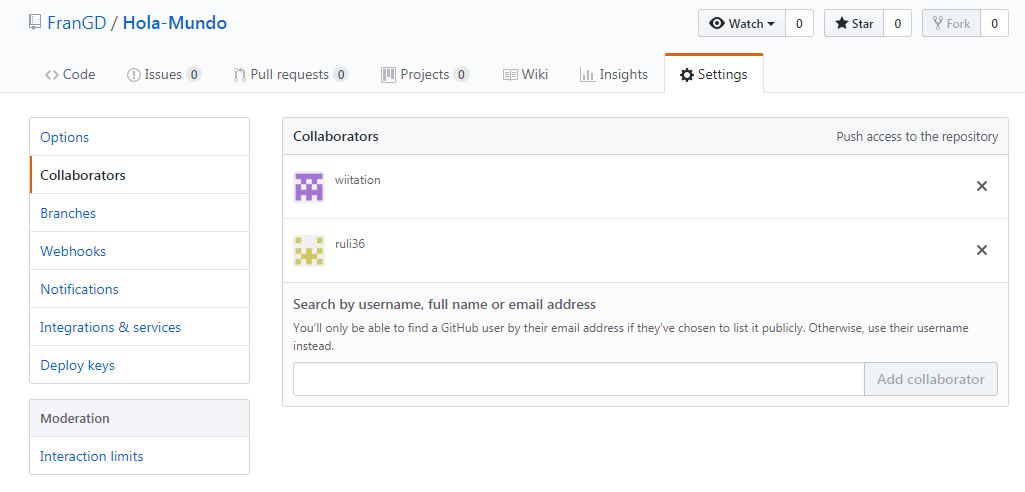


Dentro del repositorio pulsamos en la opción upload files donde se nos preguntará por el archivo que deseamos subir.

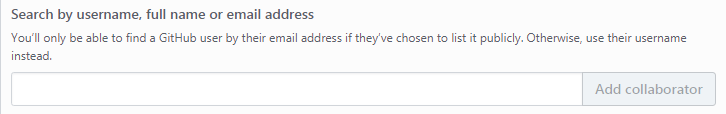
Arrastramos el archivo y pulsamos commit.



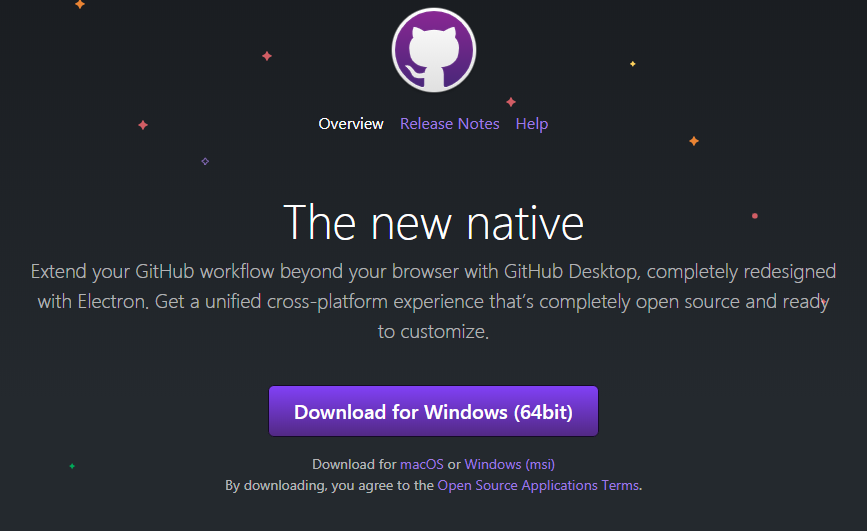
Para añadir un colaborador seleccionamos la pestaña settings en la ventana del repositorio. Abajo a la izquierda en la lista de opciones pinchamos en Collaborators.



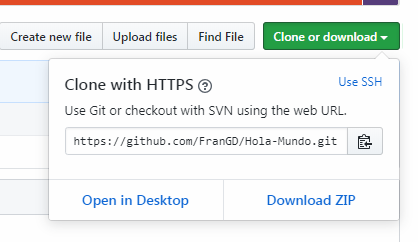
Buscamos a nuestro compañero de proyecto y lo añadimos.



Para descargar Github desktop nos dirigimos al siguiente enlace: <https://desktop.github.com/> aquí encontraremos la versión desktop de GitHub y procederemos con la descarga.

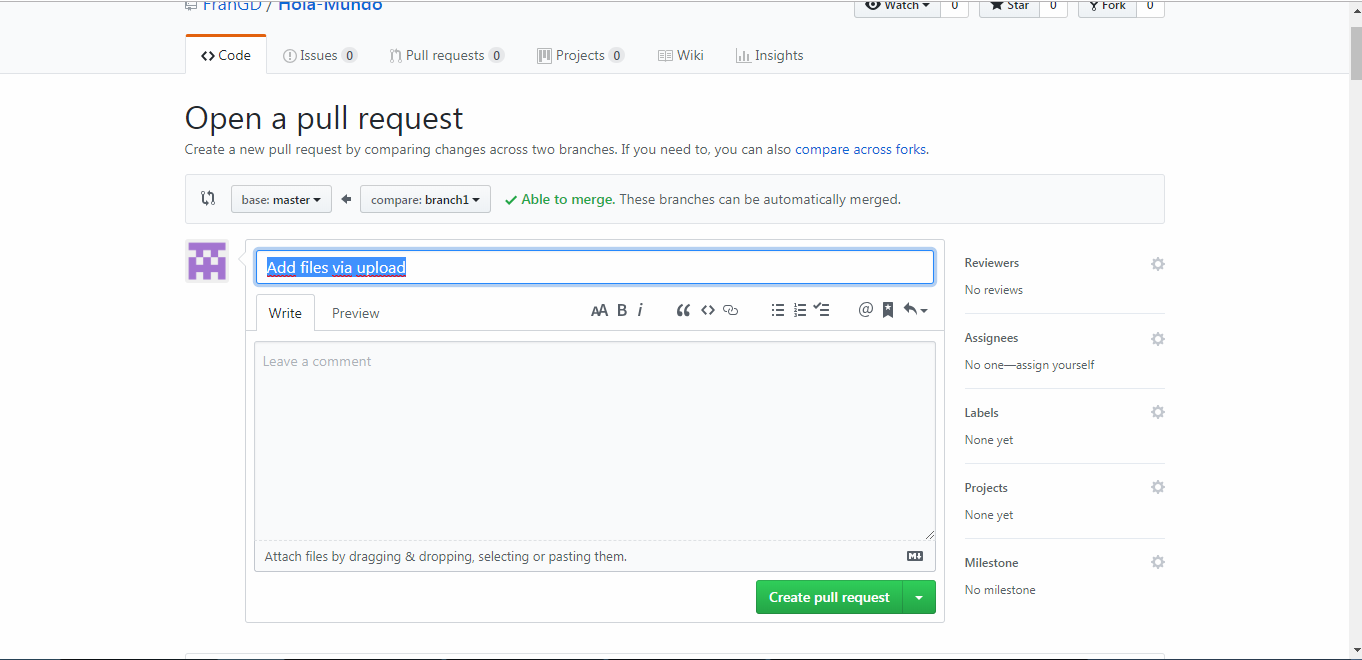


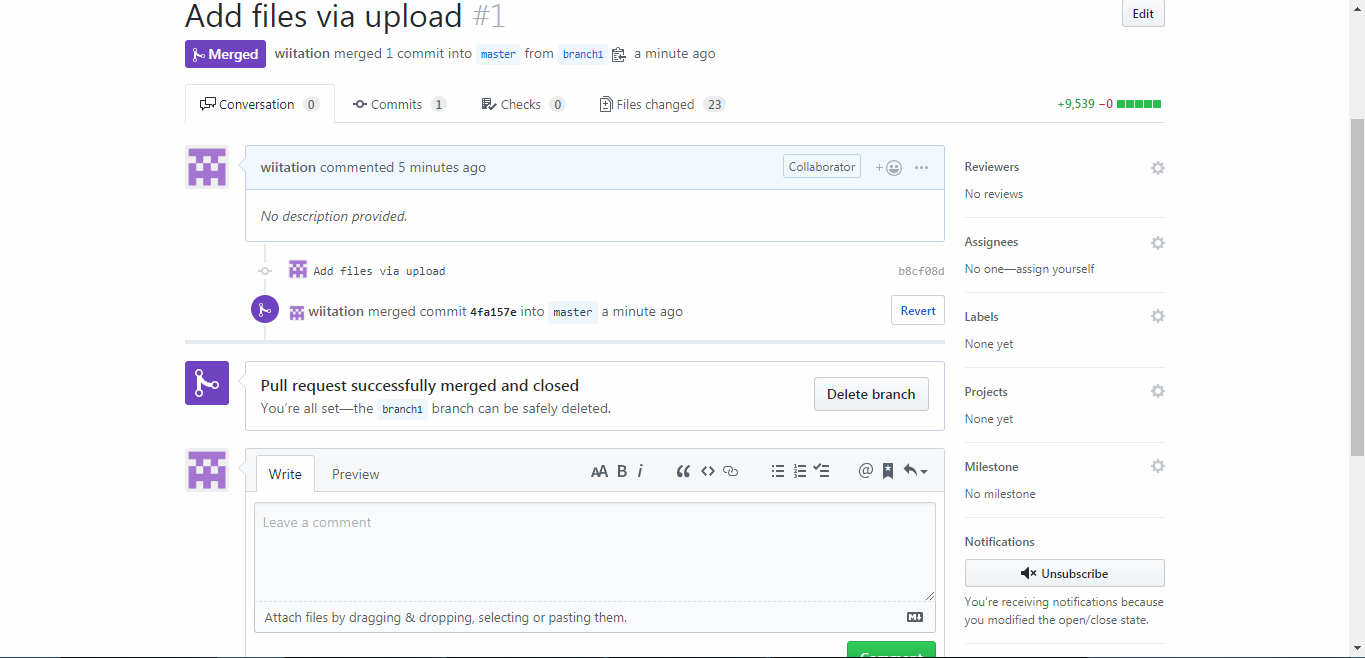
Para realizar un clone en nuestro PC solo tendremos que pulsar el botón clone or download que tenemos dentro del repositorio.



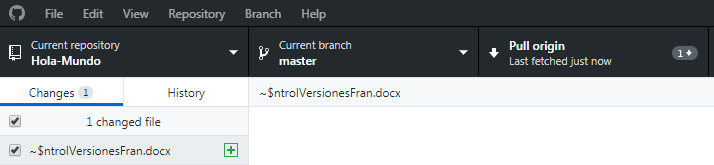
* Añade carpetas con archivos y súbelos al repositorio. **¿Qué comandos debes utilizar para añadir un archivo, confirmar el cambio y subirlo al repositorio?**

Para añadir carpetas con archivos debemos utilizar el comando pull, iniciamos una pull request y le indicamos a Githuba cuales son los archivos que necesitamos subir. Creamos la pull request y lo confirmamos con commit.

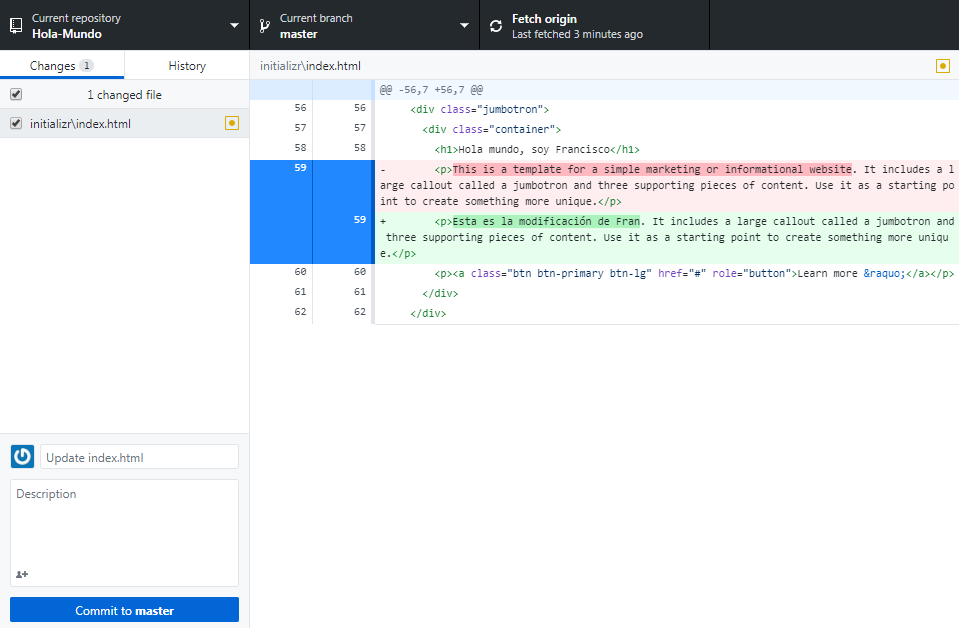




* Ambos compañeros debéis tener en vuestro ordenador una copia del proyecto en vuestro equipo local.

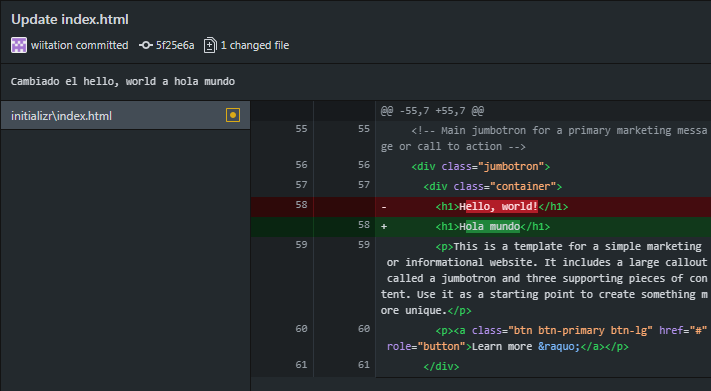


* Ambos compañeros debéis modificar un archivo distinto. Ve a la aplicación de github instalada. Comprueba que ha pasado.

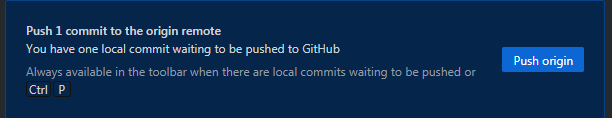




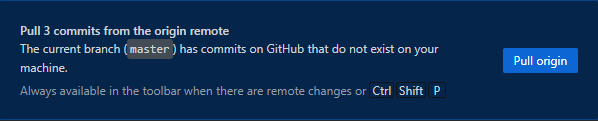
* Realiza *commit* de ese fichero modificado para que suba al repositorio. (recuerda poner una descripción antes de hacerle commit)



* Realiza un push de ese fichero para que suba al repositorio.

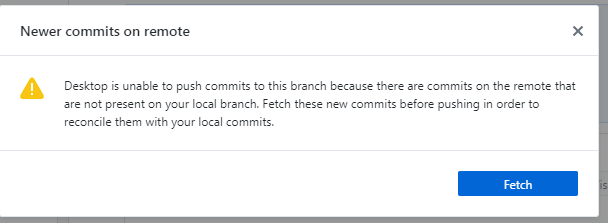


* Descarga de nuevo todos las modificaciones que se han realizado en el proyecto. Debes haber recibido en tu repositorio local (clone del proyecto) los cambios que ha realizado tu compañero y tu compañero los tuyos.

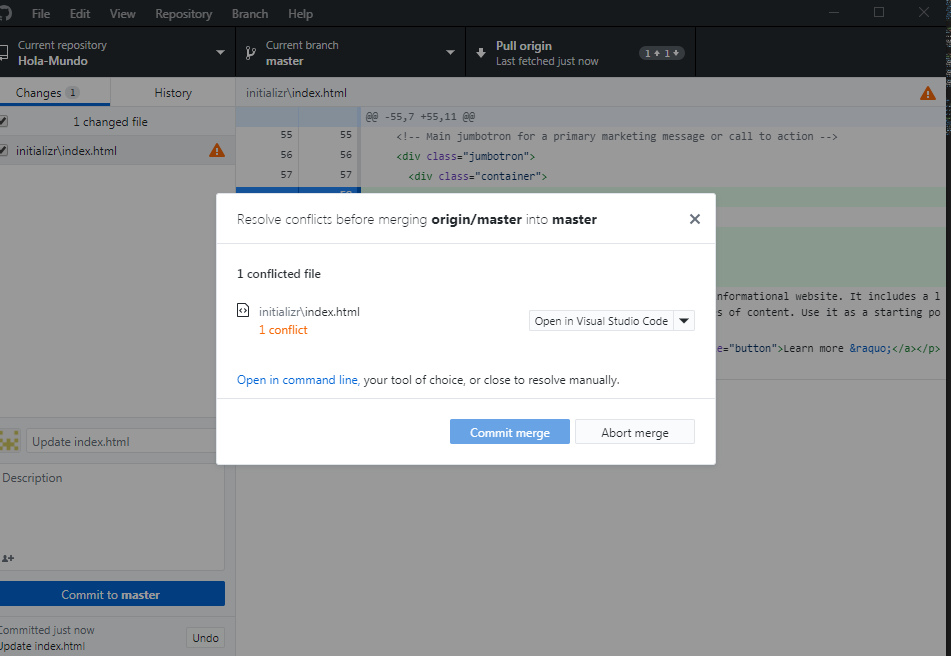


* Conflictos. Prueba a modificar el mismo archivo que tu compañero en la última versión. Hazle commit y push. **¿Qué sucede? ¿Cómo lo solucionamos?**

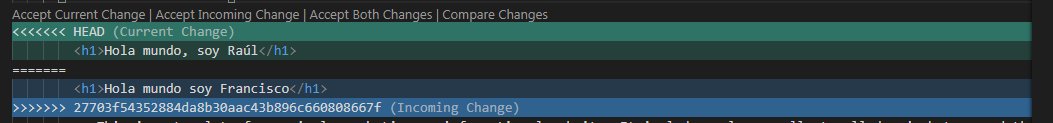
Modificamos el mismo archivo con un compañero en la ultima versión, hacemos commit y push. Nos notifica que se ha producido un conflicto.



El programa te advierte que no puede hacer un commit a la rama porque hay commits que no estan presentes en tu version del archivo



Te muestra el archivo con el conflito, al abrirlo en un editor puedes ver cual es el problema, junto a las opciones disponibles:



* Accept current change: Acepta el cambio y sobreescribe la version subida.
* Accept Incoming change: Acepta el cambio y sobreescribe tu versión.
* Accept Both Changes: Acepta ambos cambios y los incorpora a tu documento.
* Compare changes: Compara los cambios en ambos documentos para ver la diferencia entre ambos.