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

**TEMA 4. Control de versiones**

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

**Indice**

**Parte 1**

¿Qué es un control de versiones? ............................................................................................

¿Qué es git y GitHub? ...............................................................................................................

¿Qué otros sistemas de control de versiones se usan? ............................................................

¿Qué son los sistemas centralizados y distribuidos? ................................................................

Definición de conceptos ...........................................................................................................

**Parte 2**

Añadir colaborador al proyecto ...............................................................................................

Añadir carpetas y subirlas al repositorio .................................................................................

Modificar un archivo distinto ..................................................................................................

Commit y push del archivo ......................................................................................................

Descarga de modificaciones en equipo local ...........................................................................

Conflictos .................................................................................................................................

# PARTE 1

* 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.

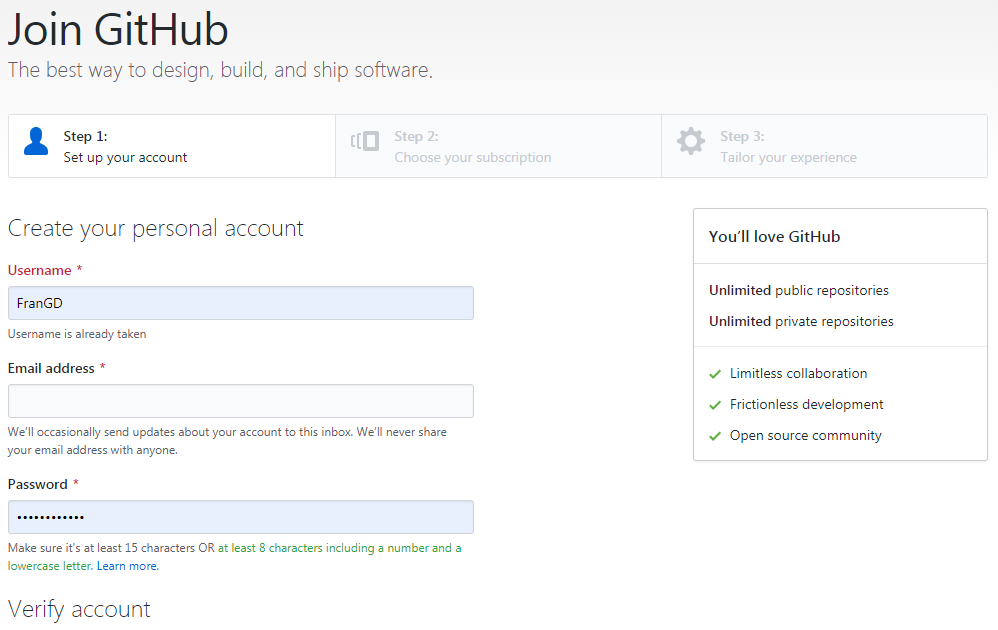
**C.F.G.S DAM Y DAW .ENTORNOS DE DESARROLLO**

**TEMA 4. Control de versiones**

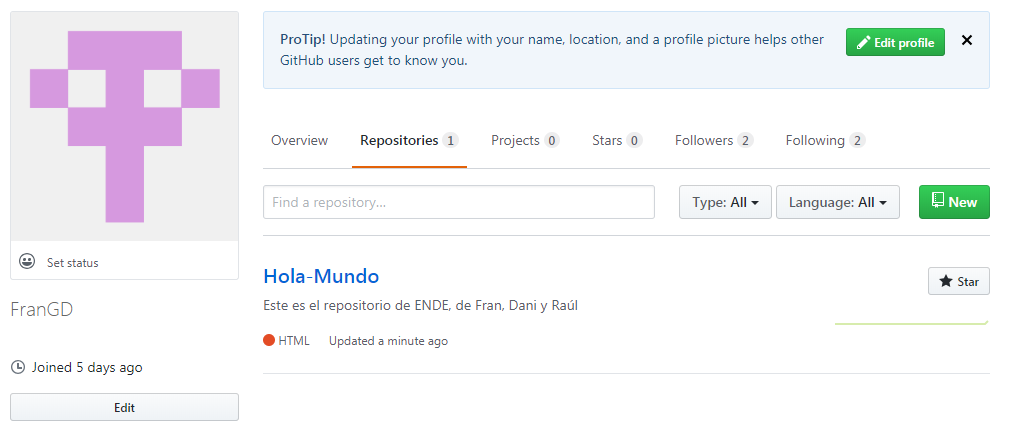
## **PARTE 2** (Para realizar esta segunda parte debes entender bien los conceptos de la primera parte)

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:

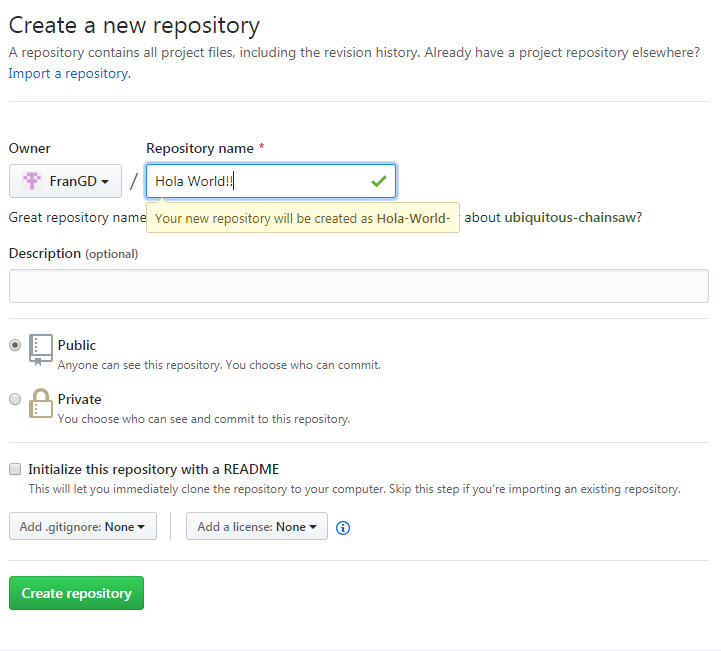
Lo primero que necesitaremos será crear una cuenta de Github donde crearemos los repositorios para ello accedemos a github.com y pulsamos en la opción de sign up e introducimos nuestros datos.

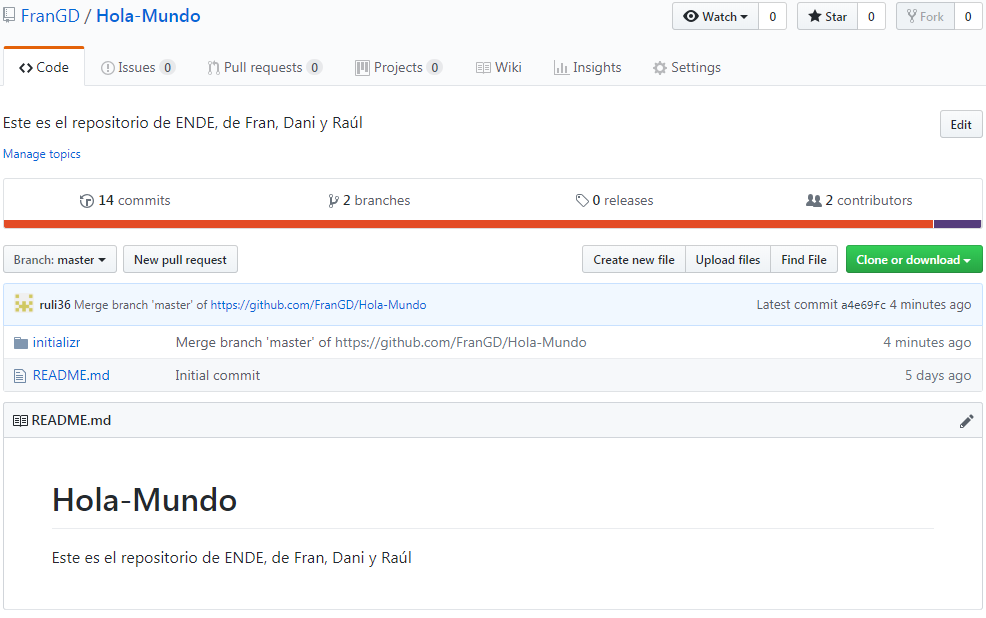


Una vez creada la cuenta crearemos nuestro primer repositorio para ello en página del perfil creado y accedemos a la pestaña Repositories. En la pestaña pulsamos en new.

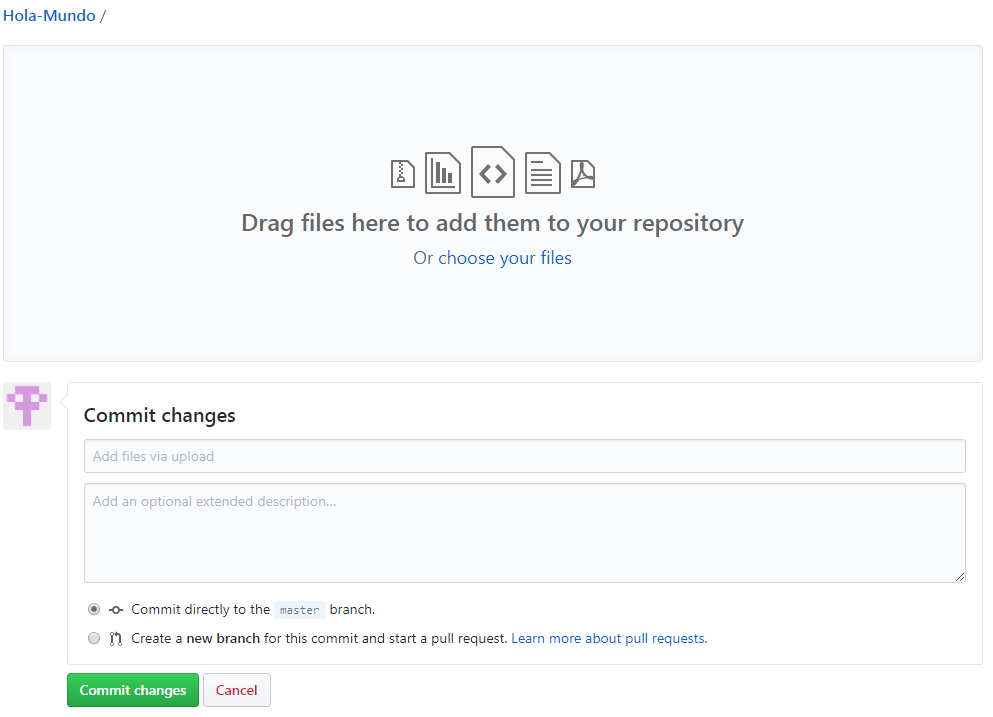


La nueva ventana nos pedirá el nombre del repositorio, una descripción opcional y si queremos que este sea publico o privado.

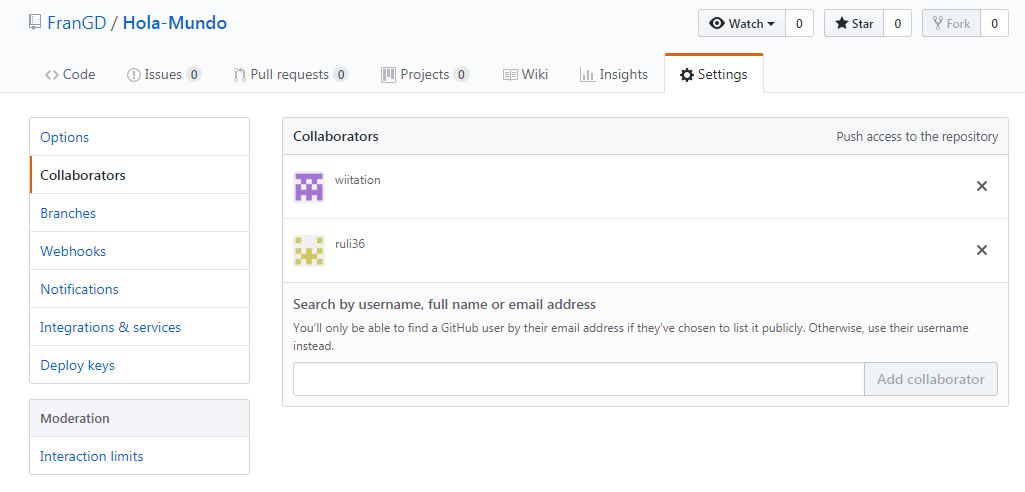


Dentro del repositorio pulsamos en la opción upload files donde se nos preguntará por el archivo que deseamos subir.

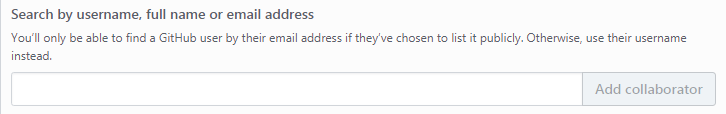
Arrastramos el archivo y pulsamos commit.



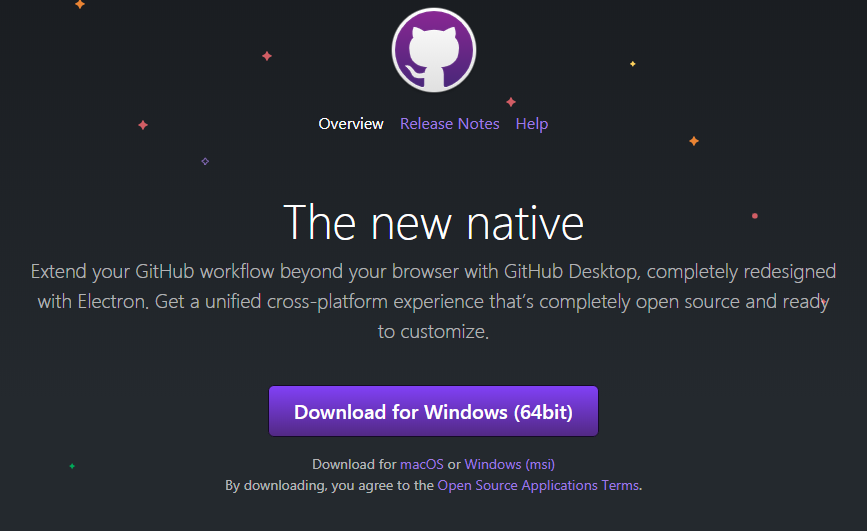
Para añadir un colaborador seleccionamos la pestaña settings en la ventana del repositorio. Abajo a la izquierda en la lista de opciones pinchamos en Collaborators.



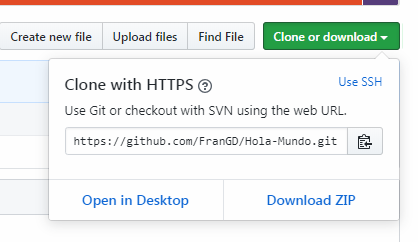
Buscamos a nuestro compañero de proyecto y lo añadimos.



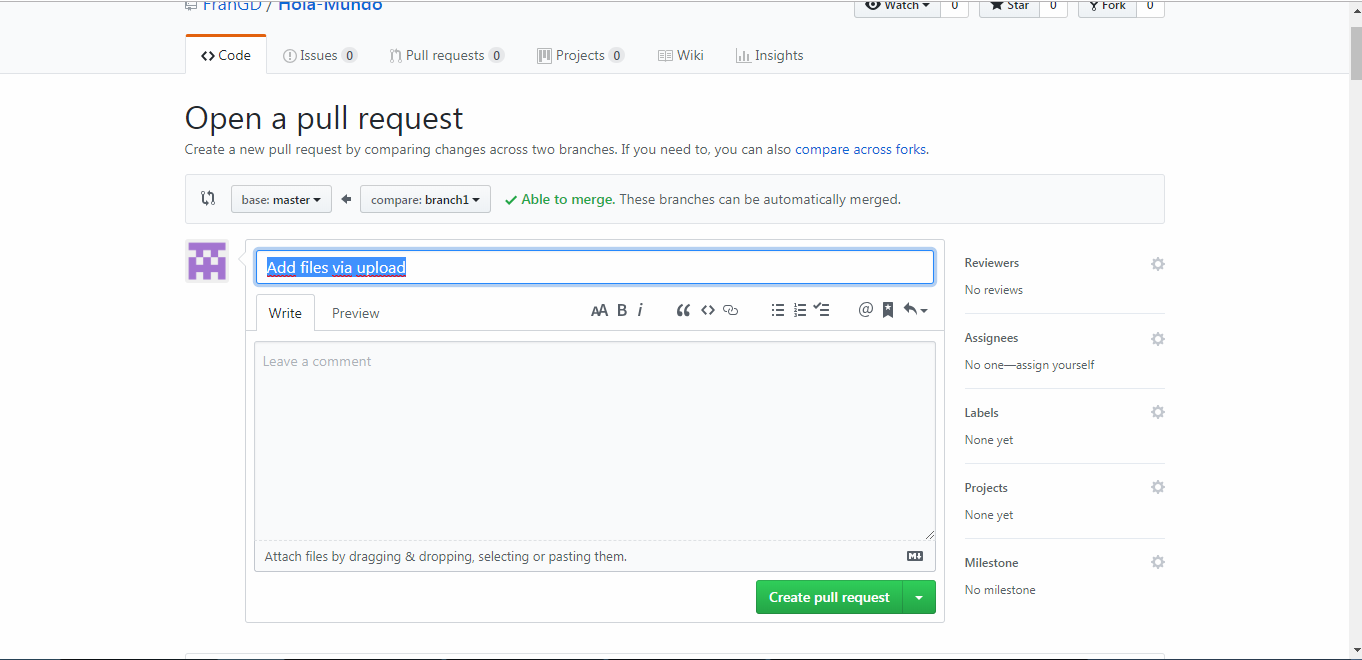
Para descargar Github desktop nos dirigimos al siguiente enlace: <https://desktop.github.com/> aquí encontraremos la versión desktop de GitHub y procederemos con la descarga.

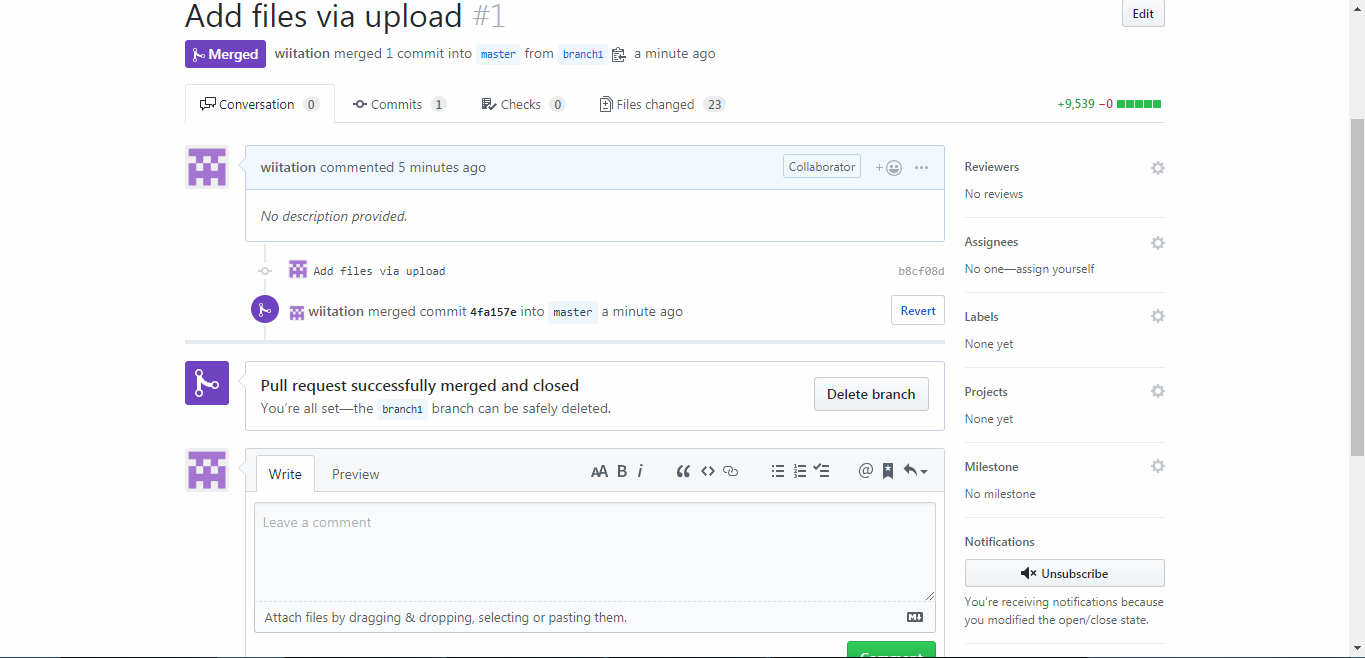


Para realizar un clone en nuestro PC solo tendremos que pulsar el botón clone or download que tenemos dentro del repositorio.



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

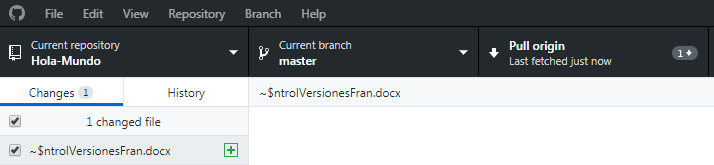




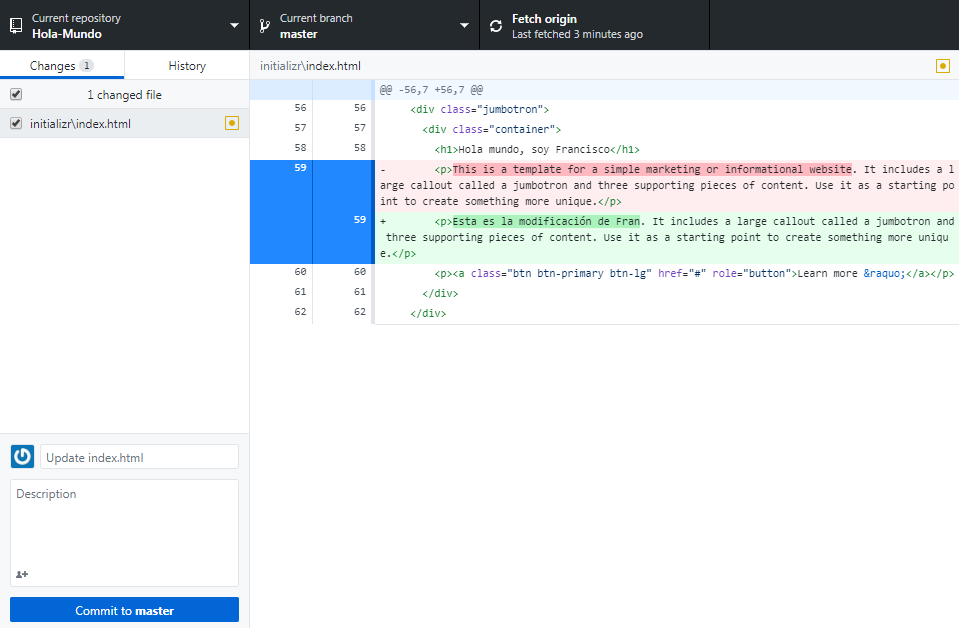
Finalmente, con el comando push lo subimos al repositorio.



En las siguientes capturas se muestra que los colaboradores tienen una copia del proyecto en su equipo local. Para ello refrescamos la página con fetch y luego pulsamos en Pull origin situado arriba a la derecha lo que descarga la ultima versión en nuestro equipo



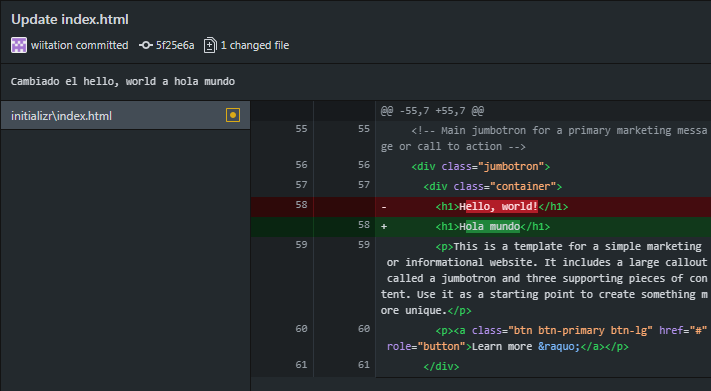
Los miembros del grupo modifican archivos diferentes, como se muestra en la captura los diferentes cambios realizados aparecen en la pestaña changes de la izquierda, debajo tendremos los diferentes archivos que se modificaran en los que podremos navegar y ver sus diferentes cambios. Mas abajo pedirá en nombre de la versión y una breve descripción de los cambios realizados.



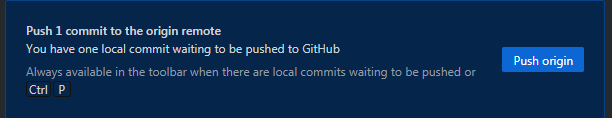
Despues de completar esta información solo quedaría pulsar sobre Commit to master y hacer un push origin.



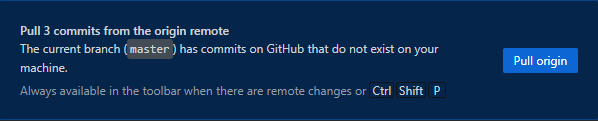
En el siguiente ejemplo podemos el nombre de la versión y su descripción tras hacer el commit.



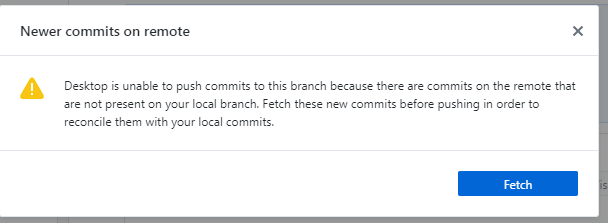
Realizamos un push para subirlo al repositorio.



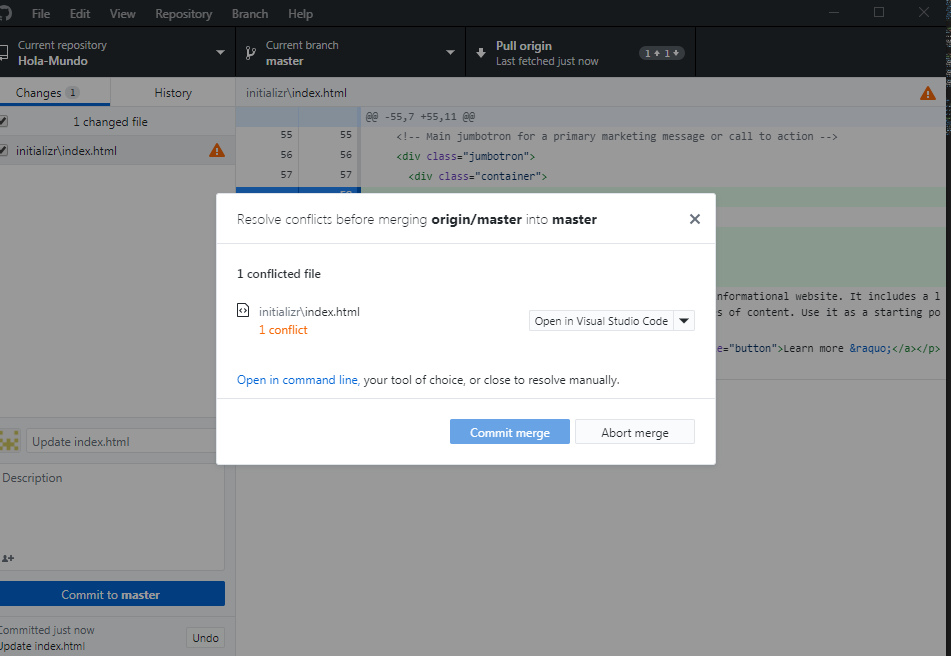
Descargamos de nuevo las modificaciones realizadas en el proyecto en nuestro equipo mediante pull origin.



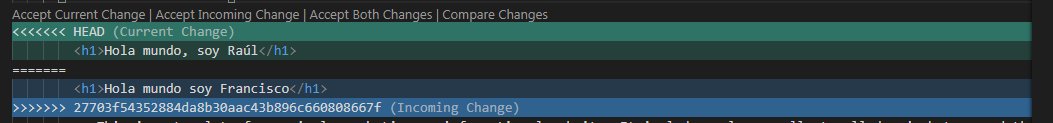
Modificamos el mismo archivo con un compañero en la ultima versión, hacemos commit y push. Nos notifica que se ha producido un conflicto.



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 sobrescribe la version subida.
* Accept Incoming change: Acepta el cambio y sobrescribe 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.