



# MANUAL GITHUB

Que es y aprendizaje del control de versiones

# INDICE

1. Introducción.
2. Que es un control de versiones.
3. Como crear un repositorio paso a paso.

## 1. Introduccion.

Este manual va a contener una guía para conocer que es y como usar un control de versiones. En este caso el control de versiones se realizará con GitHub. Además de GitHub se enlazará un ide de java para ver como enlazar el control de versiones y proyectos al ide. El Ide de java que se va a utilizar es el IntelliJ de JetBrains. También se utilizara la App de GitHub desktop para la comunicación entre GitHub y el resto.

## 2. Que es un control de versiones.

El control de versiones es la gestión y seguimiento de proyectos. Ayudan a mantener un seguimiento intensivo sobre el software que se modifica en los diferentes proyectos. Su funcionamiento es similar a un gran árbol, la parte mas importante es la llamada rama principal que es la suele tener toda la información común sobre el proyecto, y luego están llamadas las ramas del proyecto que son las diferentes versiones desarrolladas base la rama principal. Este desglose permite que la rama principal pueda servir de “backup” en el caso en el que alguna versión se corrompa o similar. La ramificación de las versiones luego permite que luego se pueda incorporar la versión que mas sea acorde con el equipo a la rama principal creando de esta manera un acople de esa rama sobre la principal. Es como en el árbol se escogiera la “rama” que queremos seguir y el resto o las mantenemos o las cortamos. De esta manera se va “controlando” de una forma ordenada que información y que cambios se van incorporando a el proyecto y se guarda también una pequeña red para por si acaso alguna parte del proyecto se pierda.

Hay 3 operaciones básicas en un control de versiones:

**Pull:** tal y como dice el nombre, es la operación que “trae” el proyecto de la rama seleccionada. Básicamente es para recuperar la información del repositorio para luego usarla en el id o donde fuere.

**Push:** al igual que la anterior, esta es la traducción literal de “empujar”. En este caso lo que ocurre es que se envía la información que tenemos a la rama seleccionada. Al igual que Pull trae la información, luego con Push la enviamos al repositorio.

**Comit:** esta es la tercera operación del control de versiones. Para mi opinión es la mas importante de todas por que es literalmente un pequeño comentario y un titulo de los archivos que se van a hacer en el Push. Es decir, aquí es donde hay que ser minucioso por que es el resumen del rastro que se va a ir dejando de nuestras modificaciones en el repositorio.

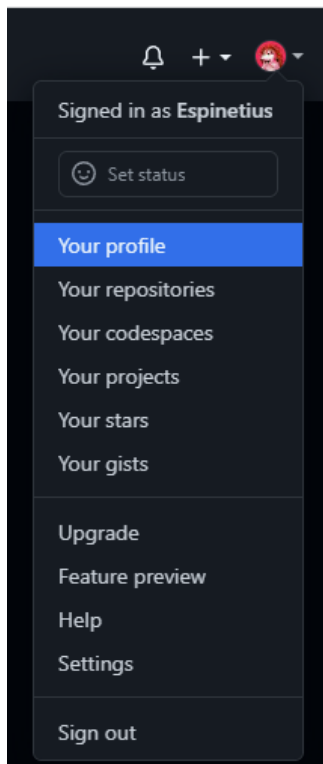
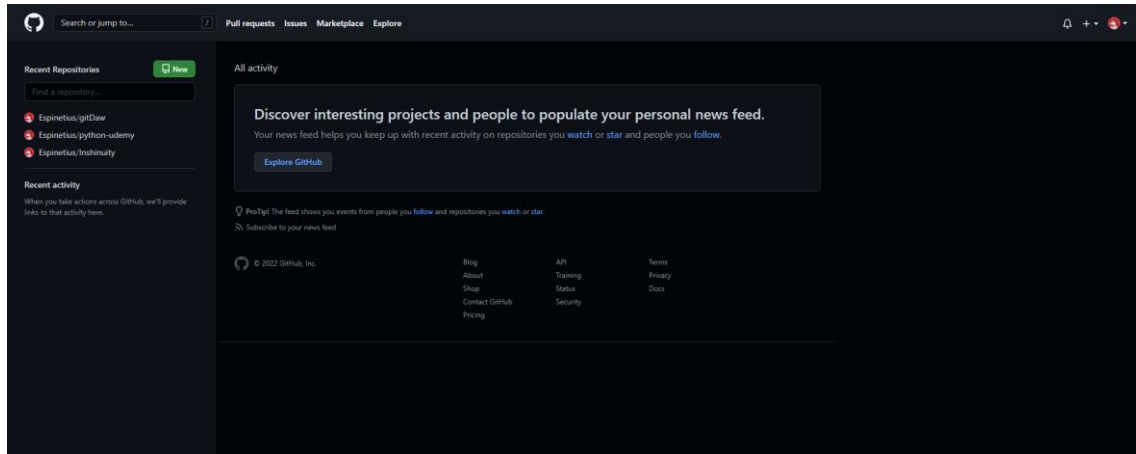
Usando estas tres operaciones podríamos empezar a usar un control de versiones. Es importante realizar las operaciones en el “orden” correcto para evitar el mal funcionamiento o la sobreescritura errónea de los archivos. Primero hay que hacer un Pull, para recuperar la información del repositorio y así ver si ha habido algún cambio o similar. Luego, al finalizar el trabajo hay que hacer el Comit y detallar bien en el cuadro de texto lo que se ha modificado o el paso realizado. Y por último hacer el Push en la rama indicada. No es recomendable hacer los Push directamente en la rama principal por lo explicado anteriormente de que la rama principal es una “red de seguridad” frente a corrupciones de los archivos. Lo recomendable es hacer los Push en una rama de desarrollo y luego realizar una operación llamada “Merge”. Esta operación lo que hará será trasladar los archivos seleccionados de la rama en la estaban a la rama principal.

La mayoría de los repositorios funcionan en inglés, de ahí que tengamos las operaciones o las partes traducidas al idioma. Las ramas se llaman “Branch”, la rama principal se llama “main” y el resto serian las operaciones ya explicadas (pull, push, comit, merge).

### 3. Como crear un repositorio paso a paso.

En este manual el repositorio que utilizare será GitHub, por lo que el primer paso es la creación de una cuenta en GitHub si no se tiene una ya.

Una vez tenemos nuestra de GitHub creada y hemos logeado en ella, veremos la siguiente pantalla:



A continuación, entraremos en el perfil, o directamente en los repositorios para poder ver los proyectos donde se trabajará.

La siguiente imagen corresponde directamente a los repositorios que tenemos en la cuenta. Se van a ver tres repositorios los cuales están identificados con un nombre. En este caso los tres repositorios son gitDaw (donde tengo todos los archivos del curso), Python-udemy (donde guardo los avances de un curso online de la plataforma de Udemy) y por ultimo un pequeño proyecto personal de una app web.



Para crear un repositorio nuevo se seleccionara en el botón de color verde que pone “New”.


Una vez pulsado ese botón, se abrirá un menú de configuración del repositorio en cuestión: el nombre que tendrá, la descripción, si será publico o privado (visibilidad al resto de la plataforma) y por ultimo 3 opciones que son añadir un “Readme” donde se podrá ver mas detallada la descripción del repositorio, “add.gitignore” donde se iran añadiendo archivos que no se quieren subir al repositorio (por ejemplo el id que estes utilizando o librerías que no afecten al proyecto, o en ocasiones un zip con los archivos originales del proyecto), y la ultima para elegir la licencia del proyecto. Por defecto se seleccionara una licencia gratuita (yo no cambiaria esta opción).

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---


**Owner \*** **Repository name \***


 Espinetus /

Great repository names are short and memorable. Need inspiration? How about [musical-adventure?](#)

**Description (optional)**

---

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

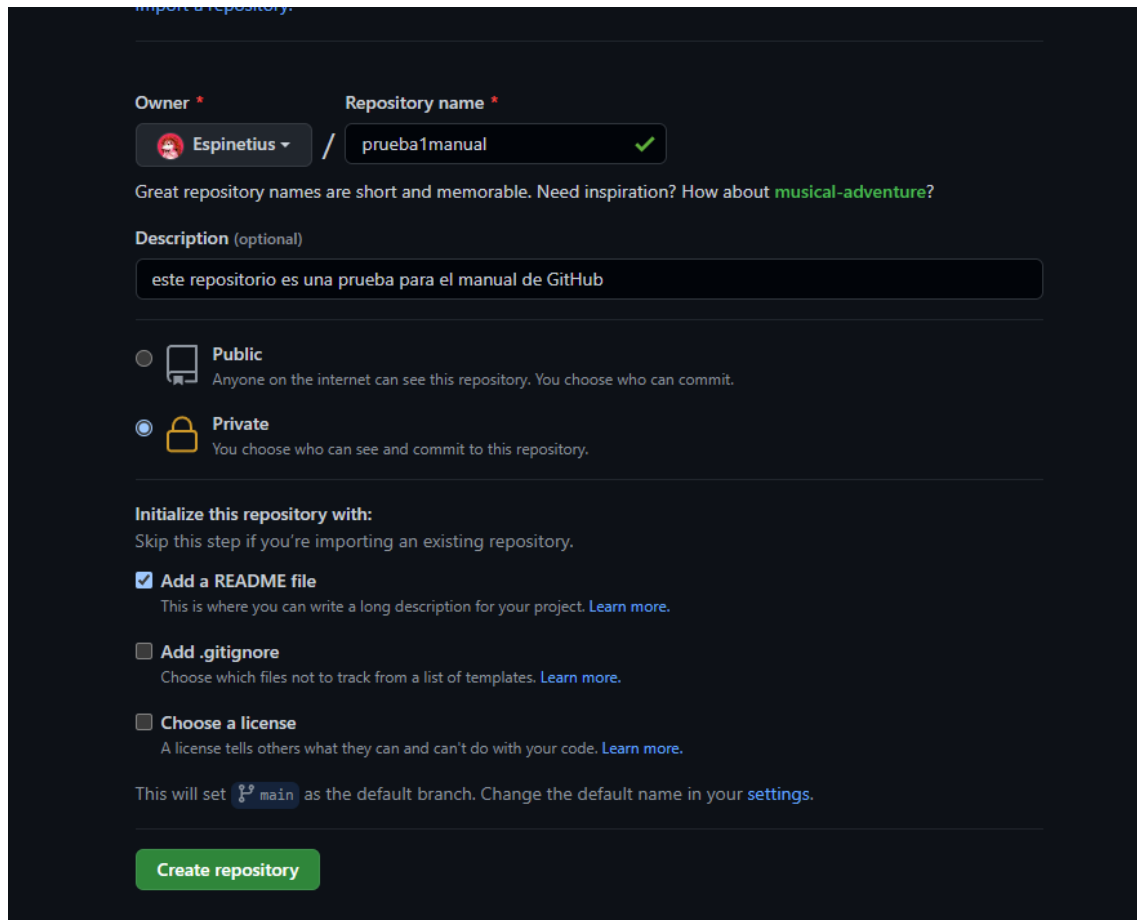
☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

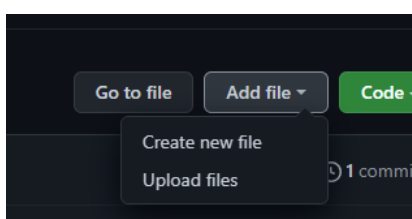
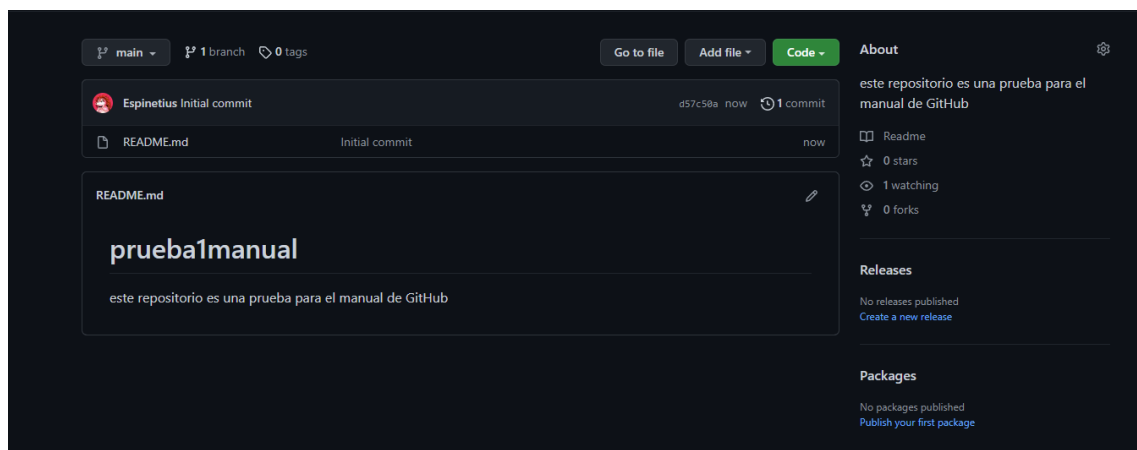
[Create repository](#)

Para hacer un ejemplo crearemos un repositorio de prueba que llamaremos prueba1manual. A continuación, se agrega un pantallazo con las opciones escogidas:



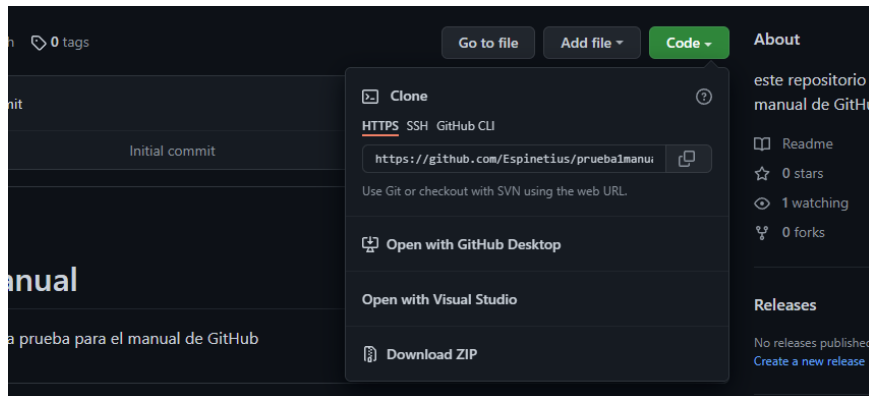
The screenshot shows the GitHub repository creation interface. At the top, there's a section for 'Owner' (Espinetius) and 'Repository name' (prueba1manual). Below this, a description field contains 'este repositorio es una prueba para el manual de GitHub'. The 'Public' option is selected, and the 'Private' option is also visible. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. Other options like 'Add .gitignore' and 'Choose a license' are unchecked. A green 'Create repository' button is at the bottom.

Tal y como se ve en la imagen se han escogido las opciones de privado y que se agregue un readme.

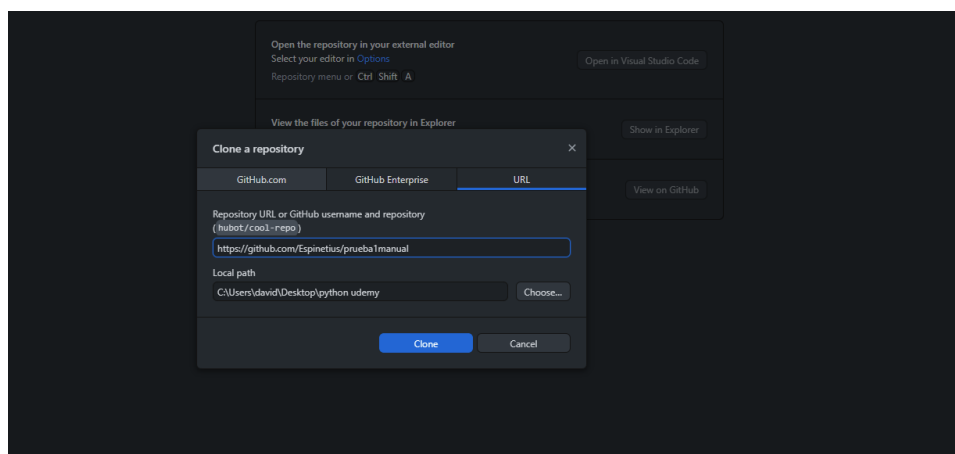


Una vez creado el repositorio se pueden añadir ficheros que se tengan para crear el principio del proyecto en la opción de "Add File". Aquí se puede subir un zip con la información del proyecto o crear los archivos de 0.

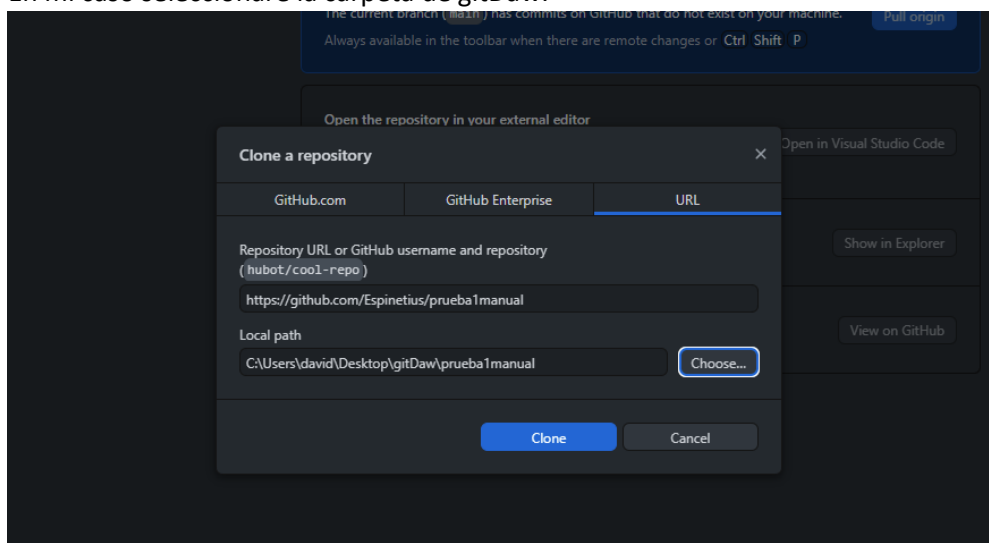
Ahora viene el momento de enlazar el repositorio en la app de GitHub desktop. Tal y como se ve en la siguiente imagen, al pulsar en el botón verde de “Code” se podrá ver una opción que es “Open with GitHub Desktop”. Al pulsar esa opción automáticamente se nos abrirá la aplicación. Si no se tiene la aplicación descargada se recomienda descargarla, es mas sencillo que de la forma tradicional.



Una vez seleccionada esa opción se nos abrirá automáticamente la ruta en GitHub desktop, y podremos ver la siguiente pantalla:

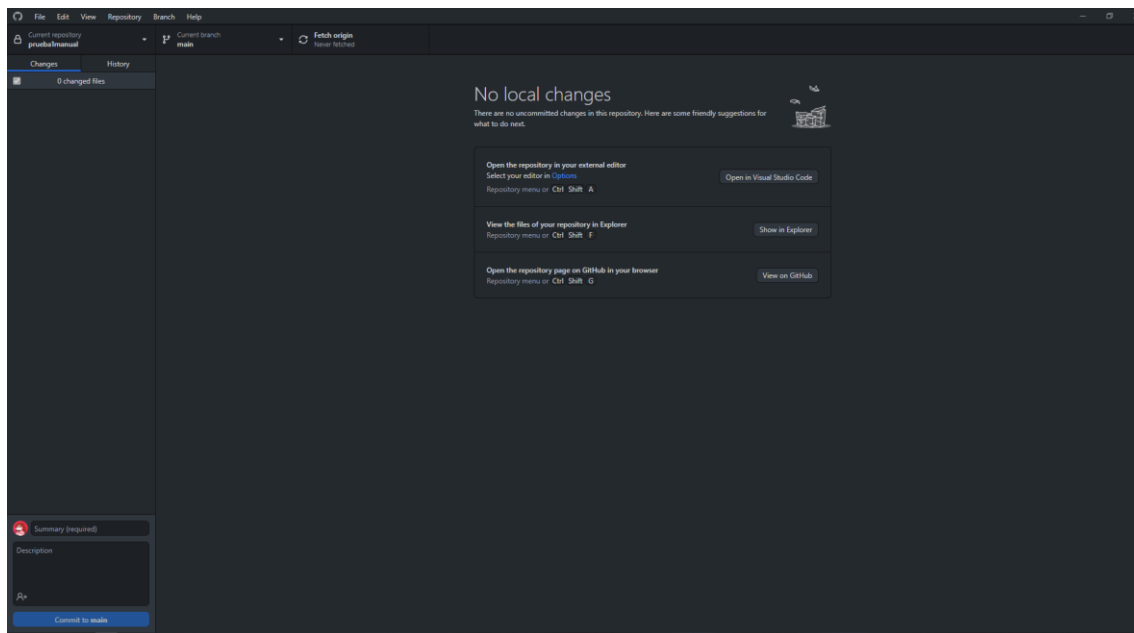


Tal y como se ve en la imagen te pide una ruta local para depositar los archivos del repositorio. En mi caso seleccionar la carpeta de gitDaw.

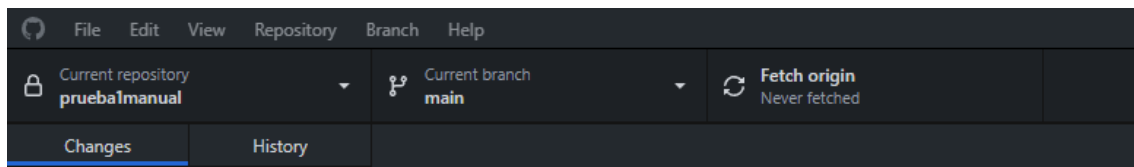




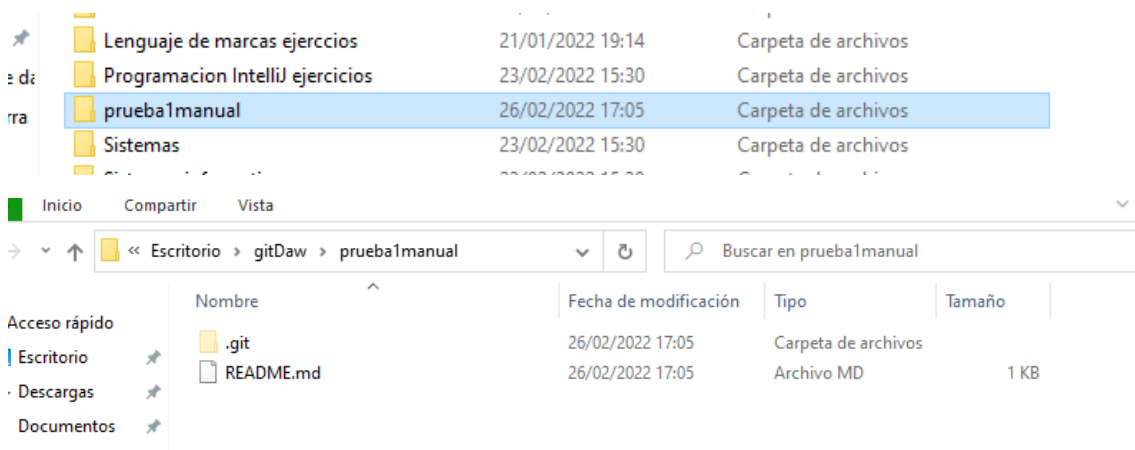
Una vez seleccionado la aplicación nos devolverá la siguiente pantalla:



En la barra superior se puede ver el repositorio en el que se está, la rama en la que se está trabajando y una opción llamada fetch. Esta operación es un chequeo de los archivos web comparándolos con los archivos locales.

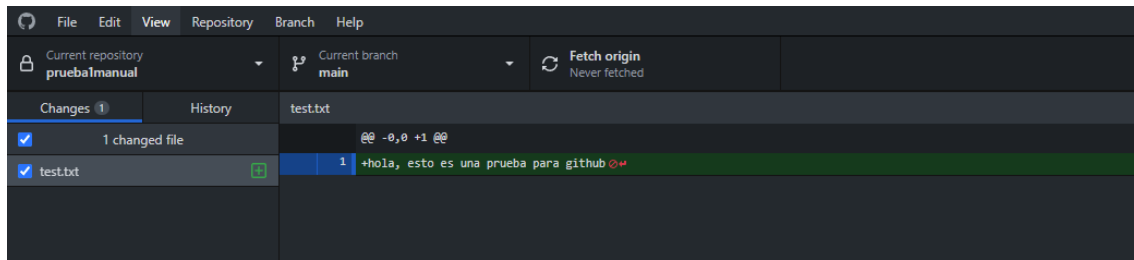


Ahora si abrimos el explorador de archivos y buscamos la ruta local podremos ver que se nos ha creado una carpeta en nuestro ordenador. Si realizamos cambios en la carpeta la aplicación de GitHub nos mostrara los cambios realizados.



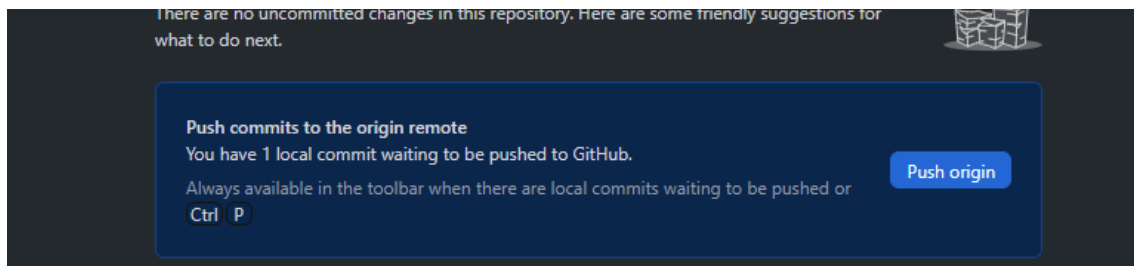
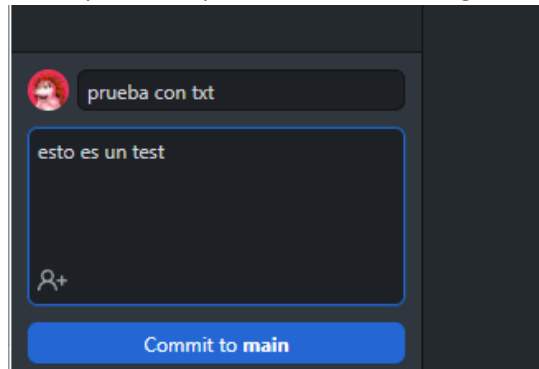
Para ver como reacciona a los cambios se creara un bloc de notas con una frase de prueba.

Automáticamente la aplicación nos reconoce los cambios realizados en la carpeta:

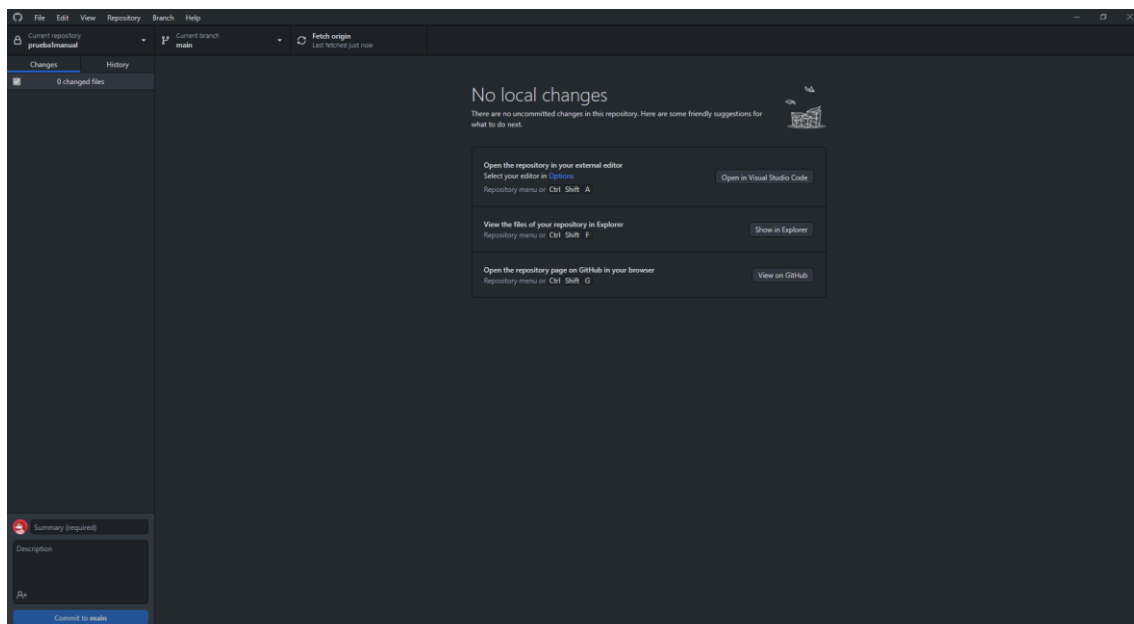


Y nos muestra en verde los cambios añadidos. En rojo nos mostrara los eliminados.

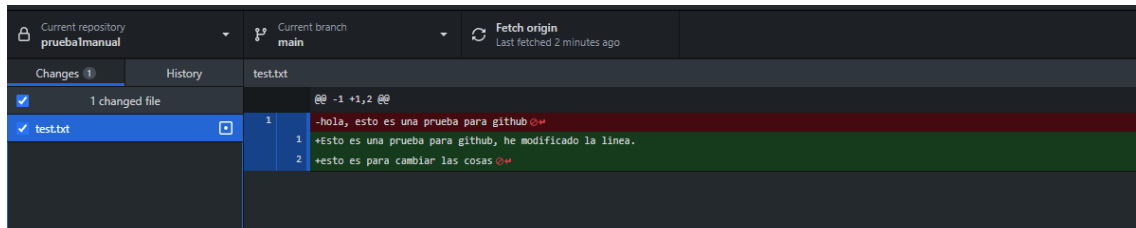
Ahora realizaremos las operaciones básicas descritas en el apartado anterior. Primero haremos un comit con una descripción de que se ha hecho. Y luego realizaremos un push.



Una vez realizados los cambios, se vera que la lista de cambios ha desaparecido:



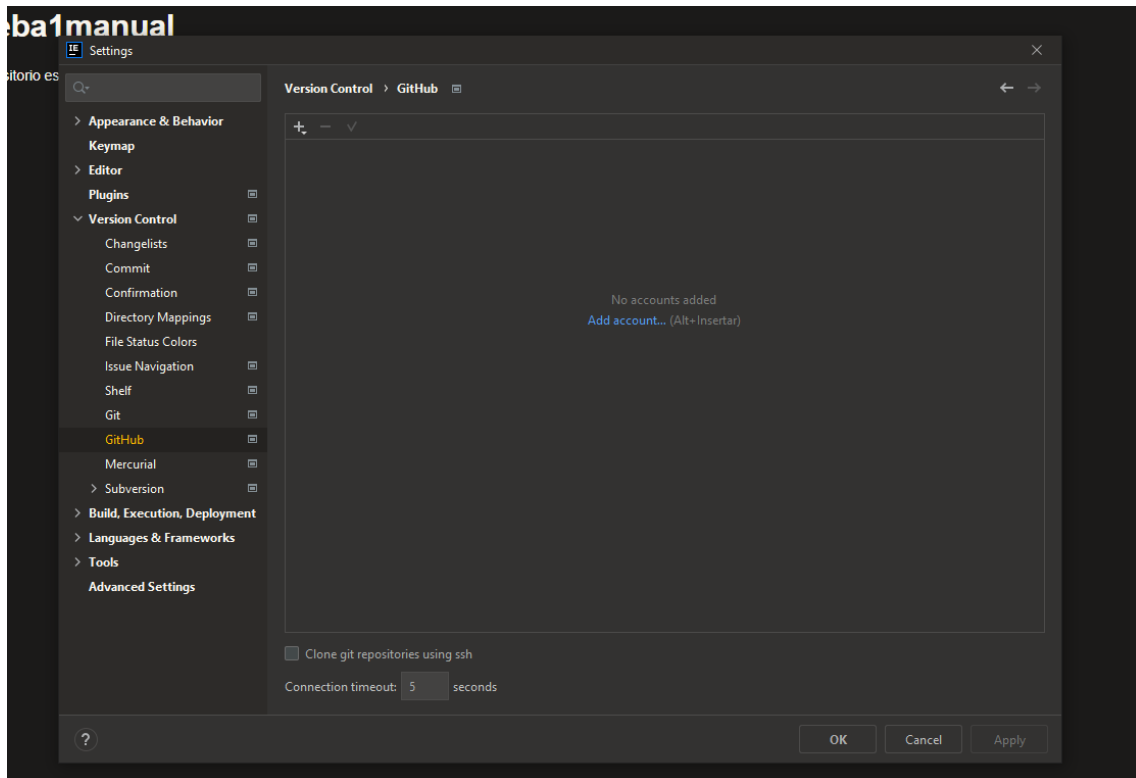
Ahora nos meteremos en la carpeta y vamos a modificar un poco el documento de texto para ver como reacciona la app.



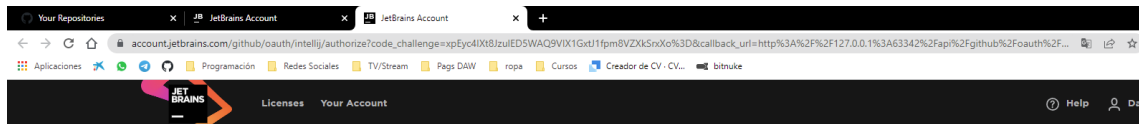
Tal y como se ve la app reconoce los cambios y nos los muestra en pantalla. En rojo lo eliminado, y en verde lo añadido.

Ahora enlazaremos git al ide que utilizaremos. En este caso estaremos utilizando Intel J de JetBrains.

Lo primero a realizar será abrir el entorno como tal, y luego ir a los ajustes en File/Settings. Una vez ahí en la barra lateral hay que desplegar la opción de versión control y de ahí seleccionaremos la opción de GitHub.

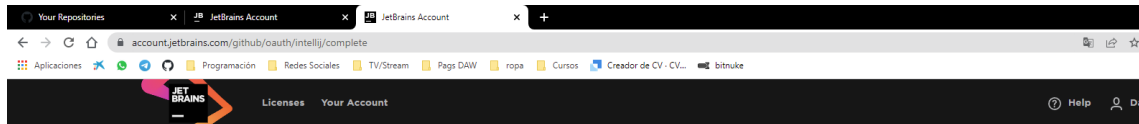


Luego seleccionaremos la opción de Add account y seleccionaremos la opción de logear via Github. Se abrirá una pag al navegador donde nos pedirá si autorizamos nuestra cuenta de JetBrains a nuestra cuenta de GitHub y le daremos a aceptar.



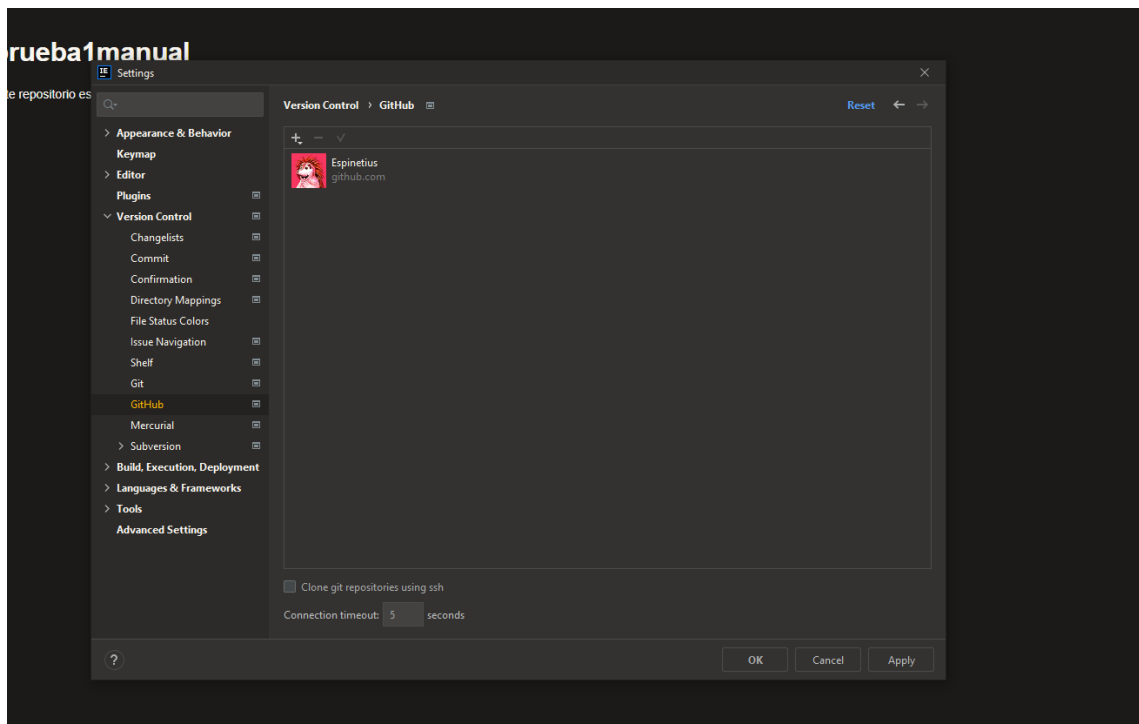
Please continue only if this page is opened from a [JetBrains IDE](#).

[Authorize in GitHub](#)

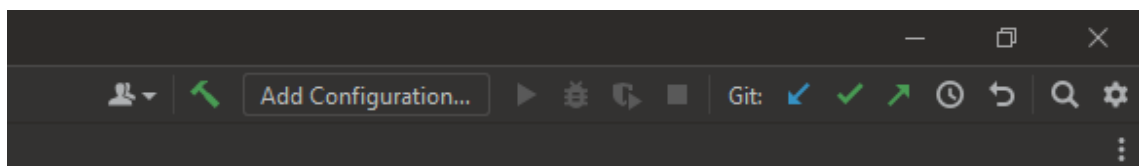


You have been successfully authorized in GitHub. You can close the page.

Una vez aceptada ya estará enlazada, y si volvemos al entorno podremos ver que nuestra cuenta esta introducida en el.

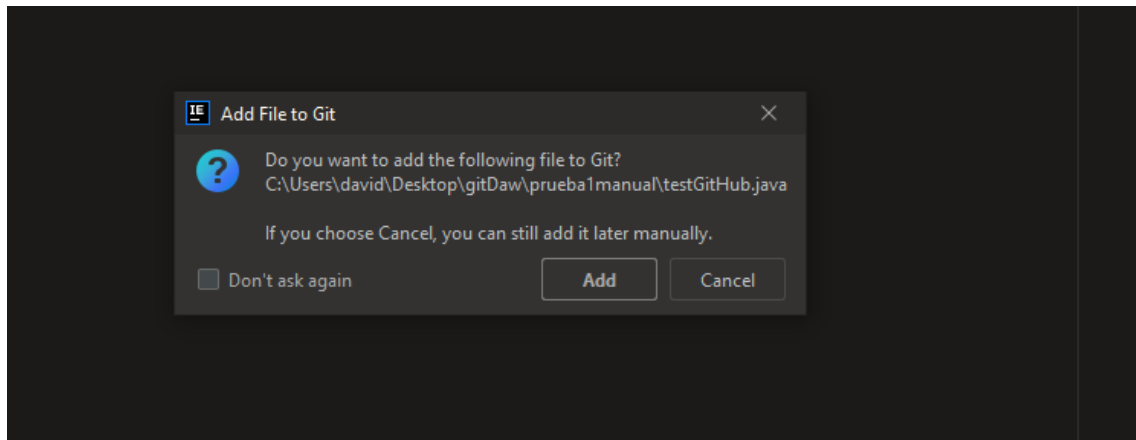


Manejar el control de versiones en Inteli J es bastante sencillo. Se puede ver en la esquina superior derecha varias “flechitas” de colores verdes o azules, y estas son las operaciones del control de versión.

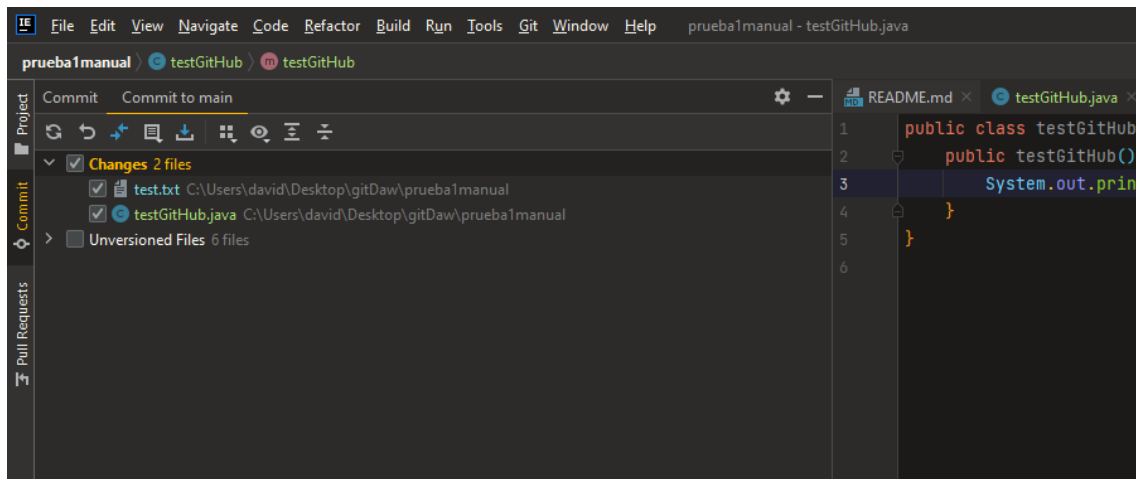


Vamos a crear un archivo java para ver como realizar los comit y los push en el id.

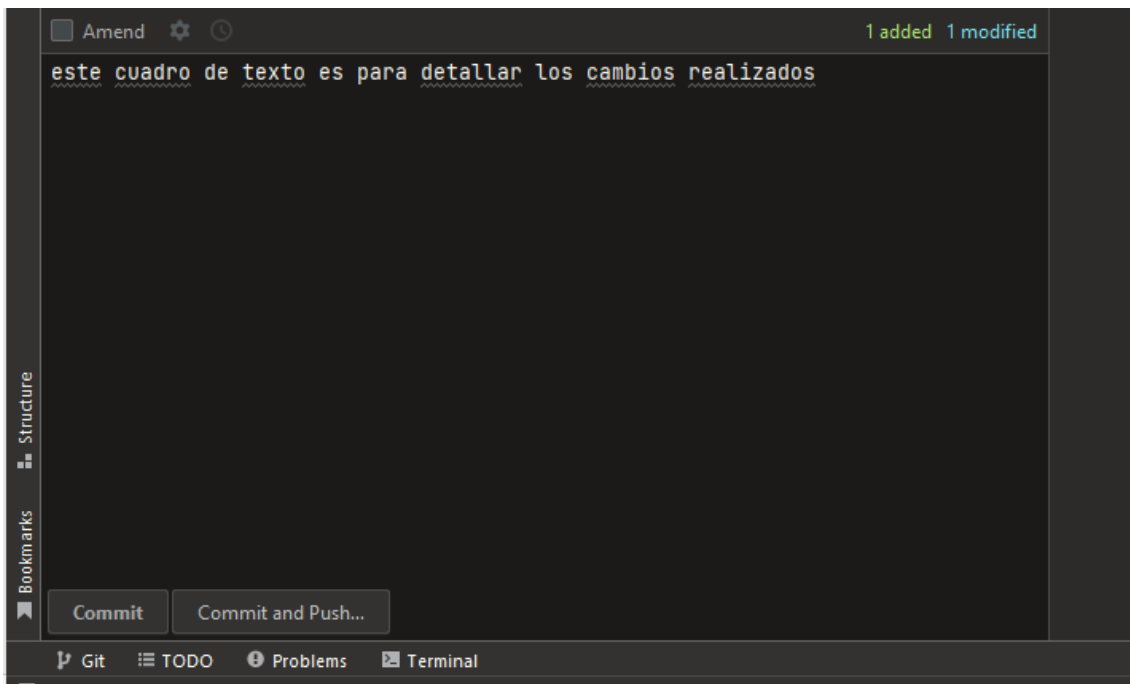
Cuando creamos cualquier archivo nuevo en el proyecto nos aparecerá un mensaje en el id preguntándonos que, si queremos incorporar un nuevo archivo al Git, le daremos a Add para añadirlo.



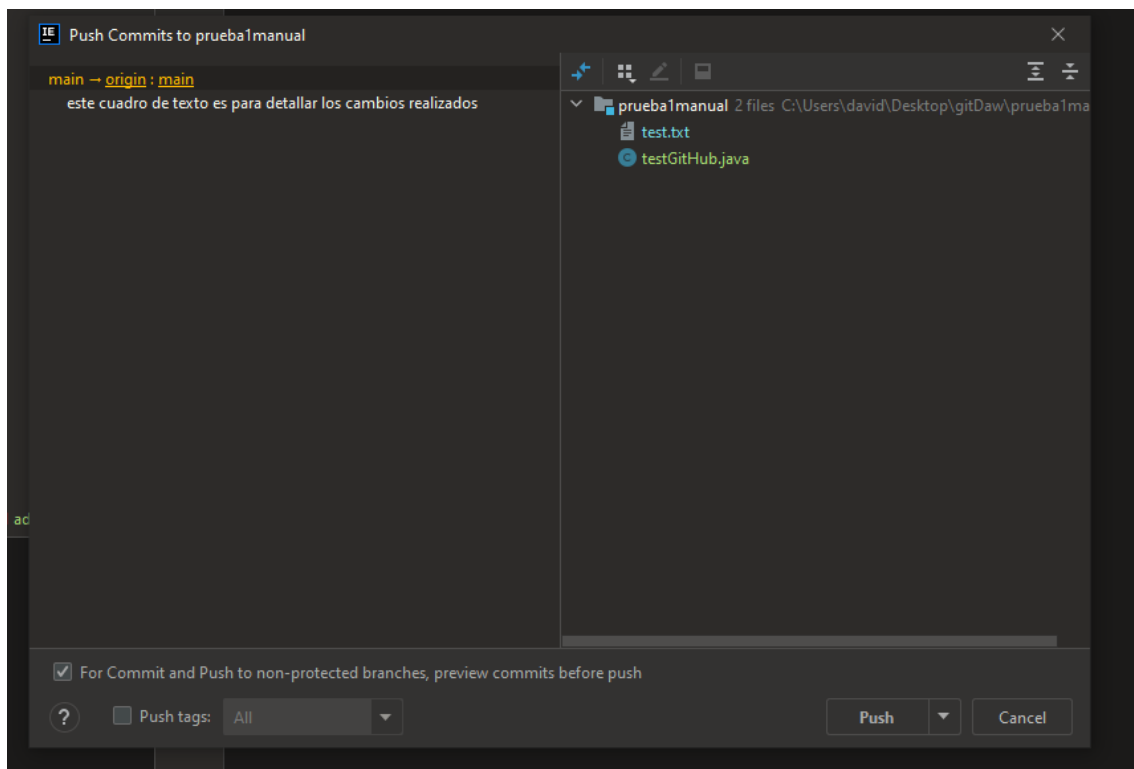
Una vez realizados los cambios en el código, de las 3 flechitas de antes primero seleccionaremos la verde que esta mas a la izquierda, es decir, la que si dejamos el ratón encima vemos que sale un mensaje que pone comit.



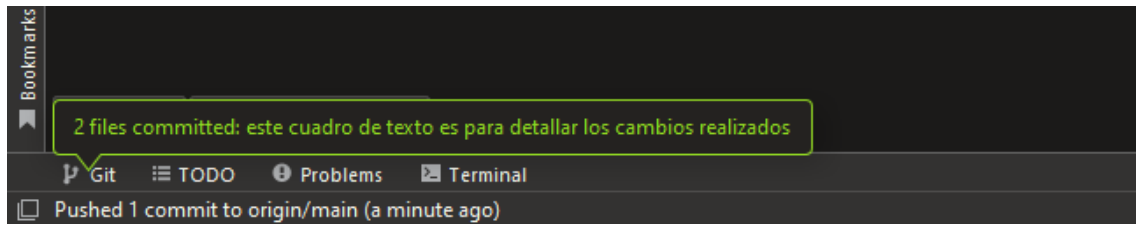
Aquí nos saldrá una lista de los archivos que han sido modificados y están seleccionados para hacer el comit, y justo debajo se vera un commit message que es donde escribiremos la descripción de los cambios realizados.



Ahora seleccionaremos la opción de commit and push, y esto nos abrirá una ventana de push para ver que se está haciendo el push y ver los cambios. Aquí es donde se puede seleccionar la rama a la que quieres hacer el push.



Una vez que se presione la opción push, se podrá observar como carga la barra del id y nos sale un mensaje de git con los cambios hechos commit y pusheados.



Si se hace el push desde el id no es necesario hacer el push desde la aplicación de GitHub desktop, es mas, en la aplicación nos saldrán que los cambios ya han sido confirmados y subidos al repositorio.