

1000 Programadorxs

>Introducción a la Programación con Python 2023

módulo 5

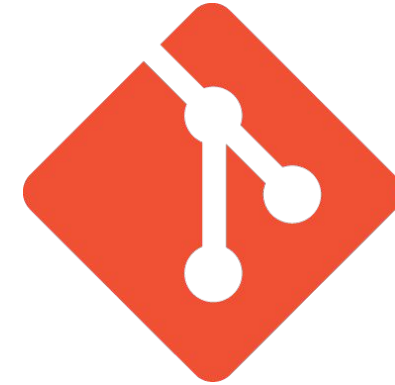
>Git en GitHub y Visual Studio Code

Introducción

En la clase pasada descubrimos cómo **gestionar** las versiones de nuestro código utilizando **Git**.

Este sistema por sí solo trabaja de manera **local** en nuestra computadora.

Ahora, para sumar el aspecto **colaborativo** de la herramienta necesitamos sumar una **plataforma en línea**.



Introducción

Existen muchas plataformas web que trabajan con el sistema de **control de versiones**. La más popular es **GitHub**, que nos permite alojar **código fuente** de forma gratuita en la **nube**.

GitHub funciona como una **red social** y nos brinda acceso a **repositorios públicos** de forma ilimitada.



Creación de Cuenta



El primer paso para utilizar **GitHub** es crear una **nueva cuenta** en donde podremos crear y alojar nuestros repositorios.

Visitamos la página oficial **github.com**. Aquí presionamos el botón **Sign Up** y seguimos los pasos para la creación, comenzando por el ingreso de nuestro email.

Welcome to GitHub!
Let's begin the adventure

Enter your email*

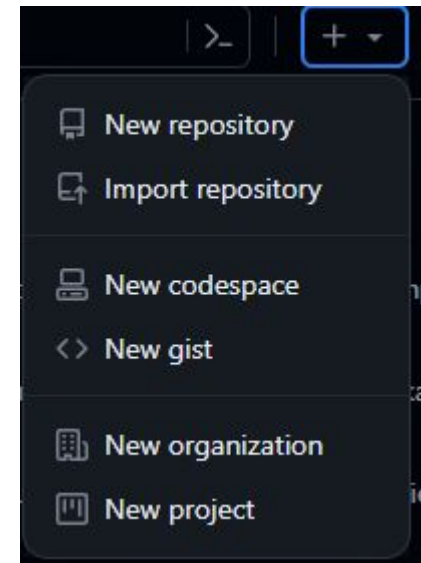
→ Continue

Página de Inicio de GitHub

Una vez creada la cuenta llegaremos a la [página principal](#) de [GitHub](#). El siguiente paso es crear nuestro [primer repositorio](#).

Arriba a la derecha tenemos varios [accesos directos](#) para empezar a trabajar. El que nos interesa explorar es el [signo +](#), donde vemos las siguientes opciones.

Para comenzar con nuestro proyecto elegimos [New repository](#).

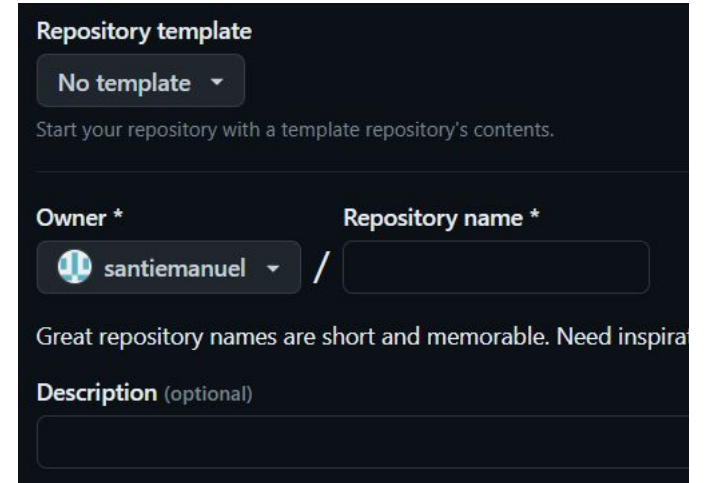


Creación del Primer Repositorio

En esta página veremos sus **secciones** una a una comenzando por el inicio. No usamos **plantillas** así que podemos omitir la primera opción.

A continuación tenemos el **propietario** (Owner) del **repositorio**. En este caso aparecerá **nuestra cuenta** actual.

Luego de la barra debemos darle el **nombre** a nuestro repositorio. Este será el nombre del **proyecto** que deseamos alojar en GitHub.

A screenshot of the GitHub repository creation interface. At the top, it says 'Repository template' with a dropdown menu set to 'No template'. Below this is a text input field for the repository name. The 'Owner' is set to 'santimanuel' with a GitHub icon. The 'Repository name' field is empty. A hint text says 'Great repository names are short and memorable. Need inspiration?'. At the bottom, there is a text input field for the 'Description (optional)'.

Repository template

No template

Start your repository with a template repository's contents.

Owner * Repository name *

santimanuel /

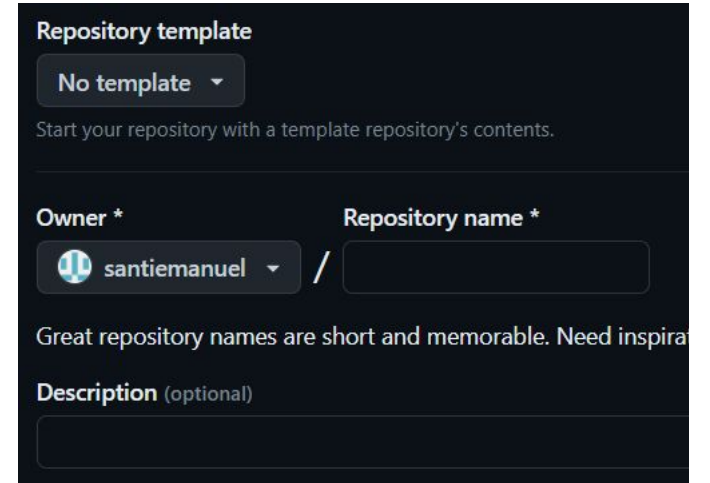
Great repository names are short and memorable. Need inspiration?

Description (optional)

Creación del Primer Repositorio

Si bien la descripción es **opcional**, se recomienda utilizarla para dar los **detalles más importantes** del proyecto en el repositorio.

Esta aparecerá en los **resultados de búsqueda** en GitHub y ayudará a aumentar las probabilidades de que nuestro repositorio **aparezca** entre los resultados.

A screenshot of the GitHub repository creation interface. At the top, it says 'Repository template' with a 'No template' dropdown. Below that, it says 'Start your repository with a template repository's contents.' The main form has two sections: 'Owner *' with a dropdown showing 'santimanuel' and a 'Repository name *' text input field. Below these, it says 'Great repository names are short and memorable. Need inspira'. At the bottom, there is a 'Description (optional)' text input field.

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * Repository name *

santimanuel /

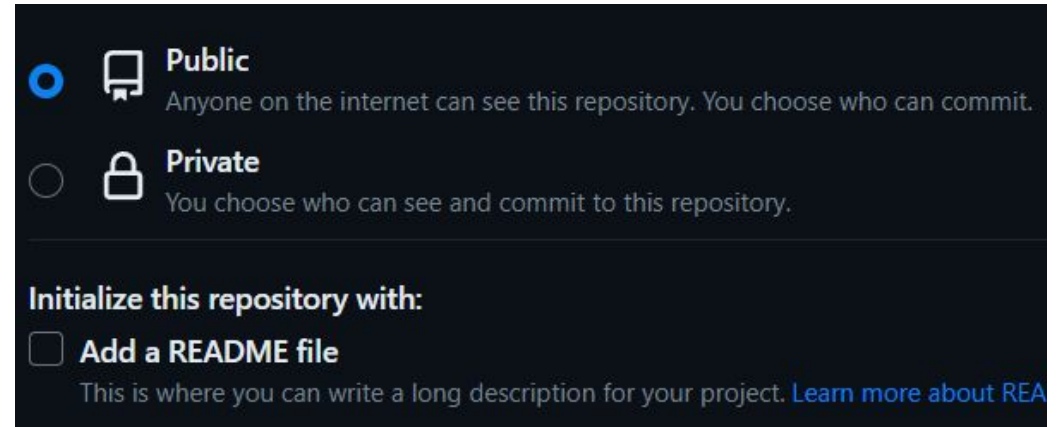

Great repository names are short and memorable. Need inspira

Description (optional)

Privacidad del Repositorio

La siguiente sección nos solicita indicar si el repositorio será **público** (accesible sin permisos) o **privado** (sólo por invitación). Para este primer **repositorio** podemos crearlo privado para realizar pruebas.

El archivo **README.md** es una buena práctica para **documentar** nuestro proyecto. Podemos marcar la opción pero también es posible crearlo **manualmente** en el futuro.



The screenshot shows the GitHub repository creation interface. The 'Private' option is selected, indicated by a blue circle next to the lock icon. The 'Public' option is unselected, indicated by a grey circle next to the open lock icon. Below the options, there is a section titled 'Initialize this repository with:' with a checkbox for 'Add a README file' which is currently unchecked. The text below the checkbox states: 'This is where you can write a long description for your project. [Learn more about README files](#)'.

☒ **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:

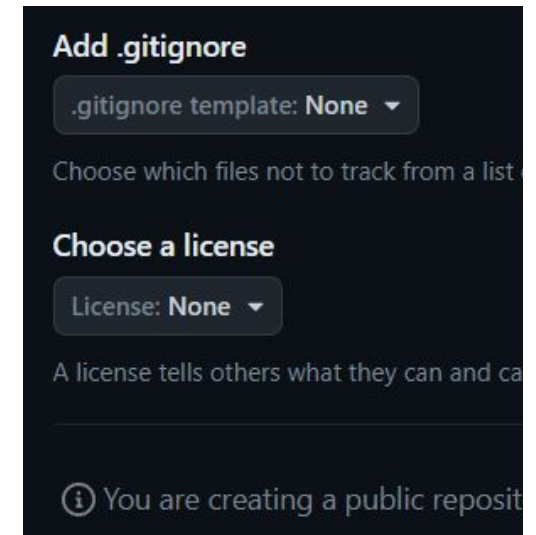
☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about README files](#)

Licencia y .gitignore



El archivo **.gitignore** es muy importante en la **creación** de nuestro proyecto. Al trabajar **localmente**, es posible que nuestro **entorno de desarrollo** o el mismo proyecto cree archivos al **ejecutar** el programa.

Estos archivos pueden ser un resultado de la **compilación del código** o información **sensible** para acceder a **bases de datos** u otras herramientas con **autenticación**.

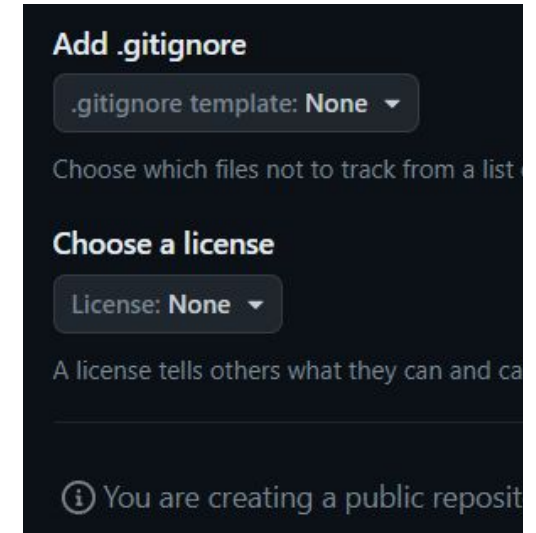


Licencia y .gitignore



Para evitar que estos archivos se **suban** a nuestro repositorio, podemos añadirlos en **.gitignore** para que... sí, git los ignore. Tenemos la posibilidad de **ignorar**, archivos, carpetas, tipos de archivos y otras herramientas avanzadas.

Para empezar podemos elegir la **plantilla Python** en la lista desplegable, que incluye todos los archivos que se generan al trabajar y ejecutar código.

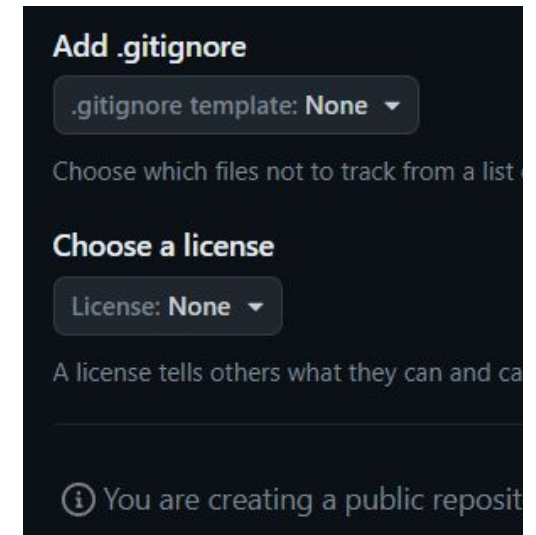


Licencia y .gitignore



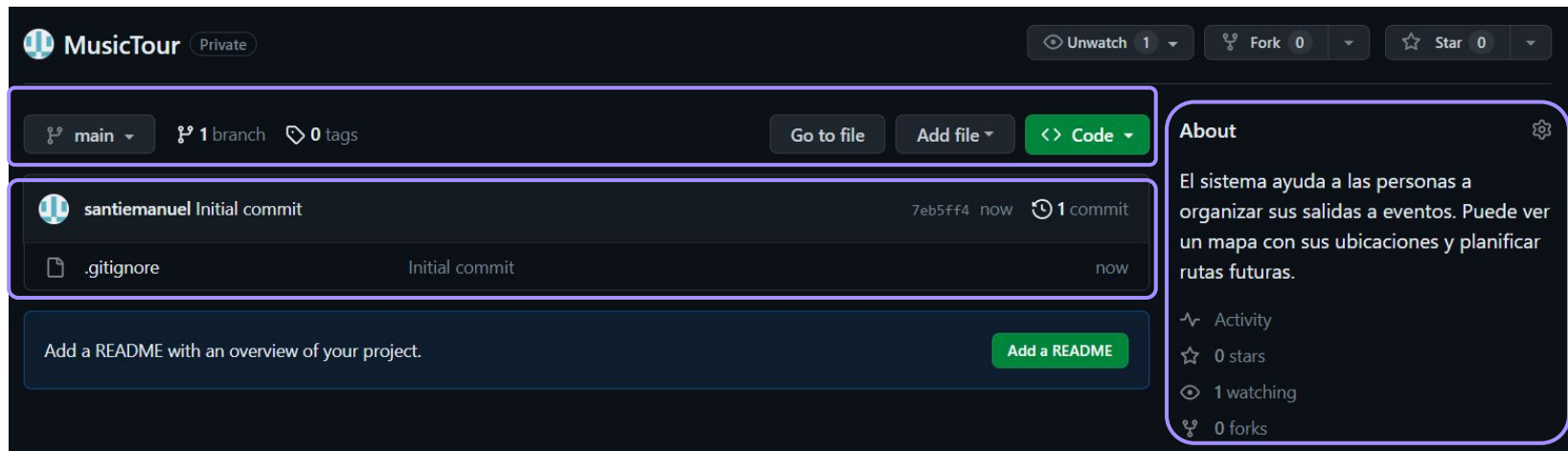
Por otro lado, la **licencia** delinea lo que otras personas pueden hacer con **nuestro código**, para el nivel que estamos viendo actualmente no es **necesario** elegir una licencia.

Si deseamos obtener más información sobre las licencias, tenemos disponible el sitio <https://choosealicense.com/> recomendado por **GitHub** para decidir la que nos conviene.



Vista Inicial de un Repositorio

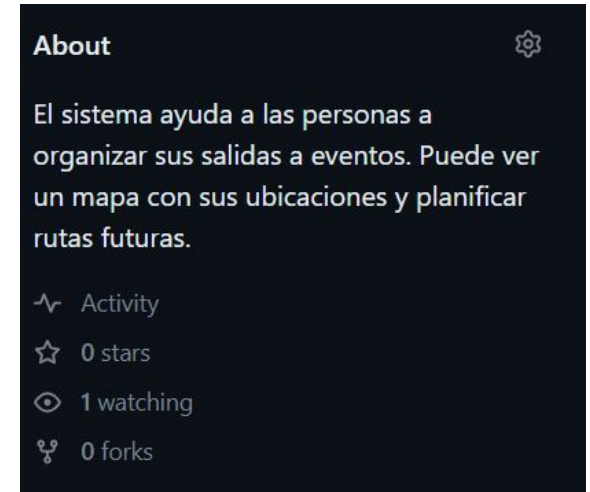
Al crear el repositorio nos redirigirá a la [página](#) del mismo. Aquí podemos ver la [información principal](#) de manera resumida.



Descripción

A la derecha vemos la **descripción** que cargamos luego de elegir el nombre del repositorio. Esto da un **resumen** de lo que trata el proyecto. La información adicional y detallada la incluiremos en el archivo **README.md**.

Luego tenemos la actividad, que es un **resumen** de los commits realizados, **estrellas** que recibió el proyecto en GitHub, y personas que lo están **siguiendo**.



Barra Superior



Debajo del nombre del repositorio tenemos información vinculada al **sistema Git**.

Primero, vemos la **rama** del repositorio que estamos viendo actualmente. Al empezar, todo repositorio **comienza** con la rama **main** y es la única que podemos ver actualmente.

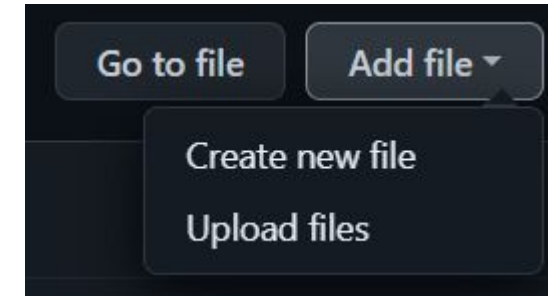
A la derecha vemos la **cantidad de ramas** que tiene nuestro repositorio.



Barra Superior - Botones



El botón **Add file** nos permite **subir** o **crear** archivos nuevos en el repositorio. Es el equivalente de usar **git add** y luego **commit** y **push** en la consola.



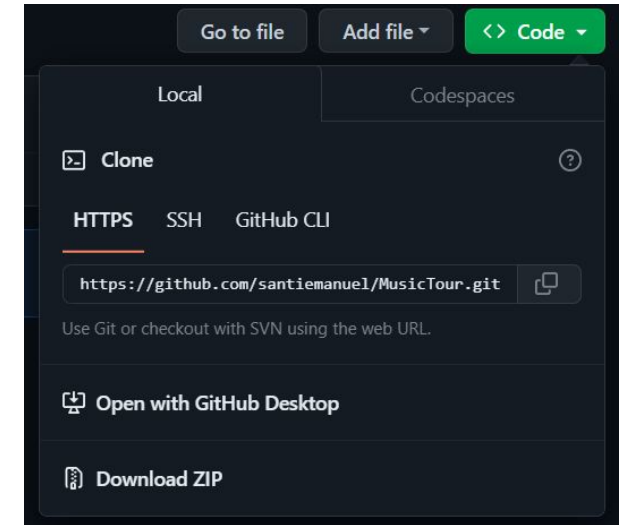
En nuestro caso no los utilizaremos porque vamos a trabajar en el **entorno de desarrollo** que se ocupará de facilitar estas tareas.

Barra Superior - Botones

El último botón, **Code**, nos permite **clonar** el repositorio en una ubicación local.

Al clonar un **repositorio**, lo que hacemos es **descargar todos** sus archivos así como su directorio **.git** que incluye toda la información de los **cambios** realizados, **ramas**, **autores**, y cualquier otra tarea la cual quede **registrada** en el mismo.

Esto es útil si deseamos clonar un **repositorio público** y luego contribuir al mismo con **mejoras** o **correcciones**.






Navegador de Archivos



Por último, tenemos una **tabla** en donde aparecerán todos los **archivos** de la **rama actual** del repositorio. Aquí vemos el **nombre** del archivo, el **mensaje** del commit que agregó este archivo al repositorio, y **la hora** en que se realizó.

Arriba de la tabla veremos el **último** usuario que realizó un commit al **repositorio** con el mensaje del commit, y a la derecha la **cantidad de commits** realizadas hasta el momento.

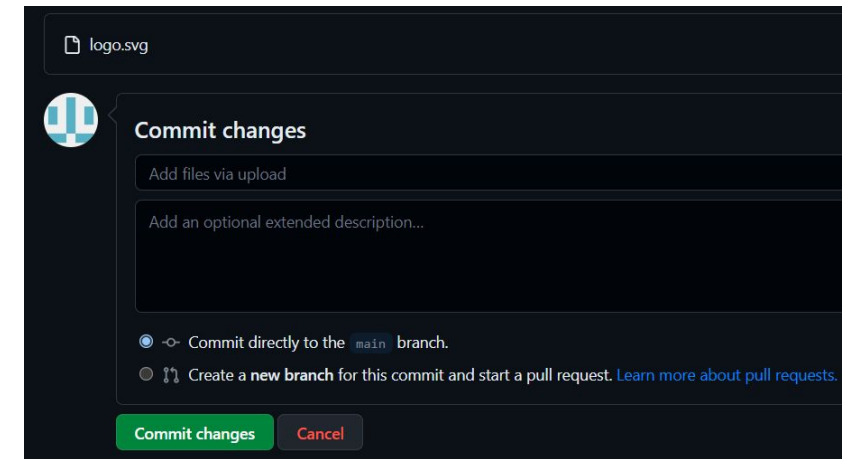
 santimanuel Initial commit	7eb5ff4 18 minutes ago	 1 commit
 .gitignore	Initial commit	18 minutes ago

Subir un Archivo

Si vamos a subir un archivo desde el botón **Add file** nos encontramos con la siguiente vista.

Aquí nos solicita la **información** para el commit a realizar para **añadir** el archivo. Debajo de la descripción tenemos un **selector** para elegir la rama de destino **main** o crear una rama nueva con un **Pull Request**.

La segunda tarea la veremos en detalle al trabajar con **Visual Studio Code** en la próxima clase.



Trabajo en Visual Studio Code



Para trabajar en el repositorio de manera **local**, debemos **descargarlo** a nuestra computadora. Esta acción se denomina **clonar** el repositorio.

Previamente debemos contar con una **carpeta** en donde guardamos nuestros proyectos. Abrimos esta carpeta con **Open Folder**, por ejemplo, sea la carpeta **python-workspace**, abrimos la misma.



Trabajo en Visual Studio Code



Ahora abrimos una terminal con **Terminal -> New Terminal** y, por ejemplo, escribimos el siguiente comando en la consola:

```
git clone https://github.com/santiemanuel/MusicTour.git
```

Este link lo tenemos disponible en el botón **Code** de GitHub como vemos.

Aquí usaremos el repositorio que creamos en **pasos anteriores** porque el repositorio de ejemplo es privado.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

Cargar los perfiles personales y de sistema tardó 1986ms.
(base) PS C:\Users\Santiago> |
```

Trabajo en Visual Studio Code



El repositorio se **descargará** en una carpeta con su **nombre**. Aquí la primera vez que lo usemos nos pedirá **completar** nuestra información de la cuenta.

Ahora debemos hacer **Open Folder** nuevamente para ubicarnos en el **directorio** del proyecto.

```
PS C:\vscode-workspace\proyect-music-tour> git clone https://github.com/santiemanuel/MusicTour.git
Cloning into 'MusicTour'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
Receiving objects: 100% (6/6) used 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), 413.05 KiB | 942.00 KiB/s, done.
```

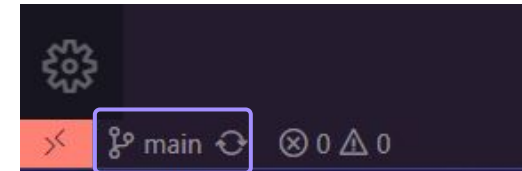
Trabajo en Visual Studio Code



Aquí vemos los archivos que **subimos** a nuestro repositorio incluyendo a **.gitignore** que se añadió en la etapa de **creación** del repositorio.



Al fondo de la ventana visualizamos la **rama actual** en la que nos encontramos. Todos los cambios que realizamos impactarán **aquí**.



Trabajo Local en el Proyecto



Desde aquí, cualquier **cambio** que realicemos dentro del proyecto será **detectado** por VS Code y veremos **indicadores** que señalan el tipo de **modificación**.

Tenemos **tres tipos** de cambios que se indican de diferentes formas:

- Creación
- Modificación
- Eliminación

Creación



Al **crear** o **copiar** un archivo al proyecto, su nombre se marcará con una **U** a la derecha. Esto significa **Untracked** (un archivo sin seguimiento).

Modificación



Cuando **editamos** un archivo y guardamos los cambios **localmente**, notaremos que aparece una **M** a la derecha. Esto significa **Modified** (un archivo modificado).

Al abrir un archivo **modificado**, VS Code nos **resalta** con colores a la izquierda las **líneas** que tuvieron modificaciones para poder **ubicarlas** rápidamente.

Eliminación

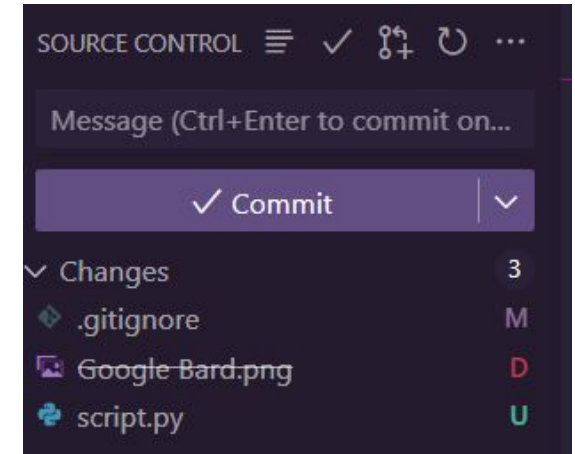


Al **eliminar** un archivo del proyecto, este desaparecerá. Veremos más información sobre esto en la pantalla de **control de versiones** de código.

Control de Código Fuente

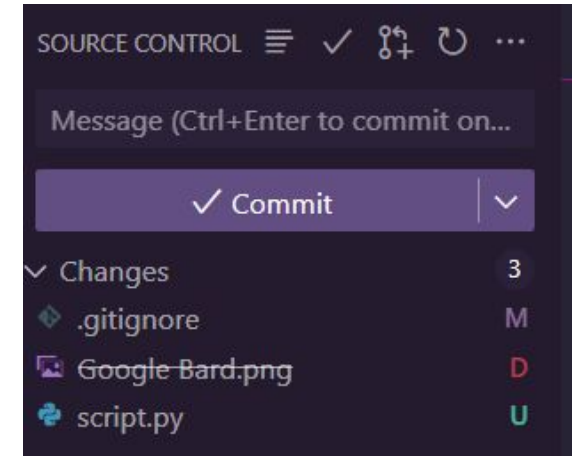
Todas las acciones vinculadas al **seguimiento** de cambios las encontraremos en la pestaña **Source Control** ubicada debajo de la función de **búsqueda**.

Aquí se listará todas las **modificaciones** que realizamos al proyecto desde que lo clonamos.



Control de Código Fuente

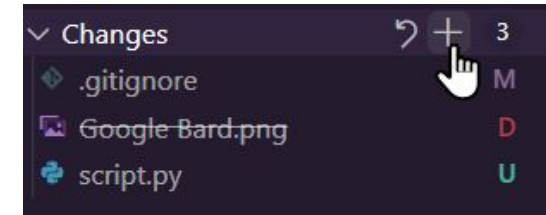
En la imagen vemos ejemplos de los **3 casos** que se pueden dar al trabajar en un proyecto. La **eliminación** de archivos aparece con una **D** (**Deleted**, eliminado).



Etapa de Staging



Antes de confirmar los cambios debemos **añadirlos**, el equivalente a **git add**.



Con el signo + en **Changes** podemos enviar todos los cambios a **Staging**, pasando el cursor por archivo podemos hacerlo individualmente pero por simplicidad añadiremos **todos** los cambios.

Etapa de Staging



Luego de añadirlos, veremos cómo aparecen **dos** categorías:

- **Staged Changes:** Los preparados para **confirmar**.
- **Changes (o unstaged):** Cambios sin preparar, que actualmente no hay ninguno. Si modificamos algo **luego** del staging, estos aparecerán aquí.

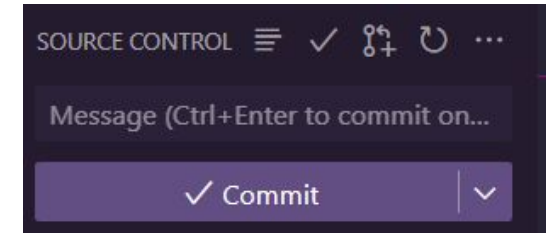


Confirmación de Cambios



El último paso es **confirmar** los cambios que tenemos en staging.

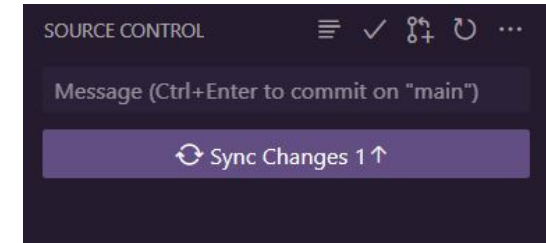
Primero escribimos el mensaje del commit en el cuadro **Message**, y presionamos **Commit**, lo equivalente a `git -m "Mensaje escrito en Message"`.



Subida de los Cambios



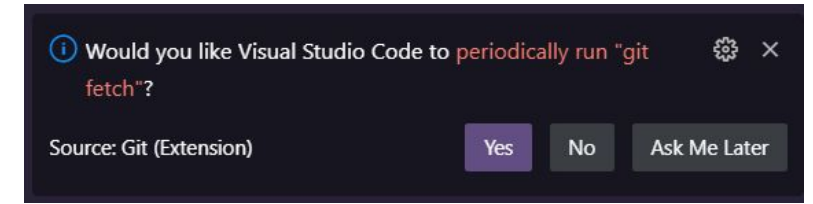
Luego de pulsar **Commit**, veremos que el botón cambia a “**Sync Changes**”. Esto en terminología Git es el comando `git push`, que sube los cambios al **repositorio remoto**. VS Code se encargará de subir los cambios a **GitHub**.



Subida de los Cambios

Al finalizar la operación **desaparecerá** todo lo que veíamos en **Source Control** y ahora el repositorio remoto tiene los cambios que realizamos **localmente**. Visual Code nos dará la siguiente **sugerencia**.

La podemos ignorar dado que realizaremos el **seguimiento** de cambios por otros **medios** o puede ser un proyecto **personal** el cual solo nosotros subiremos cambios al **repositorio remoto**.



Próximos Pasos



Hasta aquí vimos cómo trabajar con un repositorio remoto en [GitHub](#) y luego en [Visual Code](#). Con estas herramientas se puede trabajar en una rama principal (“[main](#)”) y actualizar el repositorio con [commits](#).

En la segunda parte veremos una metodología para trabajar con [ramas por características](#), y una vez completadas las ramas hacer uso de los [Pull Requests](#) para organizar el avance del proyecto.

**muchas
gracias.**