

MANUAL DE GITHUB 2ª PARTE

Contenido

1.INTRODUCCIÓN	1
2.DESCARGA DE Git	1
3.CREAR CUENTA EN GitHub	3
4. ENVIO DE ARCHIVOS A UN REPOSITORIO GitHub	6
PROCESAMIENTO DE ARCHIVOS	6
5. ACTUALIZACIÓN DE ARCHIVOS EN LÍNEA	11
6. CLONAR REPOSITORIOS.....	13
7. DESCARGA DE CONTENIDO DE GitHub	14

1.INTRODUCCIÓN

En este manual se va a explicar los principales pasos para trabajar con repositorios en línea alojados en GitHub.

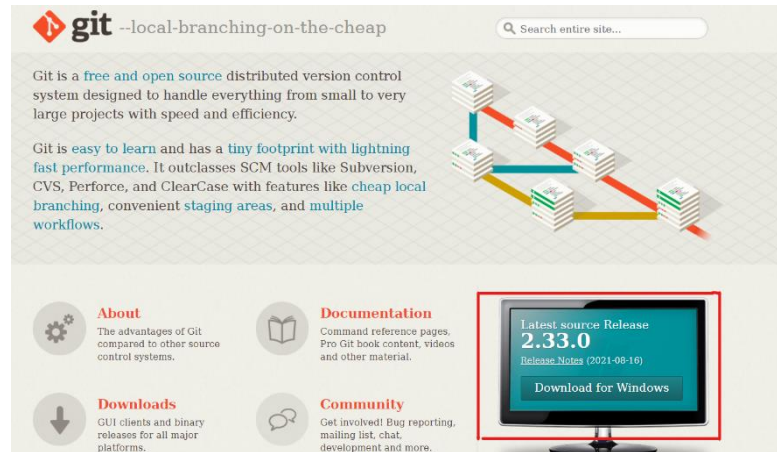
GitHub es una herramienta de código abierto que nos permite trabajar con archivos compartidos en repositorios alojados en la nube y poder acceder a ellos desde cualquier terminal. Para poder establecer una conexión desde una estación local al servidor de GitHub, se van a necesitar la instalación de los siguientes programas en el equipo: Git y Visual Studio Code.

2.DESCARGA DE Git

Git es un programa libre de control de sistemas y de código abierto de versión distribuida, diseñado para gestionar todo tipo de proyectos con agilidad y eficiencia. La función de este software es establecer comunicación entre el host y el servidor de GitHub.

Para descargar este programa nos debemos de dirigir a la dirección <https://git-scm.com/> desde el navegador instalado en el equipo, el cual nos redireccionara a la página oficial.

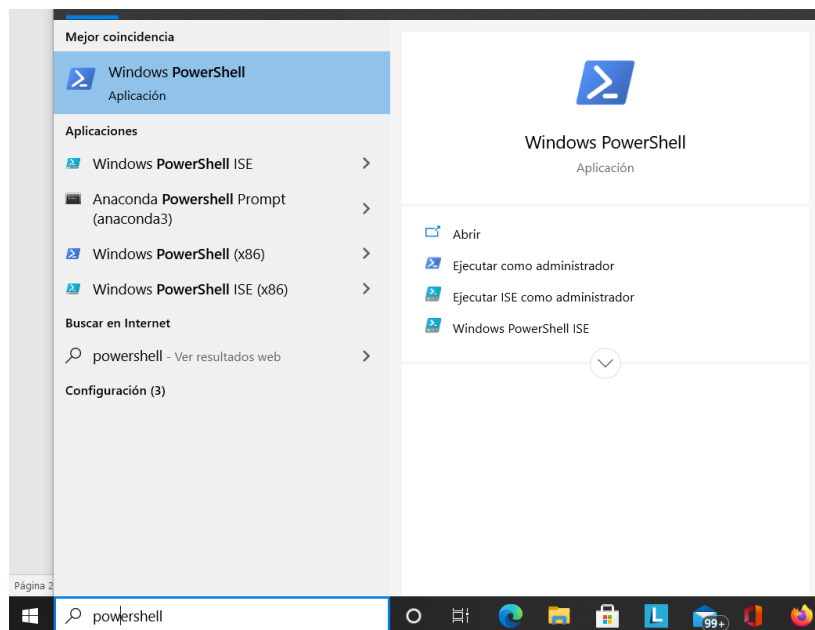
Una vez clicado en el botón señalado, automáticamente se descargará el ejecutable del programa.



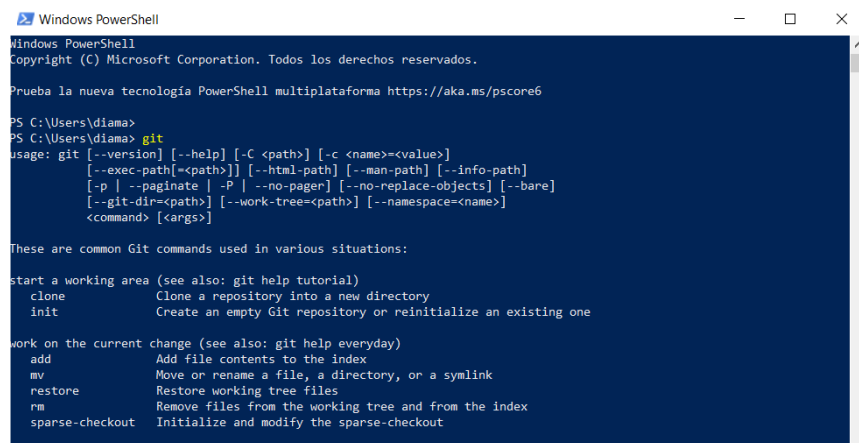
Cuando la descarga finalice, hacemos doble clic en el ejecutable y se procederá a la instalación automática. En este caso, la instalación se realizará siguiendo la configuración por defecto, por lo que **no tenemos que cambiar ningún parámetro** dentro del proceso.

***IMPORTANTE

Si queremos comprobar que el programa se ha instalado correctamente en el equipo, podemos comprobarlo usando la herramienta Powershell de nuestro ordenador. Para ejecutar esta aplicación únicamente debemos escribir en el buscador de Windows “Powershell” y clicar en la coincidencia resultante.



Una vez abierto el programa, debemos de escribir el comando **git** y si hemos seguido bien los pasos, la aplicación nos devolverá la siguiente respuesta.



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

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\diana>
PS C:\Users\diana> git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

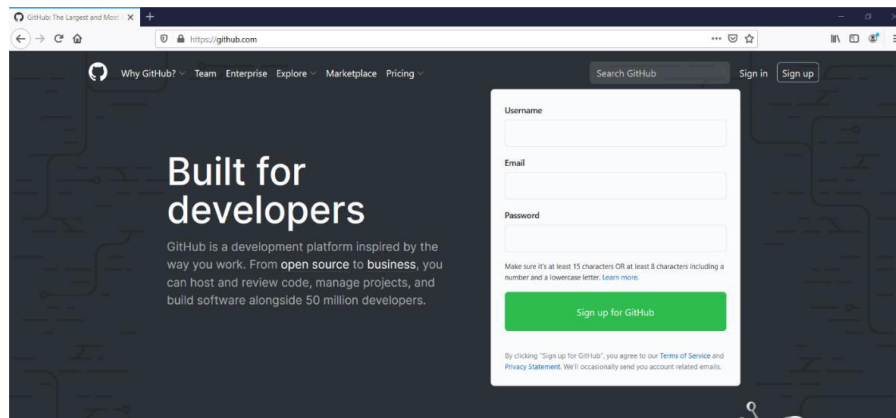
These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
  sparse-checkout  Initialize and modify the sparse-checkout
```

3.CREAR CUENTA EN GitHub

El siguiente paso en este manual (puede realizarse indistintamente en cualquier momento del proceso) será la creación en una cuenta en GitHub. Para ello, nos dirigiremos a la dirección <https://github.com/> desde nuestro navegador predeterminado.




Una vez en la página oficial, debemos registrarnos. Como se ve en la imagen, para completar el proceso de registro, debemos de elegir un nombre de usuario, un email de contacto y una contraseña. Cuando finalicemos este paso, clicamos sobre **Sign Up for GitHub** (regístrate en GitHub).


Create a new repository

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

Owner *


 NaranjoJimenezAndres2

Repository name *


Proyecto_001 

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

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

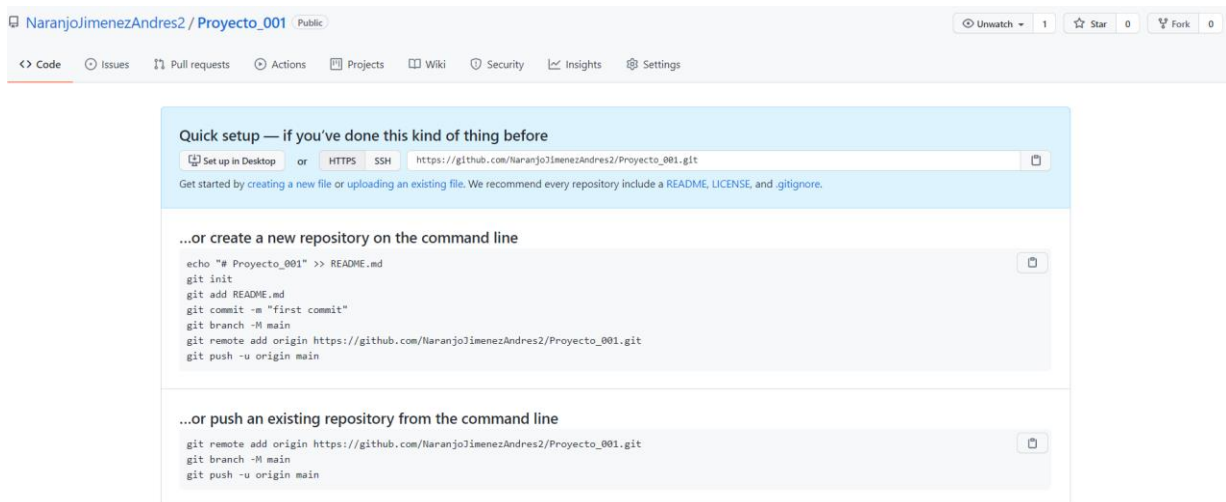
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

A

continuación, debemos de crear nuestro primer repositorio. Para ello debemos de elegir un nombre, seleccionar si queremos que este repositorio sea público o privado (lo cual afectará a la interacción que tendrán los usuarios sobre los archivos alojados) y finalizar el proceso clicando en “**Create repository**”.

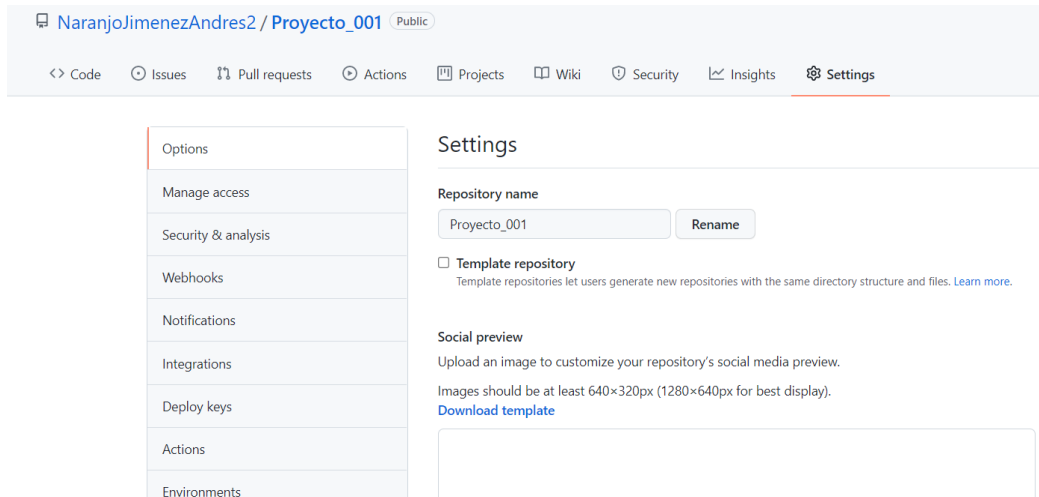
Una vez que nuestro primer reposito este creado tendremos una pantalla como esta.



Lo siguiente que nos aparecerá en pantalla será una lista de comandos cuya finalidad es la configuración y el enlace entre una carpeta del equipo local y el repositorio de GitHub recientemente creado en la nube.

***IMPORTANTE

Si posteriormente queremos eliminar el repositorio o cambiarle el nombre, debemos dirigirnos a la pestaña “**Settings**”.



Al final de esa misma ventana:

GitHub Pages

Pages settings now has its own dedicated tab! [Check it out here!](#)

Danger Zone

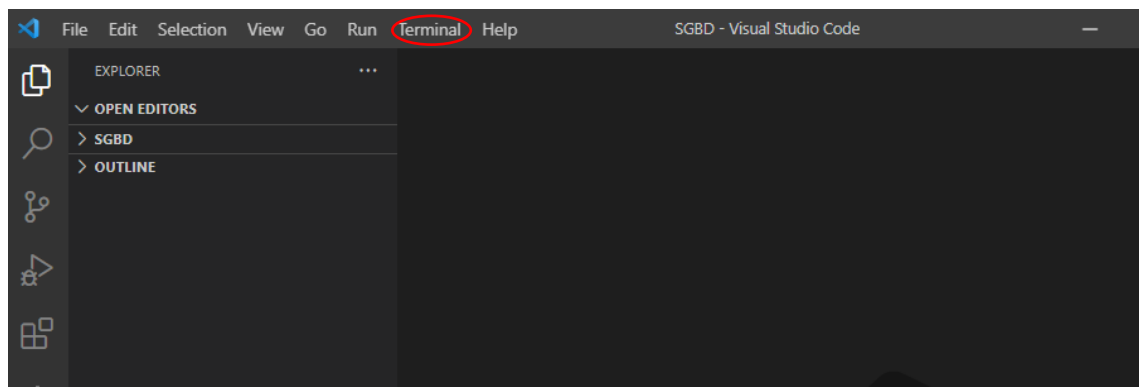
Change repository visibility This repository is currently public.	Change visibility
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	Transfer
Archive this repository Mark this repository as archived and read-only.	Archive this repository
Delete this repository Once you delete a repository, there is no going back. Please be certain.	Delete this repository

Además de su eliminación, en este apartado podemos modificar configuraciones de privacidad y seguridad entre otros.

4. ENVIO DE ARCHIVOS A UN REPOSITORIO GitHub

Para realizar el emparejamiento entre carpetas, debemos de ejecutar el programa Visual Studio Code. Seguidamente necesitamos crear una carpeta la cual se quiere emparejar.

La carpeta que vamos a emparejar es la “**proyecto_01**” pero la que vamos a introducir en el Visual Studio Code es la carpeta que en la ruta esta un escalón superior. Dicha carpeta la arrastramos con el cursor dentro de la interfaz de Visual Studio Code y deberíamos de ver esta imagen.

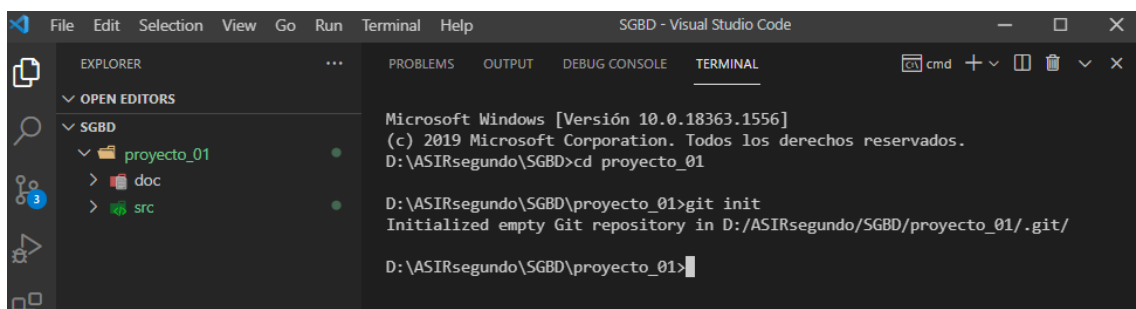


Como se puede apreciar en la imagen, la carpeta antes seleccionada se ha introducido en el programa. El siguiente paso es abrir la pestaña “**Terminal**” y seguidamente “**New Terminal**”.

PROCESAMIENTO DE ARCHIVOS

Una vez la carpeta preparada, vamos a proceder con el envío de un archivo desde el equipo local al repositorio de GitHub.

* Con **git init** inicializamos el programa.



* El paso siguiente es la comprobación de los archivos sincronizados, con “**git status**”.

```
D:\ASIRsegundo\SGBD\proyecto_01>git status
On branch master

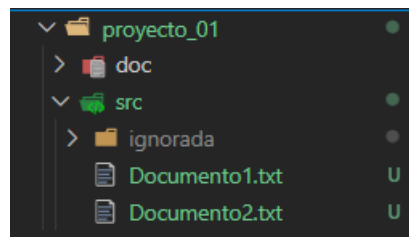
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/

nothing added to commit but untracked files present (use "git add" to track)

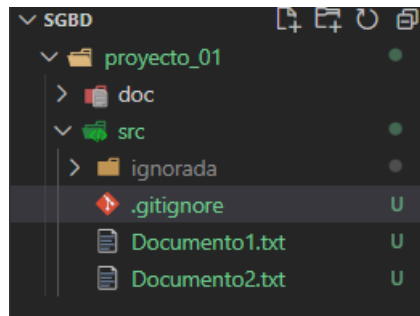
D:\ASIRsegundo\SGBD\proyecto_01>
```

La carpeta **src/** no ha sido sincronizada aun puesto que sale en rojo. Para sincronizarla debemos usar el comando **git add**. Pero antes vamos a ver que contiene esta carpeta.

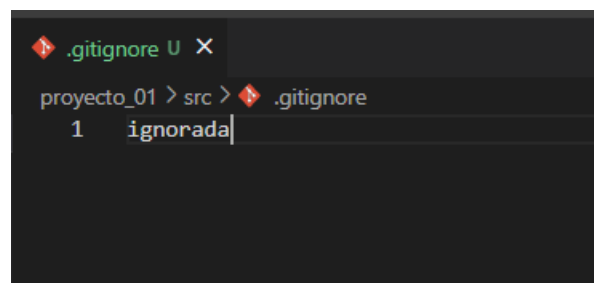


Si dentro de nuestro directorio tenemos una carpeta que debe ser ignorada, posiblemente porque no es útil o simplemente como desarrolladores no nos interesa subir su contenido a GitHub, existe una herramienta dentro de Git que se activa mediante el **archivo .gitignore**.

Para crear este archivo, tan solo tenemos que acceder al menú lateral y crear un archivo con ese nombre.

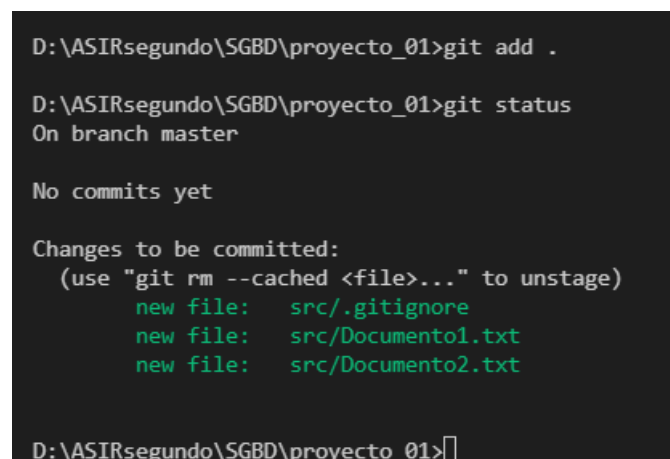


Como se puede apreciar, Git reconoce este tipo de archivo y le otorga un icono distintivo. Este archivo está vacío por defecto. En él debemos de introducir el nombre de los archivos, extensiones, o directorios que no queremos que se lancen a GitHub.



Una vez preparado nuestro proyecto, esta listo para ser subido a GitHub.

* Usamos **git add** para sincronizar cambios y prepararlos para la subida. Si queremos solo subir un solo archivo es **git add <filename>**, sin embargo si queremos subir todo, la sentencia es “**git add .**”



Cuando aplicamos **git status**, en este caso nos sale verde, ya que están sincronizados correctamente

* El siguiente paso es usar **git commit -m** para guardar los cambios en el repositorio. A este comando se le puede añadir una etiqueta para futuros cambios.

```
D:\ASIRsegundo\SGBD\proyecto_01>git commit -m "first commit"
[master (root-commit) 996f29f] first commit
3 files changed, 3 insertions(+)
create mode 100644 src/.gitignore
create mode 100644 src/Documento1.txt
create mode 100644 src/Documento2.txt
D:\ASIRsegundo\SGBD\proyecto_01>
```

* El siguiente comando sirve para indicar en que parte del proyecto queremos realizar la subida, ya que **git** se trata de un programa de control de versiones puede haber mas de una rama de trabajo. En nuestro caso siempre trabajaremos en la “**main**” usando el comando:

```
D:\ASIRsegundo\SGBD\proyecto_01>git branch -M main
```

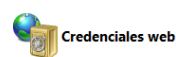
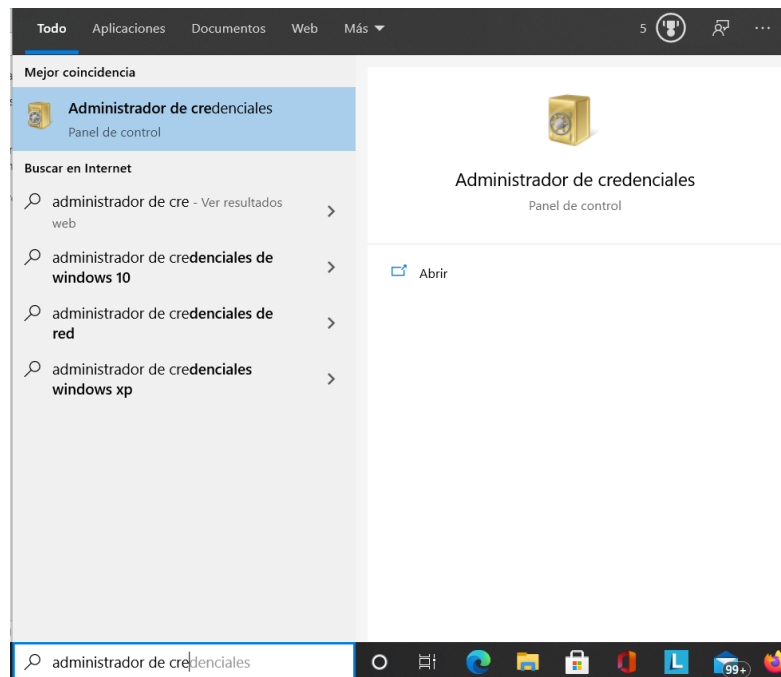
* Para enviar el archivo debemos de poner la dirección https de nuestro repositorio en GitHub en el comando **git remote add origin https://.....**

```
D:\ASIRsegundo\SGBD\proyecto_01>git remote add origin https://github.com/NaranjoJimenezAndres2/Proyecto_001.git
```

* Por último, el comando **git push -u origin main**. Una vez realizada esta acción, nos debemos de autenticar en una ventana emergente con la cuenta de GitHub.

```
D:\ASIRsegundo\SGBD\proyecto_01>git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 393 bytes | 393.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NaranjoJimenezAndres2/Proyecto_001.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

La primera vez que hagamos esto, nos saldrá una ventana emergente para acceder a GitHub. Esta cuenta queda grabada en las credenciales del sistema, por lo que para utilizar otra cuenta en un mismo PC debemos de eliminar la anterior. Para acceder a las credenciales del sistema solo tenemos que poner en el buscador de Windows “administrador de credenciales”



[Copia de seguridad de credenciales](#) [Restaurar credenciales](#)

Credenciales de Windows

[Agregar una credencial de Windows](#)

192.168.6.100	Modificado: 14/10/2020
192.168.98.80	Modificado: 13/04/2021

Credenciales basadas en certificados

[Agregar una credencial basada en certificado](#)

No hay certificados.

Credenciales genéricas

[Agregar una credencial genérica](#)

MongoDB Compass/Connections/0158d179-3fc5-44f9...	Fecha de modificación: 24/02/2021
MongoDB Compass/Connections/0bbcc1cc-701f-4633...	Fecha de modificación: 21/02/2021
MongoDB Compass/Connections/10f71552-ae8f-45c2...	Fecha de modificación: 08/02/2021
MongoDB Compass/Connections/1120f233-46ad-454c...	Fecha de modificación: 23/02/2021
MongoDB Compass/Connections/1296da10-ddd2-4ce...	Fecha de modificación: 19/11/2020
MongoDB Compass/Connections/14629973-9570-492...	Fecha de modificación: 24/02/2021
MongoDB Compass/Connections/196a8770-578c-4a6...	Fecha de modificación: 23/04/2021
MongoDB Compass/Connections/1b8f4c4a-7cb3-4be...	Fecha de modificación: 26/10/2020
MongoDB Compass/Connections/24e5e119-8026-47d...	Fecha de modificación: 10/10/2020
MongoDB Compass/Connections/ffd078cf-197a-4765-...	Fecha de modificación: 17/04/2021
vscodevscode.github-authentication/github.auth	Fecha de modificación: Hoy
git:https://github.com	Fecha de modificación: Hoy

Dirección de red o Internet: git:https://github.com

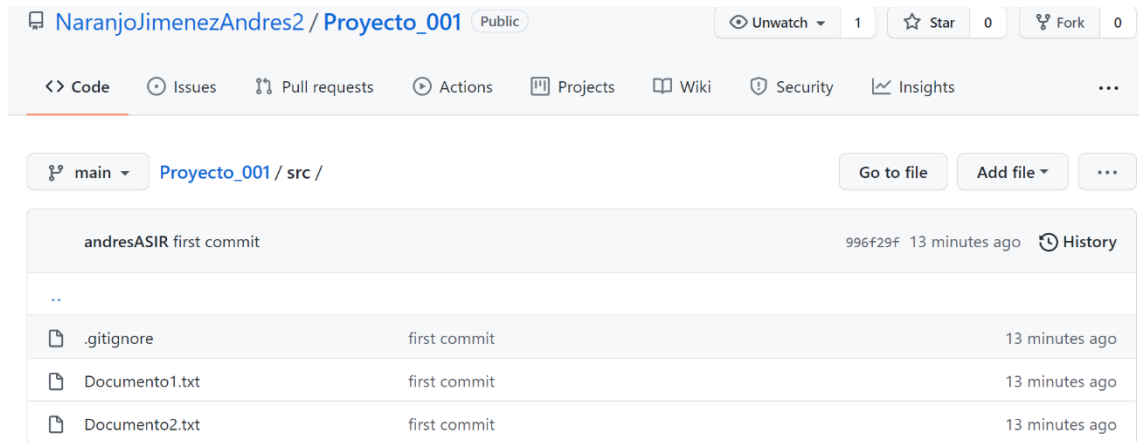
Nombre de usuario: 91050947

Contraseña:

Persistencia: Equipo local

[Editar](#) [Quitar](#)

Si nos metemos en nuestro repositorio de GitHub, veremos que los archivos se han subido correctamente.

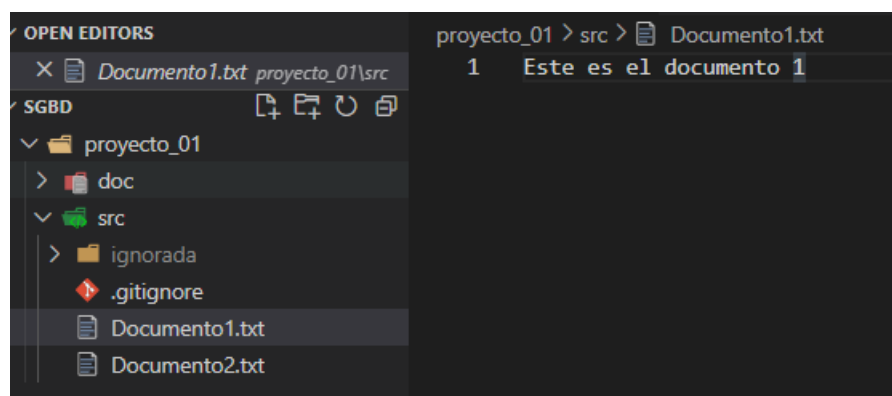


Como podemos apreciar, la carpeta “ignorada” dentro del directorio src/ no se ha subido a GitHub debido al trabajo del archivo **.gitignore**

5. ACTUALIZACIÓN DE ARCHIVOS EN LÍNEA

El archivo subido puede modificarse en línea por otros usuarios, por lo que si queremos actualizar nuestro archivo en el equipo local con los cambios realizados, debemos de introducir el comando **“git pull -v origin main”**.

DOCUMENTO1: LOCAL



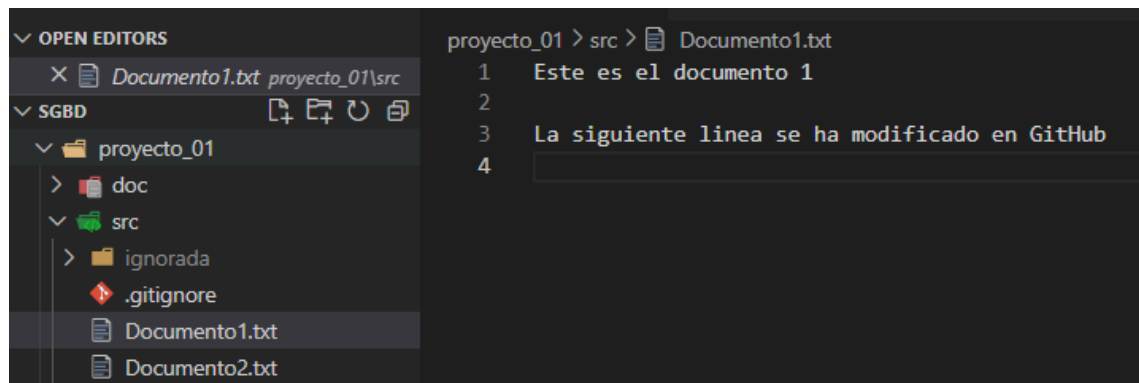
DOCUMENTO1: GitHub

The screenshot shows the GitHub interface for the repository 'NaranjoJimenezAndres2 / Proyecto_001'. The repository is marked as 'Public'. Navigation tabs include 'Code', 'Issues', 'Pull requests', 'Actions', and 'Projects'. The current view is the file 'Proyecto_001 / src / Documento1.txt' on the 'main' branch. A commit by 'NaranjoJimenezAndres2' is shown with the message 'Update Documento1.txt'. It indicates '1 contributor'. The file details show '3 lines (2 sloc)' and '70 Bytes'. The file content is displayed with line numbers: '1 Este es el documento 1', '2', and '3 La siguiente linea se ha modificado en GitHub'. The third line is circled in red.

Ejecutamos el comando mencionado anteriormente.

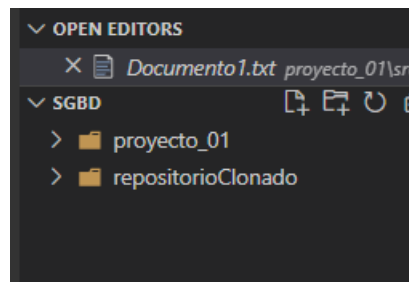
```
D:\ASIRsegundo\SGBD\proyecto_01>git pull -v origin main
POST git-upload-pack (225 bytes)
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 749 bytes | 2.00 KiB/s, done.
From https://github.com/NaranjoJimenezAndres2/Proyecto_001
* branch      main      -> FETCH_HEAD
   996f29f..7110aa3  main      -> origin/main
Updating 996f29f..7110aa3
Fast-forward
 src/Documento1.txt | 4 +++- ←
 1 file changed, 3 insertions(+), 1 deletion(-)
```

D:\ASIRsegundo\SGBD\proyecto_01>



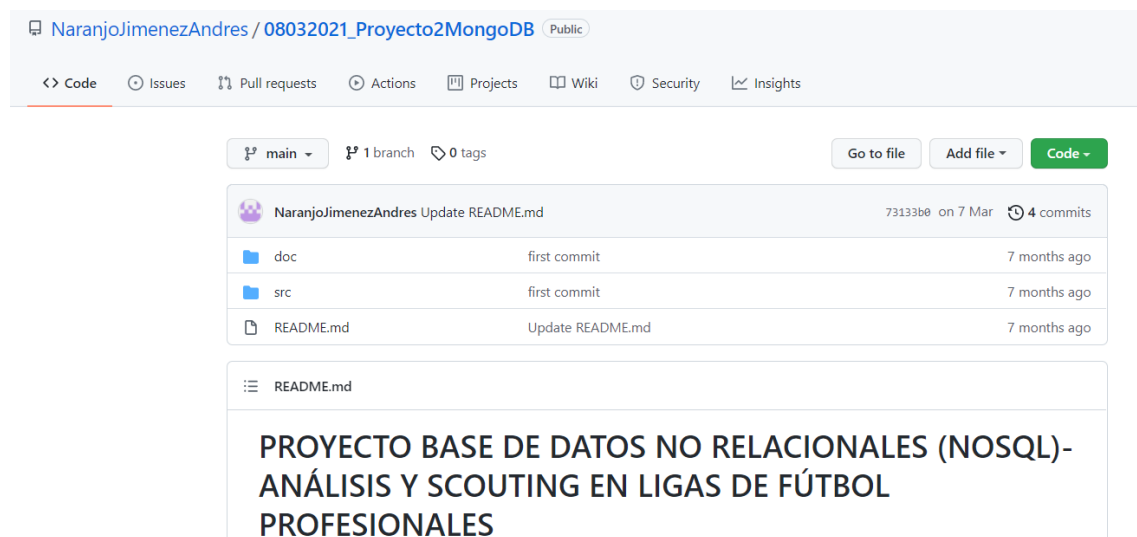
6. CLONAR REPOSITORIOS

Por último, si queremos clonar un repositorio en la nube, debemos de usar el comando **git clone https: //**

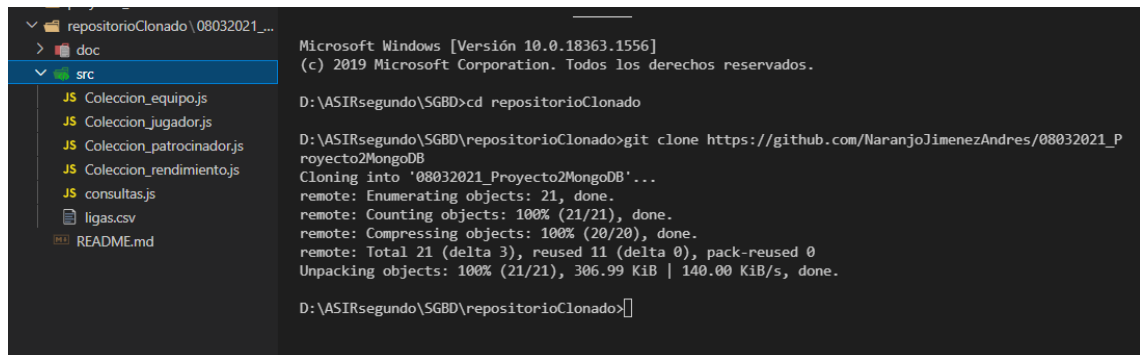


-Creamos una carpeta en nuestro equipo local para alojar el repositorio clonado de Github.

-Nos dirigimos al repositorio que queremos clonar



Abrimos una nueva terminal en nuestra nueva carpeta



```
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

D:\ASIRsegundo\SGBD>cd repositorioClonado

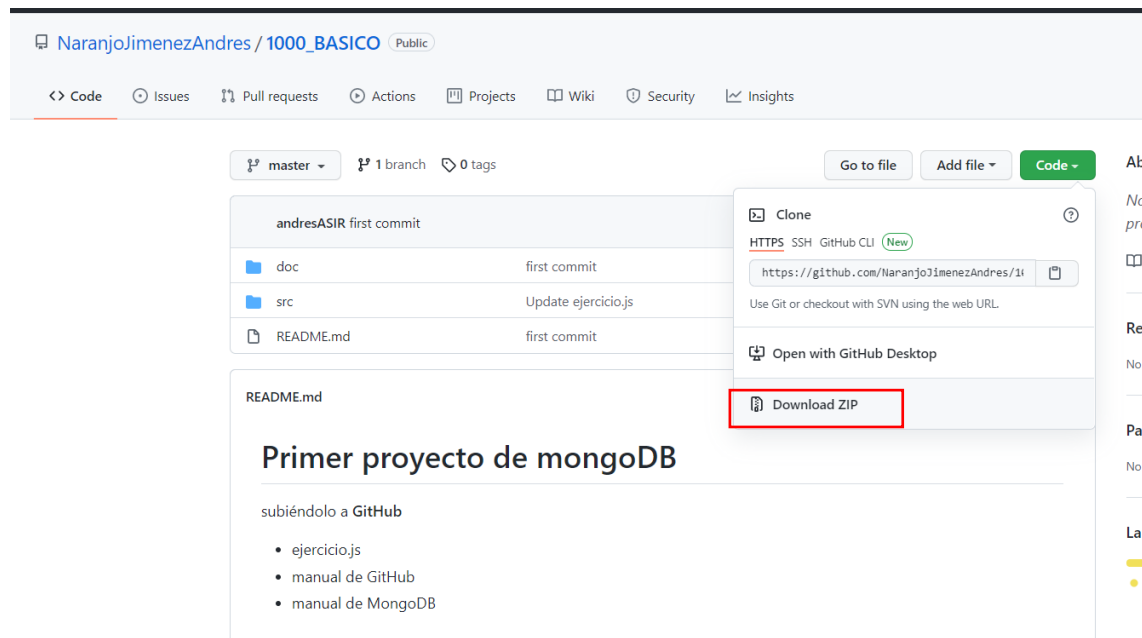
D:\ASIRsegundo\SGBD\repositorioClonado>git clone https://github.com/NaranjoJimenezAndres/08032021_Proyecto2MongoDB
Cloning into '08032021_Proyecto2MongoDB'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 21 (delta 3), reused 11 (delta 0), pack-reused 0
Unpacking objects: 100% (21/21), 306.99 KiB | 140.00 KiB/s, done.

D:\ASIRsegundo\SGBD\repositorioClonado>
```

Esta herramienta es bastante útil para poder seguir trabajando desde un equipo que no es el habitual. Tan solo se debe clonar el repositorio desde la web y seguir trabajando en él.

7. DESCARGA DE CONTENIDO DE GitHub

Si nos interesa descargar un repositorio en .zip , nos debemos dirigir al botón verde “Code” y pulsar **Download ZIP**



Esto puede ser útil para poder trasladar nuestro trabajo a lugares o equipos sin conexión a internet, por lo que no podemos usar el proceso anterior de clonado.

