# MINIPROYECTO UT1

# PLANIFICACION DE INTERFACES WEB

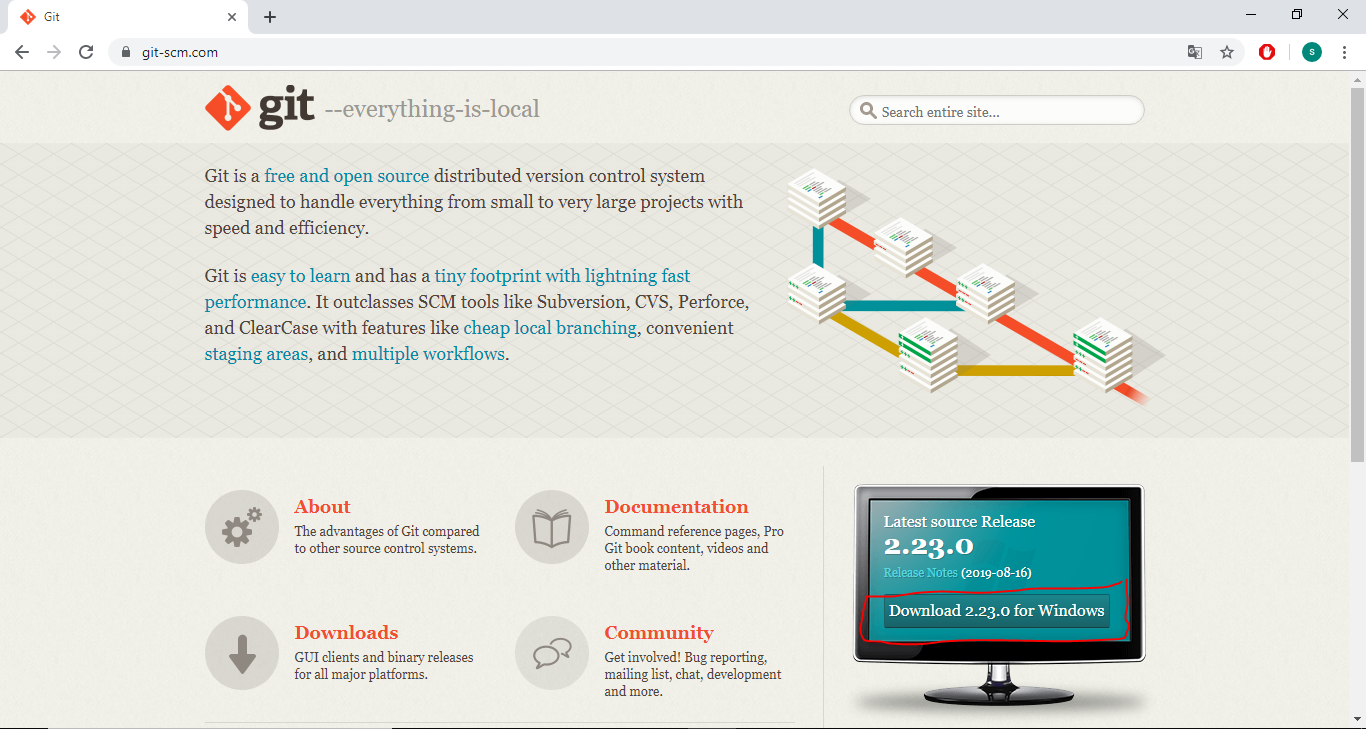
## IMPLANTACIÓN DE GIT

### INSTALACIÓN

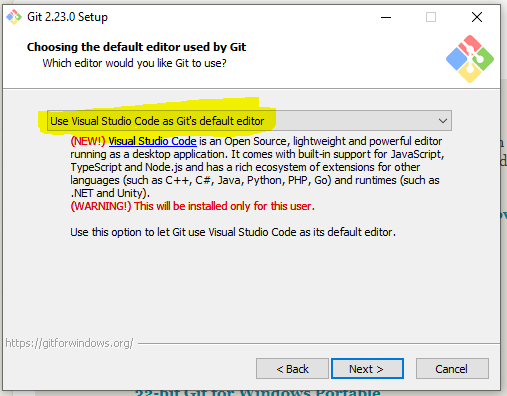
Nos vamos a centrar en la instalación de GIT para Windows, ya que es lo que hemos realizado en clase.

Hay que tener una consideración previa y es la instalación de Visual Studio Code. Esto nos va a permitir usar la terminal bash que nos proporciona Visual Studio como consola de comandos por defecto para lanzar comandos de GIT.

Una vez comentado eso, vamos a empezar con la instalación. Para ello hay que ir a la web oficial de [GIT](https://git-scm.com/) y descargar la versión para Windows:



Una vez descargado, hacemos doble clic para iniciar el instalador. Todos los valores los dejaremos por defecto salvo el editor de GIT por defecto, que lo cambiaremos y utilizaremos VS code como editor por defecto:



Ahora, antes de seguir. Hay que hablar de **que es GIT**:

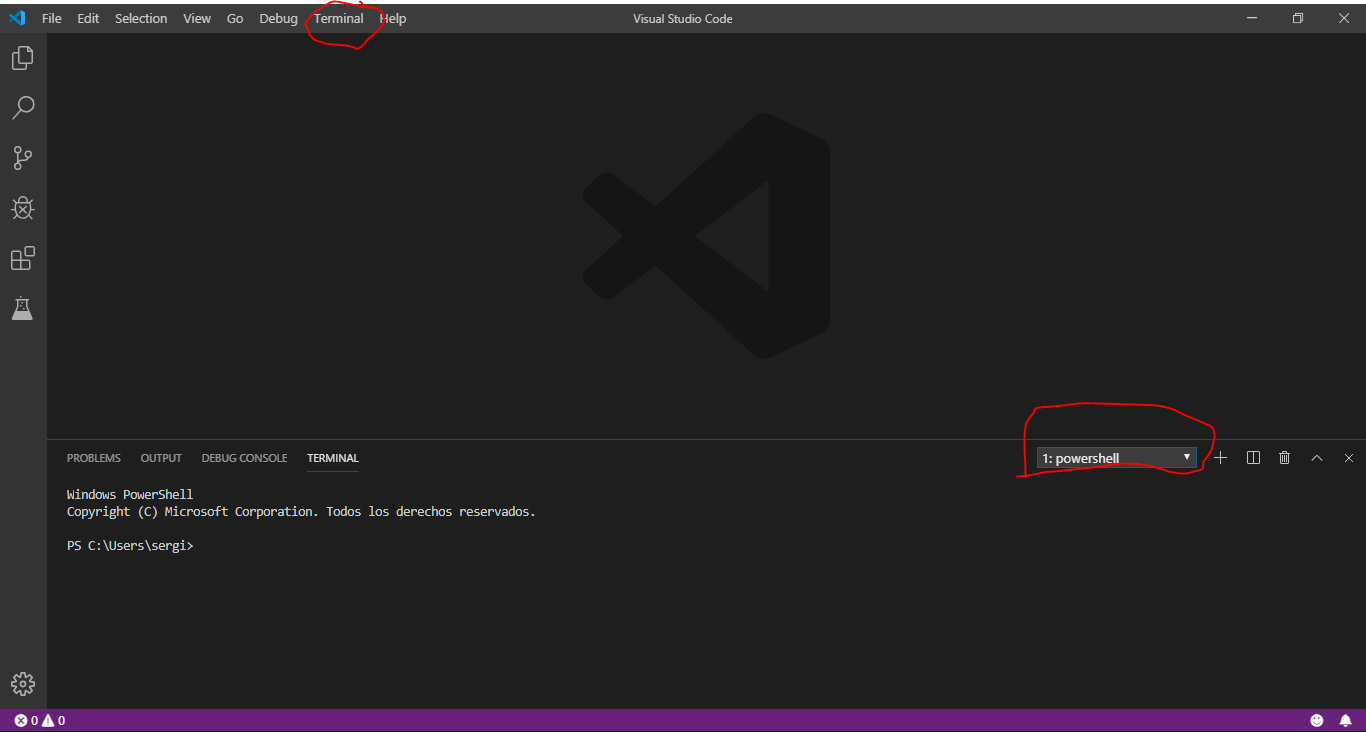
GIT es un software de control de versiones, es decir nos permite en un proyecto tener diferentes versiones del mismo según vamos desarrollando este. Esto es muy útil, ya que nos permite volver a etapas anteriores si ocurre algún error o también tiene gran utilidad para el desarrollo de proyectos grupales.

## CONFIGURACIÓN DE GIT

Ya tenemos GIT instalado en nuestro equipo, pero para poder empezar a usarlo y sobretodo para sacar todo el potencial que tiene hay que hacer algunas configuraciones.

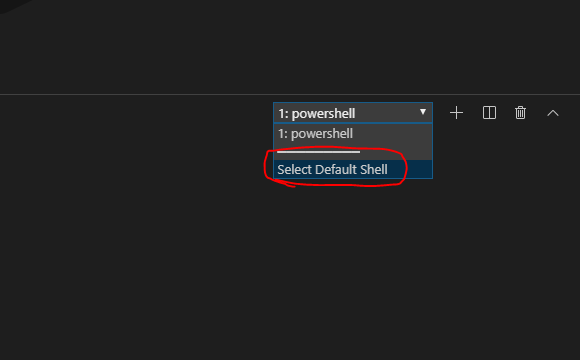
Primero y muy importante, hay que configurar la terminal de visual studio code, que por defecto viene con powershell y cambiarla para que sea bash.

Para abrir una nueva terminal hay que ir en el menú de la parte superior del programa a terminal -> new terminal:

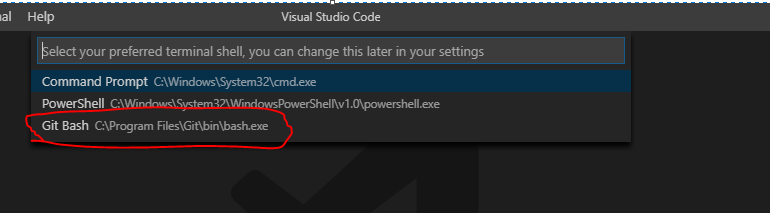


Si hacemos eso, se nos abre la terminal en la parte inferior de la zona de edición de código, pero como dije anteriormente por defecto nos aparecerá powershell.

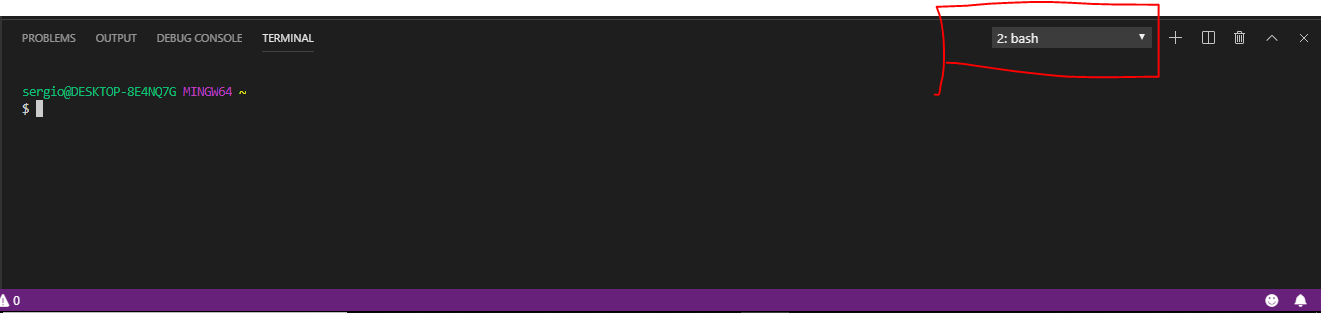
Para cambiarlo es necesario hacer clic sobre el desplegable donde apareceré powershell y seleccionar Select Default Shell



Al hacer clic se nos abre otro desplegable en la parte de arriba y aquí es donde ya elegimos bash como terminal por defecto.



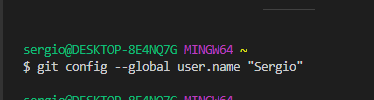
Si lo hemos hecho bien, al abrir una nueva terminal se nos abrirá con bash:



**Hay que decirle a GIT quienes somos.**

Es necesario para trabajar con GIT hacer unas configuraciones globales para identificarnos. Para ello usamos el comando git config seguido de una serie de modificadores.

Lo primero que vamos a configurar es el usuario: git config –global user.name “Sergio”



Ahora hay que establecer un correo electrónico: git config –global user.mail “sergio.pabayu@educa.jcyl.es”



Con estas dos configuraciones que acabamos de hacer, ya podríamos subir a nuestro repositorio remoto lo que tuviéramos en local.

Y en el anterior párrafo acabo de introducir uno de los conceptos mas importantes que vamos a tratar y es el de los **repositorios, tanto locales como remotos.**

Bien un repositorio a nivel remoto no es más que un espacio donde queda alojado nuestro código y quedará seccionado por las versiones y ramas que hayamos ido creando.

Un repositorio a nivel local, no sería más que aquella carpeta, que hayamos iniciado mediante unos comandos con git, donde almacenamos nuestros ficheros.

Hemos hablado de lo que es un repositorio remoto, vamos a ver ahora que aplicaciones/webs nos permite trabajar con repositorios remotos.

Hasta donde yo he trabajado conozco dos, por un lado Github, es con seguridad la más conocida y utilizada. Y por otro lado estaría Bitbucket. Nosotros en clase hemos trabajado con Github así que para este proyecto continuaré.

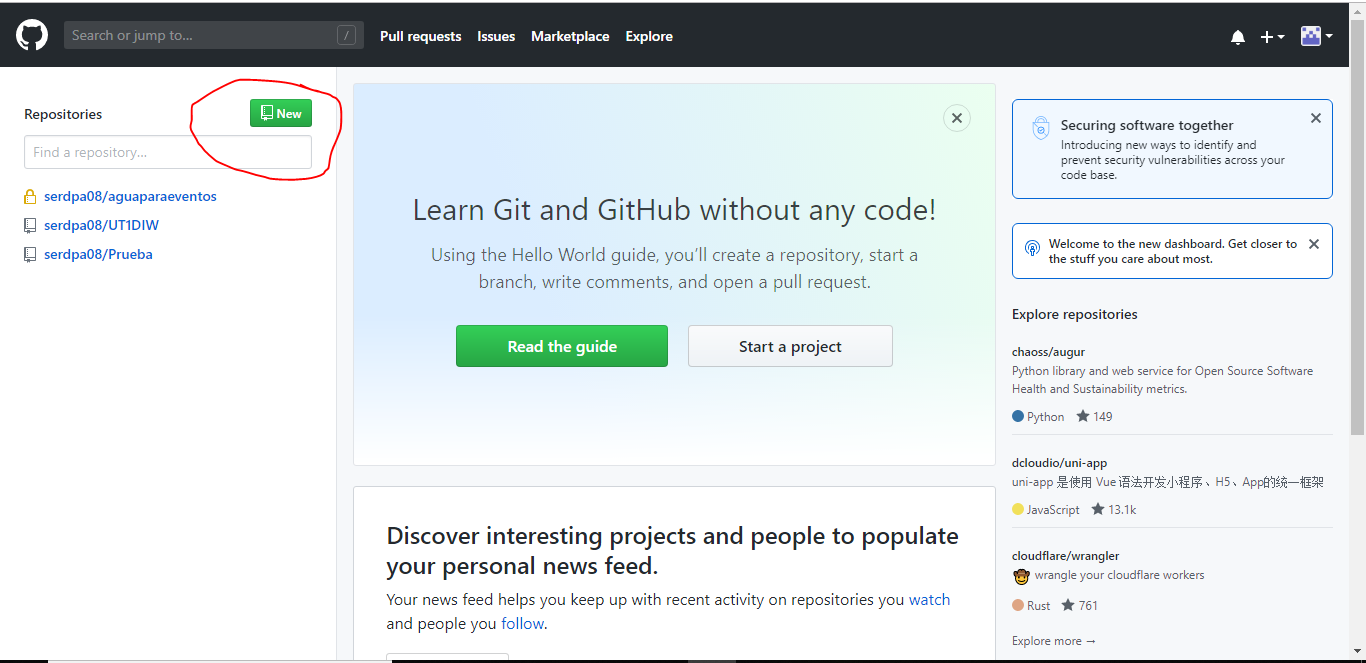
Tras esta breve introducción vamos ahora a detenernos más en como usar Github y como crear nuestro primer repositorio remoto. Tras eso lo conectaremos con nuestro repositorio local y haremos los primeros commits y push.

## GITHUB

## 

Lo primero que tenemos que hacer para poder usar Github es hacernos una cuenta. Una vez hecha, tendremos un espacio personal donde podremos empezar con la creación de repositorios a nivel remoto.

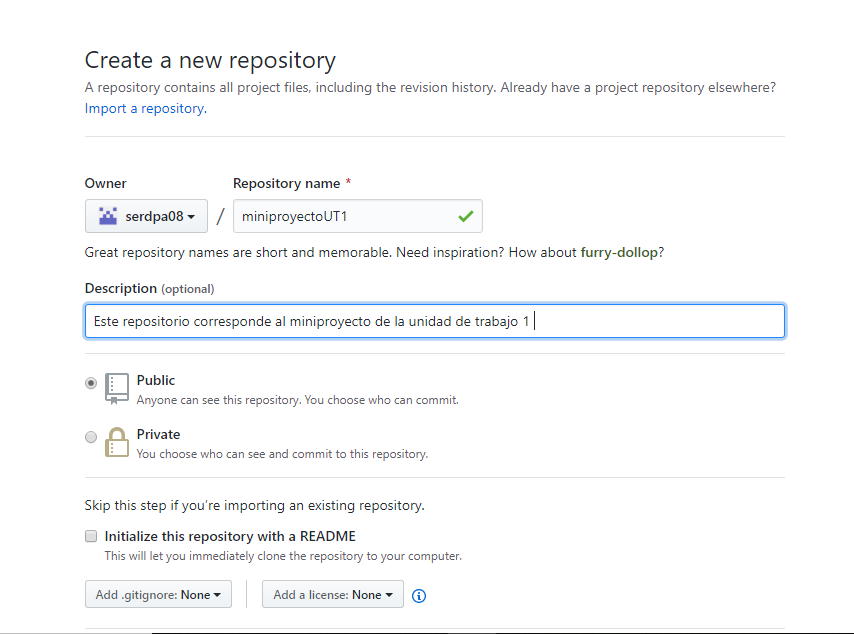
Para crear un **nuevo repositorio:** En la pagina principal de nuestro espacio personal hay un botón verde con la palabra “new”. Hacemos clic sobre él para comenzar la creación del repositorio.



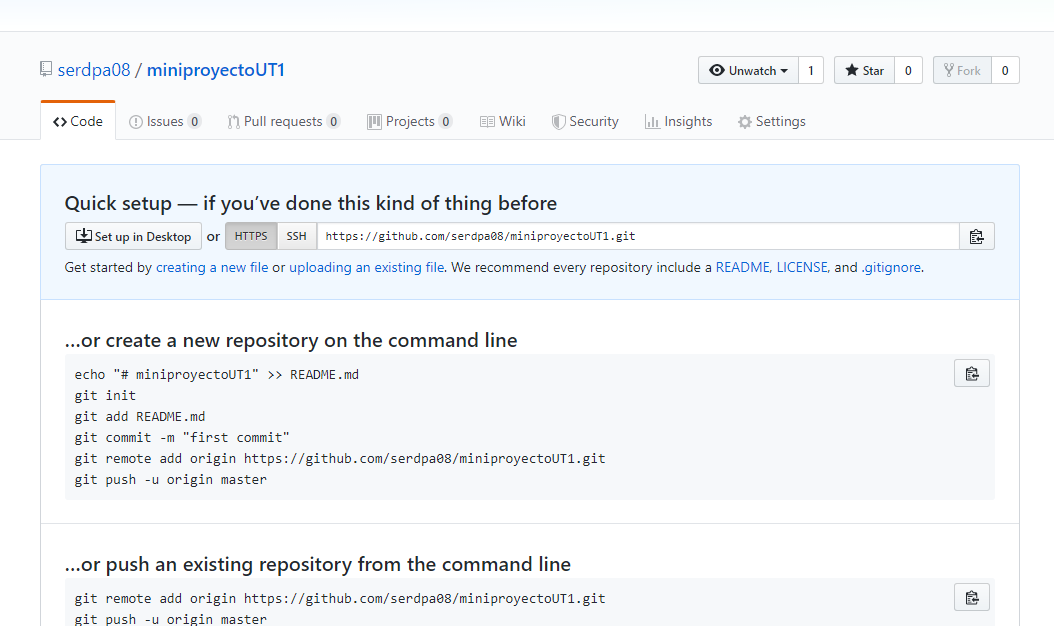
Una vez hecho clic, se nos abre una página donde podemos completar los datos del repositorio. Aquí habrá que completar:

* Nombre del repositorio.
* Descripción, que será opcional.
* Elegiremos la opción de público o privado.
* Podremos realizar un fichero explicativo del repositorio conocido como README.
* Podremos añadir la licencia que tendrá nuestro proyecto.

Yo en este ejemplo solo voy a rellenar el nombre y la descripción.

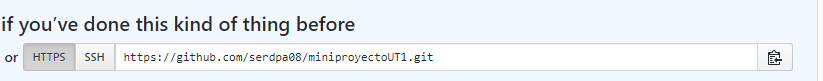


Una vez creado, Github nos lleva directamente a la página principal de nuestro repositorio.



Por supuesto, este repositorio esta vacío, pues aun no hemos subido nada desde nuestro repositorio a nivel local. Además Github nos proporciona unas pequeñas instrucciones de como subir nuestros archivos al repositorio.

Importante va a ser la dirección HTTP o SSH que nos proporciona de nuestro repositorio, pues la necesitaremos para conectarnos desde nuestra consola al repositorio:

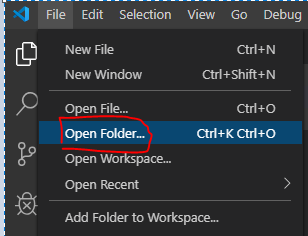


Voy a dejar para más adelante el configurar la conexión al repositorio por vía SSH y de momento voy a trabajar por HTTP.

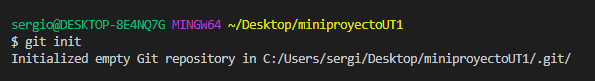
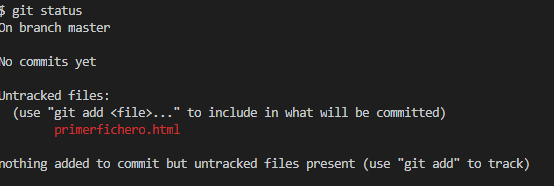
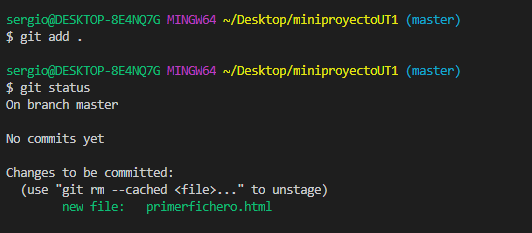
## CONECTAR NUESTRO REPOSITORIO LOCAL AL REMOTO Y PRIMER PUSH

Para este punto voy a crear una carpeta que se va a llamar miniproyectoUT1, que será la carpeta que contendrá mi repositorio local.

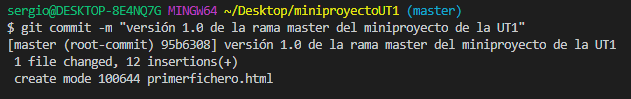
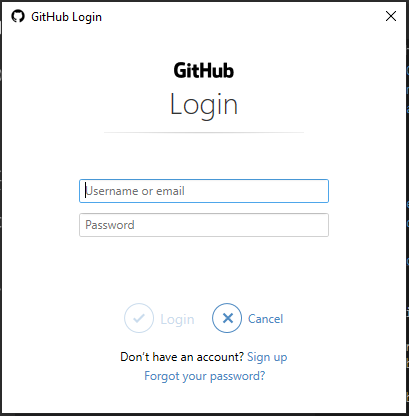
Para ello creo la carpeta en el escritorio y posteriormente la voy a abrir con visual studio code:

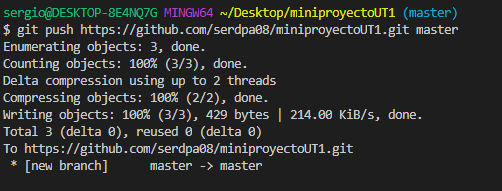


Una vez que ya tengo abierta la carpeta, voy a crear un fichero html y lo voy a subir al repositorio remoto. Para ello hay que seguir una serie de pasos y ejecutar unos comandos en el siguiente orden.

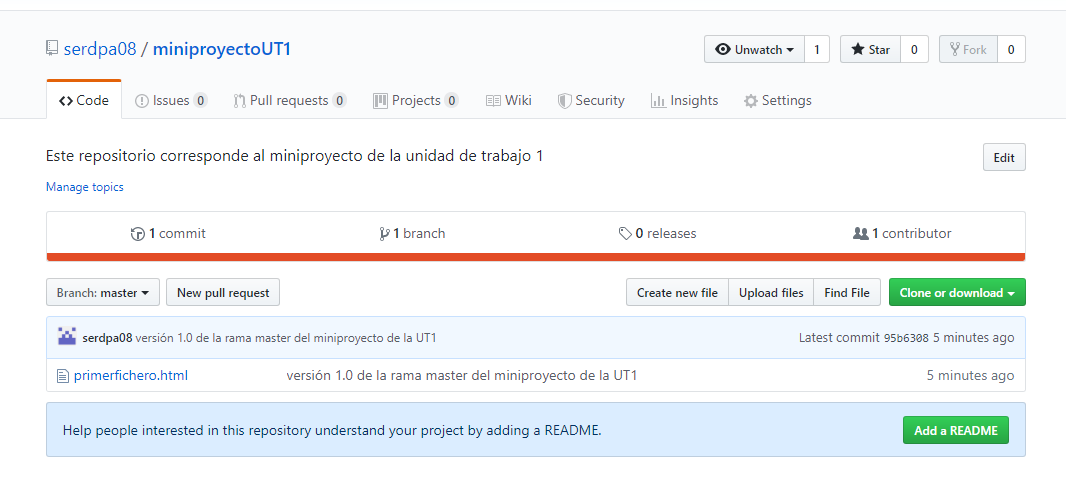
* Git init: inicializa el repositorio y añade un archivo .git
* 
* Git status: nos va a servir para ver que ficheros han sido modificados y no se les ha añadido para hacer commit. Es decir debería darnos diferentes salidas si lo ejecutamos antes y después del comando git add.
* 
* Git add: este comando añade los ficheros que queremos subir al repositorio remoto. Podemos seleccionar por nombre del fichero, por ruta o añadir todo. Dependerá de lo que nosotros especifiquemos. En este ejemplo vamos a añadir todo el contenido actual poniendo git add .
* 

Para ver que el fichero se añadió correctamente basta con hacer git status para comprobarlo.

* Git commit -m ””: con este comando empaquetamos lo que vamos a subir y además podemos poner un mensaje que nos ayudará a etiquetar esa versión.
* 
* Git push: con este comando subimos al repositorio remoto lo que hemos añadido previamente. Hay que añadir el origen y la rama que lo sube. Si usaramos SSH en vez de HTTP no sería necesario especificar la ruta del repositorio remoto. Nos pedirá loguearnos con nuestro usuario y contraseña de github antes de realizar el push.
* 



Con todo esto ya habremos hecho nuestro primer push al remositorio remoto. Para comprobar que todo funciono correctamente vamos al repositorio y comprobamos que se ha subido el fichero:



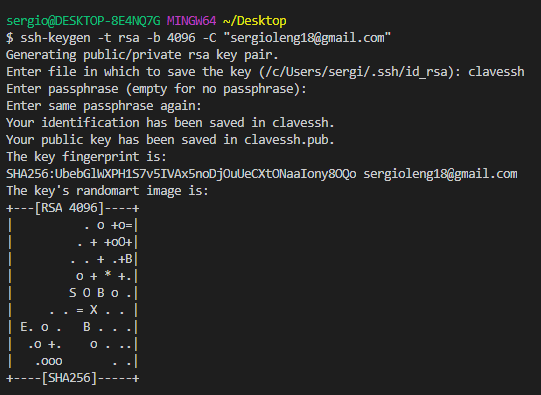
# Conectar al repositorio remoto por SSH y push por SSH

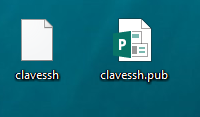
Anteriormente hemos utilizado el protocolo HTTP para conectarnos a nuestro repositorio remoto, pero una forma de hacerlo con mayor seguridad es hacerlo mediante SSH.

Para ello, lo primero que debemos hacer es generar una clave SSH a nivel local. Para poder generar esta clave vamos a utilizar la terminal bash de visual studio code.

Nos situamos en el directorio donde vayamos a guardar las claves y ejecutamos el siguiente comando: ssh-keygen -t rsa -b 4096 -C “email”

Una vez introducido nos pedirá introducir el nombre del archivo y una contraseña.

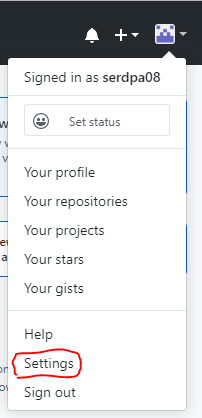




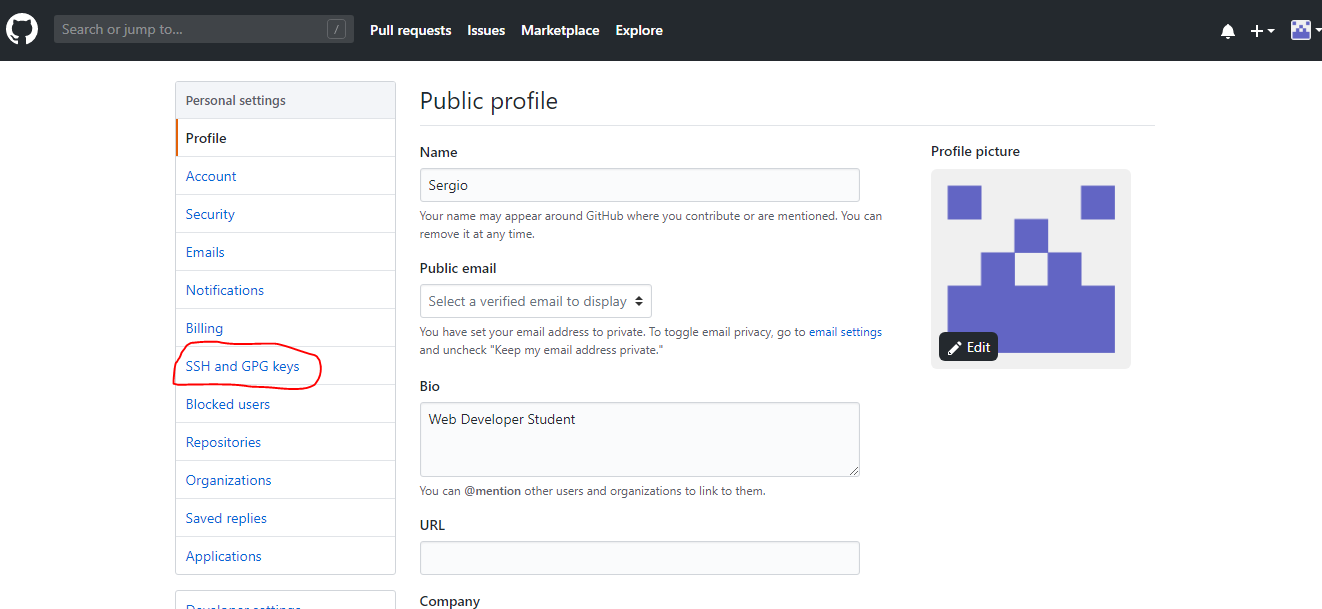
Ahora ya tenemos la clave generada a nivel local **y hay que añadirla a nuestro perfil de github.**

## Añadir clave a Github

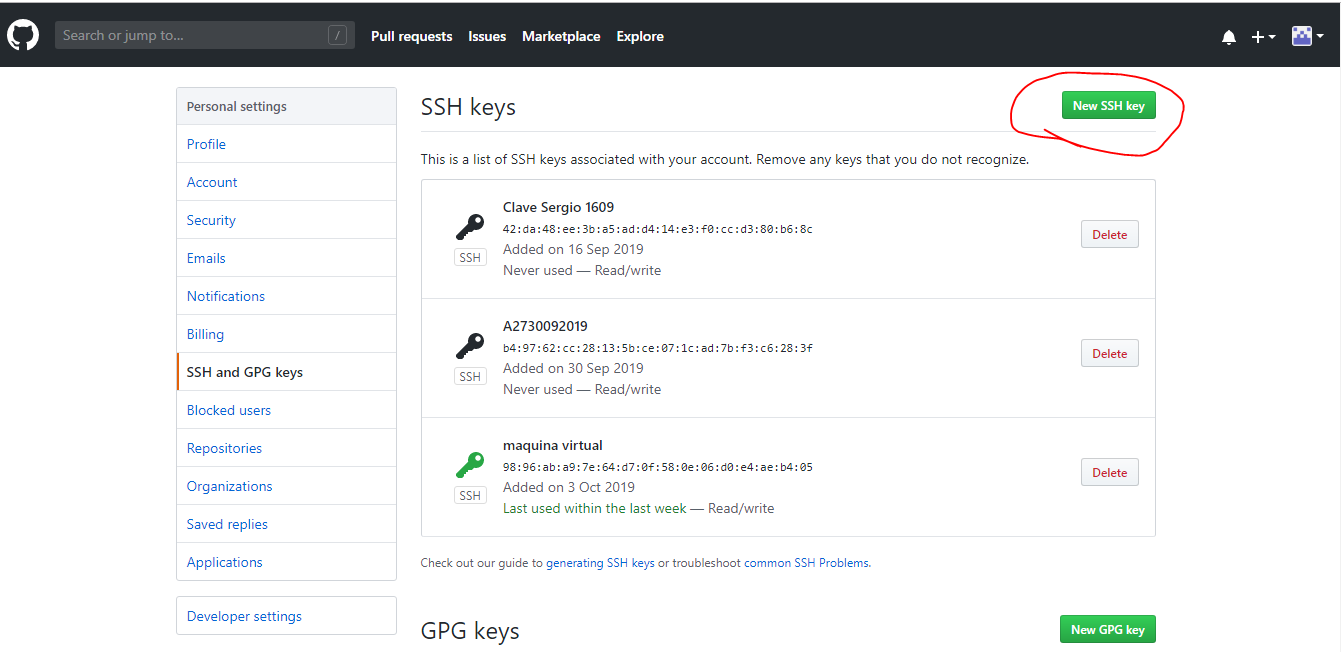
Para añadirla abrimos github y en nuestro icono de perfil desplegamos el menú y hacemos clic sobre la opción de **settings.**



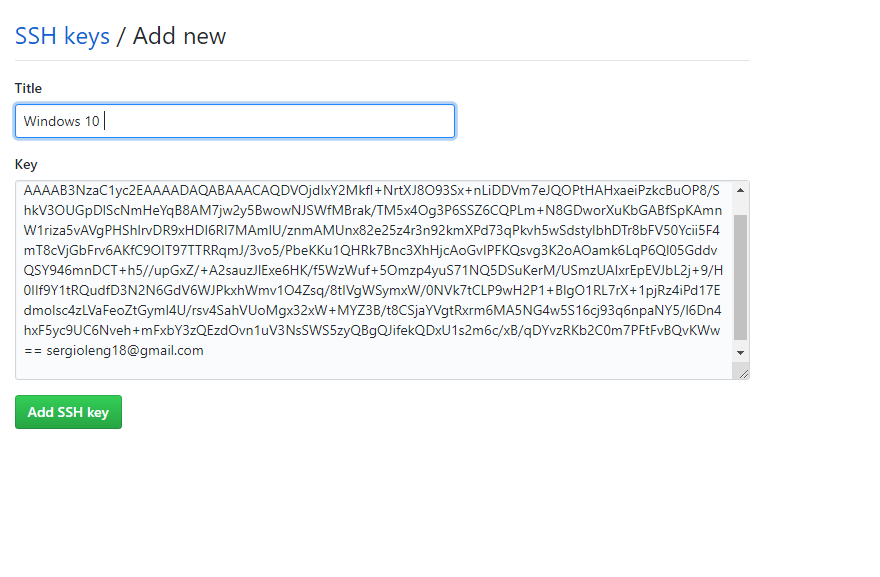
Ahora en la página que se nos abre vamos al apartado de **SSH and GPG Keys**



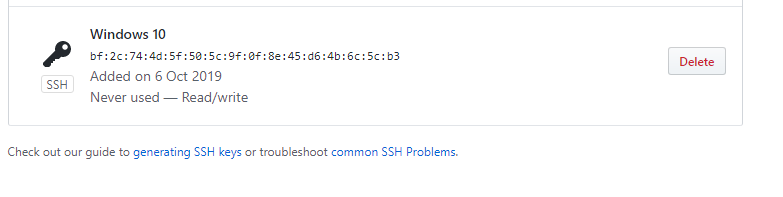
En esta página hay que hacer clic sobre el botón verde de add new Key



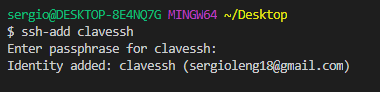
Ahora abrimos el archivo .pub que previamente creamos y que contiene la clave pública y copiamos el contenido y lo pegamos en el apartado que nos pone en github:



Podemos ver que se añadió correctamente la clave:



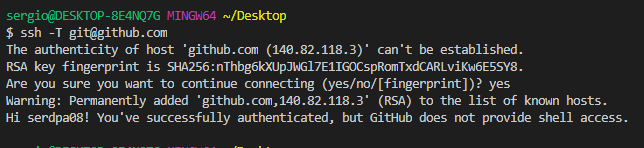
Ahora que ya hemos añadido la clave a nivel remoto, hay que añadir la clave a nivel local. Para ello en la consola de visual studio code y ejecutamos el comando ssh-add y el nombre del archivo que contiene la clave:



Nos pedirá la contraseña que introducimos en la creación y ya estaría añadida la clave.

## Probar conectividad

El paso siguiente es probar que hay conexión, para ello ejecutamos el siguiente comando: **ssh -T** [**git@github.com**](mailto:git@github.com)



## Push por SSH

Hemos probado y ya sabemos que tenemos conectividad, así que ya podemos hacer un push vía SSH.

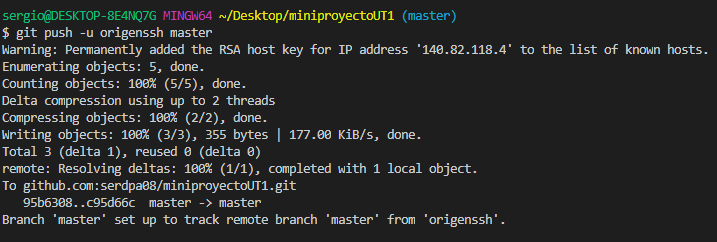
Antes de eso vamos a añadir la url que nos proporciona github como una variable para no tener que copiarla cada vez que vayamos a hacer el push.

Para ello usamos el siguiente comando:

Git remote add nombre url



Con esto ya tenemos la ruta del ssh guardada en esa variable y podemos hacer el push:

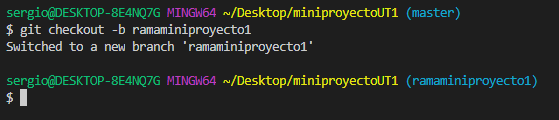


## Ramas en GIT

Por último, vamos a tratar el apartado de las ramas en GIT. Existe una rama general que es la que conocemos como **master**, esta rama será normalmente la principal del proyecto y en la que convergerán todas las demás ramas.

Como hemos dicho, la rama master es la principal y a partir de esta rama podemos ir creando ramas individuales, es decir independientes de la principal y con su propio contenido.

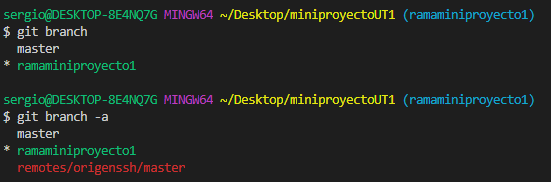
Para crear una rama usamos el comando **git checkout -b nombreRama**



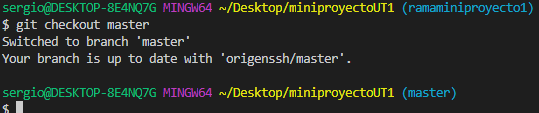
Ahora podemos ver como hemos creado la ramaminiproyecto1 y además hemos cambiado a esa rama. Los cambios que efectuemos ahora por ejemplo en el fichero, solo estarían guardados en la ramaminiproyecto1.

Algunos comandos importantes cuando estamos hablando de ramas son los siguientes:

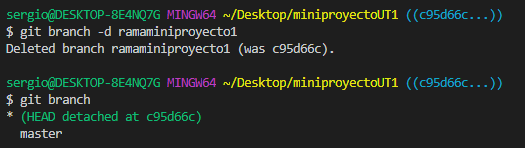
**Git Branch o git Branch -a:** nos permite ver las ramas que tiene el repositorio y con el modificador -a podemos además ver las ramas en el repositorio remoto:



**Git checkout nombreRama**: nos permite cambiar de una rama a otra:

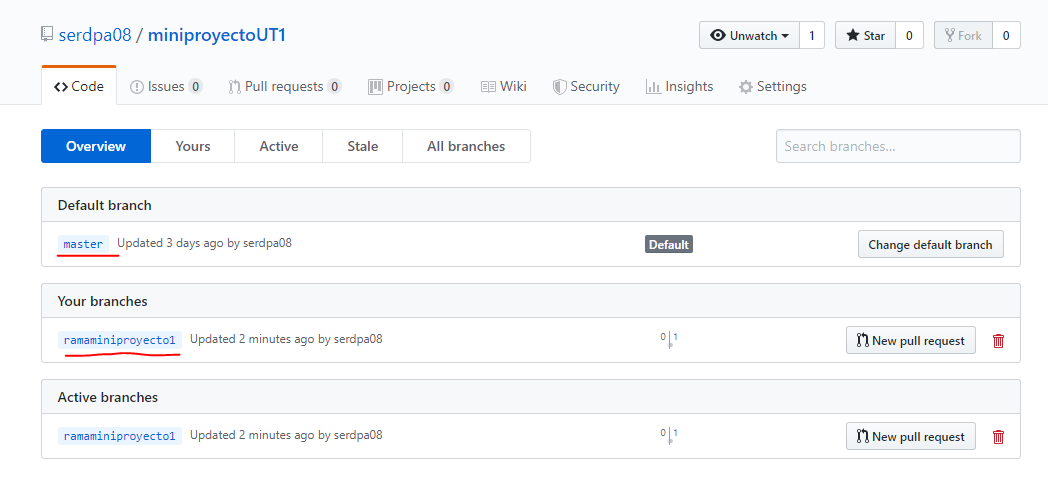


**Git brach -d nombreRama:** con este comando podemos borrar una rama.

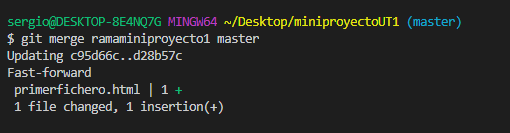


**Git merge ramaafusionar ramadestino**: el comando merge sirve para fusionar dos ramas, sirve sobretodo cuando queremos dejar la rama final del proyecto como el conjunto de las ramas.

Para probar el merge, lo primero que he hecho es volver a crear una rama y hacer cambios en un fichero, así esa rama ya es independiente de la master. Además lo he subido al repositorio remoto mediante ssh como ya hemos visto anteriormente, podemos ver que ahora en github existen dos ramas:



Ahora ya podemos hacer el merge y entonces la rama master ya tendrá los cambios que habíamos realizado en el fichero solo con la ramaminiproyecto1



Podemos ver que solo hay 1 fichero cambiado porque es donde realice el cambio.

# ELEMENTOS HTML5

Damos por finalizado el apartado de GIT por esta unidad, ahondaremos más en próximas unidades e iré ampliando conceptos.

Ahora vamos a comenzar con los elementos de marcado de HTML5.

Voy a hacer una división en los siguientes campos:

* Elementos de bloque.
* Elementos en línea.
* Formularios y sus componentes.
* Tablas.
* Elementos interactivos.