

GitHub

Entornos de Desarrollo
Eduardo Martínez Romero

1 DAW
2020-2021

Se trata de un sistema de gestión de proyectos y **control de versiones remoto**, el cual implementa herramientas sociales y colaborativas **para facilitar la planificación y el trabajo en equipo.**



GitHub

Primeros pasos

| 1

Para comenzar, vamos a **crear y acceder** a nuestra cuenta de GitHub. Este servicio es **gratuito**, pero a su vez ofrece características de pago que pueden interesar al proyecto.

Overview Repositories 2 Projects Packages

Popular repositories [Customize your pins](#)

pro
Repository del módulo de programación 20/21. IES Doñana
Python

vampiro
Forked from iesdonana/vampiro
Python

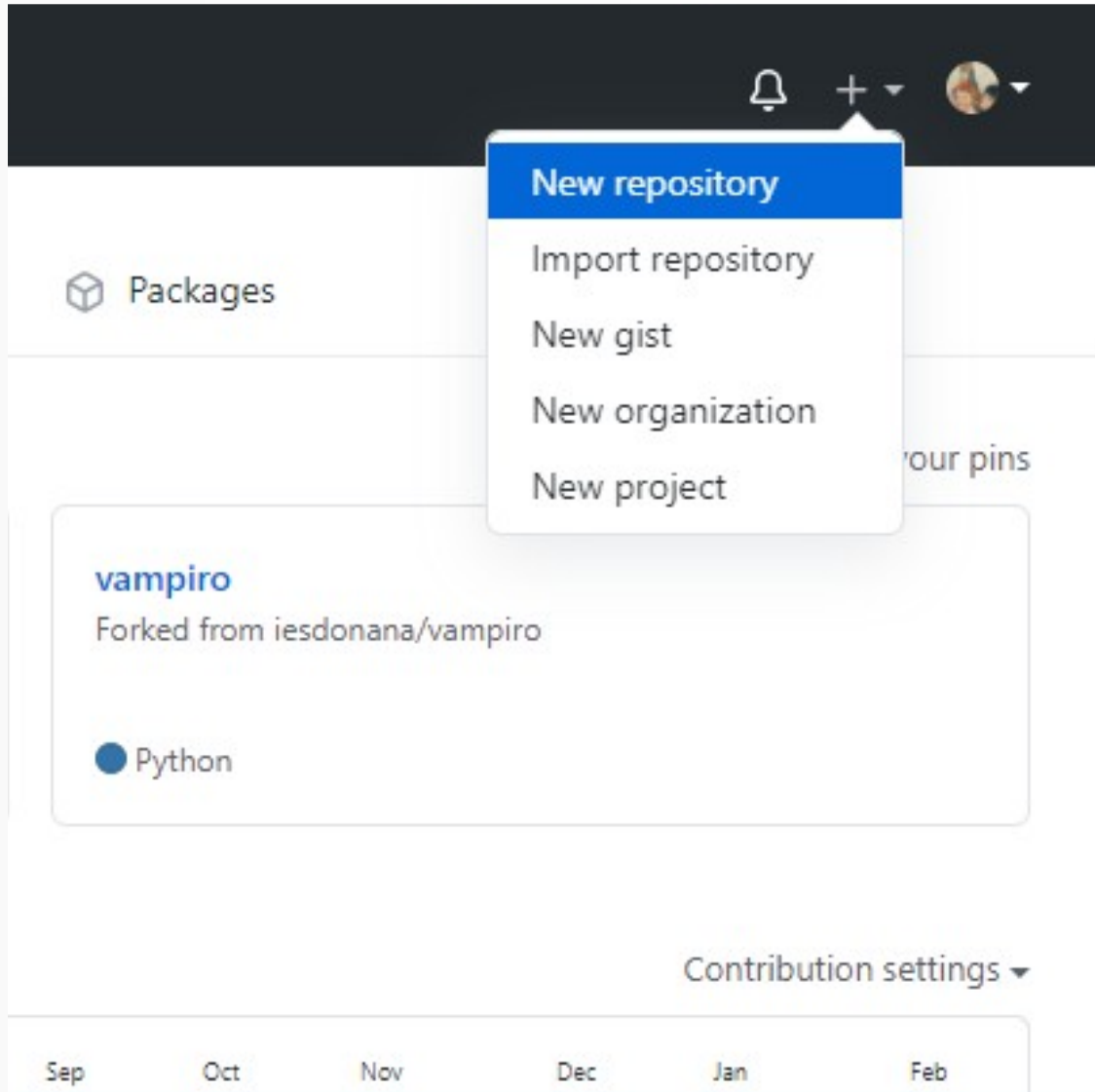
Harry Plotter
edumarrom

6 contributions in the last year [Contribution settings](#)

Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb

Creando un repositorio

| 2

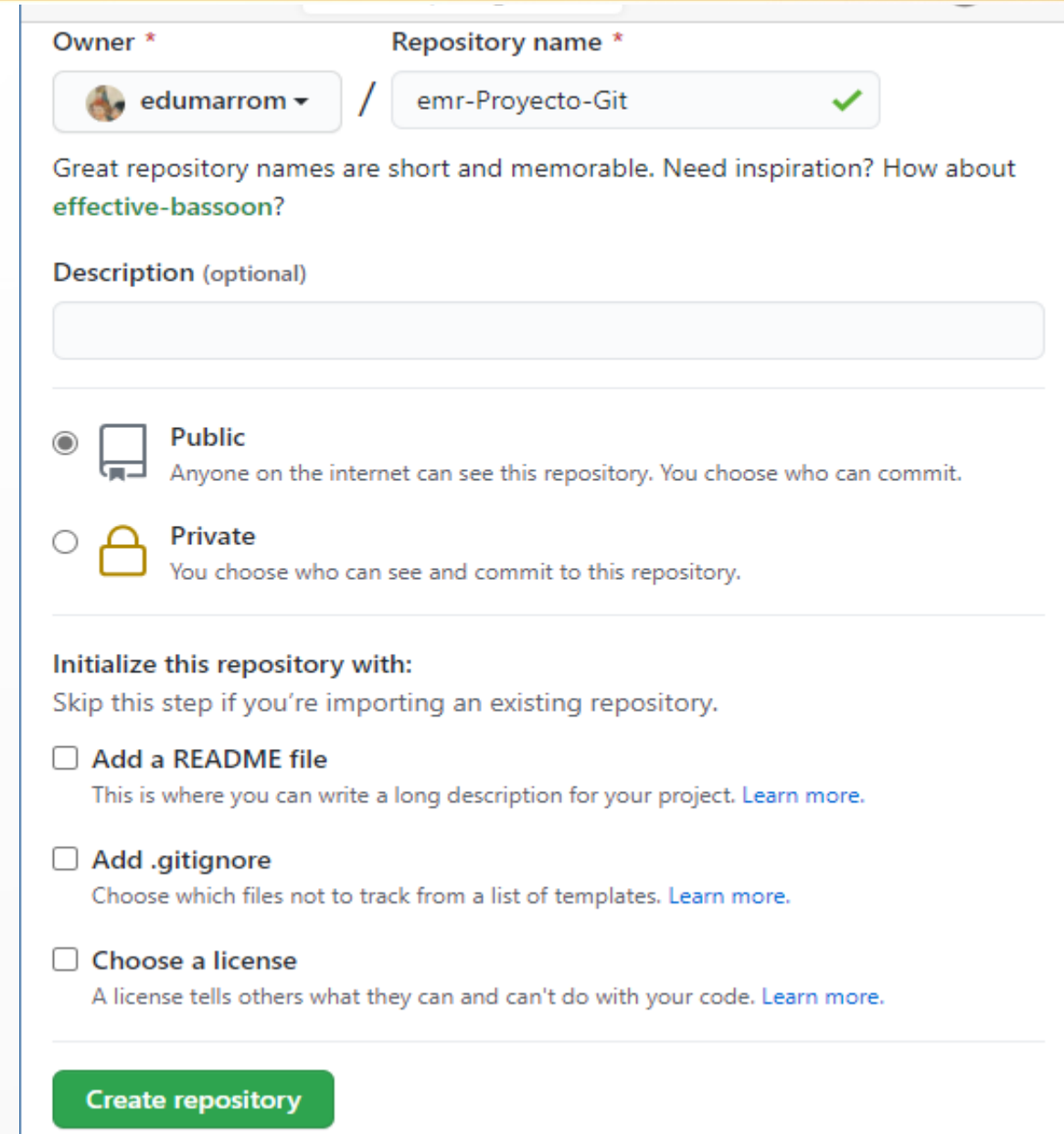


Vamos a comenzar nuestra travesía por GitHub **creando un nuevo repositorio**. La interfaz web nos pone las cosas muy fáciles. Tan pronto estamos *logeados* encontramos un botón + (más) junto a nuestra imagen de perfil. Vamos a clicar en él y en las opciones disponibles seleccionamos *New repository*.

Creando un repositorio

2

El asistente para crear un nuevo repositorio nos permite inicializar algunos archivos comunes y recomendados en un repositorio, como son *README*, *LICENSE* o *.gitignore*. Estos conceptos son ahora nuevos y por el momento los vamos a ignorar.




The screenshot shows the GitHub 'Create new repository' form. At the top, the 'Owner' is set to 'edumarrom' and the 'Repository name' is 'emr-Proyecto-Git', which is marked as valid with a green checkmark. Below this, a message states: 'Great repository names are short and memorable. Need inspiration? How about [effective-bassoon?](#)'. There is a text input field for an optional 'Description'. The 'Visibility' section has two options: 'Public' (selected with a radio button and a computer icon) and 'Private' (unselected with a radio button and a lock icon). The 'Initialize this repository with:' section includes a note to skip the step if importing an existing repository, followed by three unchecked checkboxes: 'Add a README file' (with a link to learn more), 'Add .gitignore' (with a link to learn more), and 'Choose a license' (with a link to learn more). A green 'Create repository' button is at the bottom.


Owner * Repository name *

edumarrom / emr-Proyecto-Git ✓

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

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

Importando nuestro *repo*

3

Una vez creado , la página nos ofrece información de ayuda sobre cómo crear o importar un repositorio local de nuestro equipo. En nuestro caso, nos interesa **importarle nuestro repositorio local** de las prácticas anteriores. **Añadimos un origen¹ remoto** pasando la url de nuestro repositorio remoto. **Renombraremos** nuestra rama principal por main² en lugar de master*. Finalmente vamos a **exportar³** el contenido del proyecto local al destino previamente definido.

1. `git remote add origin https://github.com/edumarrom/emr-Proyecto-GIT.git`
2. `git branch -M main`
3. `git push -u origin main`

```
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (master)
$ git remote add origin https://github.com/edumarrom/emr-Proy
ecto-GIT.git

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (master)
$ git branch -M main

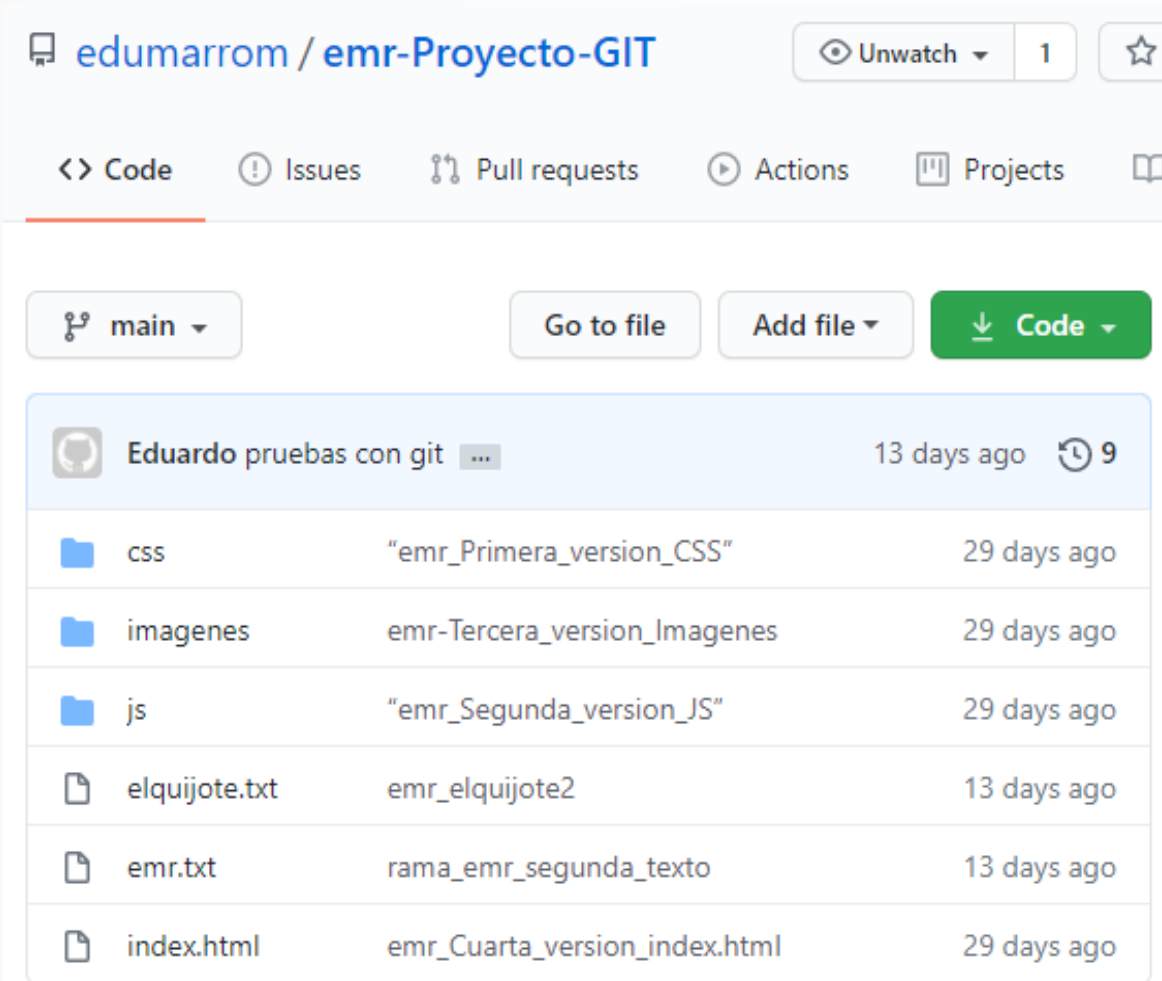
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git push -u origin main
Enumerating objects: 49, done.
Counting objects: 100% (49/49), done.
Delta compression using up to 4 threads
Compressing objects: 100% (45/45), done.
Writing objects: 100% (49/49), 867.97 KiB | 5.29 MiB/s, done.
Total 49 (delta 16), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (16/16), done.
To https://github.com/edumarrom/emr-Proyecto-GIT.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'orig
in'.

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$
```

*. Esto es una norma de estilo que GitHub recomienda, aunque no es obligatorio.

Importando nuestro *repo*

4



Si echamos un vistazo a nuestro repositorio remoto, podremos observar que **ahora posee todos los ficheros de nuestro repositorio local.**

Modificando desde GitHub | 5

Para poner a prueba la sincronización entre repositorio local y remoto, vamos a crear desde la interfaz web un archivo *README* y realizar un *commit*.

main Go to file Add file Code

Eduardo pruebas con git 13 days ago 9

css	"emr_Primer_version_CSS"	29 days ago
imagenes	emr-Tercera_version_Imagenes	29 days ago
js	"emr_Segunda_version_JS"	29 days ago
elquijote.txt	emr_elquijote2	13 days ago
emr.txt	rama_emr_segunda_texto	13 days ago
index.html	emr_Cuarta_version_index.html	29 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

emr-Proyecto-GIT / README.md in main

Edit new file

Preview

```
1 # emr-Proyecto-GIT
```


Modificando desde GitHub | 6

Una vez confirmemos los cambios, nos vamos a dirigir a nuestro repositorio local, para luego **importar los cambios** que pudieran haber en nuestro repositorio remoto de GitHub.

```
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 676 bytes | 29.00 KiB/s, done.
From https://github.com/edumarrom/emr-Proyecto-GIT
   9acc045..f80bf92  main       -> origin/main
Updating 9acc045..f80bf92
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ |
```

Commit new file

Creado archivo README

Add an optional extended description...

- ☒ Commit directly to the `main` branch.
- ☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file

Mediante la orden `git pull` podremos realizar esta actualización, que nos traerá el fichero *README.md*.

Clonando un repositorio

7

```
edumarrom@Dama-dama MINGW64 ~
$ ls emr-Proyecto-GIT/
README.md  elquijote.txt  imagenes/  js/
css/       emr.txt        index.html

edumarrom@Dama-dama MINGW64 ~
$ rm -rf emr-Proyecto-GIT/

edumarrom@Dama-dama MINGW64 ~
$ ls emr-Proyecto-GIT/
ls: cannot access 'emr-Proyecto-GIT/': No such file or directory

edumarrom@Dama-dama MINGW64 ~
$ git clone https://github.com/edumarrom/emr-Proyecto-GIT.git
Cloning into 'emr-Proyecto-GIT'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 52 (delta 17), reused 48 (delta 16), pack-reused 0
Receiving objects: 100% (52/52), 868.58 KiB | 359.00 KiB/s, done.
Resolving deltas: 100% (17/17), done.

edumarrom@Dama-dama MINGW64 ~
$
```

Vamos a ir un paso más allá en la prueba, **borrando** al completo nuestro repositorio local para después volver a importarlo desde nuestra versión remota de GitHub.

Al principio puede asustar un poco, pero no hay nada de que temer. Con la orden `git clone [URL]` **obtendremos una copia exacta** de nuestro repositorio de GitHub. Recuerda que al realizar la importación se creará una carpeta con el nombre del repositorio, por lo que no será necesario que la creamos.

Clonando un repositorio

7

Si entramos de nuevo en la carpeta de de nuestro proyecto y revisamos su estado, veremos que **todo está tal y como estaba antes de borrar** nada, y si revisamos el historial de versiones tenemos guardadas todas la versiones anteriores*.

```
edumarrom@Dama-dama MINGW64 ~  
$ cd emr-Proyecto-GIT  
  
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)  
$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
nothing to commit, working tree clean  
  
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)  
$ git log --oneline  
f80bf92 (HEAD -> main, origin/main, origin/HEAD) Creado arch  
ivo README  
9acc045 pruebas con git  
066a1f7 emr_elquijote2  
e8ddd40 emr_elquijote  
4dc0a39 rama_emr_segunda_texto  
6d0ca8d rama_emr_texto  
86d3259 emr_Cuarta_version_index.html  
3070aee emr-Tercera_version_Imagenes  
39544e0 "emr_Segunda_version_JS"  
89e7351 "emr_Primer_version_CSS"  
  
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)  
$
```

*. La fusión aparece como un cambio más en el historial, con el nombre *pruebas con git*.

Modificando desde Git

| 8 - 9

Crearemos un nuevo fichero con el nombre saludo.txt. Revisamos su estado para verificar que git continúa funcionando correctamente. Añadimos el fichero al área de preparación y confirmamos una nueva versión.

```
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ echo Hola Mundo > saludo.txt

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git status --short
?? saludo.txt

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git add saludo.txt
warning: LF will be replaced by CRLF in saludo.txt.
The file will have its original line endings in your working
directory

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git commit -m "emr_saludo"
[main edda191] emr_saludo
1 file changed, 1 insertion(+)
create mode 100644 saludo.txt
```

```
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git log --oneline
edda191 (HEAD -> main) emr_saludo
f80bf92 (origin/main, origin/HEAD) Creado archivo README
9acc045 pruebas con git
066a1f7 emr_elquijote2
e8ddd40 emr_elquijote
4dc0a39 rama_emr_segunda_texto
6d0ca8d rama_emr_texto
86d3259 emr_Cuarta_version_index.html
3070aee emr-Tercera_version_Imagenes
39544e0 "emr_Segunda_version_JS"
89e7351 "emr_Primer_version_CSS"
```

Al comprobar de nuevo el estado **observaremos la advertencia** *Your branch is ahead of 'origin/main' by 1 commit*. Ésto significa que ahora mismo nuestro repositorio local **está más actualizado** que su contraparte remota.


```
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local
object.
To https://github.com/edumarrom/emr-Proyecto-GIT.git
f80bf92..edda191 main -> main

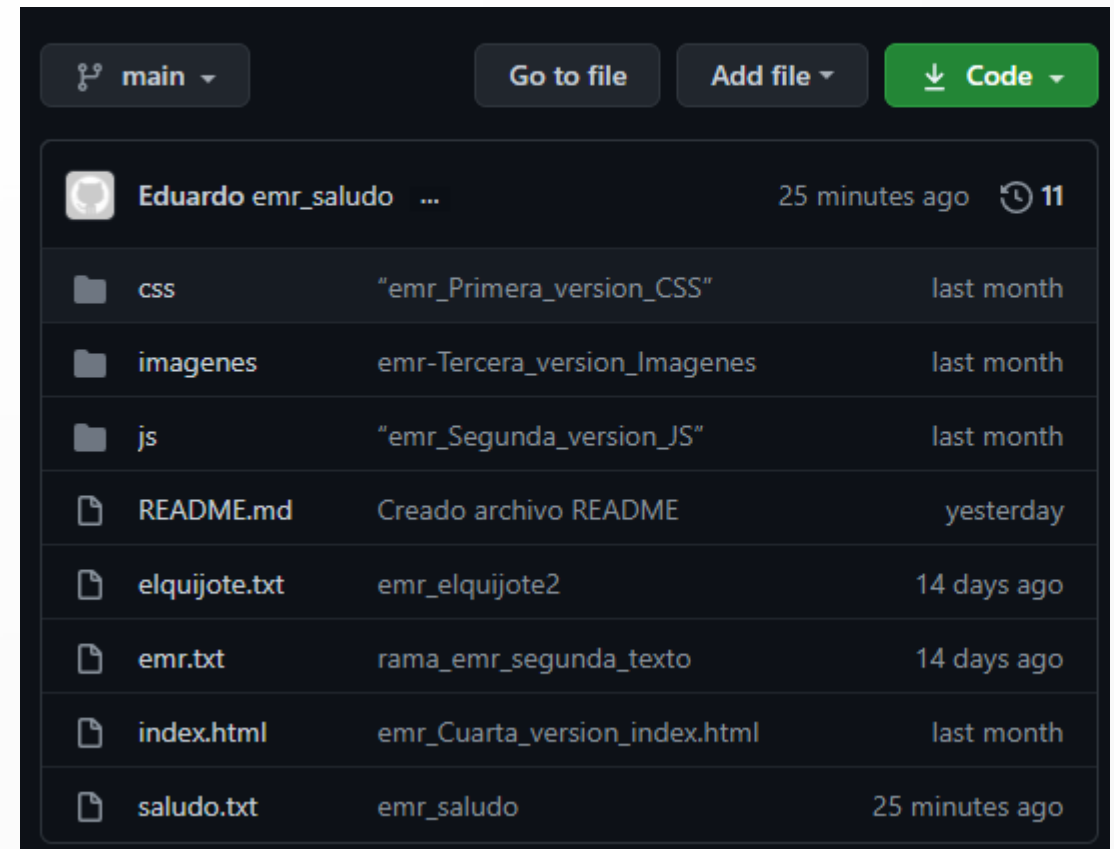
edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

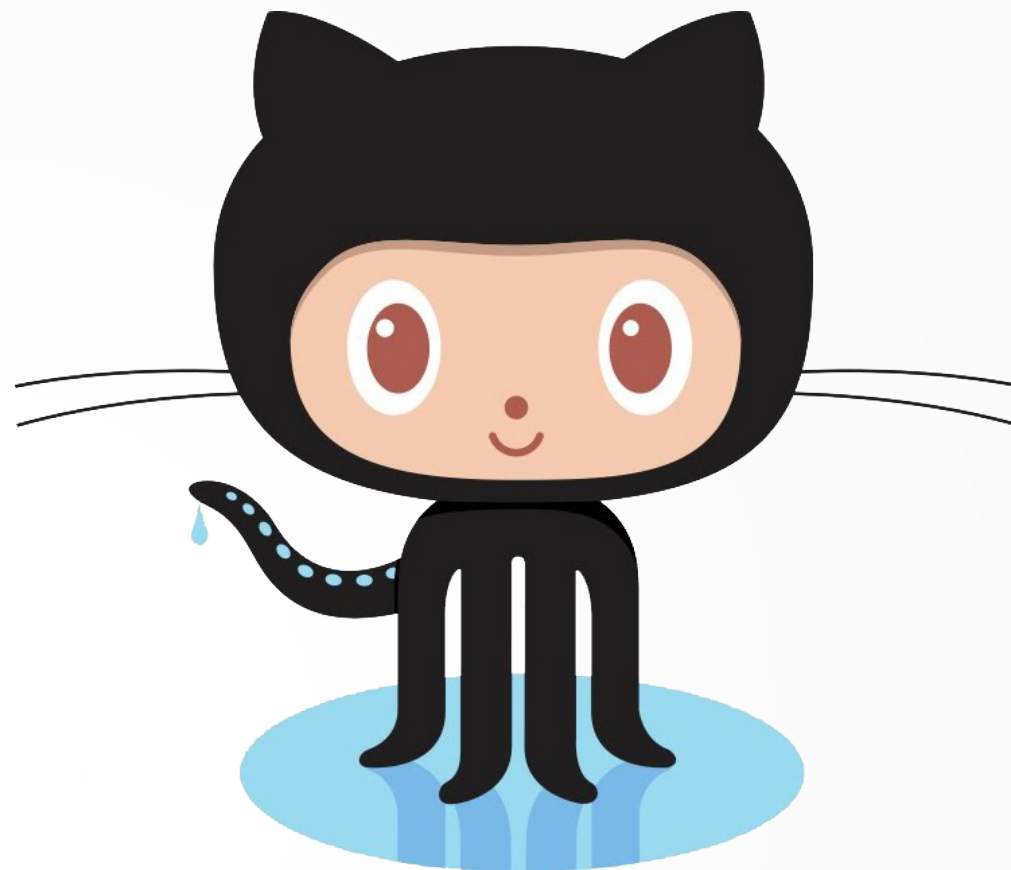
nothing to commit, working tree clean

edumarrom@Dama-dama MINGW64 ~/emr-Proyecto-GIT (main)
$ git log --oneline
edda191 (HEAD -> main, origin/main, origin/HEAD) emr_saludo
f80bf92 Creado archivo README
9acc045 pruebas con git
066a1f7 emr_elquijote2
e8ddd40 emr_elquijote
4dc0a39 rama_emr_segunda_texto
6d0ca8d rama_emr_texto
86d3259 emr_Cuarta_version_index.html
3070aee emr-Tercera_version_Imagenes
39544e0 "emr_Segunda_version_JS"
89e7351 "emr_Primer_version_CSS"
```

A continuación vamos a **exportar** nuestros cambios al repositorio remoto mediante la orden `git push`. Cuando volvamos a verificar el estado del proyecto, nos indicará que **nos encontramos al día** con nuestra contraparte remota.

Si revisamos nuestro repositorio remoto desde la interfaz de GitHub, el fichero *saludo.txt* también se encuentra aquí. Es interesante observar que podemos ver de un vistazo la **última versión** que modificó a un fichero, así como su **fecha de creación**.





Ahora que conocemos los fundamentos de git y GitHub, podremos tener un repositorio remoto, fácil de mantener y disponible donde y cuando queramos.

Gracias