

Material complementario

CODERHOUSE

Clase 11 - GITHUB

GitHub

Por ahora todo lo que venía ocurriendo en Git era de manera local, no necesitábamos nada de internet para guardar nuestros commits y nuestro repositorio.

Ahora **queremos compartir nuestro trabajo con otros** (compañeros de proyecto, clientes, etc).

Para eso utilizamos Github!

[GitHub](#)

GitHub es una plataforma de desarrollo colaborativo dedicada a alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails desarrollado por GitHub, Inc. (anteriormente conocida como Logical Awesome). El código de los proyectos alojados en GitHub se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

Creando una cuenta en GitHub

CODERHOUSE

Material complementario

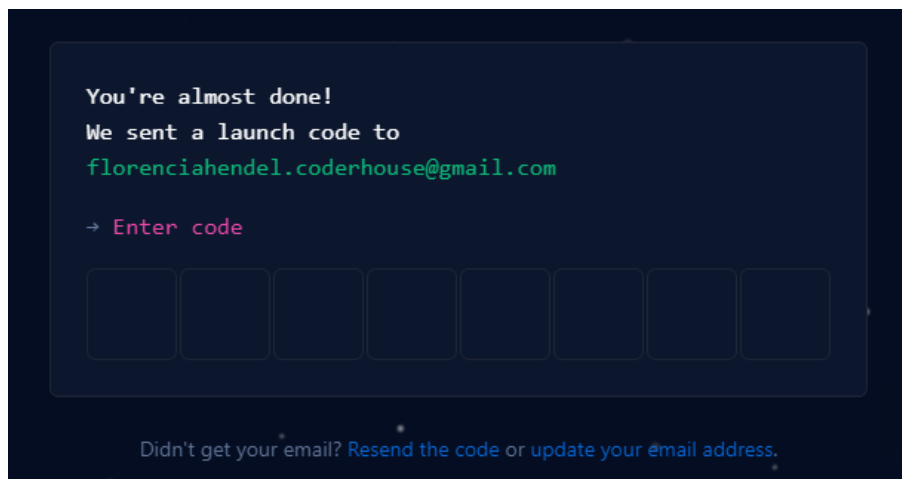
CODERHOUSE

Entra a la página de registro de GitHub <https://github.com/> y desde allí crea tu cuenta. Deberás ir completando una serie de datos:

- Email: debe ser el mismo que usaste para Git.
- Password: una contraseña que puedas recordar luego.
- Username: será el nombre con el que te encuentren los demás usuarios. Elige uno que sea fácil de recordar, te identifique, y que puedas compartir con tus colegas y clientes.
- Would you like to receive product updates and announcements via email? Puedes elegir si quieres recibir (o no) información sobre actualizaciones y publicidad de GitHub por email.

Antes de poder completar el registro, deberás realizar una verificación para demostrar que no eres un robot.

Una vez realizada, haz click en create account. Deberías ver una pantalla como la de la imagen.



Ingresa el código de 8 dígitos que fue enviado a tu email, y habrás completado la creación de tu cuenta.

¡Felicitaciones! Ya estás listo para empezar a crear tu propio repositorio online.

CODERHOUSE


Material complementario

CODERHOUSE

Crear un repositorio remoto


Una vez dentro de tu cuenta, podrás ver en la parte superior una serie de pestañas. La opción “Repositories” te permite ver tus repositorios creados o, en este caso, crear tu primer repositorio.


Owner * Repository name *

 FlorHCoderhouse ▾ / mi_repositorio ✓

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

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.

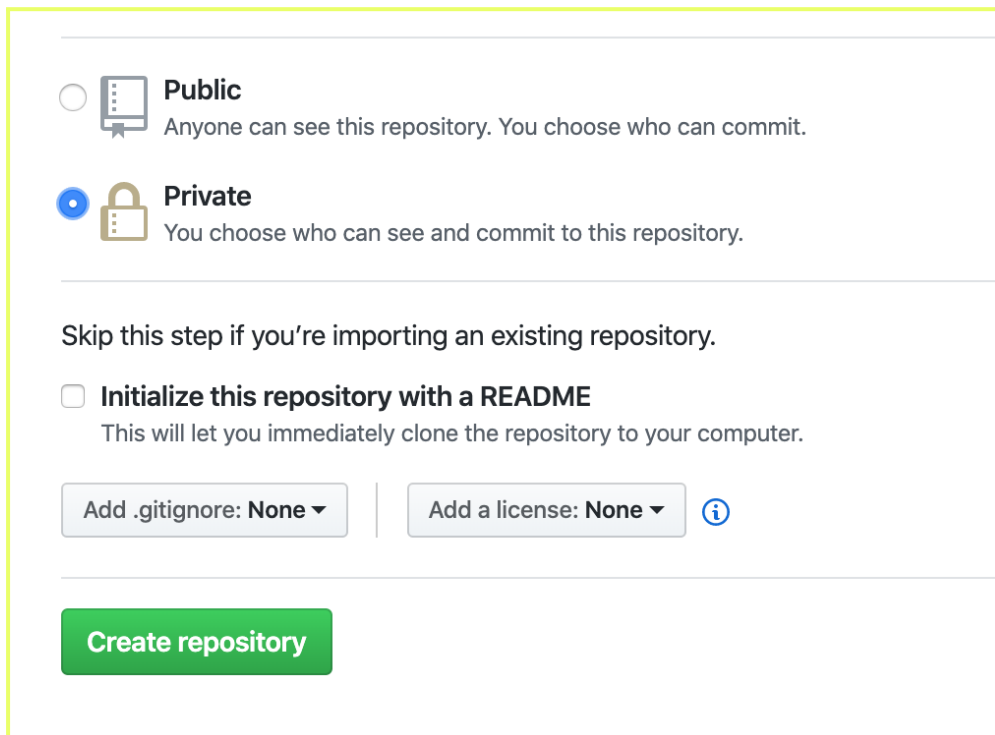
Por ejemplo, podría ser llamado “mi_repositorio”, para que pruebes con los archivos que trabajaste en el desafío de GIT.


CODERHOUSE


Material complementario

CODERHOUSE

Configuración del repositorio




☐  **Public**
Anyone can see this repository. You choose who can commit.

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

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼ | Add a license: **None** ▼ 

Create repository

Elegimos “público” o “privado”. Si bien con privado limitamos el acceso a cualquier persona, no nos permitirá mostrar nuestro código como página web, por lo que elegimos “público”.

README

El Readme podría ser el primer archivo de nuestro repositorio en donde le contemos a la comunidad de GitHub **de qué trata nuestro proyecto**.

También podemos adjudicar algún tipo de licencia a nuestro proyecto. Si quieres saber más sobre licencias:

[Licencia de software – Wikipedia, la enciclopedia libre](#)

[Choose a License](#)

CODERHOUSE

Material complementario

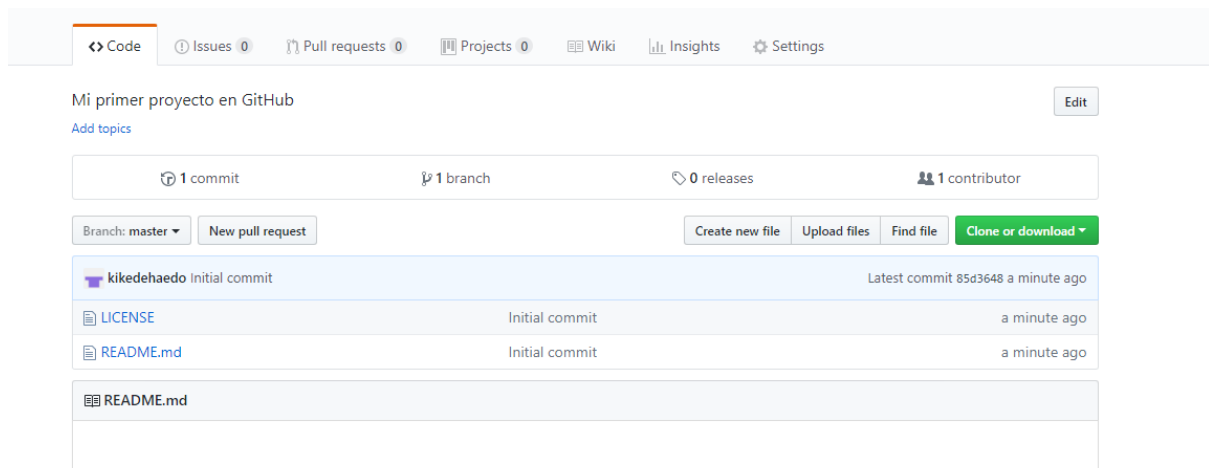
CODERHOUSE

Luego de ingresar esa información dar click en Crear repositorio (Create repository)

Clonando repositorios remotos

Git clone

Una vez creado vamos a encontrar el detalle de nuestro nuevo repositorio



En la parte superior derecha encontraremos un botón verde, el cual nos va a dar las siguientes opciones:

- Si queremos clonar este repositorio
- Si solamente queremos descargarlo en un archivo Zip.

Para clonarlo tenemos dos opciones:

-HTTPS

-SSH

Elijamos por el momento la versión HTTPS, debemos copiar la URL que nos proporcione.

CODERHOUSE

Material complementario

CODERHOUSE

Una vez copiado este enlace, volvamos a nuestra terminal y con el comando *git clone* + [LA URL QUE COPIAMOS DE NUESTRO REPOSITORIO] clonamos los archivos del proyecto dentro de la carpeta, en este caso, llamada: "miproyecto" :

```
john@MyShopSolutions MINGW64 /c/  
$ git clone /*Aquí la URL*/  
Cloning into 'mi_repositorio'...  
remote: Counting objects: 4, done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (4/4), done.
```

Si listamos los archivos en consola vamos a ver los archivos de nuestro repositorio:

```
john@MyShopSolutions MINGW64 /c/miproyecto (master)  
$ dir  
LICENCE README.md
```

Git remote

Si queremos **generar un enlace entre un repositorio local y un repositorio remoto** podemos hacerlo por medio del comando *git remote*

Vamos a vincular nuestro repositorio local "mi_repositorio" a nuestro nuevo repositorio en gitHub:

```
/* Paso 1: Debes ubicarte en tu repositorio local*/  
john@MyShopSolutions :~$ cd  
Documents/Proyectos_Coder/mi_repositorio  
  
/* Paso 2: Indica cuál será tu nuevo repositorio remoto. Debes  
reemplazar miuser por tu nombre de usuario elegido en github */  
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$  
git remote add origin https://github.com/miuser/mi_repositorio.git
```

CODERHOUSE

Material complementario

CODERHOUSE

```
/*Paso 3: Este comando renombra la rama master a main, es el  
nombre que usan los repositorios modernos*/  
john@MyShopSolutions: ~$ git branch -M main
```

En resumen:

`git remote add [origin] [url del repo]` Conecta un repositorio con nuestro equipo local.

`git remote -v` Lista las conexiones existentes.

`git remote remove [origin]` Elimina una conexión con algún repositorio.

Como buenas prácticas, se recomienda lo siguiente:

- El nombre del repositorio creado en Github (u otra plataforma) debe ser el mismo que el nombre de la carpeta del proyecto.
- Primero crear el repositorio remoto en Github, luego clonarlo al repositorio local en tu máquina y de ahí empezar a trabajar.

Git pull - Git fetch

Ahora vamos a manejar los cambios para que se vean respaldados en GitHub.

Lo primero que vamos a hacer es **traernos archivos desde el repositorio remoto en GitHub**. Para eso vamos a usar el comando `git fetch`

```
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio  
(main) $ git fetch origin main  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (8/8), done.  
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
From https://github.com/miuser/mi_repositorio.git  
* branch          main      -> FETCH_HEAD
```

CODERHOUSE

Material complementario

CODERHOUSE

```
* [new branch]      main      -> origin/main
```

Le decimos a nuestra terminal que se traiga los archivos del repositorio llamado "origin" y de la rama "main"

Nos ha creado una nueva rama (origin/master), no se han mezclado, ahora tenemos que **mezclar los cambios que están en esta nueva rama**, y para eso usamos el ya conocido git merge

```
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio  
(main) $ git merge origin/main
```

Ahora los cambios (nuevos archivos, cambios en archivos, etc) se han fusionado con nuestros archivos locales.

En caso de tener algún tipo de error al momento de fusionar recuerda que puedes agregar al comando *git merge*... lo siguiente siguiente:

```
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio  
(main)$ git merge origin/main --allow-unrelated-histories
```

Este anexo a nuestro comando *merge* básicamente **permite la fusión cuando se niega a fusionar historias que no comparten un antecesor común**. Esta opción se puede utilizar para anular esta seguridad al fusionar historias de dos proyectos que comenzaron sus vidas de forma independiente. Como es una ocasión muy rara, no existe ninguna variable de configuración para habilitar esto por defecto y no se agregará por el momento.

¿Qué tal si hubiera una forma que hiciera estas dos cosas al mismo tiempo? Para ello existe el comando *git pull*

```
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio  
(main)$ git pull origin main
```

CODERHOUSE

Material complementario

CODERHOUSE

Luego nos aparece en la terminal **confirmación de este merge** ¡y listo! ya tenemos los cambios en nuestro repositorio Local. Todo con un solo comando en la terminal.

Git Push

Ahora vamos a **enviar nuevos cambios a nuestro repositorio remoto**. Para ello vamos a usar un comando llamado *git push*, para empujar nuevos cambios.

```
/* Este comando envía todos los cambios que hayan sido
commiteados, al repositorio remoto*/
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$
git push -u origin main

/*En este punto, se abre una ventana que nos permite loguearnos y
autenticarnos desde el navegador, o a través de un código. Ambas
opciones terminan en un botón de autorizar a Git Credential
Manager*/
/*Una vez que completamos la autenticación, veremos el siguiente
resultado*/

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 407 bytes | 407.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/miuser/mi_repositorio.git
 * [new branch]      main -> main
Branch 'main' set up to track 'origin/main'.
```

Si chequeamos en nuestra página de GitHub vamos a encontrar este nuevo archivo en la rama main del proyecto.

CODERHOUSE

Material complementario

CODERHOUSE

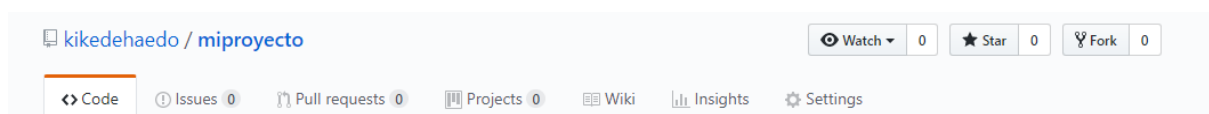
Podemos enviar otras ramas:

```
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$git  
t pull responsive
```

De esta manera enviamos la rama llamada “responsive” anteriormente creada con el comando *git branch*

Más Propiedades De Github

Como plataforma colaborativa, GitHub **ofrece a sus usuarios una gran cantidad de funcionalidades para la gestión de proyectos**, todas ellas apoyadas por la comunidad. Por esta razón, a lo mejor dentro un año tenga agregadas nuevas características que le permitan a los usuarios un mejor desenvolvimiento en el desarrollo de código.



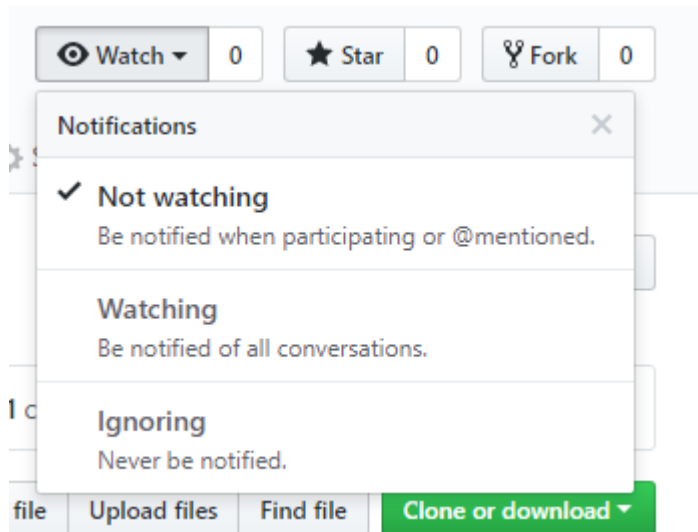
En la parte superior podríamos seguir o dejar de seguir al proyecto en donde estemos situados: tenemos las opciones Watching/Watch, Ignoring o Not Watching (Ver/Viendo, Ignorando, No Verlo)

En caso de que nos interesen los avances del proyecto, situarlo en Watching para así **poder ver los avances del mismo en nuestra página de inicio de GitHub.**

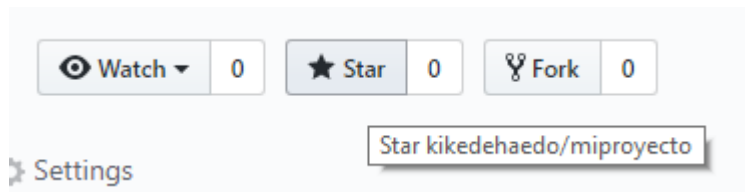
CODERHOUSE

Material complementario

CODERHOUSE



También podríamos “darle like” (Star) al proyecto



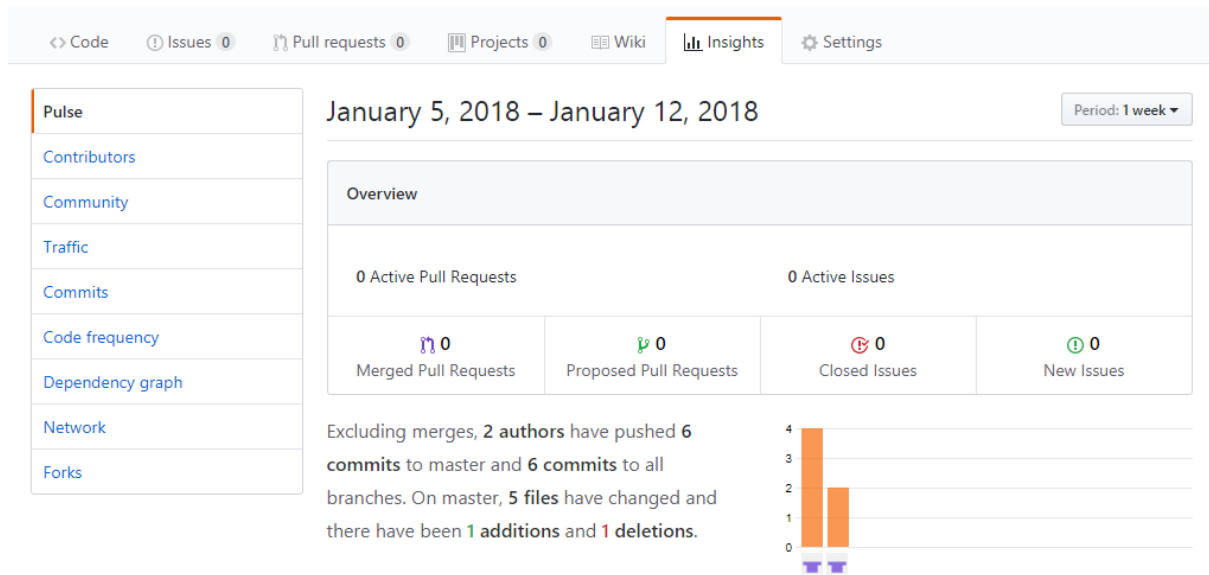
Y como última opción podríamos hacerle un Fork al proyecto (en caso de que sea un proyecto de otro usuario) y copiarlo. Este comando nos va a clonar el proyecto y crear un nuevo proyecto en nuestro GitHub.

En “Insights” podemos ver las estadísticas de nuestro proyecto:

CODERHOUSE

Material complementario

CODERHOUSE



En los Settings de nuestro repositorio podemos configurar distintos aspectos muy interesantes

The screenshot shows the GitHub Settings page for a repository. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings (selected). The left sidebar contains an 'Options' section with links to Collaborators, Branches, Webhooks, Integrations & services, and Deploy keys. The main content area is titled 'Settings'. Under the 'Repository name' section, the name 'miproyecto' is displayed with a 'Rename' button. The 'Features' section includes two checked options: 'Wikis' (with a description: 'GitHub Wikis is a simple way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support, or anything you wish.') and 'Restrict editing to collaborators only' (with a description: 'Public wikis will still be readable by everyone.').

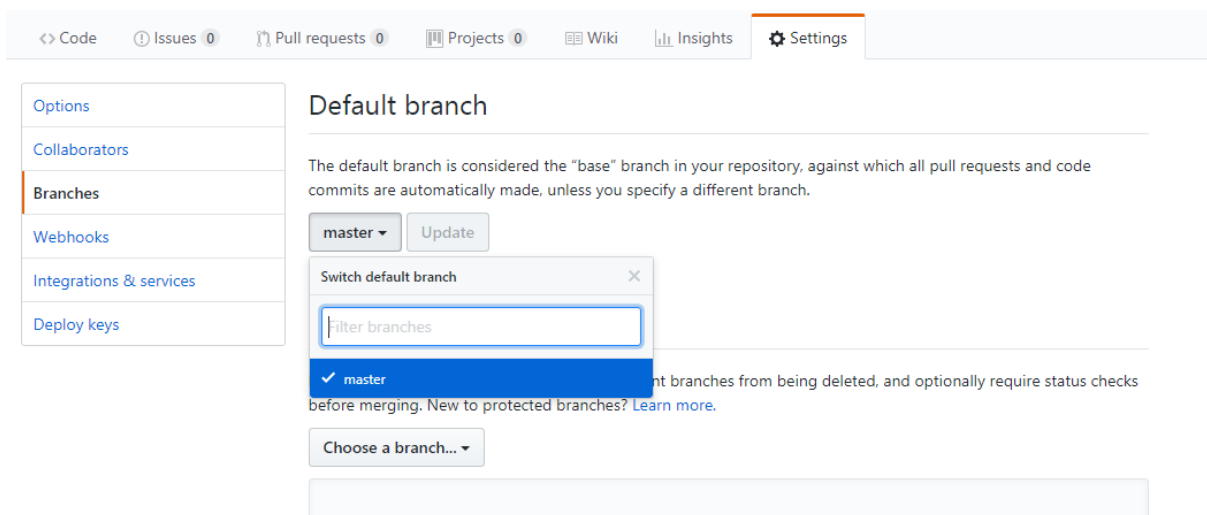
CODERHOUSE

Material complementario

CODERHOUSE

Por ejemplo, en la solapa “Collaborators” **podríamos sumar a otro usuario para que nos ayude con nuestro proyecto** y empiece a colaborar con commits sobre nuestros distintos documentos.

Otra opción a remarcar es la llamada “Branches”, para cambiar cuál sería nuestra rama principal



Y también podríamos **proteger a nuestra rama “master” (actualmente llamada “main”)**, de esa manera al protegerla mediante un “pull request” cada actualización que afecte la rama maestra queda a disposición del administrador del proyecto a ser cotejada y autorizada.

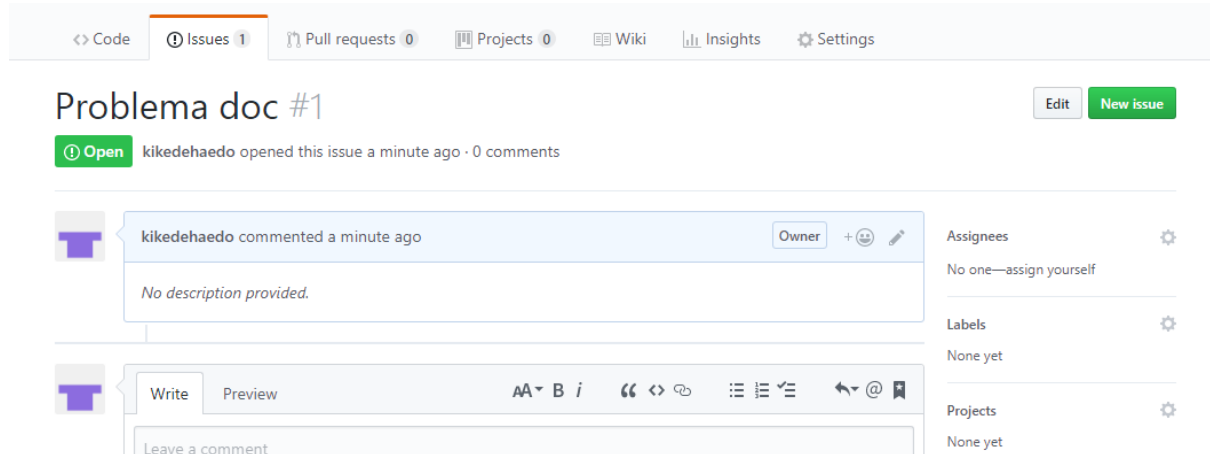
En la solapa “Issues” los diferentes colaboradores del proyecto **pueden remarcar**

CODERHOUSE

Material complementario

CODERHOUSE

y alertar acerca de diferentes “bugs” o “problemas” dentro del proyecto.



Para enterarte de todas las características de GitHub puedes visitar el siguiente link:

<https://help.github.com/>

GitHub Pages

GitHub nos **permite publicar nuestros proyectos online**. Para generar un GitHub page debemos ir a los “Settings” de nuestro repositorio y activar nuestro GitHub page, seleccionar qué rama queremos usar, guardamos los cambios y GitHub cumplirá la función básica de cualquier otro Hosting.

Importante: El proyecto solo de archivos estáticos, ningún archivo que requiera de BackEnd especial.

CODERHOUSE

Material complementario

CODERHOUSE

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾

Save

Select source

master branch

Use the master branch for GitHub Pages.

master branch /docs folder

Use only the /docs folder for GitHub Pages.

✓ None

Disable GitHub Pages.

Luego de seleccionar y salvar los cambios la página recargara y nos mostrará cuál es la URL de nuestro proyecto

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at <https://kikedehaedo.github.io/miproyecto/>.

Source

Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

master branch ▾

Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

CODERHOUSE

Material complementario

CODERHOUSE

NodeJS

Node.js es un entorno en tiempo de ejecución JavaScript, de código abierto y multiplataforma. Funciona para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript. Es asíncrono, con I/O (entrada y salida) de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Desarrollado por Ryan Dahl en 2009, y su evolución está apadrinada por la empresa Joyent.

Instalar NodeJS

Para compilar Sass vía la línea de comandos, primero necesitamos instalar NodeJS. Descárgalo del sitio oficial nodejs.org, abre el paquete y sigue el asistente de instalación.

Más información acerca de NODEJS y NPM:

[About | Node.js](#)

[Node.js NPM](#)

[Node Package Manager Guide: Install npm + Use Commands & Modules](#)

CODERHOUSE