

CURSO DE PROGRAMACIÓN FULL STACK

TUTORIAL: GUÍA DE GIT Y GITHUB CON RAMAS

video 1:

<https://www.youtube.com/watch?v=zQYKx3H06Rc&list=PLgwlfcqa5h3xypUOclgVoHegHt9xYL81U&index=1>

video 2:

<https://www.youtube.com/watch?v=Ww4wOBlrit4&list=PLgwlfcqa5h3xypUOclgVoHegHt9xYL81U&index=2>

video 3:

https://www.youtube.com/watch?v=_ELbWMFBBZE&list=PLgwlfcqa5h3xypUOclgVoHegHt9xYL81U&index=3



REPASO DE COMANDOS

Veamos un cuadro con los comandos que ya aprendimos

Comando	Explicación
Git Init	Para iniciar el trackeo de los archivos de una carpeta.
Git Status	Nos da toda la información necesaria sobre la rama actual.
Git Add	Para incluir los cambios del o de los archivos en tu siguiente commit.
Git Commit	Es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario. Es como un punto de guardado en un videojuego.
Git Push	Enviar archivos incluidos en el commit al repositorio local.
Git Push Origin	Enviar archivos incluidos en el commit al repositorio remoto.



Objetivos de la Guía

En esta guía aprenderemos a:

- Crear una nueva rama
- Clonar una rama
- Moverse entre ramas
- Fusionar ramas
- Borrar una rama
- Utilizar los nuevos comandos - pull request

RAMAS EN GIT

En la guía anterior vimos que una rama en Git es similar a la rama de un árbol. De manera análoga, una rama de árbol está unida a la parte central del árbol llamada tronco. Si bien las ramas pueden generarse y caerse, el tronco permanece compacto y es la única parte por la cual podemos decir que el árbol está vivo y en pie. De manera similar, una rama en Git es una forma de seguir desarrollando y codificando una nueva función o modificación del software y aún así no afectar la parte principal del proyecto. También vimos que **la rama principal o predeterminada en Git es la rama maestra o main.**

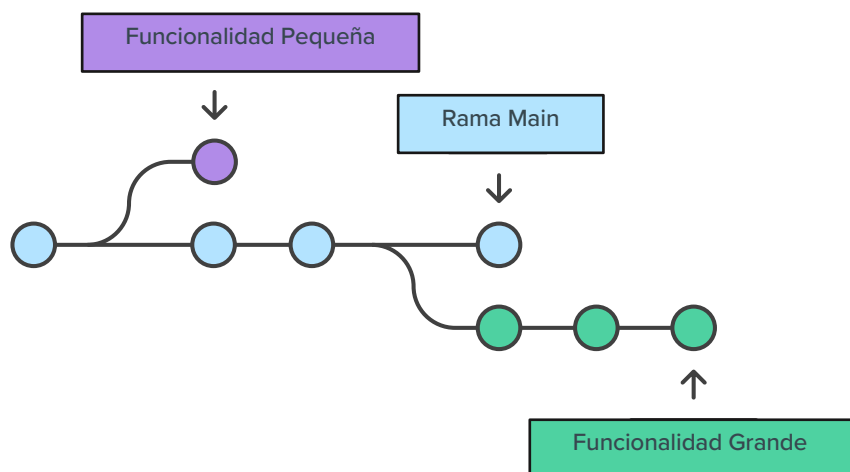
En esta guía vamos a ver como nos pueden ayudar para el trabajo grupal y de que manera podemos crear nuestra propia rama.

RAMAS PARA EL TRABAJO GRUPAL

Imaginemos que tenemos la siguiente situación, estamos trabajando x personas en un mismo proyecto y cada persona tiene que encargarse de una parte concreta del proyecto, ¿como hacemos para que no estemos todos pisando sobre el mismo proyecto? Aquí es donde entran en función las ramas.

Una rama en el trabajo grupal se ve como una **línea independiente de desarrollo**. Las ramas sirven como una abstracción de los procesos de cambio, preparación y confirmación (commit). Puedes concebirlas como una forma de solicitar un nuevo directorio de trabajo o un nuevo entorno de ensayo.

Las ramas son **espacios de desarrollo independientes que emergen de la rama principal** cuando quieres añadir una nueva función o solucionar un error, independientemente de su tamaño, generas una nueva rama para alojar estos cambios. Esto hace que resulte más complicado que el **código inestable se fusione con el código base principal**, y te da la oportunidad de limpiar tu historial futuro antes de fusionarlo con la rama principal.



El diagrama anterior representa un repositorio con dos líneas de desarrollo aisladas, una para una funcionalidad pequeña y otra para una funcionalidad más extensa. Al desarrollarlas en ramas, no solo es posible **trabajar con las dos de forma paralela**, sino que también se evita que el código dudoso se fusione con la rama main.

Explicándolo de otra manera, la rama main va a ser donde este alojado el proyecto final, el proyecto que va a tener todo lo que se trabaje en las demás ramas y las ramas van a ser los lugares en donde cada persona trabajara su parte del proyecto sin afectar al proyecto final que se encuentra en la rama main.

Ahora veremos como se crea una rama propia, como borrarlas y como unir nuestras ramas con la rama principal (rama main)

GIT BRANCH Y GIT CHECKOUT

Para trabajar con ramas son varios los comandos los que podemos usar, pero dos de los más importantes van a ser **git branch**, que nos va a servir para ver todas las ramas existentes, crear ramas, borrar ramas, renombrar ramas, etc. En cambio, **git checkout** va a ser nuestro aliado para cuando queramos movernos entre ramas.

VER RAMAS

Para ver las ramas en un repositorio Git, ejecuta el comando:

```
git branch
```

Para ver tanto las ramas de seguimiento remoto como las ramas locales, ejecuta el comando:

```
git branch -a
```

Habrás un asterisco (*) junto a la rama en la que te encuentras actualmente.



¿NECESITAS UN EJEMPLO?

```
prueba_git — git branch -a — git — less ◀ git branch -a — 80x23
develop
* master
service
remotes/origin/HEAD -> origin/master
remotes/origin/develop
remotes/origin/master
remotes/origin/service
(END)
```

A veces para salir de esta pantalla deberemos escribir **:wq**.

MOVERSE ENTRE RAMAS

Para moverse a una rama existente, ejecutamos el comando:

```
git checkout <nombre-de-tu-rama>
```

Generalmente, Git no permitirá moverse a otra rama a menos que el directorio en el que estás trabajando esté limpio, porque podrías perder cualquier cambio en el directorio de trabajo que no esté confirmado.

Podemos obtener una descripción más detallada de las ramas con este otro comando:

```
git show-branch
```

Esto nos muestra todas las ramas del proyecto con sus commits realizados. La salida sería como la de la siguiente imagen.



¿NECESITAS UN EJEMPLO?

```
[→ Git git:(ramaGit) git show-branch
! [master] Primer commit
* [ramaGit] Primer commit
--
+* [master] Primer commit
→ Git git:(ramaGit)
```

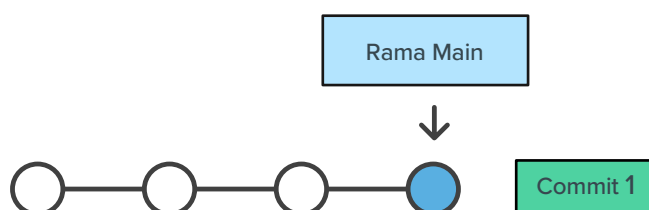


Revisemos lo aprendido hasta aquí

- Ver todas las ramas
- Moverse entre ramas

CREAR UNA RAMA NUEVA

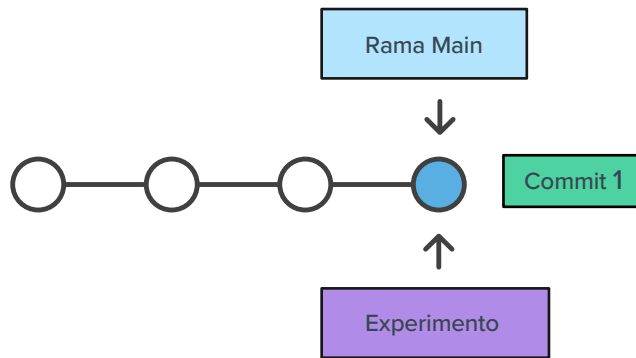
Es importante comprender que las ramas son solo punteros a los commits. Cuando creas una rama, todo lo que Git tiene que hacer es crear un nuevo puntero, no modifica el repositorio de ninguna otra forma. Esto quiere decir que la rama se va a crear con una copia del ultimo commit de la rama main. Si empiezas con un repositorio que tiene este aspecto:



Y, a continuación, creas una rama con el siguiente comando:

```
git branch experimento
```

Nos quedará una rama con el mismo commit y parado en el espacio de tiempo que el ultimo cambio que llegó a la rama main:



Ten en cuenta que este comando **solo crea la nueva rama** tendrás que usar `git checkout` para **moverte a ella** y a continuación, utilizar los comandos estándar `git add` y `git commit`. Pero, **hay un atajo para crear y moverte** a la nueva rama al mismo tiempo. Puedes pasar la **opción b** (para rama) con `git checkout`. Los siguientes comandos hacen lo mismo:

Método de 2 pasos

```
git branch <nombre-de-tu-rama>
```

```
git checkout <nombre-de-tu-rama>
```

Atajo

```
git checkout -b <nombre-de-tu-rama>
```

Si nos fijamos **no solo** creamos una nueva rama local, sino que ahora nos paramos en la nueva rama que creamos.



```
[→ Git git:(master) git checkout -b ramaGit
Switched to a new branch 'ramaGit'
→ Git git:(ramaGit) █
```

Si corriéramos un `git show-branch` podríamos ver la rama nueva ya tiene el primer commit que realizamos en la rama main porque como explicamos más arriba, estamos clonando la rama main.



```
[→ Git git:(ramaGit) git show-branch
! [master] Primer commit
* [ramaGit] Primer commit
--
+* [master] Primer commit
→ Git git:(ramaGit) █
```

```
EUGE@DESKTOP-IR3PA1E MINGW64 ~/Documents/NetBeansProjects/Guia 6 - Git enc 13 (master)
$ git branch
```

```
EUGE@DESKTOP-IR3PA1E MINGW64 ~/Documents/NetBeansProjects/Guia 6 - Git enc 13 (master)
$ git branch -a
```

```
EUGE@DESKTOP-IR3PA1E MINGW64 ~/Documents/NetBeansProjects/Guia 6 - Git enc 13 (master)
$ git checkout -b enc12
Switched to a new branch 'enc12'
```



Revisemos lo aprendido hasta aquí

- Crear una rama

CAMBIOS EN LA RAMA

De momento en nuestro ejemplo las dos ramas tenían exactamente el mismo contenido, pero ahora podríamos empezar a hacer cambios en el proyecto ramaGit y sus correspondientes commit y entonces los archivos tendrán códigos diferentes, de modo que puedas ver que al pasar de una rama a otra hay cambios en los archivos.

Al igual que explicamos antes cada vez que quieras subir un cambio a tu branch sitúate en ella y ejecuta los comandos:

```
git add nombre_del_archivo .  
git commit -m "Mensaje de los cambios"
```

No vamos a hacer un push todavía porque eso lo vamos a explicar más adelante, ya que eso hará que nuestra rama aparezca en el repositorio remoto.

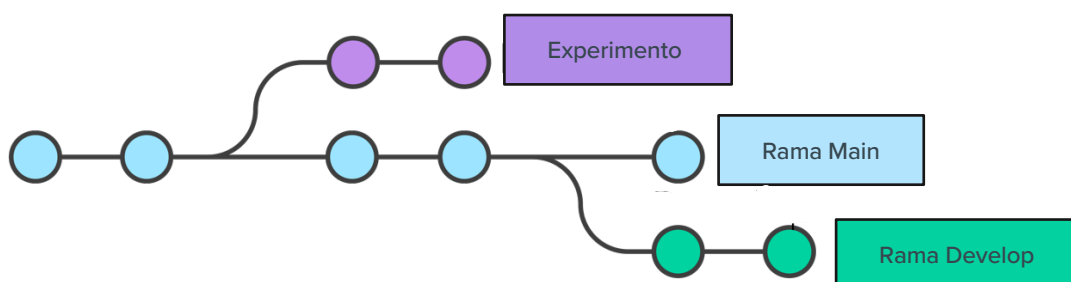
Habiendo hecho un commit en nuestra nueva rama, observarás que al hacer el `show-branches` te mostrará nuevos datos:

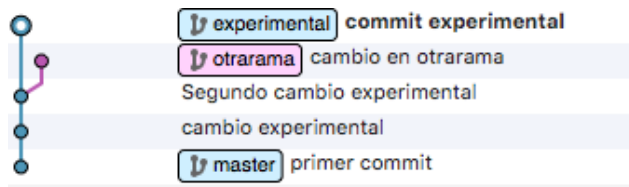
```
[→ Git git:(ramaGit) git show-branch  
! [master] Primer commit  
* [ramaGit] Segundo commit  
--  
* [ramaGit] Segundo commit  
+* [master] Primer commit
```

Como podras ver, el proyecto puede tener varios estados en un momento dado y tú podrás moverte de uno a otro con total libertad y sin tener que cambiar de carpeta ni nada parecido.



¿NECESITAS UN EJEMPLO?





Revisemos lo aprendido hasta aquí

- Enviar cambios a la rama

SUBIR UNA RAMA AL REPOSITORIO REMOTO

A pesar de que cuando crear una rama y las puedas ver en tu repositorio local con `git branch`, las ramas no se publicarán en GitHub. Para que esto ocurra tienes que realizar específicamente la acción de subir una rama determinada.

Para subir una rama en remoto la haces mediante el comando `push`, indicando el nombre de la rama que deseas subir. Por ejemplo, de esta manera:

```
git push origin <nombre-de-tu-rama>
```

Así estamos haciendo un push, empujando hacia origin (que es el nombre que se suele dar al repositorio remoto).

Si **no quieres poner siempre origin** y el nombre de tu rama en tus push, tienes que sumarle al push anterior, **-u** antes de la palabra origin. Esto hará que puedas poner `git push` solamente y vaya siempre a esa rama.

Es **importante asegurarse que lo hagamos en una rama nuestra y no en main**, ya que podríamos mandar cambios a la rama main pensando que iban a la nuestra.

Esto sería así:

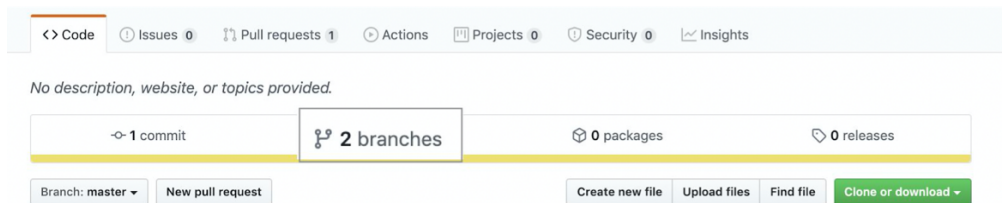
```
git push origin -u <nombre-de-tu-rama>
```

Una vez esto hecho esto podríamos pararnos en nuestra rama y simplemente escribir:

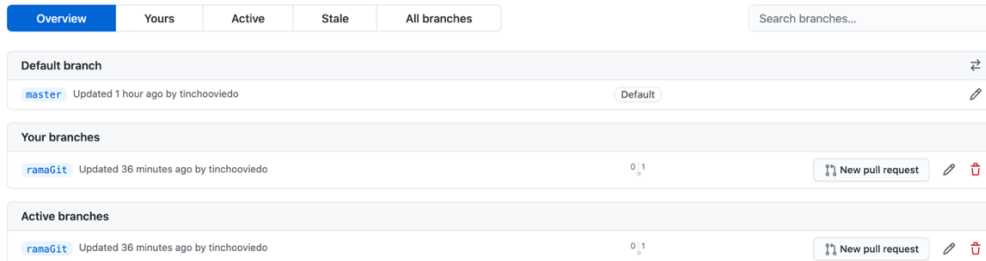
```
git push
```

¿Y COMO SE VEN LAS RAMAS EN GITHUB?

Una vez que hagamos el push para ver las ramas, primero iremos a nuestro repositorio y después a branches:



Y ahí podremos ver las dos ramas



Podes subir tanto commits como creas convenientes a tu rama antes de fusionar tus cambios con la rama main, siempre es **mejor pequeños y frecuentes cambios que pocos y grandes.**

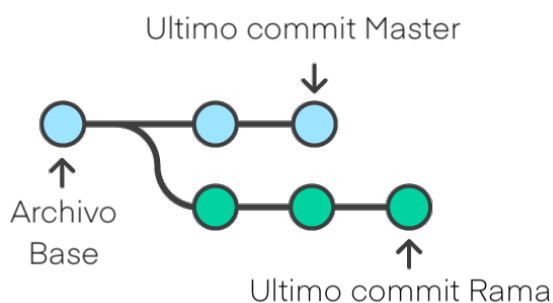


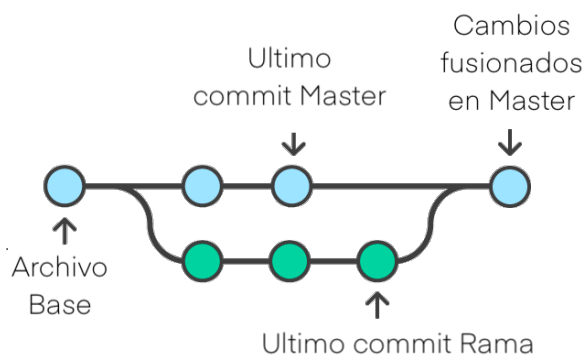
Revisemos lo aprendido hasta aquí

- Subir una rama al repositorio remoto

FUSIONAR RAMAS

A medida que creas ramas y cambies el estado de las carpetas o archivos de tu proyecto empezará a divergir de una rama a otra. Llegará el momento en el que **te interese fusionar ramas para poder incorporar el trabajo realizado a la rama main.**





El proceso de fusión se conoce como **merge** y puede llegar a ser muy simple o más complejo si se encuentran cambios que Git no pueda procesar de manera automática. Git para procesar los *merge* usa un antecesor común y comprueba los cambios que se han introducido al proyecto desde entonces, combinando el código de ambas ramas.

Para hacer un merge nos situamos en una rama, en este caso la "main", y decimos con qué otra rama se debe fusionar el código.

El siguiente comando, lanzado desde la rama "main", permite fusionarla con la rama "ramaGit".

```
git merge <nombre-de-tu-rama>
```

Un *merge* necesita un mensaje, igual que ocurre con los *commit*, por lo que al realizar ese comando se abrirá "Vim" (o cualquier otro editor de consola que tengas configurado) para que introduzcas los comentarios que juzgues oportuno. Salir de Vim lo consigues pulsando la tecla ESC y luego escribiendo **:q** y pulsando ENTER para aceptar ese comando. Esta operativa de indicar el mensaje se puede resumir con el comando:

```
git merge ramagit -m "Esto es un merge con mensaje"
```

Luego podremos comprobar que nuestra rama main tiene todo el código nuevo de la ramaGit y podremos hacer nuevos commits en main para seguir el desarrollo de nuestro proyecto ya con la rama principal, si es nuestro deseo.



Revisemos lo aprendido hasta aquí

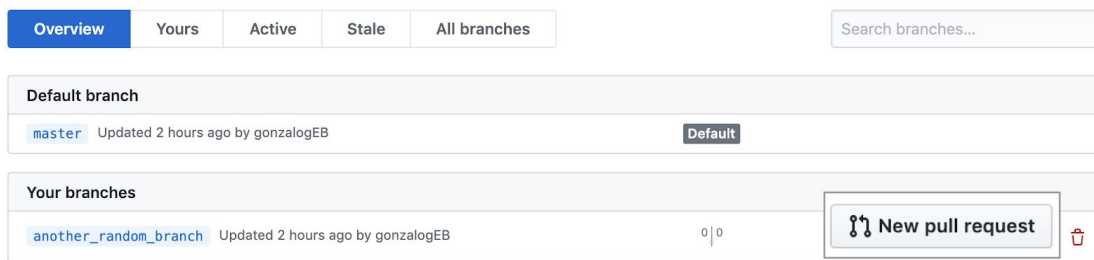
- Fusionar una rama

PULL REQUEST

Previamente habíamos fusionado nuestras ramas a través del comando **merge**, pero GitHub nos permite fusionar nuestras ramas y además ver los cambios que hay entre una rama y otra, gracias al **Pull Request**.

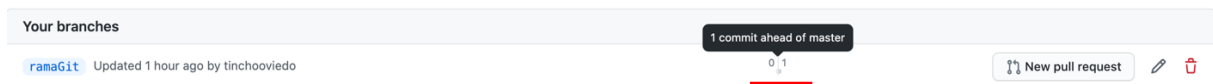
Vamos a pararnos de nuevo en una etapa en donde no hemos mergeado las ramas. Hasta ese punto habíamos logrado independizar nuestros cambios de los del resto del equipo, pero se acercaba la hora de publicar nuestros cambios y surge la necesidad de conocer y/o validar **cuán diferente es nuestra versión y de ver que todo está bien fusionar esos cambios**. Aquí es donde la herramienta de pull request viene al rescate.

Para crear un pull request debemos ir a la sección de branches, buscar nuestro branch y clicar en el botón **New pull request**.



Antes de hacer nuestro pull request, podemos ver, **cuántos commits (cambios) son los que separan a otra rama de main**. Si nos fijamos nos salen dos ceros, pero si main estuviera un commit por delante de alguna rama saldrían un uno en el cero de la izquierda y así se incrementa el número según la cantidad de commits que este por delante main. Ahora, si el número estuviera a la derecha, sería que la otra rama está x commits por delante main.

Aca podemos ver un ejemplo con ramaGit:



Como podemos ver ramaGit está un commit por delante de main, por lo que si mergeamos las ramas, sería solo un commit el que se le aplicaría a main.

GonzaGr92 / node_server Private Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Security 0 Insights

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ← compare: another_random_branch 1 Able to merge. These branches can be automatically merged.

Migrate to sequelize

Write Preview H B I

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

2

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Use Closing keywords in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#) 3

1 commit ± 4 files changed 0 commit comments 1 contributor

Commits on Jun 18, 2020

gonzalogEB Migrate to sequelize 78f236f

4 Showing 4 changed files with 158 additions and 11 deletions. Unified Split

25 app.js

```
@@ -7,6 +7,9 @@ const _ = require('lodash');
7 7 const QUERY_ALL = 'ALL';
8 8 const QUERY_GET = 'GET';
```

Nos mostrará algo similar a lo siguiente.

Referencias:

1. A la izquierda se selecciona la rama de destino a la cual vamos a querer mergear los cambios, a la derecha nuestra rama actual. Siempre podemos crear un pull request y cambiar las ramas que queremos mergear.
2. En esta sección podremos poner un título informativo de que se tratan nuestros cambios y una descripción como documentación adicional. Detalle de los cambios propuestos, test plan, etc.
3. En esta sección nos muestra un resumen de los commits y archivos modificados.
4. En la última sección nos muestra el detalle de los archivos modificados. Para visualizar los cambios podemos alternar entre los modos de vista “Unified” y “Split”

Hasta este momento aún no hemos creado nada, solo estamos viendo un resumen previo, para continuar clickeamos en el botón “Create pull request”. A continuación, veremos el pull request creado de la siguiente manera.

Code Issues 0 Pull requests 1 Actions Projects 0 Security 0 Insights

Migrate to sequelize #3

gonzalogEB wants to merge 1 commit into master from another_random_branch

Conversation 0 Commits 1 Checks 0 Files changed 4 +158 -11

gonzalogEB commented now

Este es un comentario

Migrate to sequelize 78f236f

Add more commits by pushing to the **another_random_branch** branch on **GonzaGr92/node_server**.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Write Preview H B I

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

Notifications Customize

Unsubscribe

You're receiving notifications because you authored the thread.

Por otro lado, podemos usar la pestaña de “Files changed” para hacer code review.

Migrate to sequelize #3

gonzalogEB wants to merge 1 commit into master from another_random_branch

Conversation 0 Commits 1 Checks 0 Files changed 4 +158 -11

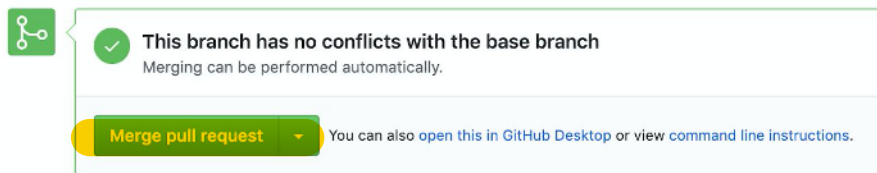
Changes from all commits File filter... Jump to... 0 / 4 files viewed Review changes +

app.js

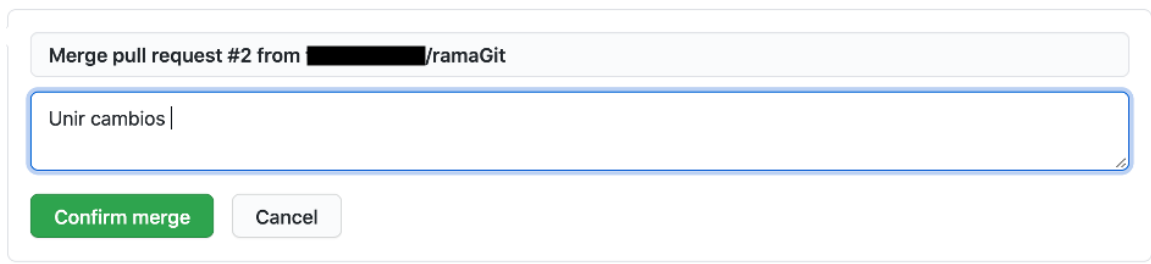
```

@@ -7,6 +7,9 @@ const _ = require('lodash');
7   const QUERY_ALL = 'ALL';
8   const QUERY_GET = 'GET';
9
10  // Connect to sqlite db
11  let db = new sqlite3.Database('./db/chinook.db', (err) => {
12    if (err) {
13      // Project self dependencies
14      + var Artist = require('./models/Artist');
15      +
16      // Connect to sqlite db
17      let db = new sqlite3.Database('./db/chinook.db', (err) => {
18        if (err) {
19          const page = _get(req, 'query.page', 0);
20          const limit = 20;
21
22          Artist.findAll({
23            where: {
24              ArtistId: 2
25            }
26          });
27
28          let sql = `SELECT * FROM artists
29            LIMIT ${limit}
30            OFFSET ${limit * page}`;
31
32          const page = _get(req, 'query.page', 0);
33          const limit = 20;
34
35          Artist.findAll({
36            where: {
37              ArtistId: 2
38            }
39          });
40
41          let sql = `SELECT * FROM artists
42            LIMIT ${limit}
43            OFFSET ${limit * page}`;
44
45          const page = _get(req, 'query.page', 0);
46          const limit = 20;
47
48          Artist.findAll({
49            where: {
50              ArtistId: 2
51            }
52          });
53
54          let sql = `SELECT * FROM artists
55            LIMIT ${limit}
56            OFFSET ${limit * page}`;
57
58          const page = _get(req, 'query.page', 0);
59          const limit = 20;
60
61          Artist.findAll({
62            where: {
63              ArtistId: 2
64            }
65          });
66
67          let sql = `SELECT * FROM artists
68            LIMIT ${limit}
69            OFFSET ${limit * page}`;
70
71          const page = _get(req, 'query.page', 0);
72          const limit = 20;
73
74          Artist.findAll({
75            where: {
76              ArtistId: 2
77            }
78          });
79
80          let sql = `SELECT * FROM artists
81            LIMIT ${limit}
82            OFFSET ${limit * page}`;
83
84          const page = _get(req, 'query.page', 0);
85          const limit = 20;
86
87          Artist.findAll({
88            where: {
89              ArtistId: 2
90            }
91          });
92
93          let sql = `SELECT * FROM artists
94            LIMIT ${limit}
95            OFFSET ${limit * page}`;
96
97          const page = _get(req, 'query.page', 0);
98          const limit = 20;
99
100         Artist.findAll({
101           where: {
102             ArtistId: 2
103           }
104         });
105
106         let sql = `SELECT * FROM artists
107           LIMIT ${limit}
108           OFFSET ${limit * page}`;
109
110         const page = _get(req, 'query.page', 0);
111         const limit = 20;
112
113         Artist.findAll({
114           where: {
115             ArtistId: 2
116           }
117         });
118
119         let sql = `SELECT * FROM artists
120           LIMIT ${limit}
121           OFFSET ${limit * page}`;
122
123         const page = _get(req, 'query.page', 0);
124         const limit = 20;
125
126         Artist.findAll({
127           where: {
128             ArtistId: 2
129           }
130         });
131
132         let sql = `SELECT * FROM artists
133           LIMIT ${limit}
134           OFFSET ${limit * page}`;
135
136         const page = _get(req, 'query.page', 0);
137         const limit = 20;
138
139         Artist.findAll({
140           where: {
141             ArtistId: 2
142           }
143         });
144
145         let sql = `SELECT * FROM artists
146           LIMIT ${limit}
147           OFFSET ${limit * page}`;
148
149         const page = _get(req, 'query.page', 0);
150         const limit = 20;
151
152         Artist.findAll({
153           where: {
154             ArtistId: 2
155           }
156         });
157
158         let sql = `SELECT * FROM artists
159           LIMIT ${limit}
160           OFFSET ${limit * page}`;
161
162         const page = _get(req, 'query.page', 0);
163         const limit = 20;
164
165         Artist.findAll({
166           where: {
167             ArtistId: 2
168           }
169         });
170
171         let sql = `SELECT * FROM artists
172           LIMIT ${limit}
173           OFFSET ${limit * page}`;
174
175         const page = _get(req, 'query.page', 0);
176         const limit = 20;
177
178         Artist.findAll({
179           where: {
180             ArtistId: 2
181           }
182         });
183
184         let sql = `SELECT * FROM artists
185           LIMIT ${limit}
186           OFFSET ${limit * page}`;
187
188         const page = _get(req, 'query.page', 0);
189         const limit = 20;
190
191         Artist.findAll({
192           where: {
193             ArtistId: 2
194           }
195         });
196
197         let sql = `SELECT * FROM artists
198           LIMIT ${limit}
199           OFFSET ${limit * page}`;
200
201         const page = _get(req, 'query.page', 0);
202         const limit = 20;
203
204         Artist.findAll({
205           where: {
206             ArtistId: 2
207           }
208         });
209
210         let sql = `SELECT * FROM artists
211           LIMIT ${limit}
212           OFFSET ${limit * page}`;
213
214         const page = _get(req, 'query.page', 0);
215         const limit = 20;
216
217         Artist.findAll({
218           where: {
219             ArtistId: 2
220           }
221         });
222
223         let sql = `SELECT * FROM artists
224           LIMIT ${limit}
225           OFFSET ${limit * page}`;
226
227         const page = _get(req, 'query.page', 0);
228         const limit = 20;
229
230         Artist.findAll({
231           where: {
232             ArtistId: 2
233           }
234         });
235
236         let sql = `SELECT * FROM artists
237           LIMIT ${limit}
238           OFFSET ${limit * page}`;
239
240         const page = _get(req, 'query.page', 0);
241         const limit = 20;
242
243         Artist.findAll({
244           where: {
245             ArtistId: 2
246           }
247         });
248
249         let sql = `SELECT * FROM artists
250           LIMIT ${limit}
251           OFFSET ${limit * page}`;
252
253         const page = _get(req, 'query.page', 0);
254         const limit = 20;
255
256         Artist.findAll({
257           where: {
258             ArtistId: 2
259           }
260         });
261
262         let sql = `SELECT * FROM artists
263           LIMIT ${limit}
264           OFFSET ${limit * page}`;
265
266         const page = _get(req, 'query.page', 0);
267         const limit = 20;
268
269         Artist.findAll({
270           where: {
271             ArtistId: 2
272           }
273         });
274
275         let sql = `SELECT * FROM artists
276           LIMIT ${limit}
277           OFFSET ${limit * page}`;
278
279         const page = _get(req, 'query.page', 0);
280         const limit = 20;
281
282         Artist.findAll({
283           where: {
284             ArtistId: 2
285           }
286         });
287
288         let sql = `SELECT * FROM artists
289           LIMIT ${limit}
290           OFFSET ${limit * page}`;
291
292         const page = _get(req, 'query.page', 0);
293         const limit = 20;
294
295         Artist.findAll({
296           where: {
297             ArtistId: 2
298           }
299         });
300
301         let sql = `SELECT * FROM artists
302           LIMIT ${limit}
303           OFFSET ${limit * page}`;
304
305         const page = _get(req, 'query.page', 0);
306         const limit = 20;
307
308         Artist.findAll({
309           where: {
310             ArtistId: 2
311           }
312         });
313
314         let sql = `SELECT * FROM artists
315           LIMIT ${limit}
316           OFFSET ${limit * page}`;
317
318         const page = _get(req, 'query.page', 0);
319         const limit = 20;
320
321         Artist.findAll({
322           where: {
323             ArtistId: 2
324           }
325         });
326
327         let sql = `SELECT * FROM artists
328           LIMIT ${limit}
329           OFFSET ${limit * page}`;
330
331         const page = _get(req, 'query.page', 0);
332         const limit = 20;
333
334         Artist.findAll({
335           where: {
336             ArtistId: 2
337           }
338         });
339
340         let sql = `SELECT * FROM artists
341           LIMIT ${limit}
342           OFFSET ${limit * page}`;
343
344         const page = _get(req, 'query.page', 0);
345         const limit = 20;
346
347         Artist.findAll({
348           where: {
349             ArtistId: 2
350           }
351         });
352
353         let sql = `SELECT * FROM artists
354           LIMIT ${limit}
355           OFFSET ${limit * page}`;
356
357         const page = _get(req, 'query.page', 0);
358         const limit = 20;
359
360         Artist.findAll({
361           where: {
362             ArtistId: 2
363           }
364         });
365
366         let sql = `SELECT * FROM artists
367           LIMIT ${limit}
368           OFFSET ${limit * page}`;
369
370         const page = _get(req, 'query.page', 0);
371         const limit = 20;
372
373         Artist.findAll({
374           where: {
375             ArtistId: 2
376           }
377         });
378
379         let sql = `SELECT * FROM artists
380           LIMIT ${limit}
381           OFFSET ${limit * page}`;
382
383         const page = _get(req, 'query.page', 0);
384         const limit = 20;
385
386         Artist.findAll({
387           where: {
388             ArtistId: 2
389           }
390         });
391
392         let sql = `SELECT * FROM artists
393           LIMIT ${limit}
394           OFFSET ${limit * page}`;
395
396         const page = _get(req, 'query.page', 0);
397         const limit = 20;
398
399         Artist.findAll({
400           where: {
401             ArtistId: 2
402           }
403         });
404
405         let sql = `SELECT * FROM artists
406           LIMIT ${limit}
407           OFFSET ${limit * page}`;
408
409         const page = _get(req, 'query.page', 0);
410         const limit = 20;
411
412         Artist.findAll({
413           where: {
414             ArtistId: 2
415           }
416         });
417
418         let sql = `SELECT * FROM artists
419           LIMIT ${limit}
420           OFFSET ${limit * page}`;
421
422         const page = _get(req, 'query.page', 0);
423         const limit = 20;
424
425         Artist.findAll({
426           where: {
427             ArtistId: 2
428           }
429         });
430
431         let sql = `SELECT * FROM artists
432           LIMIT ${limit}
433           OFFSET ${limit * page}`;
434
435         const page = _get(req, 'query.page', 0);
436         const limit = 20;
437
438         Artist.findAll({
439           where: {
440             ArtistId: 2
441           }
442         });
443
444         let sql = `SELECT * FROM artists
445           LIMIT ${limit}
446           OFFSET ${limit * page}`;
447
448         const page = _get(req, 'query.page', 0);
449         const limit = 20;
450
451         Artist.findAll({
452           where: {
453             ArtistId: 2
454           }
455         });
456
457         let sql = `SELECT * FROM artists
458           LIMIT ${limit}
459           OFFSET ${limit * page}`;
460
461         const page = _get(req, 'query.page', 0);
462         const limit = 20;
463
464         Artist.findAll({
465           where: {
466             ArtistId: 2
467           }
468         });
469
470         let sql = `SELECT * FROM artists
471           LIMIT ${limit}
472           OFFSET ${limit * page}`;
473
474         const page = _get(req, 'query.page', 0);
475         const limit = 20;
476
477         Artist.findAll({
478           where: {
479             ArtistId: 2
480           }
481         });
482
483         let sql = `SELECT * FROM artists
484           LIMIT ${limit}
485           OFFSET ${limit * page}`;
486
487         const page = _get(req, 'query.page', 0);
488         const limit = 20;
489
490         Artist.findAll({
491           where: {
492             ArtistId: 2
493           }
494         });
495
496         let sql = `SELECT * FROM artists
497           LIMIT ${limit}
498           OFFSET ${limit * page}`;
499
500         const page = _get(req, 'query.page', 0);
501         const limit = 20;
502
503         Artist.findAll({
504           where: {
505             ArtistId: 2
506           }
507         });
508
509         let sql = `SELECT * FROM artists
510           LIMIT ${limit}
511           OFFSET ${limit * page}`;
512
513         const page = _get(req, 'query.page', 0);
514         const limit = 20;
515
516         Artist.findAll({
517           where: {
518             ArtistId: 2
519           }
520         });
521
522         let sql = `SELECT * FROM artists
523           LIMIT ${limit}
524           OFFSET ${limit * page}`;
525
526         const page = _get(req, 'query.page', 0);
527         const limit = 20;
528
529         Artist.findAll({
530           where: {
531             ArtistId: 2
532           }
533         });
534
535         let sql = `SELECT * FROM artists
536           LIMIT ${limit}
537           OFFSET ${limit * page}`;
538
539         const page = _get(req, 'query.page', 0);
540         const limit = 20;
541
542         Artist.findAll({
543           where: {
544             ArtistId: 2
545           }
546         });
547
548         let sql = `SELECT * FROM artists
549           LIMIT ${limit}
550           OFFSET ${limit * page}`;
551
552         const page = _get(req, 'query.page', 0);
553         const limit = 20;
554
555         Artist.findAll({
556           where: {
557             ArtistId: 2
558           }
559         });
560
561         let sql = `SELECT * FROM artists
562           LIMIT ${limit}
563           OFFSET ${limit * page}`;
564
565         const page = _get(req, 'query.page', 0);
566         const limit = 20;
567
568         Artist.findAll({
569           where: {
570             ArtistId: 2
571           }
572         });
573
574         let sql = `SELECT * FROM artists
575           LIMIT ${limit}
576           OFFSET ${limit * page}`;
577
578         const page = _get(req, 'query.page', 0);
579         const limit = 20;
580
581         Artist.findAll({
582           where: {
583             ArtistId: 2
584           }
585         });
586
587         let sql = `SELECT * FROM artists
588           LIMIT ${limit}
589           OFFSET ${limit * page}`;
590
591         const page = _get(req, 'query.page', 0);
592         const limit = 20;
593
594         Artist.findAll({
595           where: {
596             ArtistId: 2
597           }
598         });
599
600         let sql = `SELECT * FROM artists
601           LIMIT ${limit}
602           OFFSET ${limit * page}`;
603
604         const page = _get(req, 'query.page', 0);
605         const limit = 20;
606
607         Artist.findAll({
608           where: {
609             ArtistId: 2
610           }
611         });
612
613         let sql = `SELECT * FROM artists
614           LIMIT ${limit}
615           OFFSET ${limit * page}`;
616
617         const page = _get(req, 'query.page', 0);
618         const limit = 20;
619
620         Artist.findAll({
621           where: {
622             ArtistId: 2
623           }
624         });
625
626         let sql = `SELECT * FROM artists
627           LIMIT ${limit}
628           OFFSET ${limit * page}`;
629
630         const page = _get(req, 'query.page', 0);
631         const limit = 20;
632
633         Artist.findAll({
634           where: {
635             ArtistId: 2
636           }
637         });
638
639         let sql = `SELECT * FROM artists
640           LIMIT ${limit}
641           OFFSET ${limit * page}`;
642
643         const page = _get(req, 'query.page', 0);
644         const limit = 20;
645
646         Artist.findAll({
647           where: {
648             ArtistId: 2
649           }
650         });
651
652         let sql = `SELECT * FROM artists
653           LIMIT ${limit}
654           OFFSET ${limit * page}`;
655
656         const page = _get(req, 'query.page', 0);
657         const limit = 20;
658
659         Artist.findAll({
660           where: {
661             ArtistId: 2
662           }
663         });
664
665         let sql = `SELECT * FROM artists
666           LIMIT ${limit}
667           OFFSET ${limit * page}`;
668
669         const page = _get(req, 'query.page', 0);
670         const limit = 20;
671
672         Artist.findAll({
673           where: {
674             ArtistId: 2
675           }
676         });
677
678         let sql = `SELECT * FROM artists
679           LIMIT ${limit}
680           OFFSET ${limit * page}`;
681
682         const page = _get(req, 'query.page', 0);
683         const limit = 20;
684
685         Artist.findAll({
686           where: {
687             ArtistId: 2
688           }
689         });
690
691         let sql = `SELECT * FROM artists
692           LIMIT ${limit}
693           OFFSET ${limit * page}`;
694
695         const page = _get(req, 'query.page', 0);
696         const limit = 20;
697
698         Artist.findAll({
699           where: {
700             ArtistId: 2
701           }
702         });
703
704         let sql = `SELECT * FROM artists
705           LIMIT ${limit}
706           OFFSET ${limit * page}`;
707
708         const page = _get(req, 'query.page', 0);
709         const limit = 20;
710
711         Artist.findAll({
712           where: {
713             ArtistId: 2
714           }
715         });
716
717         let sql = `SELECT * FROM artists
718           LIMIT ${limit}
719           OFFSET ${limit * page}`;
720
721         const page = _get(req, 'query.page', 0);
722         const limit = 20;
723
724         Artist.findAll({
725           where: {
726             ArtistId: 2
727           }
728         });
729
730         let sql = `SELECT * FROM artists
731           LIMIT ${limit}
732           OFFSET ${limit * page}`;
733
734         const page = _get(req, 'query.page', 0);
735         const limit = 20;
736
737         Artist.findAll({
738           where: {
739             ArtistId: 2
740           }
741         });
742
743         let sql = `SELECT * FROM artists
744           LIMIT ${limit}
745           OFFSET ${limit * page}`;
746
747         const page = _get(req, 'query.page', 0);
748         const limit = 20;
749
750         Artist.findAll({
751           where: {
752             ArtistId: 2
753           }
754         });
755
756         let sql = `SELECT * FROM artists
757           LIMIT ${limit}
758           OFFSET ${limit * page}`;
759
760         const page = _get(req, 'query.page', 0);
761         const limit = 20;
762
763         Artist.findAll({
764           where: {
765             ArtistId: 2
766           }
767         });
768
769         let sql = `SELECT * FROM artists
770           LIMIT ${limit}
771           OFFSET ${limit * page}`;
772
773         const page = _get(req, 'query.page', 0);
774         const limit = 20;
775
776         Artist.findAll({
777           where: {
778             ArtistId: 2
779           }
780         });
781
782         let sql = `SELECT * FROM artists
783           LIMIT ${limit}
784           OFFSET ${limit * page}`;
785
786         const page = _get(req, 'query.page', 0);
787         const limit = 20;
788
789         Artist.findAll({
790           where: {
791             ArtistId: 2
792           }
793         });
794
795         let sql = `SELECT * FROM artists
796           LIMIT ${limit}
797           OFFSET ${limit * page}`;
798
799         const page = _get(req, 'query.page', 0);
800         const limit = 20;
801
802         Artist.findAll({
803           where: {
804             ArtistId: 2
805           }
806         });
807
808         let sql = `SELECT * FROM artists
809           LIMIT ${limit}
810           OFFSET ${limit * page}`;
811
812         const page = _get(req, 'query.page', 0);
813         const limit = 20;
814
815         Artist.findAll({
816           where: {
817             ArtistId: 2
818           }
819         });
820
821         let sql = `SELECT * FROM artists
822           LIMIT ${limit}
823           OFFSET ${limit * page}`;
824
825         const page = _get(req, 'query.page', 0);
826         const limit = 20;
827
828         Artist.findAll({
829           where: {
830             ArtistId: 2
831           }
832         });
833
834         let sql = `SELECT * FROM artists
835           LIMIT ${limit}
836           OFFSET ${limit * page}`;
837
838         const page = _get(req, 'query.page', 0);
839         const limit = 20;
840
841         Artist.findAll({
842           where: {
843             ArtistId: 2
844           }
845         });
846
847         let sql = `SELECT * FROM artists
848           LIMIT ${limit}
849           OFFSET ${limit * page}`;
850
851         const page = _get(req, 'query.page', 0);
852         const limit = 20;
853
854         Artist.findAll({
855           where: {
856             ArtistId: 2
857           }
858         });
859
860         let sql = `SELECT * FROM artists
861           LIMIT ${limit}
862           OFFSET ${limit * page}`;
863
864         const page = _get(req, 'query.page', 0);
865         const limit = 20;
866
867         Artist.findAll({
868           where: {
869             ArtistId: 2
870           }
871         });
872
873         let sql = `SELECT * FROM artists
874           LIMIT ${limit}
875           OFFSET ${limit * page}`;
876
877         const page = _get(req, 'query.page', 0);
878         const limit = 20;
879
880         Artist.findAll({
881           where: {
882             ArtistId: 2
883           }
884         });
885
886         let sql = `SELECT * FROM artists
887           LIMIT ${limit}
888           OFFSET ${limit * page}`;
889
890         const page = _get(req, 'query.page', 0);
891         const limit = 20;
892
893         Artist.findAll({
894           where: {
895             ArtistId: 2
896           }
897         });
898
899         let sql = `SELECT * FROM artists
900           LIMIT ${limit}
901           OFFSET ${limit * page}`;
902
903         const page = _get(req, 'query.page', 0);
904         const limit = 20;
905
906         Artist.findAll({
907           where: {
908             ArtistId: 2
909           }
910         });
911
912         let sql = `SELECT * FROM artists
913           LIMIT ${limit}
914           OFFSET ${limit * page}`;
915
916         const page = _get(req, 'query.page', 0);
917         const limit = 20;
918
919         Artist.findAll({
920           where: {
921             ArtistId: 2
922           }
923         });
924
925         let sql = `SELECT * FROM artists
926           LIMIT ${limit}
927           OFFSET ${limit * page}`;
928
929         const page = _get(req, 'query.page', 0);
930         const limit = 20;
931
932         Artist.findAll({
933           where: {
934             ArtistId: 2
935           }
936         });
937
938         let sql = `SELECT * FROM artists
939           LIMIT ${limit}
940           OFFSET ${limit * page}`;
941
942         const page = _get(req, 'query.page', 0);
943         const limit = 20;
944
945         Artist.findAll({
946           where: {
947             ArtistId: 2
948           }
949         });
950
951         let sql = `SELECT * FROM artists
952           LIMIT ${limit}
953           OFFSET ${limit * page}`;
954
955         const page = _get(req, 'query.page', 0);
956         const limit = 20;
957
958         Artist.findAll({
959           where: {
960             ArtistId: 2
961           }
962         });
963
964         let sql = `SELECT * FROM artists
965           LIMIT ${limit}
966           OFFSET ${limit * page}`;
967
968         const page = _get(req, 'query.page', 0);
969         const limit = 20;
970
971         Artist.findAll({
972           where: {
973             ArtistId: 2
974           }
975         });
976
977         let sql = `SELECT * FROM artists
978           LIMIT ${limit}
979           OFFSET ${limit * page}`;
980
981         const page = _get(req, 'query.page', 0);
982         const limit = 20;
983
984         Artist.findAll({
985           where: {
986             ArtistId: 2
987           }
988         });
989
990         let sql = `SELECT * FROM artists
991           LIMIT ${limit}
992           OFFSET ${limit * page}`;
993
994         const page = _get(req, 'query.page', 0);
995         const limit = 20;
996
997         Artist.findAll({
998           where: {
999             ArtistId: 2
1000           }
1001         });
1002
1003         let sql = `SELECT * FROM artists
1004           LIMIT ${limit}
1005           OFFSET ${limit * page}`;
1006
1007         const page = _get(req, 'query.page', 0);
1008         const limit = 20;
1009
1010         Artist.findAll({
1011           where: {
1012             ArtistId: 2
1013           }
1014         });
1015
1016         let sql = `SELECT * FROM artists
1017           LIMIT ${limit}
1018           OFFSET ${limit * page}`;
1019
1020         const page = _get(req, 'query.page', 0);
1021         const limit = 20;
1022
1023         Artist.findAll({
1024           where: {
1025             ArtistId: 2
1026           }
1027         });
1028
1029         let sql = `SELECT * FROM artists
1030           LIMIT ${limit}
1031           OFFSET ${limit * page}`;
1032
1033         const page = _get(req, 'query.page', 0);
1034         const limit = 20;
1035
1036         Artist.findAll({
1037           where: {
1038             ArtistId: 2
1039           }
1040         });
1041
1042         let sql = `SELECT * FROM artists
1043           LIMIT ${limit}
1044           OFFSET ${limit * page}`;
1045
1046         const page = _get(req, 'query.page', 0);
1047         const limit = 20;
1048
1049         Artist.findAll({
1050           where: {
1051             ArtistId: 2
1052           }
1053         });
1054
1055         let sql = `SELECT * FROM artists
1056           LIMIT ${limit}
1057           OFFSET ${limit * page}`;
1058
1059         const page = _get(req, 'query.page', 0);
1060         const limit = 20;
1061
1062         Artist.findAll({
1063           where: {
1064             ArtistId: 2
1065           }
1066         });
1067
1068         let sql = `SELECT * FROM artists
1069           LIMIT ${limit}
1070           OFFSET ${limit * page}`;
1071
1072         const page = _get(req, 'query.page', 0);
1073         const limit = 20;
1074
1075         Artist.findAll({
1076           where: {
1077             ArtistId: 2
1078           }
1079         });
1080
1081         let sql = `SELECT * FROM artists
1082           LIMIT ${limit}
1083           OFFSET ${limit * page}`;
1084
1085         const page = _get(req, 'query.page', 0);
1086         const limit = 20;
1087
1088         Artist.findAll({
1089           where: {
1090             ArtistId: 2
1091           }
1092         });
1093
1094         let sql = `SELECT * FROM artists
1095           LIMIT ${limit}
1096           OFFSET ${limit * page}`;
1097
1098         const page = _get(req, 'query.page', 0);
1099         const limit = 20;
1100
1101         Artist.findAll({
1102           where: {
1103             ArtistId: 2
1104           }
1105         });
1106
1107         let sql = `SELECT * FROM artists
1108           LIMIT ${limit}
1109           OFFSET ${limit * page}`;
1110
1111         const page = _get(req, 'query.page', 0);
1112         const limit = 20;
1113
1114         Artist.findAll({
1115           where: {
1116             ArtistId: 2
1117           }
1118         });
1119
1120         let sql = `SELECT * FROM artists
1121           LIMIT ${limit}
1122           OFFSET ${limit * page}`;
1123
1124         const page = _get(req, 'query.page', 0);
1125         const limit = 20;
1126
1127         Artist.findAll({
1128           where: {
1129             ArtistId: 2
1130           }
1131         });
1132
1133         let sql = `SELECT * FROM artists
1134           LIMIT ${limit}
1135           OFFSET ${limit * page}`;
1136
1137         const page = _get(req, 'query.page', 0);
1138         const limit = 20;
1139
1140         Artist.findAll({
1141           where: {
1142             ArtistId: 2
1143           }
1144         });
1145
1146         let sql = `SELECT * FROM artists
1147           LIMIT ${limit}
1148           OFFSET ${limit * page}`;
1149
1150         const page = _get(req, 'query.page', 0);
1151         const limit = 20;
1152
1153         Artist.findAll({
1154           where: {
1155             ArtistId: 2
1156           }
1157         });
1158
1159         let sql = `SELECT * FROM artists
1160           LIMIT ${limit}
1161           OFFSET ${limit * page}`;
1162
1163         const page = _get(req, 'query.page', 0);
1164         const limit = 20;
1165
1166         Artist.findAll({
1167           where: {
1168             ArtistId: 2
1169           }
1170         });
1171
1172         let sql = `SELECT * FROM artists
1173           LIMIT ${limit}
1174           OFFSET ${limit * page}`;
1175
1176         const page = _get(req, 'query.page', 0);
1177         const limit = 20;
1178
1179         Artist.findAll({
1180           where: {
1181             ArtistId: 2
1182           }
1183         });
1184
1185         let sql = `SELECT * FROM artists
1186           LIMIT ${limit}
1187           OFFSET ${limit * page}`;
1188
1189         const page = _get(req, 'query.page', 0);
1190         const limit = 20;
1191
1192         Artist.findAll({
1193           where: {
1194             ArtistId: 2
1195           }
1196         });
1197
1198         let sql = `SELECT * FROM artists
1199           LIMIT ${limit}
1200           OFFSET ${limit * page}`;
1201
1202         const page = _get(req, 'query.page', 0);
1203         const limit = 20;
1204
1205         Artist.findAll({
1206           where: {
1207             ArtistId: 2
1208           }
1209         });
1210
1211         let sql = `SELECT * FROM artists
1212           LIMIT ${limit}
1213           OFFSET ${limit * page}`;
1214
1215         const page = _get(req, 'query.page', 0);
1216         const limit = 20;
1217
1218         Artist.findAll({
1219           where: {
1220             ArtistId: 2
1221           }
1222         });
1223
1224         let sql = `SELECT * FROM artists
1225           LIMIT ${limit}
1226           OFFSET ${limit * page}`;
1227
1228         const page = _get(req, 'query.page', 0);
1229         const limit = 20;
1230
1231         Artist.findAll({
1232           where: {
1233             ArtistId: 2
1234           }
1235         });
1236
1237         let sql = `SELECT * FROM artists
1238           LIMIT ${limit}
1239           OFFSET ${limit * page}`;
1240
1241         const page = _get(req, 'query.page', 0);
1242         const limit = 20;
1243
1244         Artist.findAll({
1245           where: {
1246             ArtistId: 2
1247           }
1248         });
1249
1250         let sql = `SELECT * FROM artists
1251           LIMIT ${limit}
1252           OFFSET ${limit * page}`;
1253
1254         const page = _get(req, 'query.page', 0);
1255         const limit = 20;
1256
1257         Artist.findAll({
1258           where: {
1259             ArtistId: 2
1260           }
1261         });
1262
1263         let sql = `SELECT * FROM artists
1264           LIMIT ${limit}
1265           OFFSET ${limit * page}`;
1266
1267         const page = _get(req, 'query.page', 0);
1268         const limit = 20;
1269
1270         Artist.findAll({
1271           where: {
1272             ArtistId: 2
1273           }
1274         });
1275
1276         let sql = `SELECT * FROM artists
1277           LIMIT ${limit}
1278           OFFSET ${limit * page}`;
1279
1280         const page = _get(req, 'query.page', 0);
1281         const limit = 20;
1282
1283         Artist.findAll({
1284           where: {
1285             ArtistId: 2
1286           }
1287         });
1288
1289         let sql = `SELECT * FROM artists
1290           LIMIT ${limit}
1291           OFFSET ${limit * page}`;
1292
1293         const page = _get(req, 'query.page', 0);
1294
```

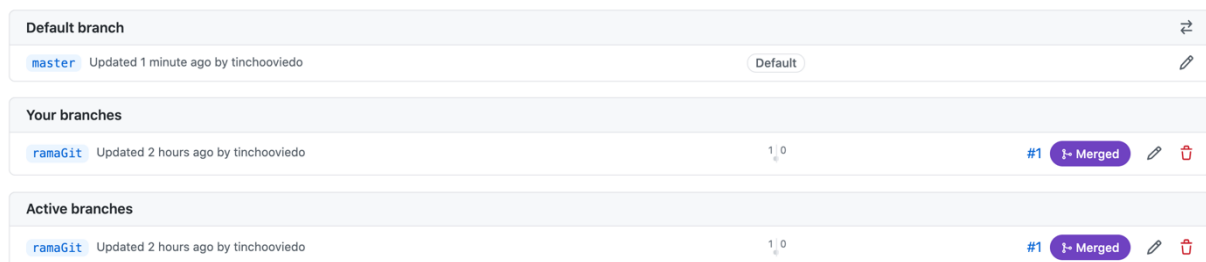
Si esta todo bien y no hay conflictos podemos mergear nuestro branch a main clickeando en el botón “Merge pull request” y de eso modo finaliza el ciclo del branch.



Finalmente tenemos la opción de aprobar los cambios para que el autor del pull request mergee su cambio.



Una vez que unamos los cambios en nuestra ramas nos va a salir que la rama ha sido mergeada.



Revisemos lo aprendido hasta aquí

- Utilizar el pull request para fusionar ramas

DESCARGAR CONTENIDO DESDE UN REPOSITORIO REMOTO

En la guía anterior vimos como clonar un repositorio, pero imaginemos que ya estamos trabajando en dicho repositorio con un grupo de personas. A veces ocurre que las personas del equipo generan sus propias ramas en remoto, donde trabajarán su parte del proyecto por ejemplo cuando han sido creadas por otros usuarios y subidas al hosting de repositorios.

Ahora supongamos **necesitamos acceder a las ramas y su contenido en local para verificar los cambios o continuar el trabajo**. En principio esas ramas en remoto creadas por otros usuarios no están disponibles para nosotros en local, pero las podemos descargar.

Para esto usaremos el comando **git pull**

GIT PULL

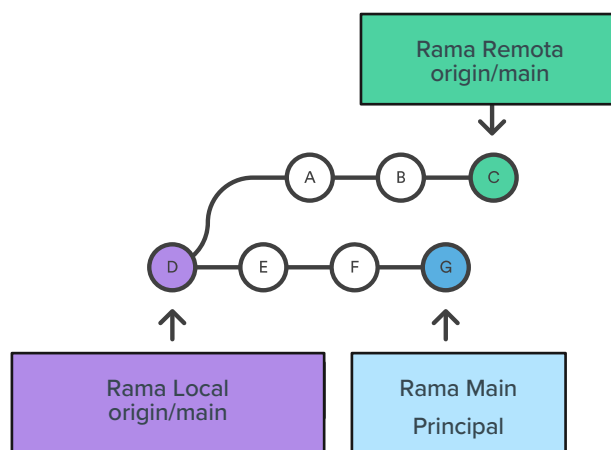
Git pull se utiliza para mantener el repositorio actualizado con los cambios que haya habido. No siempre, pero en muchos casos hay más de una persona trabajando en un repositorio, por lo que **cundo una persona hace cambios en un repositorio remoto con un git push, nuestro repositorio local estará desactualizado y tendremos que traernos los cambios que se hayan hecho**. Supongamos que un compañero nuestro hace un cambio y nosotros debemos seguir trabajando sobre ese cambio, para esto nos sirve usar el git pull

Git pull comprueba si hay cambios en el repositorio remoto y, en caso de que los haya, se trae esos archivos a tu repositorio local y actualiza tu espacio de trabajo (tu IDE, tus archivos).

Es uno de los cuatro comandos que solicita interacción de red por Git. Por default, git pull hace dos cosas.

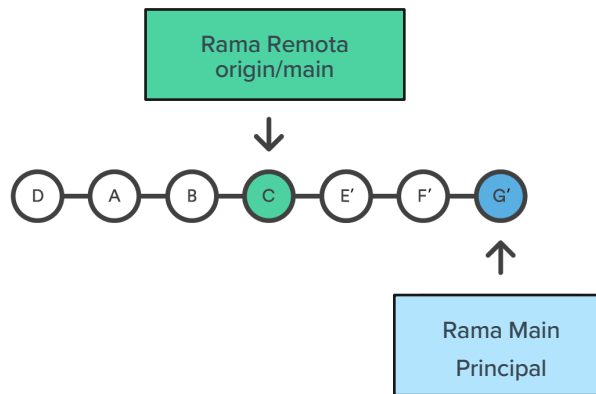
1. **Actualiza la rama de trabajo actual** (la rama a la que se ha cambiado actualmente)
2. **Actualiza las referencias de rama remota para todas las demás ramas.**

git pull recupera (git fetch) los nuevos commits y las fusiona (git merge) en tu rama local.



Nuestra rama está en el commit inicial que es commit D, la rama de otra persona está en el commit C y el Main está en el commit G.

En este caso, git pull descargará todos los cambios desde el punto de separación de la rama local y la rama principal. En el ejemplo de arriba, ese punto es E. **El comando git pull recuperará los commits remotos divergentes**, que son A, B y C. A continuación, el proceso de incorporación de cambios creará otro de fusión local que incluya el contenido de las nuevas confirmaciones remotas divergentes.



Ahora podemos ver que la incorporación mediante cambio de base no ha creado la confirmación H, sino que el cambio de base ha copiado los commits remotos A, B y C y ha reescrito los commits locales E, F y G para que aparezcan después de ellas en el historial de commits principales o de origen locales.

¿QUÉ ES UNA RAMA REMOTA?

Las ramas remotas son referencias al estado de las ramas en tus repositorios remotos. Son ramas locales que no puedes mover; se mueven automáticamente cuando estableces comunicaciones en la red. Las ramas remotas funcionan como marcadores, para recordarte en qué estado se encontraban tus repositorios remotos la última vez que conectaste con ellos.

Suelen referenciarse como (remoto)/(rama). Por ejemplo, si quieres saber cómo estaba la rama main en el remoto origin, puedes revisar la rama **origin/main**. O si estás trabajando en un problema con un compañero y este envía (push) una rama **develop**, tú tendrás tu propia rama de trabajo local **develop**; pero la rama en el servidor apuntará a la última confirmación (commit) en la rama **origin/develop**.

Esto puede ser un tanto confuso, pero intentemos aclararlo con un ejemplo. Supongamos que tienes un servidor Git en tu red, en **git.ourcompany.com**. Si haces un clon desde ahí, Git automáticamente lo denominará origin, traerá (pull) sus datos, creará un apuntador hacia donde esté en ese momento su rama main y denominará la copia local origin/main. Git te proporcionará también tu propia rama main, apuntando al mismo lugar que la rama main de origin; de manera que tengas donde trabajar.

SINTAXIS GIT PULL

La sintaxis de este comando es el siguiente:

```
git pull <nombre-remoto> <nombre-de-tu-rama>
```

Nombre-remoto: es el nombre de tu repositorio remoto. Por ejemplo: origin.

Nombre-de-tu-rama: es el nombre de tu rama. Por ejemplo: develop.

Por ejemplo:

```
git pull origin ramaGit
```

Nota:

Si tienes cambios no confirmados, la parte de fusión del comando git pull fallará y tu rama local quedará intacta.

Por lo tanto, *siempre deberías confirmar tus cambios en una rama antes de actualizar nuevas confirmaciones de un repositorio remoto.*



Revisemos lo aprendido hasta aquí

- Utilizar el comando Git Pull
- Descargar ramas del repositorio remoto

BORRAR UNA RAMA

En ocasiones puede ser necesario **eliminar una rama del repositorio**, por ejemplo, porque nos hayamos equivocado en el nombre al crearla. Aquí la operativa puede ser diferente, dependiendo de si hemos subido ya esa rama a remoto o si todavía solamente está en local.

BORRADO DE LA RAMA EN LOCAL

Esto lo conseguimos con el comando `git branch`, solamente que ahora usamos la opción **-d** para indicar que esa rama queremos borrarla.

```
git branch -d <rama-a-borrar>
```

Sin embargo, puede que esta acción no nos funcione porque hayamos hecho **cambios que no se hayan salvado en el repositorio remoto**, o **no se hayan fusionado con otras ramas**. En el caso que queramos **forzar** el borrado de la rama, para eliminarla independientemente de si se ha hecho el push o el merge, tendrás que usar la opción **-D**.

```
git branch -D <rama-a-borrar>
```

Debes prestar especial atención a esta opción **"-D"**, ya que al eliminar de este modo puede haber cambios que ya no se puedan recuperar. Como puedes apreciar, es bastante fácil de confundir con **"-d"**, opción más segura, ya que no permite borrado de ramas en situaciones donde se pueda perder código.

ELIMINAR UN BRANCH EN REMOTO

Si **la rama que queremos eliminar está en el repositorio remoto**, la operativa es un poco diferente. Tenemos que hacer un push, indicando la opción **--delete**, seguida de la rama que se desea borrar.

```
git push origin --delete <rama-a-borrar>
```



Revisemos lo aprendido hasta aquí

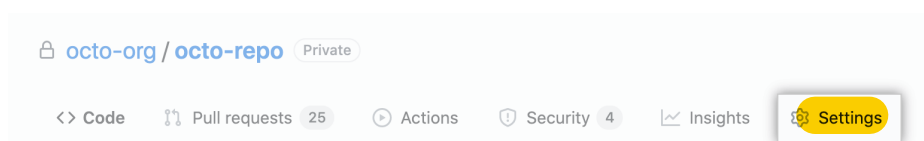
- Eliminar ramas del repositorio local y remoto

INVITAR COLABORADORES A UN REPOSITORIO PERSONAL

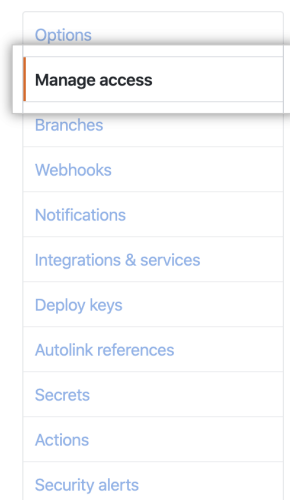
Nosotros vimos como clonar un repositorio para poder usar su código y si quisiéramos hacer cambios y subir esos cambios. Todos los usuarios de GitHub pueden ver y clonar tu repositorio, siempre y cuando sea un repositorio público. Pero, no todas las personas que clonan tu repositorio pueden subir sus cambios, ya que GitHub entiende que los repositorios son de nuestra propiedad y somos los únicos que podemos modificarlo.

Ahora, como hacemos cuando queremos que varias personas trabajen en un mismo repositorio y queremos que GitHub les deje subir esos cambios. Para ese dilema GitHub nos deja invitar colaboradores a nuestro proyecto. Esto se hará de la siguiente manera:

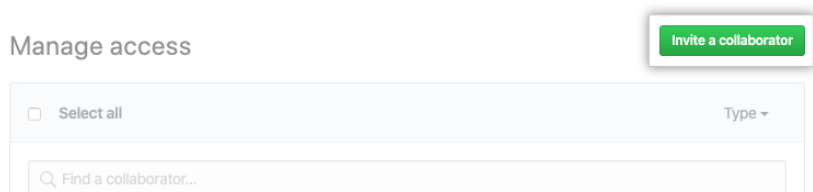
1. Solicita el nombre de usuario de la persona que estás invitando como colaboradora.
2. En GitHub, visita la página principal del repositorio.
3. Debajo de tu nombre de repositorio, da clic en **Configuración**.



4. En la barra lateral izquierda, da clic en Administrar acceso.

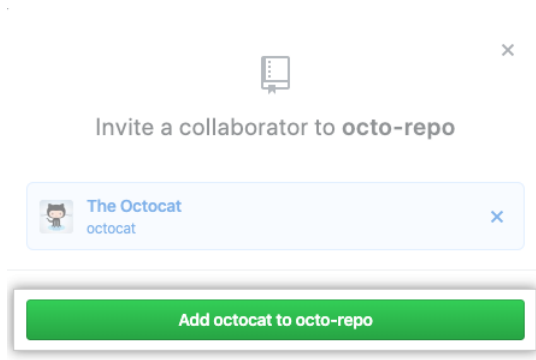


5. Da clic en Invitar un colaborador.



6. En el campo de búsqueda, comienza a teclear el nombre de la persona que quieres invitar, luego da clic en un nombre de la lista de resultados.

7. Da clic en **Añadir a nombreRepositorio**.



8. El usuario recibirá un correo electrónico invitándolo al repositorio. Una vez que acepte la invitación, tendrá acceso de colaborador a tu repositorio. Las invitaciones pendientes caducarán después de 7 días. Esto restablecerá cualquier licencia sin reclamar.

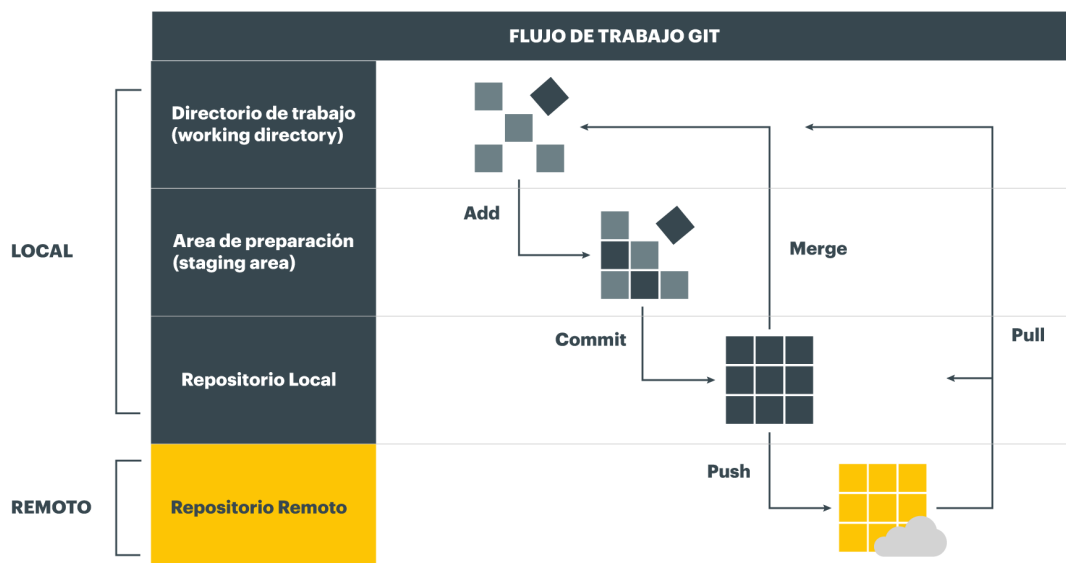


Revisemos lo aprendido hasta aquí

- Invitar colaboradores a tu repositorio

FLUJO DE TRABAJO GIT

En la guía anterior vimos un flujo de trabajo de Git con GitHub, pero en esta guía aprendimos nuevos comandos que se sumarán a este flujo de trabajo. Así que veremos el mismo diagrama con estos nuevos comandos.



EJERCICIOS DE APRENDIZAJE

Ahora es momento de poner en práctica todo lo visto en la guía.



VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

1. Para el siguiente ejercicio, van a tener que trabajar en equipo, con sus compañeros de mesa.
 - a) El facilitador de cada equipo debe **crear un repositorio público** con el nombre `practica_github` seleccionando la opción Initialize this repository with a README.
 - b) Una vez creado el repositorio el facilitador **debe invitar a los integrantes** de su mesa al mismo. Clickear en el botón Invite a collaborator y buscar a los miembros de su mesa por username o email.
 - c) Cada miembro debe **aceptar la invitación al repositorio**. Checkear la invitación el email y clickear en View Invitation.
 - d) Clonar el repositorio. Cada miembro del equipo debe clonar el repositorio con el archivo ReadMe. Luego de aceptar la invitación github te redirige al repositorio.
 - e) Cada miembro de la mesa, incluido el facilitador, debe crear su propia rama para trabajar sobre el archivo ReadMe
 - f) Ahora cada miembro de la mesa debe incluir su nombre en el archivo ReadMe de manera local y subirlo a su rama.
 - g) Cuando todos los miembros de la mesa han agregado su nombre al archivo ReadMe, de uno en uno, ir uniendo en la rama main todos vuestros cambios. Al final les debería quedar un archivo ReadMe con todos sus nombres.
2. Ahora van a continuar trabajando como mesa. Vuestra tarea ahora es que cada miembro de la mesa, incluido el facilitador, debe crear su branch y crear una de las siguientes clases: Gato, Perro, Caballo, Conejo, Pájaro y Pato. Cada uno le va a poner los atributos que desee.

El facilitador va a tener que crear el repositorio y subir un proyecto de Java vacío para que los miembros de la mesa puedan clonar y crear su clase. Una vez que cada miembro haya creado su clase en su respectiva rama, deberán unir todas las clases en la rama main, para que quede el proyecto final.

GLOSARIO

1. Puntero: **Un puntero no es más que una variable, en la cual se almacena una dirección de memoria.** Al ser una dirección de memoria, le podemos decir a un puntero que en ese lugar donde apunta queremos almacenar un valor, por ejemplo, un número. En el caso de git la rama va a almacenar la dirección de memoria de un commit de la rama main.