educación IT
CODE YOUR FUTURE

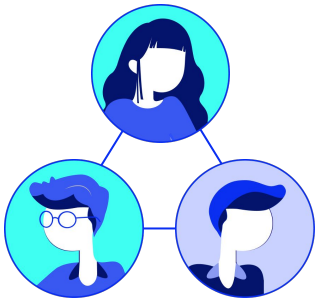
Git: Desarrollo colaborativo

Módulo 4

Desarrollo colaborativo

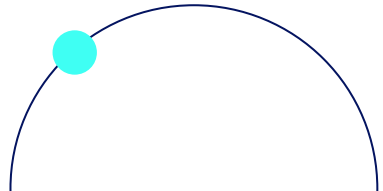
Desarrollo colaborativo

GitHub está diseñado alrededor de un flujo de **trabajo focalizado en `Pull Requests`**. Esto permite que los **desarrolladores colaboren** tanto en repositorios unificados por grupo, como en aquellos que son distribuidos en forma global a través de compañías o incluso personas desconocidas.



Por lo general, la acción de colaborar funciona de la siguiente manera:

1. Hacemos un **`Fork`** de un proyecto.
2. Creamos un **`branch`** nuevo desde el **`branch master`**.
3. Realizamos algunos **`commits`** para mejorar el proyecto.
4. Hacemos un **`push`** del **`branch`** al proyecto de GitHub.
5. Abrimos un **`Pull Request`** en GitHub.
6. Generamos una discusión al respecto de nuestro trabajo realizado y, en forma alternativa, seguimos haciendo más **`commits`**.
7. Sincronizamos el **`branch master`** nuevamente a nuestro **`Fork`**.



Git fetch

Como ya hemos mencionado, el comando **`git clone`** agrega implícitamente un **`origin`** (origen) remoto en el repositorio local.

Veremos el caso en el que se trabaje con otros desarrolladores.

Si en el repositorio local, no tenemos el historial de cambios realizados por otros usuarios, podemos obtener esa información con:

```
> git fetch <alias_remoto>
```

`git fetch` traerá todas las referencias a objetos **`commit`** que estén presentes en el repositorio remoto - que no tengamos copiado en el local -.

Esto brinda la posibilidad de inspeccionar el repositorio con posterioridad o bien, si estamos seguros de que es un trabajo testeado, unirlo con nuestro trabajo actual.

En caso de estar trabajando **con un repositorio clonado, simplemente se puede correr el comando `git fetch origin`** (ya que, como vimos, clonar un repositorio remoto crea automáticamente un remoto local con alias "**`origin`**").

Es importante notar que el comando **`git fetch`** solo descarga el historial de cambios al repositorio local, sin embargo **no realiza ningún `merge` con el repositorio actual.** Para ello, tenemos que integrar ambas ramas de trabajo en forma manual.

Git pull

Si la rama actual está configurada para apuntar a una rama remota, podemos alternativamente usar el siguiente comando:

```
> git pull
```

A diferencia del comando **`git fetch`**, **`git pull`** **descargará todos los cambios nuevos** que no tengamos en nuestro repositorio local desde el repositorio remoto **y va a intentar integrar automáticamente nuestro trabajo local actual con el que acaba de traer.**

Esto puede ser de gran ventaja para el flujo de trabajo ya que **no dependemos de una integración manual** en caso de estar seguros de lo que estemos descargando desde el proyecto remoto.

Pull Request

Pull Request

Imaginemos que tenemos un **Fork** de algún proyecto en nuestra cuenta de usuario. Queremos realizar modificaciones en él y sugerirlos como cambios potenciales dentro del proyecto original.

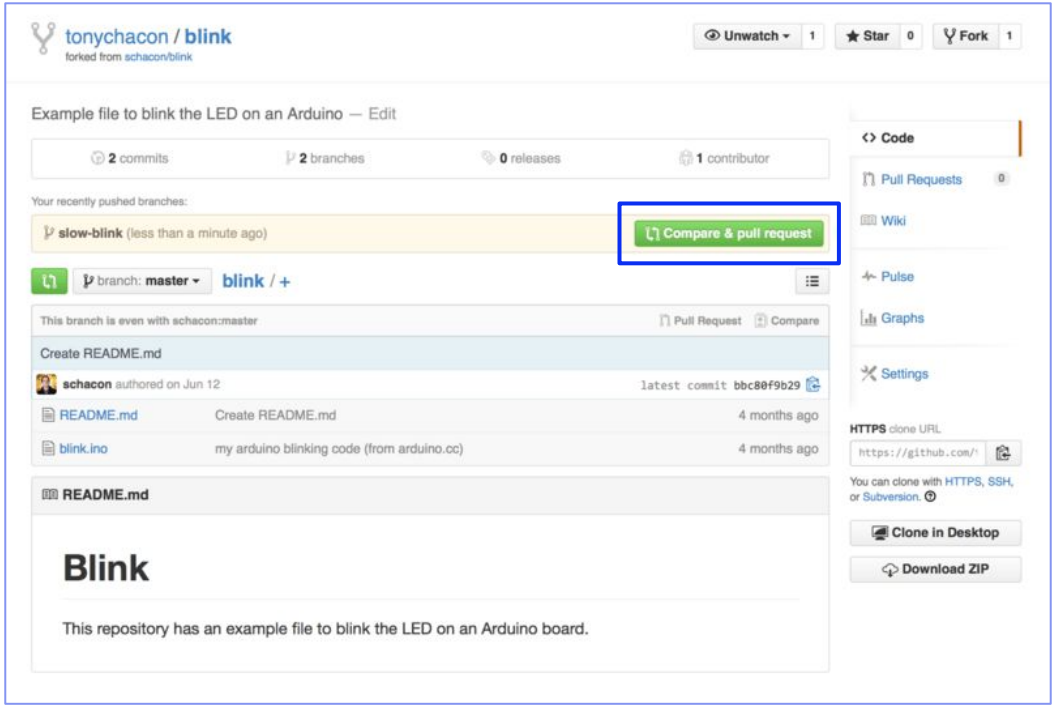
Pasos a seguir:

1. Generar el **`Fork`** del proyecto.
2. **Clonarlo** en forma local.
3. Crear una **rama de trabajo nueva.**
4. **Realizar cambios** de código.
5. **Verificar** que los cambios estén bien y hayan sido **testeados.**
6. Confirmar nuestros cambios con un **`commit`**.
7. Realizar un **`push`** de esa rama al **`Fork`** que tenemos en GitHub.

Tu primer Pull Request

Una vez dentro de GitHub podemos observar cómo la página reconoce que fue creado y subido un nuevo **`branch`**. Además, va a aparecer con un gran botón verde con la leyenda ***Compare & pull request***, que nos permitirá generar un primer **`Pull Request`**.

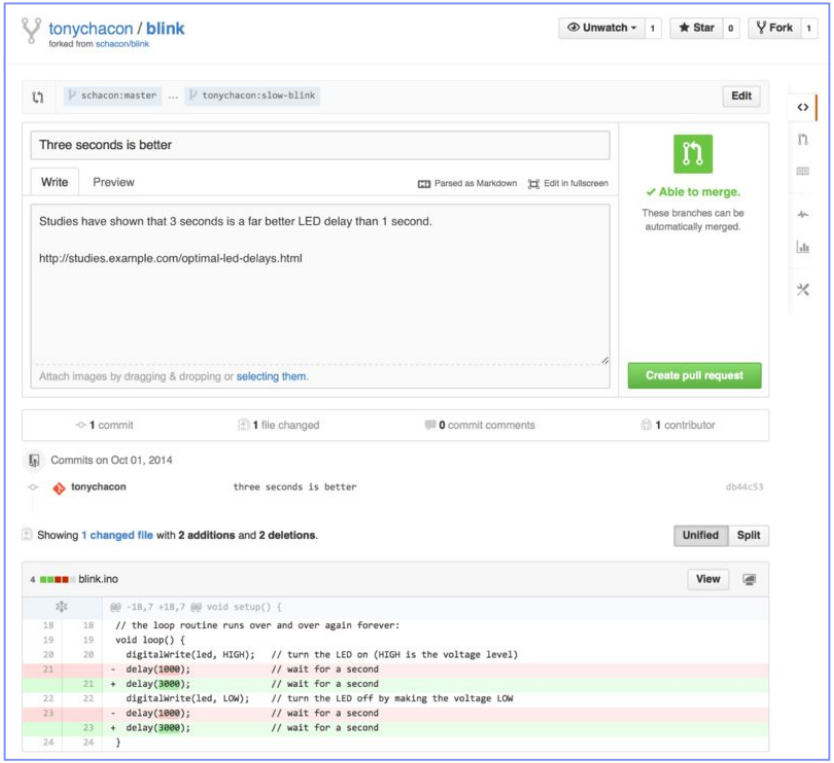
Verás en pantalla:



Si hacemos clic en el botón, accedemos a una ventana que **nos pedirá un título y una descripción** de nuestro **`Pull Request`**. Usualmente, vale la pena esforzarnos en pensar ambos datos, ya que va a ser de mucha ayuda para el dueño del proyecto original a determinar qué cambios estuvimos haciendo y si los tiene que aceptar o no.

También podemos ver una lista de uno o más **`commits`** en nuestro **`branch`** que están por sobre **(“ahead”)** el **`branch master`** y un visor de diferencias unificadas que van a tomar efectos en caso de que nuestro trabajo sea aceptado.

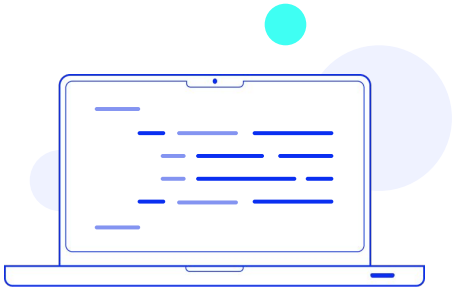
Verás la lista así:



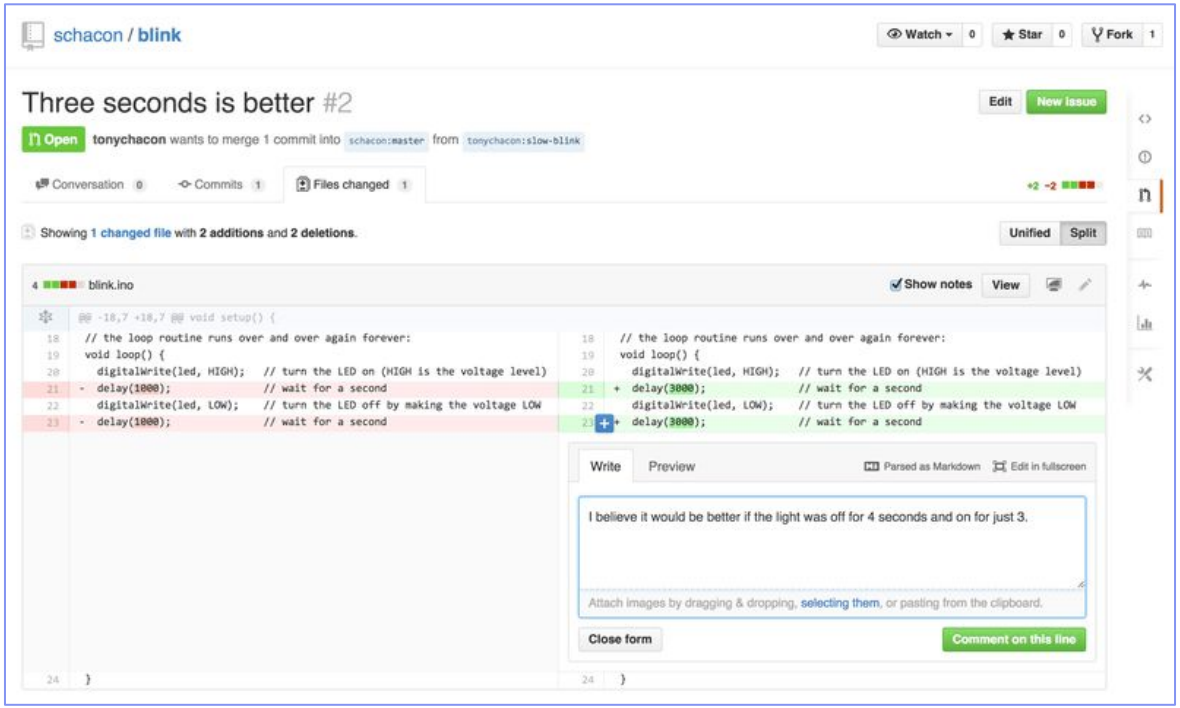
Cuando hagamos clic en el botón ***Create pull request***, el dueño del proyecto del que hicimos el **Fork** va a recibir una notificación de que alguien está sugiriendo cambios y va a poder acceder a toda la información de los cambios que propusimos.

En este punto, el dueño del proyecto puede ver los cambios sugeridos, integrarlos con el repositorio original, rechazarlos o realizar comentarios sobre ellos.

En la **vista de diferencias,** el dueño del proyecto puede hacer clic en cualquier línea de cambio propuesta para dejar un comentario sobre alguna de ellas.



Aparece en pantalla:



Una vez que este cambio fue comentado, el usuario que abrió el **`Pull Request`** (y cualquier otra persona que le esté dando seguimiento al repositorio) va a recibir una notificación. Si tenemos configurada la cuenta para poder recibir notificaciones por e-mail, podríamos ver un correo similar al siguiente:


Re: [blink] Three seconds is better (#2)

Scott Chacon <notifications@github.com> 10:55 AM (18 minutes ago)

to schacon/blink, me

In blink.ino:

```
>    digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
> -  delay(1000);              // wait for a second
> +  delay(3000);              // wait for a second
```

I believe it would be better if the light was off for 4 seconds and on for just 3.

—

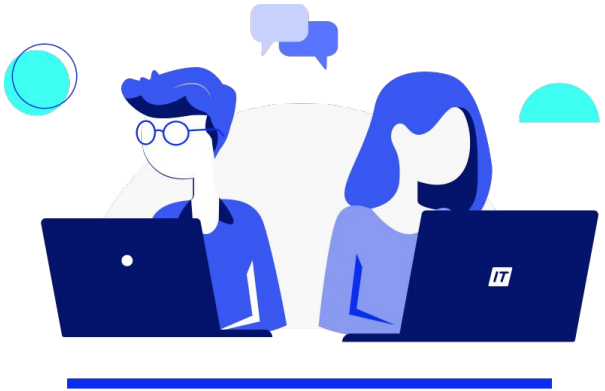
Reply to this email directly or view it on GitHub.

Además, cualquier usuario puede dejar comentarios en “generales” en el **`Pull Request`**.

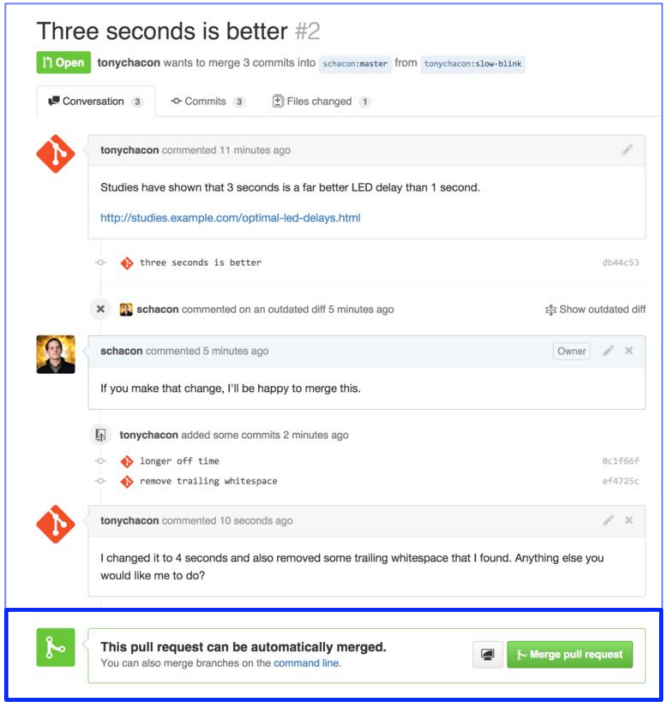
Ahora el usuario que está contribuyendo al proyecto original simplemente puede realizar más **`commits`** en el `branch` en cuestión y volver a hacer un **`push`**.

Esta acción actualizará automáticamente el `Pull Request`. Si se afectan líneas de código que habían sido previamente comentadas por alguien, se verán colapsadas en la página de la discusión del **`Pull Request`**.

Agregar **`commits`** a un **`Pull Request`** existente no activa una notificación. Una vez que se hayan realizado los cambios o corregido el trabajo, lo ideal es **dejar un nuevo comentario para informar sobre ellos al dueño del proyecto**.



Ejemplo:



Algo interesante es que **GitHub verifica que el `Pull Request` pueda ser integrado de manera limpia y sin problemas.** En ese caso, se mostrará un botón de atajo para realizar esta operación.

¡Sigamos trabajando!