

## Guía de Estudio Github

Sitio: [Instituto Superior Politécnico Córdoba](#)  
Curso: Programador Full Stack - TSDWAD - 2022  
Libro: Guía de Estudio Github

Imprimido por: Pablo Matias Jose MONTOYA  
Día: martes, 18 octubre 2022, 7:47

# Tabla de contenidos

## 1. Introducción

## 2. Github

2.1. GitHub Projects

2.2. Project Board

2.3. Issues

2.4. Milestones

2.5. GitHub Flow

# 1. Introducción

Esta sección, tiene por objeto introducir Github como herramienta de Gestión de Proyectos de Desarrollo de Software:

Objetivos

- Crear y gestionar un proyecto de software a través de un repositorio en Github, generando un flujo de trabajo con ramificaciones

## 2. Github

### ¿Qué es GitHub?

GitHub es una plataforma de colaboración formal e informal de desarrollo de software (conocida también como plataforma de *social coding*). En esta plataforma se pueden publicar repositorios remotos que funcionan bajo el sistema de control de versiones Git. La plataforma configura los proyectos nuevos como de código abierto, por lo que cualquier persona puede verlos, pero esto es configurable.



Para comprenderlo mejor, te invitamos a mirar el siguiente video:

#### What is GitHub?



Visita el sitio oficial en: <https://github.com/>



A continuación, te invitamos a tomar un cafecito y mirar el siguiente video a fin de identificar las herramientas nos provee GitHub que pueden ayudarnos a la Gestión de Proyectos.

#### Professional Guides: Managing Projects



**Nota:** Activa los subtítulos en Español.

### ¿Cómo empezar?

Lo primero para comenzar en GitHub es **crear una cuenta**, para ello:

- Accede a <https://github.com/> y luego, clic en el botón **Sign up**.
- Selecciona el plan personal gratuito con repositorio público y sigue las instrucciones. No te olvides de terminar la verificación mediante correo electrónico.

Si quieres profundizar un poco más, visita [The First Day on GitHub](#)

### ¿Cómo crear nuevo repositorio en Github?

Para crear un repositorio:



- Accede a tu cuenta de GitHub y luego, hace clic en el **menú +** (ubicado en la parte superior derecha de la pantalla) y seleccionar la opción **“New repository”** del menú contextual o bien, hacer clic en el botón repository **“New”**. Finalmente, completa el formulario.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner

Repository name \*

 enium / diw 

Great repository names are short and memorable. Need inspiration? How about **silver-goggles**?

Description (optional)

Proyecto DIW



**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



**Actividad:** A continuación, te invitamos a observar los repositorios de [Bootstrap](#) y [Angular](#). Navega por las

pestañas **“issues”** y **“project”**. Luego, en función de lo observado y según tu opinión define *¿Qué es un issue? ¿Qué es un proyecto?*

Para profundizar un poco más, click en el siguiente enlace: <https://docs.github.com/es/github/creating-cloning-and-archiving-repositories>

## 2.1. GitHub Projects

GitHub permite, a los equipos de desarrollo coordinar, trackear y actualizar las tareas de un proyecto desde cualquier lugar, a fin de mantener los proyectos transparentes no sólo para con los integrantes del equipo de desarrollo sino también clientes, usuarios, etc. Todo desde un mismo lugar y en una misma plataforma.



Para comprender mejor esto, veamos un video:



**Nota:** Activa los subtítulos en Español.

Los proyectos en GitHub permiten organizar incidencias (issues) y notas en categorías mediante tarjetas (cards) en columnas. Estas tarjetas se pueden arrastrar entre las columnas según los estados en los que se encuentren las tareas que representan.

Los paneles de proyectos son personalizables y, por tanto, adaptables a las necesidades de cada proyecto. Dentro de los paneles se pueden reordenar columnas y cartas según los criterios que mejor se adapten al proyecto o la organización. Dentro de las columnas o incluso de las tarjetas se pueden crear notas o comentarios que ayuden a entender las incidencias asociadas o que aporten información relevante al proyecto.

En GitHub existen tres tipos de tableros de proyectos:

- Pertenecientes al usuario: pueden tener incidencias de cualquier repositorio personal.
- De proyectos a nivel de organización: pueden contener incidencias de cualquier repositorio que pertenece a una organización.
- Tableros por repositorio: están enfocados en las incidencias de un único repositorio. Pueden incluir referencias o notas a incidencias de otros repositorios.

Se pueden vincular hasta 25 repositorios a cada tablero de proyecto. Vincular repositorios a proyectos facilita agregar informes de problemas al tablero a través del botón + o desde la barra lateral de la pestaña *Issues*.

### ¿Cómo crear un proyecto en GitHub?

Para crear un tablero de proyecto:

- Selecciona la pestaña **Proyecto (Projects)**.
- Hace click en el botón **New Project**. A continuación, completa el formulario con un nombre, una descripción (opcional) y una plantilla. La plantilla más común es **Basic Kanban**

fyalva / codeER

Unwatch 1 Star 0 Fork 0

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

## Create a new project

Project board name

fsER

Description (optional)

Crear una web para promoción de turismo de la provincia de Entre Ríos

Project template

Save yourself time with a pre-configured project board template.

Template: None

Create project

Finalmente, si elegiste la opción **Basic Kanban** podrás agregar incidencias, notas, etc. a las columnas del proyecto .

fyalva / codeER

Unwatch 1 Star 0 Fork 0

Code Issues 1 Pull requests Actions Projects 1 Wiki Security Insights Settings

fsER

Updated 1 hour ago

Filter cards

4 To do + ...

0 In progress + ...

0 Done

DER

#4 opened by fyalva

testing

Sprint 0

Welcome to GitHub Projects

+ Add cards

is:open

You can use the filters available in [issue search](#).

Search results

**Recuerda!!** El tablero recientemente creado se asociará automáticamente al repositorio en cuestión.



**Actividad:** Investiga, ¿Qué es un tablero KANBAN? ¿Para qué sirve?

Para profundizar, visita el sitio: <https://docs.github.com/es/issues/organizing-your-work-with-project-boards/managing-project-boards/creating-a-project-board>

## 2.2. Project Board

La herramienta Project board (o tablero de proyecto) permite organizar y priorizar el trabajo de un equipo de desarrollo.

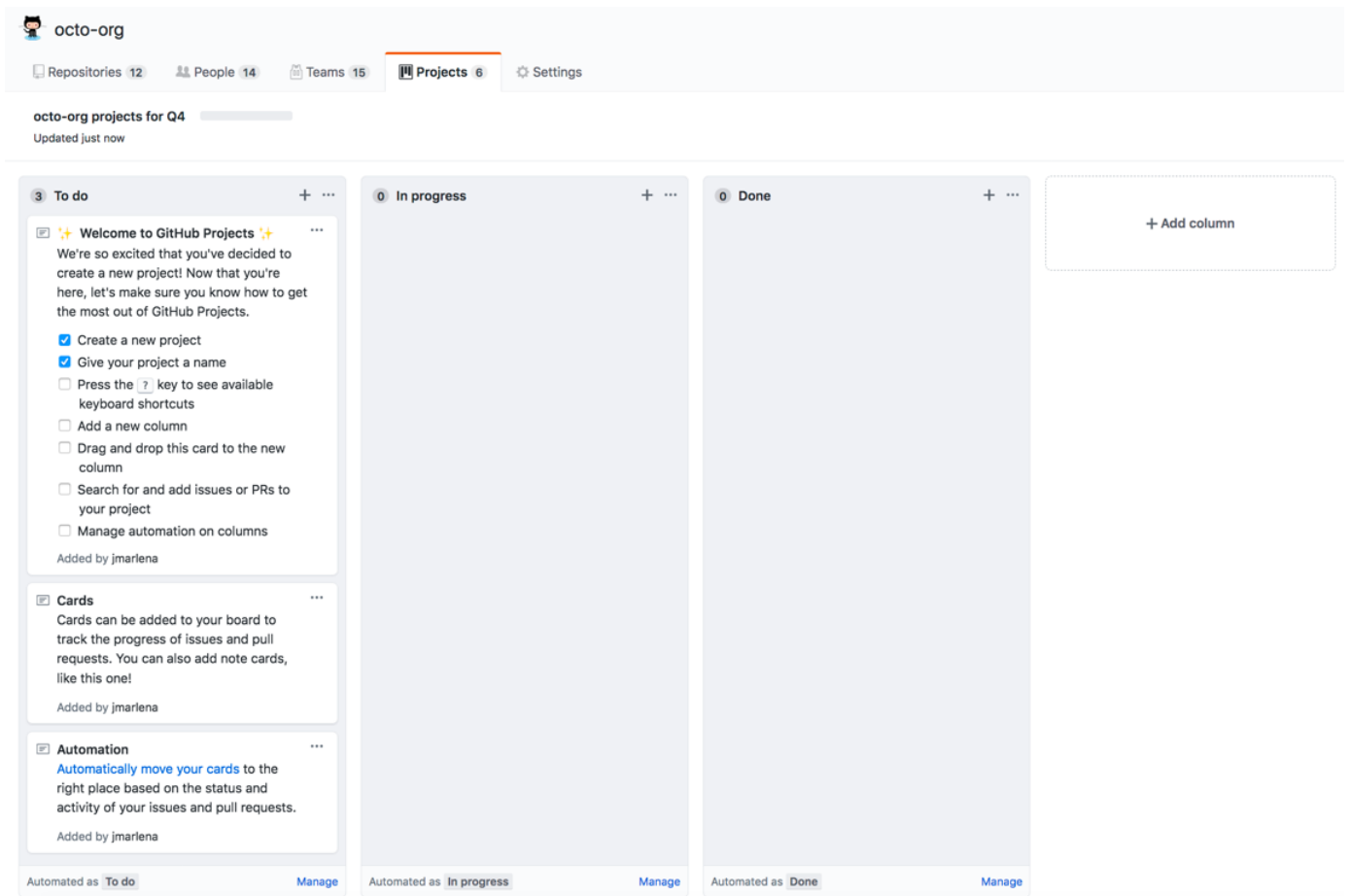
Los mismos están compuestos por incidencias y notas que se categorizan como tarjetas en columnas a tu elección. Puedes arrastrar y soltar o usar los atajos del teclado para reordenar las tarjetas dentro de una columna, mover tarjetas de columna a columna y cambiar el orden de las columnas.

También puedes crear notas dentro de las columnas para servir como recordatorios de tarea, referencias a propuestas, etc. desde cualquier repositorio en GitHub.com, o agregar información relacionada con tu tablero de proyecto.

Esta herramienta, te permite gestionar tu proyecto como desees. Es decir, puedes crear un tablero por proyecto, por sprint, etc. Además, puedes adaptarlo al flujo de trabajo que el equipo necesite.

### ¿Cómo crear un Tablero de Proyecto?

Para crear un tablero de proyecto, hacer clic en el siguiente enlace: [Instructivo Crear Tablero de Proyecto](#)



Tablero Kanban, recuperado de: <https://docs.github.com/es/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards>

Para más información hacer clic en el enlace de arriba.

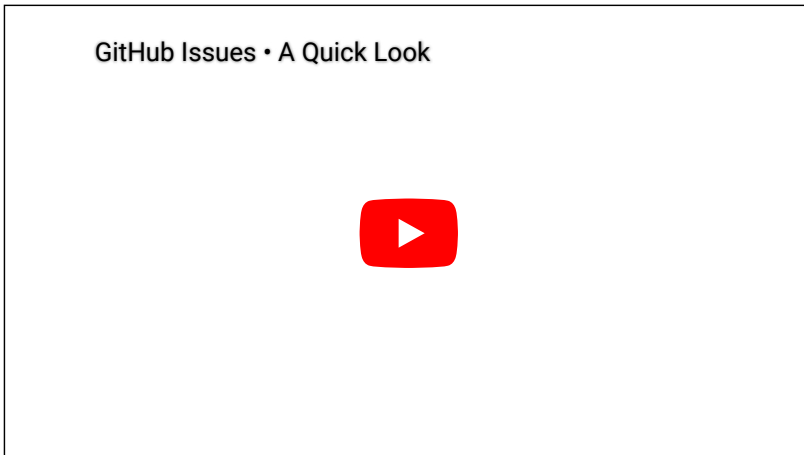


## 2.3. Issues

Una incidencia (issue) en GitHub es una unidad de trabajo designada para realizar una mejora en un sistema informático. Puede ser el arreglo de un fallo, una característica pedida, una tarea, una solicitud de documentación en específico y todo tipo de ideas o sugerencias al equipo de desarrollo.



Para comprenderlo un poco más, te invitamos a ver el siguiente video:



### ¿Cómo abrir una incidencia?

Para abrir un issue, hacer clic en el botón “New issue” y a continuación completar el siguiente formulario:

No olvides adjuntar **TODA** la información relevante al issue. Pueden ser documentos de texto, imágenes, videos, audios, etc.

Metadatos

Una unidad de trabajo o tarea, se crea completando con un título (obligatorio) y una descripción. Si la persona es miembro del equipo, opcionalmente puede asignar una serie de metadatos: etiquetas (labels), hitos (milestone), proyecto al que pertenece o responsables encargados de cerrar la incidencia.

Una vez creado, se asignará un número.

Además, si la persona que crea la incidencia es miembro del equipo con permisos sobre el repositorio, puede opcionalmente asignar etiquetas, hitos, proyecto al que pertenece o responsables encargados de atender la incidencia.

### Metadatos

#### Assignees (opcional)

Permite asignar 1 hasta 10 personas a la tarea.

#### Milestone (opcional)

Son categorías que se utilizan para tener un filtro más adecuado de la información. Cada milestone puede tener una fecha programada indicando el tiempo que es necesario para cumplir cierta tarea.

#### Label (opcional)

Especifica el tipo de issue. Pudiendo ser: bug, documentation, duplicate, invalid, etc. Es posible crear nuevas etiquetas.

**"Etiquetar los issues permite buscarlos rápidamente más tarde".**

Project (opcional)

Permite configurar el proyecto al que pertenece la incidencia.

Para más información, consulta el siguiente enlace: <https://docs.github.com/en/issues>



**Actividad:** Ahora ya sabemos qué es un incidente (issue). Sin embargo, puede que existan recomendaciones al momento de escribir un issue ¿no?. *¿Habrá mejores formas de escribir un issue? ¿Cuáles podrían ser esas recomendaciones?*

## 2.4. Milestones

Los hitos (*Milestones*) son grupos de incidencias y que ayudan a seguir el progreso de estas. Desde la página de detalle de un hito se pueden observar los siguientes datos:

Una descripción del hito proporcionada por el usuario, que puede incluir información como una descripción general del proyecto, equipos relevantes y fechas de vencimiento proyectadas.

- La fecha de vencimiento del hito
- El porcentaje de finalización del hito
- La cantidad de incidencias abiertas y cerradas asociadas con el hito.
- Una lista de incidencias abiertas y cerradas asociadas con el hito.



**Actividad:** Accede al repositorio de [Angular](#), analiza y observa los milestone y responde: En un proyecto de software: ¿Qué entiendes por hito (milestone)? ¿Cuándo deberíamos crear uno?

### ¿Cómo crear hitos (milestone)?

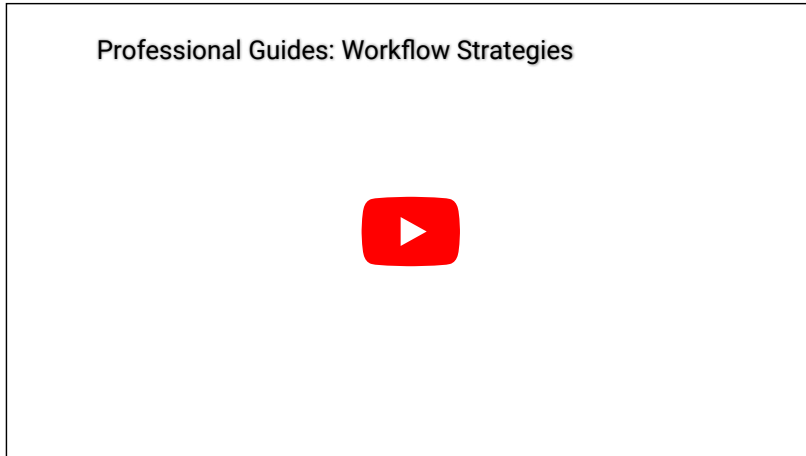
Para ello, hace click en el siguiente enlace [Instructivo para crear un Hito](#)

## 2.5. GitHub Flow

GitHub propone un **flujo de trabajo** ideado especialmente para trabajar de manera colaborativa en un proyecto de desarrollo de software.



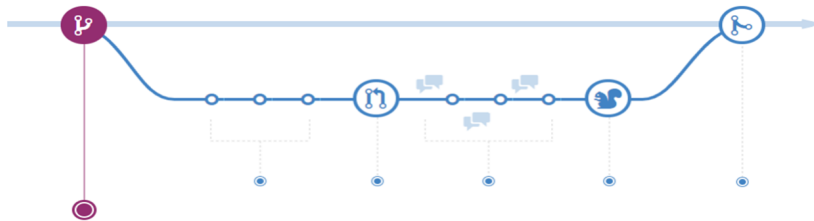
Para comprenderlo, veamos un video:



**Nota:** Activa los subtítulos en Español.

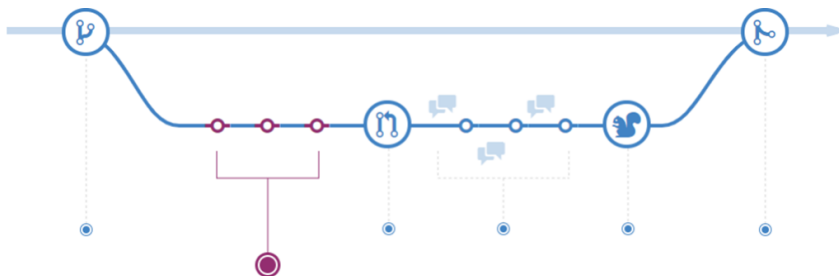
### ¿Cuáles son los eventos que propone GitHub Flow?

1. **Crear una rama (branch).** Un branch, es una línea de tiempo que nos provee un entorno de trabajo para trabajar en una nueva funcionalidad, bugs, etc. sin afectar el branch principal.



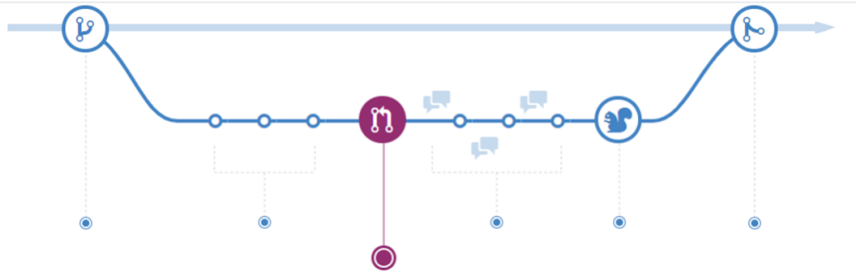
2. **Hacer cambios y crear confirmaciones (commits).** Un commit, es una unidad de cambio. Es importante hacer una descripción de los cambios efectuados. Los commit permiten crear una historia transparente del trabajo que otros, pueden leer para comprender mejor lo que se ha hecho y porqué.

Además, cada confirmación se considera una unidad de cambio separada. Esto le permite revertir los cambios si se encuentra un error o si decide ir en una dirección diferente. Los mensajes de confirmación son importantes, especialmente porque Git rastrea sus cambios y luego los muestra como confirmaciones una vez que se envían al servidor. Al escribir mensajes de compromiso claros, puede facilitar que otras personas lo sigan y brinden comentarios.



3. **Abrir una solicitud de incorporación (pull request).** Un pull request, es una comparación entre dos branches y permite invitar a otro desarrollador ya sea porque estamos atascados y necesitamos ayuda o, para hacer una revisión de código. Para ello, sólo debes utilizar el sistema de menciones de GitHub que consiste en simplemente, escribir el nombre del usuario que estamos invitando agregando previamente el signo @ previo a su nombre. Ej.: @juan\_lopez en el mensaje de

solicitud o comentarios.



## ¿Qué es un pull request?



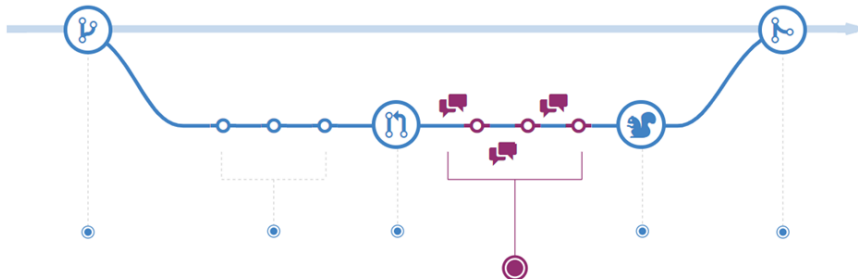
Para comprenderlo mejor, veamos un video:

### Pull Requests • GitHub & Git Foundations

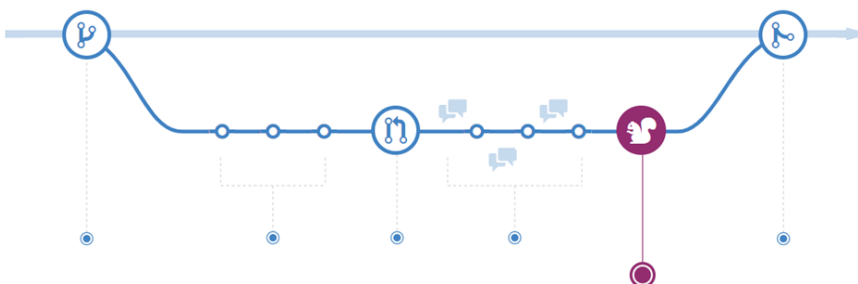


**Nota:** Activa los subtítulos en Español.

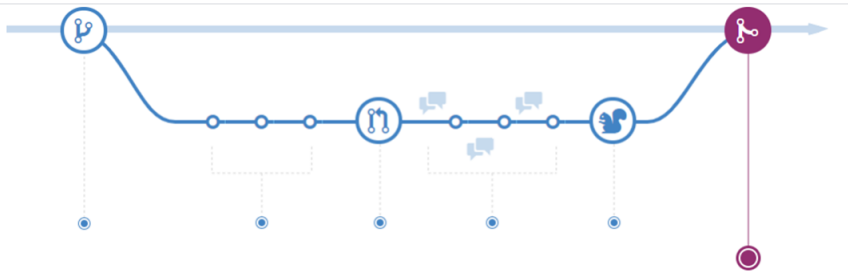
4. **Discutir y revisar el código (code review)**, una vez que se ha abierto una solicitud de incorporación, la persona o el equipo que revisa sus cambios puede tener preguntas o comentarios. Quizás el estilo de codificación no coincide con las pautas del proyecto, al cambio le faltan pruebas unitarias o tal vez todo se ve bien y los accesorios están en orden. Las solicitudes de incorporación están diseñadas para fomentar y capturar este tipo de discusiones. También puede continuar fusionando su rama luego de la discusión y los comentarios sobre sus confirmaciones. Si alguien comenta que se olvidó de hacer algo o si hay un error en el código, puede solucionarlo en su rama y fusionar el cambio. GitHub mostrará sus nuevas confirmaciones y cualquier comentario adicional que pueda recibir en la vista *Pull Request* unificada.



5. **Desplegar (deploy)**, con GitHub, se puede desplegar desde una rama para la prueba final en producción antes de fusionarse con **main**. Una vez que se haya revisado su solicitud de incorporación y la rama pase las pruebas, puede desplegar sus cambios para verificarlos en producción. Si su rama causa problemas, puede revertirla implementando la rama principal existente en producción. Los diferentes equipos pueden tener diferentes estrategias de despliegue. Para algunos, puede ser mejor desplegar en un entorno de prueba especialmente provisto. Para otros, el despliegue directamente en producción puede ser la mejor opción en función de los otros elementos de su flujo de trabajo.



6. **Fusionar (merge).** Una vez que los cambios se han verificado en producción, es posible hacer el merge. Una vez completado esto, GitHub conserva un registro de los cambios históricos.



7. **Eliminar el branch (rama)** creada inicialmente.

Figuras: Flujo de trabajo propuesto por GitHub, Recuperado de: <https://guides.github.com/introduction/flow/>

Para profundizar un poco más consulta el enlace de arriba.