Github

GitHub es una plataforma de colaboración formal e informal de desarrollo de software (conocida también como plataforma de social coding). en esta 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,

**Incidencias**

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

##### **Hitos**

Los hitos ( Milestones ) son grupos de incidencias y que ayudan a seguir el progreso de estas. veamos el siguiente video de **como crear un hito** en Github.

Desde la página de detalle de un hito se pueden observar los siguientes datos:

●       Una descripción del hito proporcionado 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.

##### **Etiquetas**

Cada incidencia puede ser etiquetada bajo categoría (error, documentación, duplicado, inválido, etc) de manera que estas etiquetas pueden identificar rápidamente a qué grupo o tipo de tarea corresponde cada incidencia. Además del conjunto por defecto de etiquetas que tiene GitHub, se pueden crear otras que sean específicas para la organización o del proyecto.

## Tarea desafío a subir del módulo

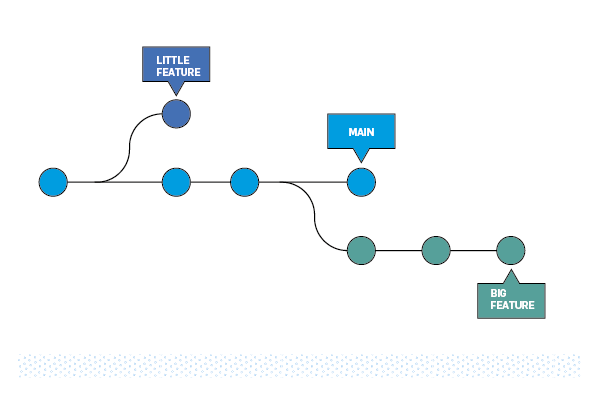
Te recordamos e invitamos a que utilices este conocimiento para crear los repositorios en github para taplicación y lo subas en la sección: **Desafío a subir del módulo**(ver imagen de referencia).  
El objetivo es poner en práctica todo lo aprendido. Para ello deberás considerar tanto el material de lectura como los contenidos desarrollados en los encuentros en vivo y los aportes de otros compañeros.

## Flujos de trabajo Github

#### **Flujos de Trabajo**

##### **Ramas**

La creación de ramas es una función disponible en la mayoría de los sistemas de control de versiones modernos. La creación de ramas en otros sistemas de control de versiones puede tanto llevar mucho tiempo como ocupar mucho espacio de almacenamiento. En Git, las ramas son parte del proceso de desarrollo diario. Las ramas de Git son un puntero eficaz para las instantáneas de tus cambios. 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. Con git, la gestión de ramas se hace a través del comando **git branch**.



Representación de un repositorio con dos líneas de desarrollo aisladas, una para una función pequeña y otra para una función más extensa.

La implementación que subyace a las ramas de Git es mucho más sencilla que la de otros modelos de sistemas de control de versiones. En lugar de copiar archivos entre directorios, Git almacena una rama como referencia a una confirmación. En este sentido, una rama representa el extremo de una serie de confirmaciones, es decir, no es un contenedor de confirmaciones. El historial de una rama se extrapola de las relaciones de confirmación.

Durante la lectura, recuerda que las ramas de Git no son como las ramas de SVN. Las ramas de SVN solo se usan para capturar el esfuerzo de desarrollo a gran escala ocasional, mientras que las ramas de Git son una parte integral del flujo de trabajo diario. El siguiente contenido amplía la información sobre la arquitectura interna de creación de ramas de Git.

##### **Funcionamiento**

Una rama representa una línea independiente de desarrollo. Las ramas sirven como una abstracción de los procesos de cambio, preparación y confirmación. Pueden concebirse como una forma de solicitar un nuevo directorio de trabajo, un nuevo entorno de ensayo o un nuevo historial de proyecto. Las nuevas confirmaciones se registran en el historial de la rama actual, lo que crea una bifurcación en el historial del proyecto.

El comando **git branch** te permite crear, enumerar, cambiar el nombre y eliminar ramas. No te permite cambiar entre ramas o volver a unir un historial bifurcado. Por este motivo, **git branch** está estrechamente relacionado con los comandos **git checkout** y **git merge**.

##### **Opciones comunes**

El comando principal, sin parámetros, permite listar todas las ramas del repositorio

**git branch**

Para crear una rama nueva se usa el comando como se muestra a continuación:

**git branch <branch>**

Para eliminar una rama, es necesario agregar la opcion **-d**

**git branch -d <branch>**

Si se desea cambiar el nombre de una rama, se puede utilizar la opción **-m**, es importante mencionar que cambia el nombre de la rama sobre la cual se está trabajando.

**git branch -m <branch>**

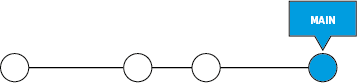
Finalmente, para listar todas las ramas en el repositorio remoto se puede utilizar el comando:

**git branch –a**

## Creación de ramas

##### **Creación de ramas**

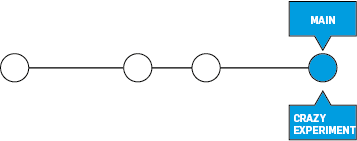
Es importante comprender que las ramas son solo punteros a las confirmaciones. Cuando creas una rama, todo lo que Git tiene que hacer es crear un nuevo puntero, no modifica el repositorio de ninguna otra forma. Si empiezas con un repositorio que tiene este aspecto:



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

**git *branch crazy-experiment***

El historial del repositorio no se modificará. Todo lo que necesitas es un nuevo puntero de la confirmación actual:



Ten en cuenta que este comando solo crea la nueva rama. Para empezar a añadir confirmaciones, necesitas seleccionarla con el comando **git checkout** y, a continuación, utilizar los comandos estándar **git add** y **git commit**.

##### **Creación de ramas remotas**

Por ahora, todos los ejemplos han ilustrado operaciones de ramas locales. El comando **git branch** también funciona con ramas remotas. Para trabajar en ramas remotas, primero hay que configurar un repositorio remoto y añadirlo a la configuración del repositorio local.

**git remote *add <remote-repo> <remote-repo-URL>***

**git push <remote-repo> crazy-experiment~**

Este comando enviará una copia de la rama local crazy-experiment al repositorio remoto **＜remote-repo＞**

##### **Eliminación de ramas**

Una vez que hayas terminado de trabajar en una rama y la hayas fusionado con el código base principal, puedes eliminar la rama sin perder ninguna historia:

**git branch -d crazy-experiment**

No obstante, si la rama no se ha fusionado, el comando anterior mostrará un mensaje de error: “error: The branch 'crazy-experiment' is not fully merged. If you are sure you want to delete it, run 'git branch -D crazy-experiment'.”

Esto te protege ante la pérdida de acceso a una línea de desarrollo completa. Si realmente quieres eliminar la rama (por ejemplo, si se trata de un experimento fallido), puedes usar el indicador -D (en mayúscula):

**git branch -D crazy-experiment**

Este comando elimina la rama independientemente de su estado y sin avisos previos, así que úsalo con cuidado.

Los comandos anteriores eliminarán una copia local de la rama. La rama seguirá existiendo en el repositorio remoto. Para eliminar una rama remota, ejecuta estos comandos.

**git push origin --delete crazy-experiment**

o bien con el comando:

**git push origin :crazy-experiment**

Enviarán una señal de eliminación al repositorio de origen remoto que desencadena la eliminación de la rama remota crazy-experiment.