MANUAL DE GITHUB

FLUJO DE DESARROLLADOR

Instructivo para el equipo de desarrollo

Gerencia de Soluciones

CONTENIDO

¡Hola! Soy Anna, aquí aprenderás los pasos básicos que debe realizar un desarrollador:

1. Requerimientos.
2. Conceptos generales.
3. Roles.
4. Prefijos.
5. Solicitud de repositorio
6. Solicitud de rama.
7. Arquitectura de ramas.
8. Estrategia de versionamiento
9. GitHub desktop.
10. ¿Cómo clonar un repositorio?
11. Readme.
12. ¿Cómo realizar un commit?

REQUERIMIENTOS

A continuación, se listarán los elementos necesarios para poder llevar acabo la ejecución del instructivo:

- Conexión estable a internet.
- Poseer cuenta GitHub ligada al correo institucional (si no posee, crearla).
- Tener habilitada la autenticación de dos factores (2FA) en su cuenta GitHub.
- Instalar Git desde su página oficial.
- GitHub Desktop.

CONCEPTOS GENERALES

Antes de iniciar, debemos conocer un poco sobre los comandos, términos o acciones más comunes en las tareas que se desempeñen, tales como:

- **REPOSITORIO GITHUB**: comprende un espacio de almacenamiento virtual para la gestión de las versiones de código de un proyecto.
- **CLONAR UN REPOSITORIO**: comprende la creación de una copia en nuestro ambiente local (equipo personal) del repositorio remoto.
- **GIT COMMIT**: comando más utilizado para guardar cualquier cambio efectuado en las ramas de esta herramienta.
- **PULL REQUEST**: petición que necesita aprobación para integrar los cambios de código de una rama a otra.

- **RAMA MAIN**: es la **rama principal** de nuestro repositorio (ambiente producción).
- **RAMA QA**: es la **rama principal** utilizada para la certificación de los cambios (ambiente QA).
- **RAMA DEVELOP**: es la **rama principal** utilizada para el desarrollo de nuevas funcionalidad y/o características del producto (ambiente desarrollo).
- **RAMAS _QA y _DEVELOP**: son las **ramas secundarias**, **creadas a partir de sus ramas principales** (develop y qa), utilizadas para procesar los cambios desarrollados (_develop) y que, posteriormente, serán certificados (_qa).

ROLES

Dentro de este flujo, si bien existen acciones y/o comandos, también existen quienes ejecutan estas acciones, las cuales son personas con los siguientes roles:

- **DESARROLLADOR**: será quien podrá trabajar sobre el código del proyecto.
- **REVIEWER**: será la persona encargada de las revisiones y aprobaciones de código nuevo.
- **ASIGNEES**: será el administrador quien realizará el paso de un ambiente a otro mediante el **PULL REQUEST**.

PREFIJOS

Para la identificación del cambio que se va realizar, utilizaremos los siguientes prefijos:

- **feature**: utilizado cuando se esté desarrollando una **nueva** característica o funcionalidad.
- **refactor**: utilizado cuando se esté realizando una **mejora** de una característica o funcionalidad existente.
- **fix**: utilizado para identificar los cambios que requieren corrección de errores. **(ambiente desarrollo).**
- **hotfix**: utilizado para identificar los cambios que requieren corrección de errores. **(ambiente producción).**
- **integration**: utilizado para identificar las ramas donde se estarán comparando cambios de 2 o más desarrolladores para la resolución de conflictos.

SOLICITUD DE REPOSITORIO

Para la creación de un **nuevo repositorio**, es necesario realizar una solicitud al equipo administrador, considerando lo siguiente:

1. Completar el **Formato de Solicitud de Repositorio**. El formato se encuentra en el siguiente enlace:
2. Enviar el **Formato de Solicitud de Repositorio (vía correo)** a los administradores de GitHub**.**
3. El equipo administrador realizará lo siguiente:
   a) Revisión de la solicitud enviada.
   b) Si la solicitud está correcta, se procede con la creación del repositorio solicitado.
4. Se enviará la **Resolución de la Solicitud (vía correo)**.

IMPORTANTE: esta solicitud solo debe ser realizada por parte de **líder de equipo / líder de proyecto**.

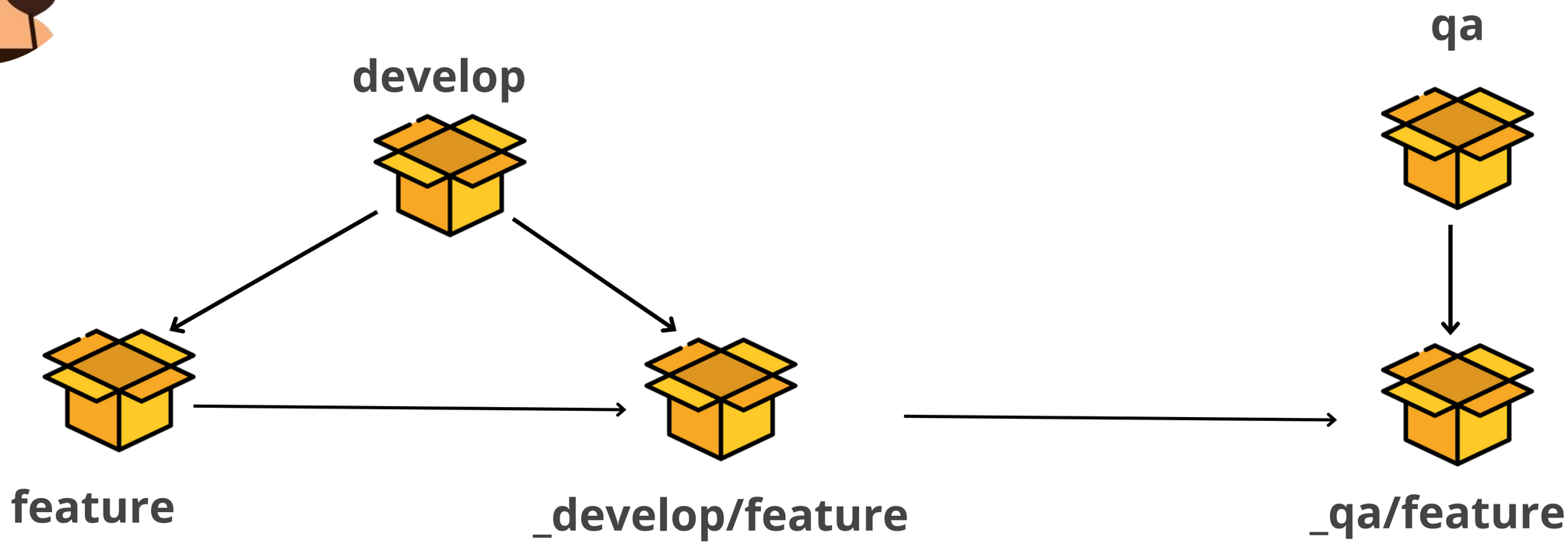
SOLICITUD DE RAMA

Para tener acceso a un repositorio y la creación de una **nueva rama**, es necesario realizar una solicitud al equipo administrador, considerando lo siguiente:

1. Completar el **Formato de Solicitud de Rama**. El formato se encuentra en el siguiente enlace:
2. Enviar el **Formato de Solicitud de Rama (vía correo)** a los administradores de GitHub**.**
3. El equipo administrador realizará lo siguiente:
   a) Revisión de la solicitud enviada.
   b) Si la solicitud está correcta, se procede con la creación de la rama solicitada.
4. Se enviará la **Resolución de la Solicitud (vía correo)**.

ARQUITECTURA DE RAMAS

La estrategia de versionamiento contará con reglas y requerimientos por cada uno de los ambientes.



ESTRATEGIA DE VERSIONAMIENTO

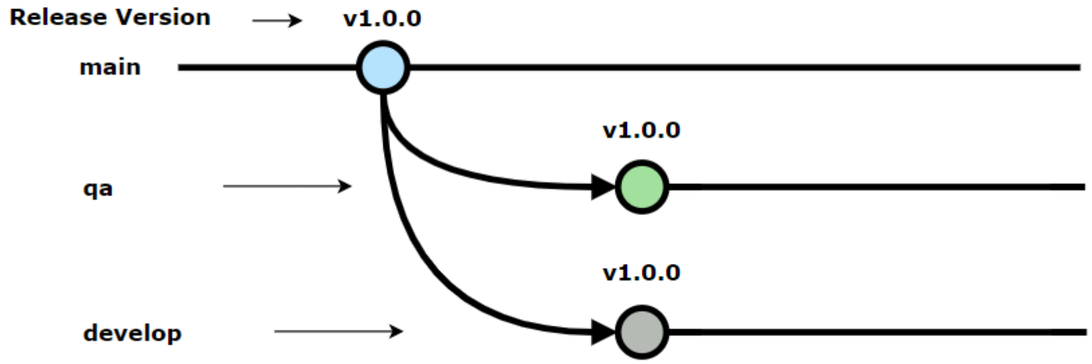
Como parte de la estrategia, nuestro repositorio inicialmente contará con **3 ramas principales**:

- **main**: es la rama que apunta a nuestro **ambiente de producción**.

- **qa**: es la rama que apunta a nuestro **ambiente de certificación**. Esta rama será creada a partir de la rama **main.**

- **develop**: es la rama que apunta a nuestro **ambiente de desarrollo.** Al igual que la rama de **qa,** será creada a partir de la rama **main.**

Creación de ramas principales



IMPORTANTE: en las **ramas principales**, **NO está permitido** integrar o realizar cambios directamente; esto sin excepciones.

Algunas de las restricciones de las ramas principales son:

1. Estas **nunca deben ser afectadas directamente** (commits o merge directos), ya que de esta forma se mantienen con la versión más estable (según ambiente producción).

2. Solo se afectan por medio de una **integración de cambios** (Pull Request).

3. La integración de cambios que se realicen a estas ramas serán gestionados por el personal encargado de la administración del repositorio.

Para crear una funcionalidad, realizar una mejora o algún ajuste, se tomará en cuenta lo siguiente:

1. Mediante la **Solicitud de Gestión de Repositorio,** se debe indicar la **información de los usuarios** que requieren el acceso, así como el **detalle de los cambios que se van realizar**. Dependiendo del tipo de cambio, se utilizará algún prefijo que sea acorde al cambio a realizar **(feature, refactor, fix, hotfix, integration).**

2. Una vez validada la gestión y el detalle del cambio por parte del personal administrativo, se procederá con la creación de la rama o las ramas donde se estarán desarrollando las funcionalidades.

NOMENCLATURA DE RAMAS

La estructura de una rama creada por el administrador para un desarrollador, estará definida de la siguiente manera:

[prefijo]/[region]/[tarjeta-proyecto]/[usuario]_[descripcion]

prefijo:

feature
refactor
fix
hotfix
integration

región:

hn
sv
ni
be

tarjeta ó proyecto:

se coloca el número de tarjeta o proyecto vinculado

usuario desarrollador

detalle de lo que se va desarrollar

EJEMPLOS:

fix/hn/1000123/BA_dmeza_dev_correccion_pantalla_usuarios

feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios

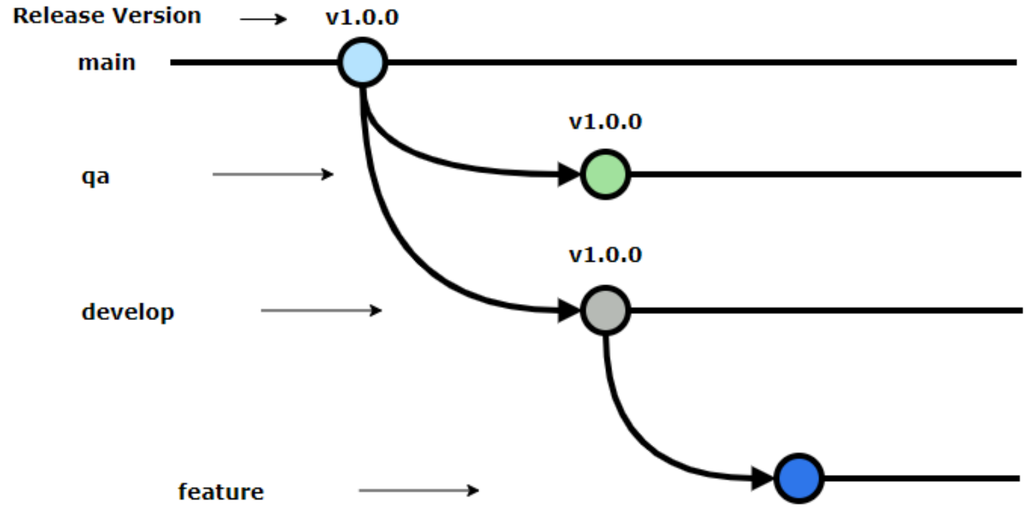
Una vez creada la rama, la estructura de nuestro proyecto a nivel de GitHub se verá de la siguiente manera:

- Para nuestro ejemplo, la rama del desarrollador **(feature),** se crea a partir de la **rama principal de desarrollo (develop).**

- Los cambios del desarrollador, se irán integrando **(commits)** a la rama que se creó específicamente para el cambio solicitado.

- Esta rama **(feature),** sólo puede ser utilizada por el desarrollador que la solicitó.

Creación de rama para el desarrollador



PROYECTOS

Consideraciones al momento de integrar los cambios a los distintos ambientes:

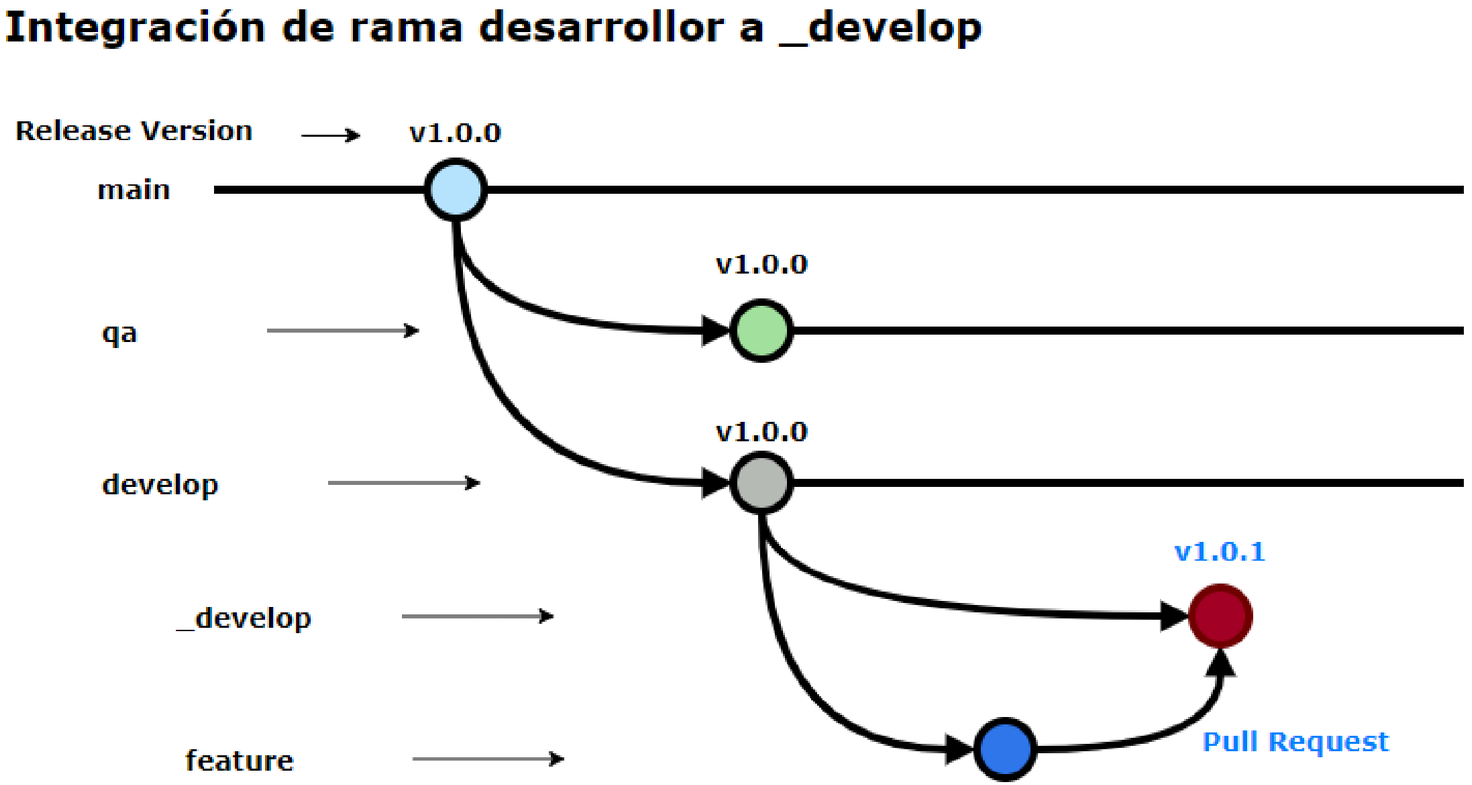
- Para la integración al ambiente de desarrollo, se creará, por parte del administrador, una rama especial para ese cambio. La rama a utilizar tendrá la siguiente estructura:

  develop/[prefijo]/[region]/[tarjeta-proyecto]/[usuario][descripcion]

  Ejemplo: _develop/feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios

- Si al momento de la **integración de los cambios** al ambiente de desarrollo **(_develop),** existiera algún conflicto por alguna nueva versión, el desarrollador deberá descargar esos cambios a su rama de desarrollo **(merge develop -> feature)** y validar los conflictos existentes. Luego de solventarlos, se solicitará la integración nuevamente.

Cuando se finalicen los cambios por parte del desarrollador, se integrará a su respectiva rama **_develop**, así como se ve en el siguiente ejemplo:



- Finalizando la integración en el ambiente de desarrollo **(_develop)**; si no existen conflictos, se procederá con la integración al ambiente de certificación **(_qa).**

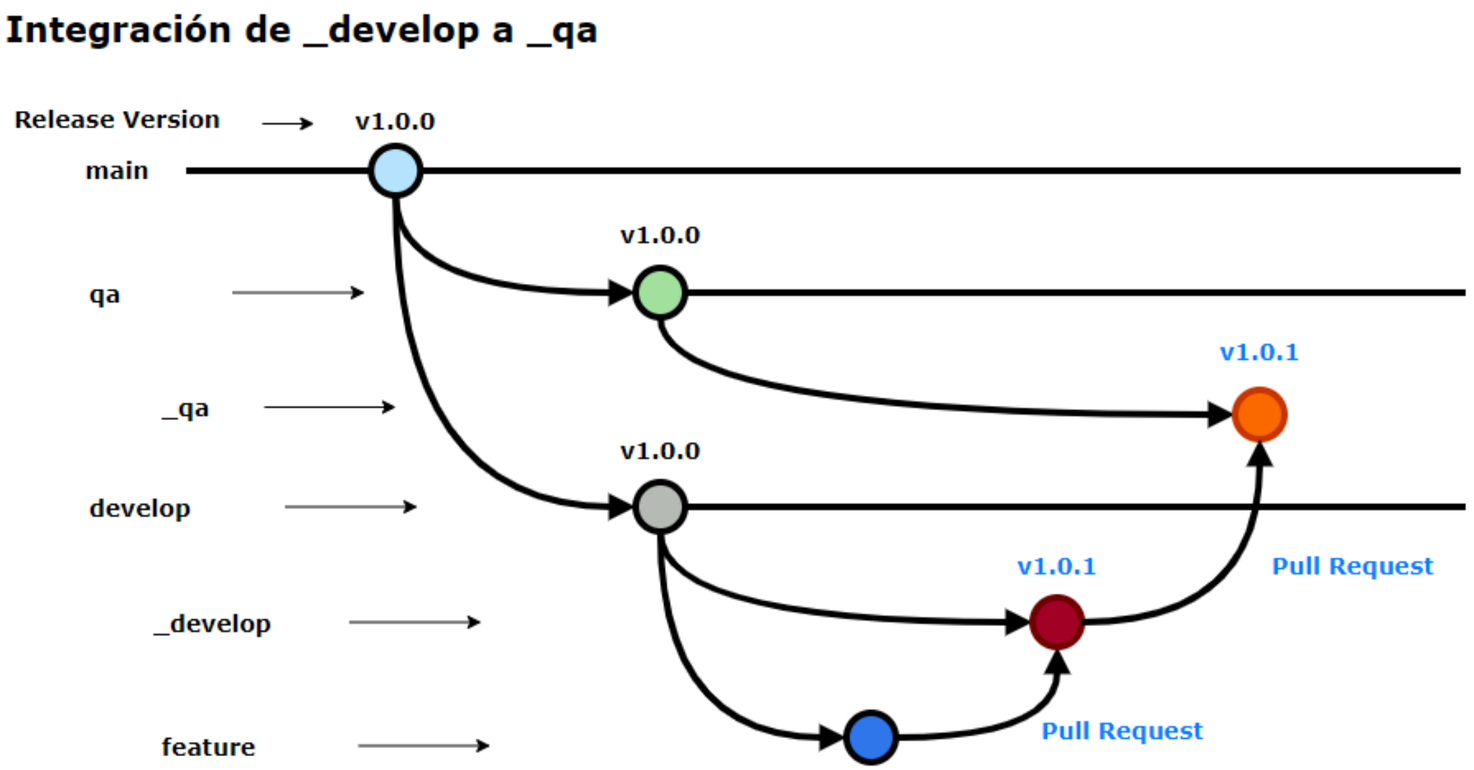
- Al igual que la integración al ambiente de desarrollo, se creará por parte del administrador, una rama especial para su certificación. La rama a utilizar tendrá la siguiente estructura:

  qa/[prefijo]/[region]/[tarjeta-proyecto]/[usuario][descripcion]

  Ejemplo: _qa/feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios

1. Si al momento de la **integración de los cambios** al ambiente de certificación **(_qa),** existiera algún conflicto por alguna nueva versión, el desarrollador deberá descargar esos cambios a su rama de desarrollo **(merge develop -> feature)** y validar los conflictos existentes. Luego de solventarlos, se solicitará la integración nuevamente.

- Este es un ejemplo de cómo será la integración del ambiente de desarrollo **(_develop)** al ambiente de certificación **(_qa).**



En esta etapa, nuestro cambio se encuentra en proceso de certificación. Algunas de las consideraciones en esta etapa son:

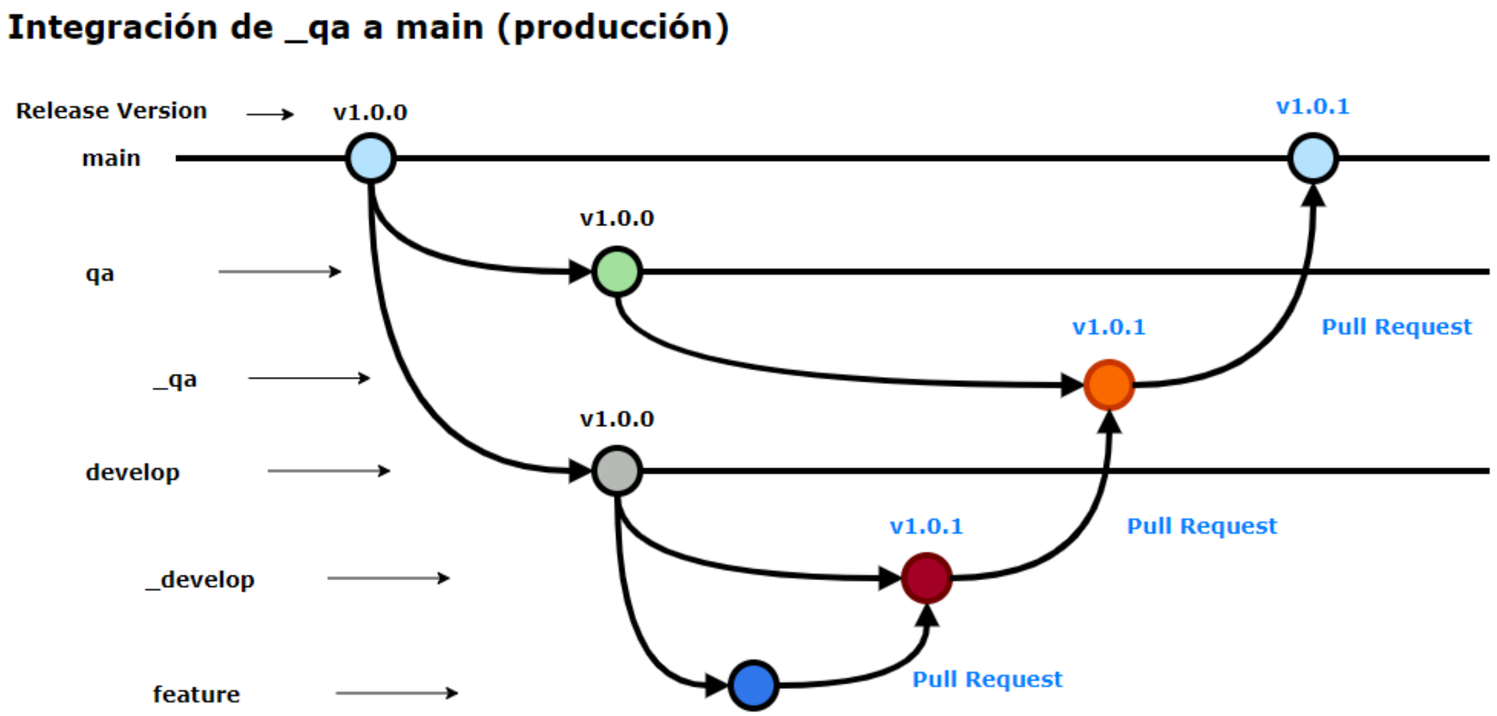
- Si se presentan inconvenientes o errores en la etapa de certificación, para poder solventar los errores, el desarrollador deberá utilizar la rama asignada:

  Ejemplo: **feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios**

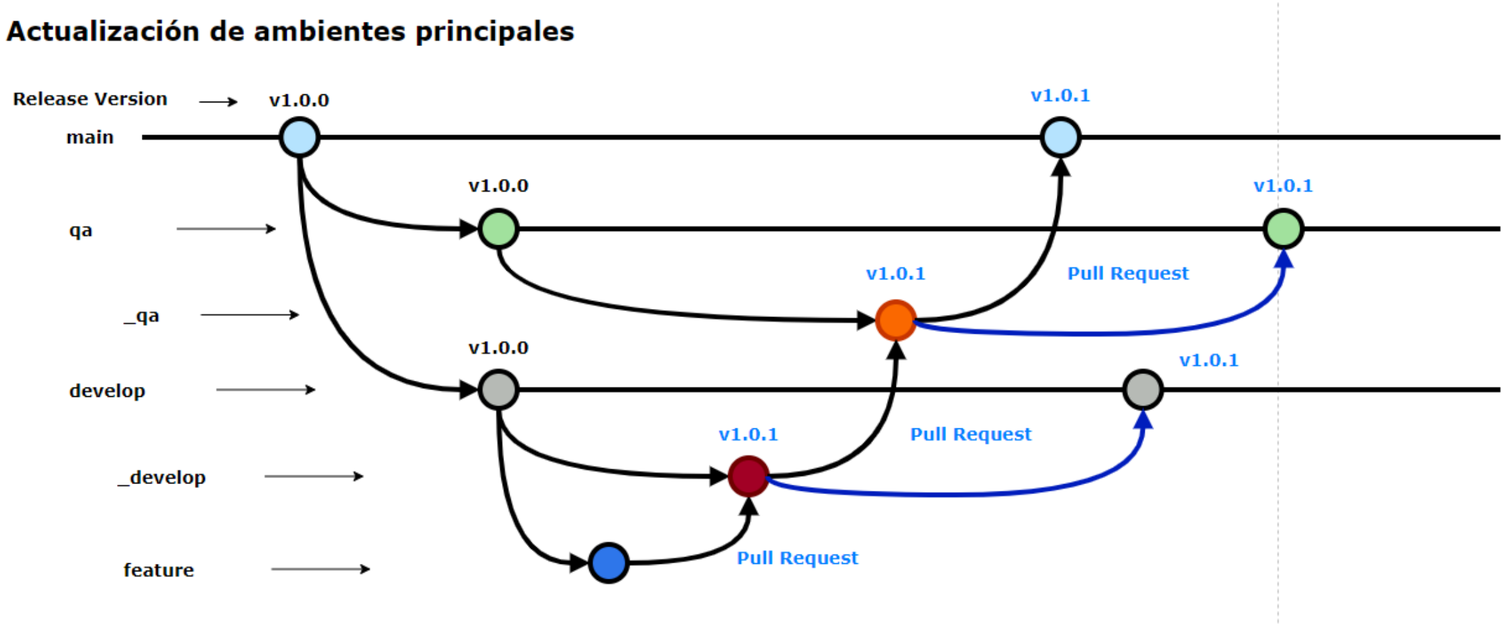
- Una vez finalizadas las correcciones, se solicitará nuevamente la integración al ambiente de desarrollo **(_develop)** y, posteriormente, al ambiente de certificación **(_qa).**

IMPORTANTE: las ramas utilizadas para el desarrollo de la funcionalidad, **NO se eliminarán** sino hasta que el cambio sea **implementado en ambiente de producción y el periodo de garantía haya finalizado**.

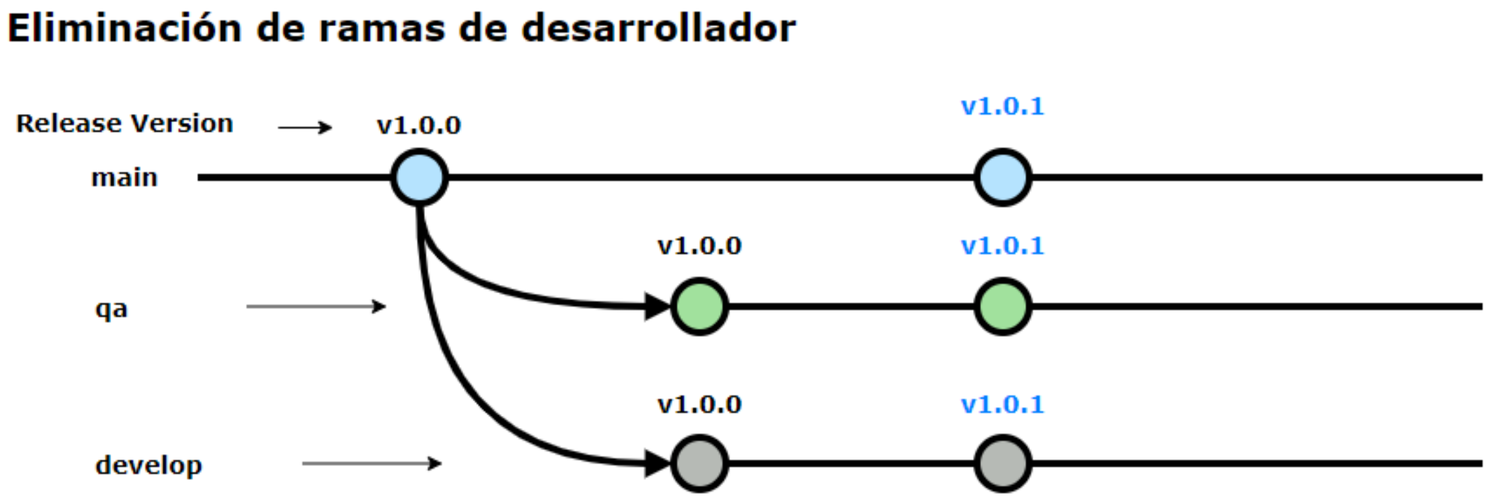
- Una vez certificado el cambio, se realizará la integración al ambiente de producción **(main)**.



- Una vez el cambio **se integre al ambiente de producción y finalice el tiempo de garantía**, se deberán actualizar los ambientes principales de desarrollo **(merge _develop -> develop)** y de certificación **(merge _qa -> qa)**, con la nueva versión estable.



- Cuando los ambientes principales de desarrollo **(develop)** y el de certificación **(qa)** estén actualizados, **el administrador eliminará las ramas** que se crearon para el desarrollo **(feature, _develop y _qa)**.



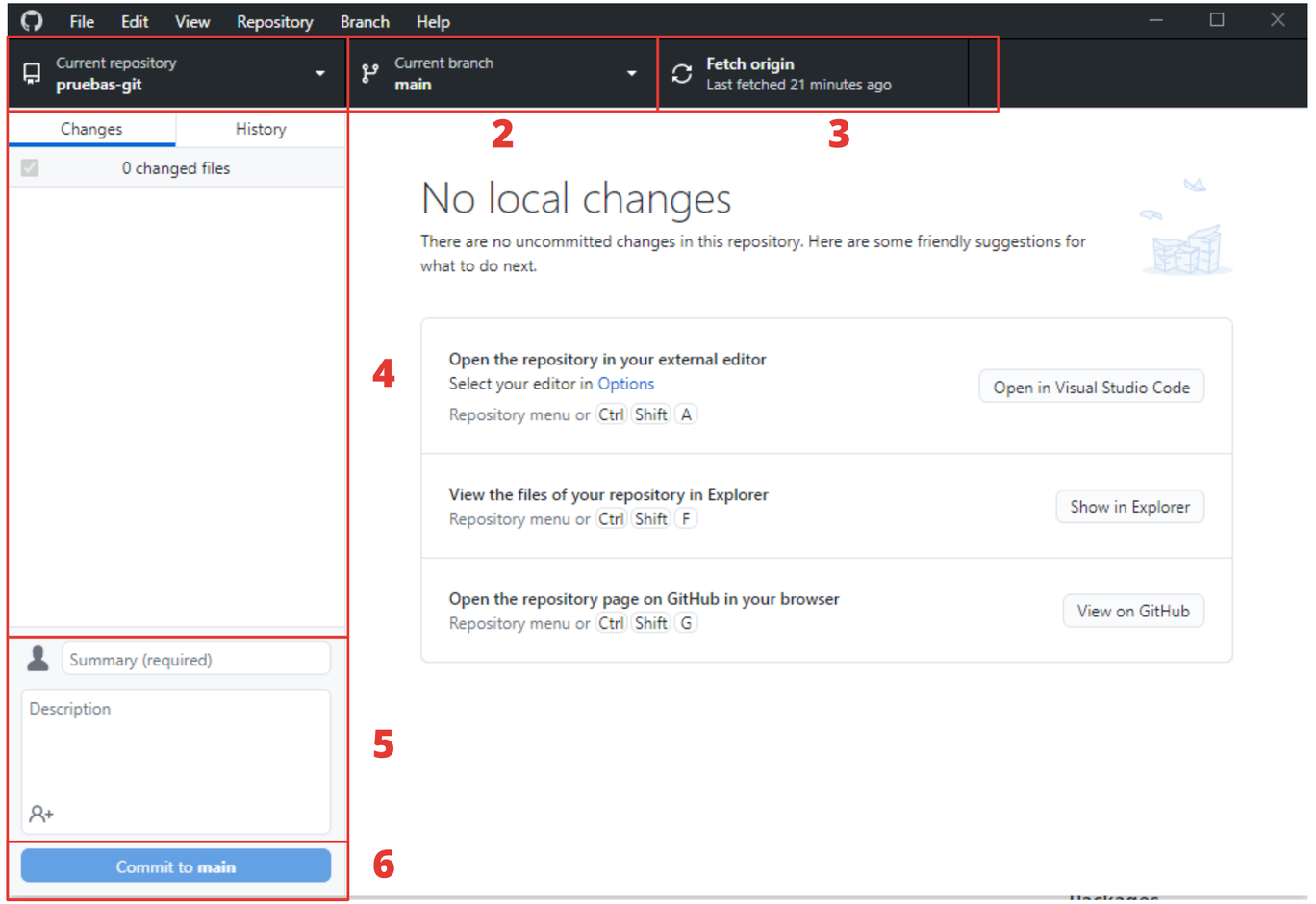
INCIDENTES / PROBLEMAS

Para el manejo de **incidentes y problemas**, se tomará en consideración la misma estrategia mencionada para los **proyectos**, con las siguientes diferencias:

1. Para el caso de los desarrollos que se consideran **mejoras o ajustes a aplicaciones existentes (Incidentes / Problemas)**, el proceso parte de una **etapa de priorización**, considerando lo siguiente:
   a) Si se tienen 2 o más desarrollos en paralelo, donde se estén realizando cambios en los mismos archivos, procesos o servicios, se debe entrar en una etapa de **priorización de cambios**. Se debe valorar qué cambio es más importante que sea certificado y posteriormente puesto en producción. Se opta por este proceso; ya que, de lo contrario, podría provocar que se envíen cambios a certificación y/o producción que no están finalizados.

b) Para los proyectos, **no será necesario** el envió de formato de solicitud de gestión de ramas cuando se requiera realizar la **integración de cambios de la rama del desarrollador a la rama _develop**. Para el caso de **Incidentes / Problemas,** el envío de esta solicitud es **obligatoria**.
c) Para los **Incidentes / Problemas**, los accesos a los repositorios serán **temporales**. Una vez se finalice con la mejora o ajuste, se removerá al usuario del repositorio.

GITHUB DESKTOP



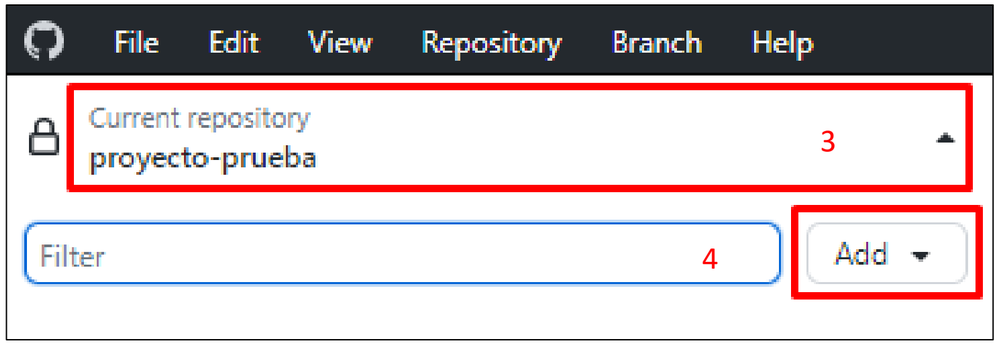
Este es nuestro GitHub Desktop, su pantalla inicial se compone de **6 ítems principales**, los cuales se detallarán en el siguiente paso.

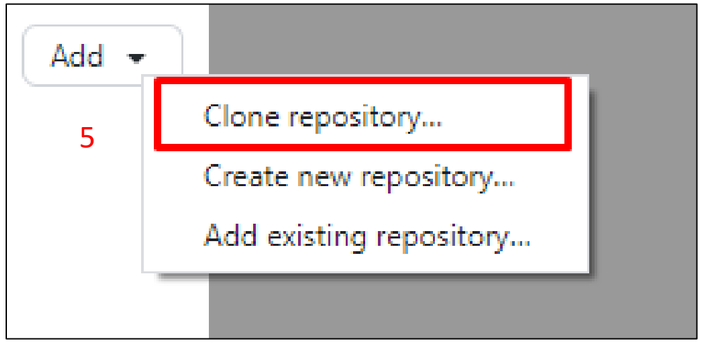
1. Repositorio al que se apunta.
2. Rama actual del repositorio donde se trabaja.
3. Botón de acción **Fetch** (Actualizar), **Push** (Enviar) o **Pull** (Recibir).
4. Lista de cambios realizados para la creación del **commit.**
5. Área de descripción o comentario para subida de **commit**.
6. Botón de creación de **commit**.

¿CÓMO CLONAR UN REPOSITORIO?

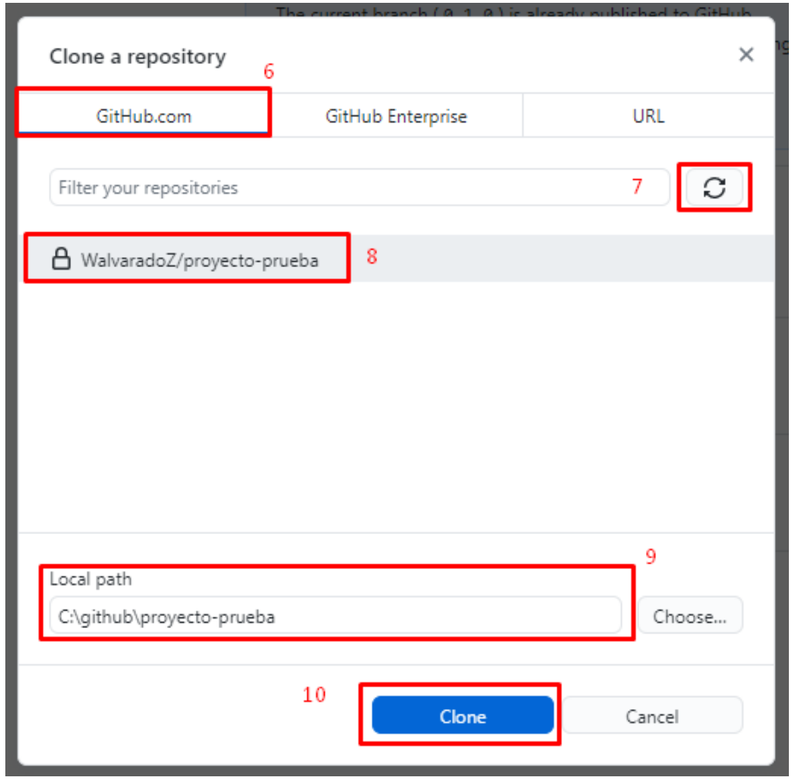
A continuación, se brindan los pasos para **clonar un repositorio a nivel local** desde la herramienta **GitHub Desktop**:

1. Abrir la aplicación **GitHub Desktop**.
2. Iniciar sesión con la **cuenta ligada a tu correo institucional** (si es primera vez).
3. Clic sobre la opción **Current repository** (Repositorio actual).
4. Clic sobre la opción **Add** (Agregar).
5. Clic sobre la opción **Clone repository** (Clonar repositorio).





Se mostrará el siguiente formulario **Clone a repository** (Clonar un repositorio), en el cual se deberán seguir los siguientes pasos:



6. Ubicarse en la pestaña de **Github.com**.
7. Clic en el botón **Refresh the list of repositories** (actualizar lista de repositorios).
8. Seleccionar el repositorio a clonar.
9. Seleccionar el **Local path** (ruta local). Para este ejemplo la ruta será **C:\github\proyecto-prueba**
10. Clic sobre la opción **Clone** (Clonar).

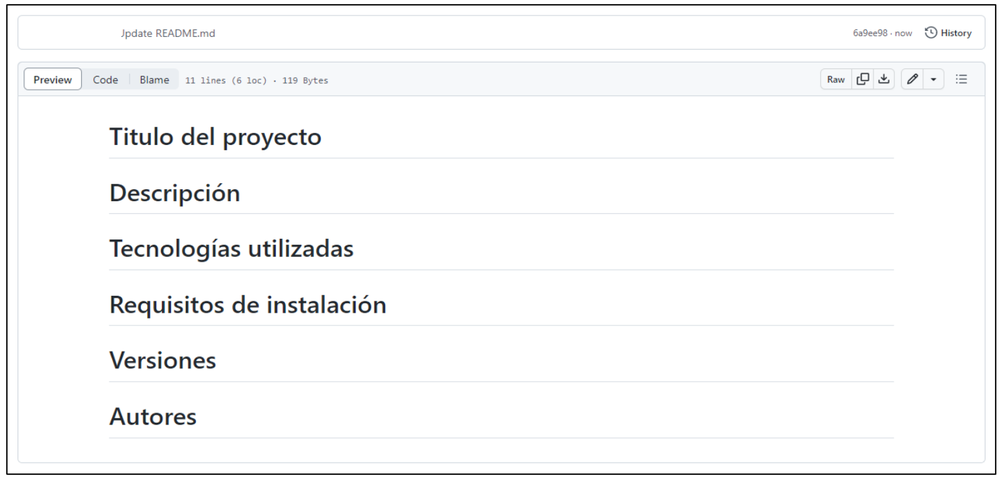
IMPORTANTE

- Todos los repositorios de GitHub se deberán clonar en la ruta **C:\github**
- Si son repositorios relacionados a un mismo proyecto, se debe utilizar la siguiente regla **C:\github\nombre-proyecto** y dentro de esa carpeta colocar todos los relacionados.

README

El archivo **README** es muy importante, ya que contiene información sobre nuestro proyecto, como ser:

1. Titulo del proyecto.
2. Descripción.
3. Tecnologías utilizadas.
4. Requisitos de instalación.
5. Versiones.
6. Autores.



IMPORTANTE: el archivo **README** debe ser actualizado con dicha información por parte del personal de desarrollo.

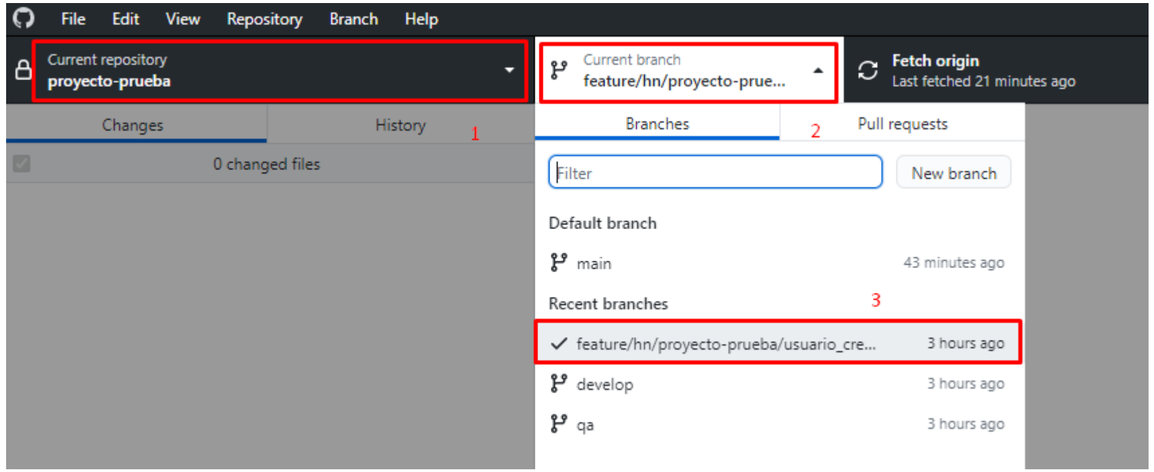
¿CÓMO REALIZAR UN COMMIT?

A continuación, te mostraremos como realizar un commit o integración de cambios a tu rama desde la aplicación **GitHub Desktop.**

1. Dentro de la aplicación **GitHub Desktop,** en la opción de **Current repository** (repositorio actual), verificar que estamos en el repositorio correcto **(ejemplo: proyecto-prueba).**

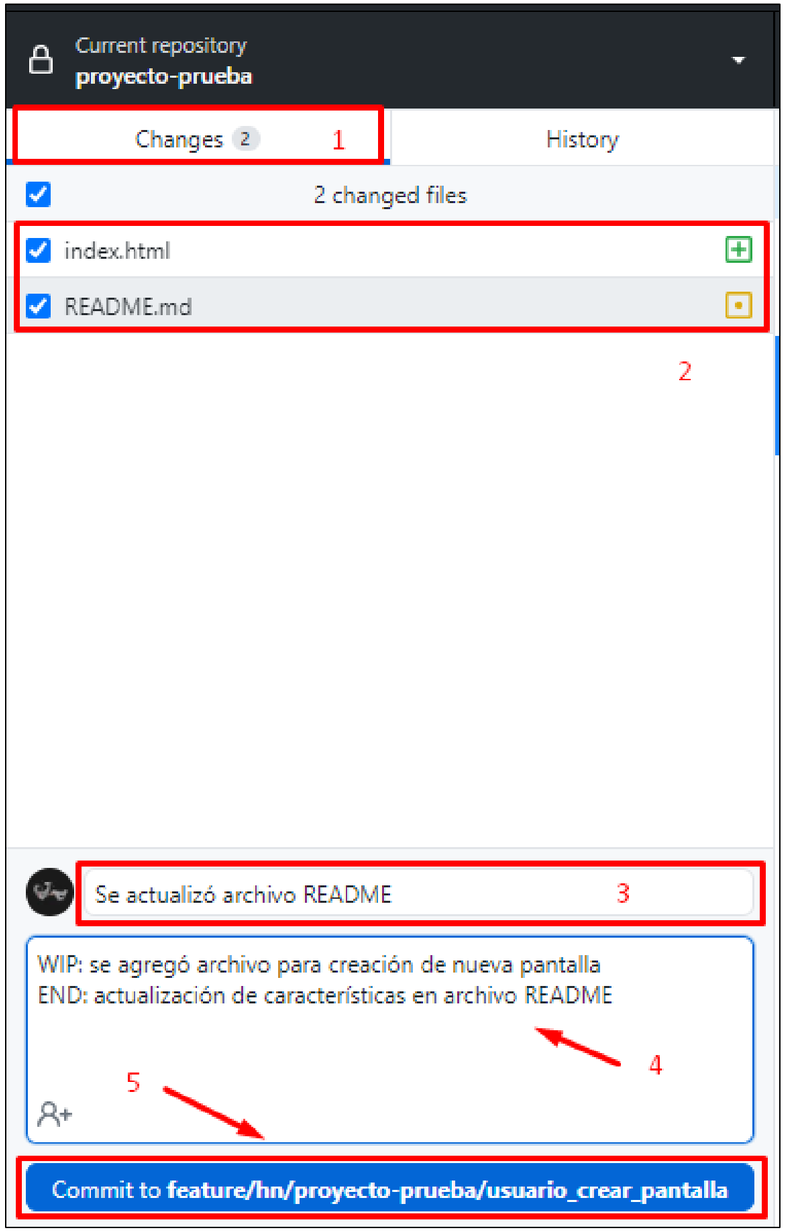
2. En la opción de **Current branch** (rama actual), verificar que estamos en el rama correcta **(ejemplo: feature/hn/proyecto/prueba/usuario_crear_pantalla).**

3. En el listado de las ramas actuales, aparecerá la rama donde estamos ubicados **(icono check al lado izquierdo del nombre).**

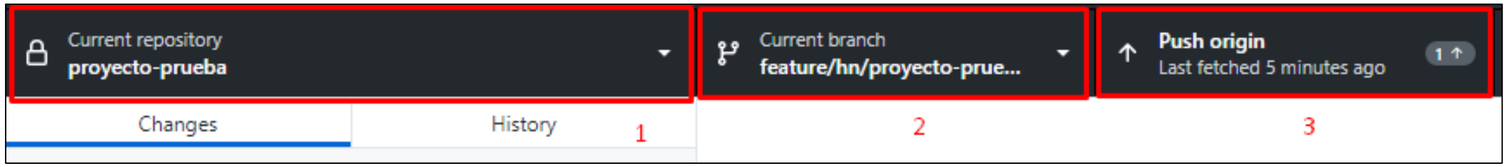


IMPORTANTE: esto es importante que lo verifiques antes de realizar cualquier integración de nuevos cambios.

1. **Changes** (cambios): se encuentran todos los cambios realizados a nivel de la rama.
2. **Changed files** (archivos modificados): en la parte inferior, se pueden visualizar los archivos **actualizados, agregados o eliminados** a nivel de la rama.
3. **Commit title** (titulo del commit): en los **commits,** es necesario agregar un título para poder identificar el cambio.
4. **Commit description** (descripción del commit): para la descripción de los **commits**, utilizaremos **2 acrónimos**:
   a) **WIP** (Work in progress): son los cambios que están siendo trabajados y que **NO han sido finalizados**.
   b) **END** (Ending): son los **cambios finalizados** que están listos para ser probados o certificados.
5. **Commit to [nombre rama]:** enviar los cambios a la rama indicada**.**



Para enviar los cambios de mi **repositorio local** al **repositorio remoto**, debemos validar lo siguiente:



1. **Current repository** (repositorio actual): que el repositorio seleccionado sea el correcto **(ejemplo: proyecto-prueba).**
2. **Current branch** (rama actual): que la rama seleccionada sea la correcta **(ejemplo: feature/hn/proyecto/prueba/usuario_crear_pantalla).**
3. **Push origin** (enviar a origen): si todo está correcto, podemos enviar los cambios al repositorio remoto.

Muchas gracias por haber prestado algo de tu tiempo.

¡Ahora la práctica hace al maestro!

Continúa aprendiendo más sobre GitHub.