



Argentina
programa

Desarrollo de aplicaciones JAVA

Guía I

“Sistemas de Control
de Versiones”

QUE ES GIT?

Git es un sistema de control de versiones de código abierto que permite gestionar cambios en el código fuente y otros archivos de un proyecto. Fue desarrollado por Linus Torvalds en el año 2005 y se ha convertido en una de las herramientas más populares entre los desarrolladores de software.

Git funciona mediante la creación de una "snapshot" o instantánea de los archivos de un proyecto en un momento determinado. Estas snapshots se almacenan en un repositorio de Git, y cada vez que se realizan cambios en los archivos, se crea una nueva snapshot que se agrega al repositorio.

Además, Git permite trabajar en equipo en un proyecto, ya que los desarrolladores pueden colaborar en el mismo repositorio y gestionar los cambios de forma organizada y sin conflictos. También facilita el trabajo en diferentes ramas del proyecto, lo que permite experimentar sin afectar la rama principal y facilita la integración de los cambios.

En resumen, Git es una herramienta de control de versiones esencial para cualquier desarrollador de software que desee gestionar los cambios de forma efectiva, colaborar con otros desarrolladores y mantener un historial detallado de los cambios realizados en un proyecto.

Existen otros “Sistemas de Control de Versiones” aparte de GIT?

Sí, existen varios sistemas de control de versiones aparte de Git, algunos de los más populares son:

- Subversion (SVN): es un sistema de control de versiones centralizado, en el cual todos los archivos se almacenan en un repositorio central. A diferencia de Git, SVN no es distribuido, lo que significa que cada usuario trabaja directamente con el repositorio central.
- Mercurial (Hg): es un sistema de control de versiones distribuido similar a Git, que permite trabajar en ramas de manera muy sencilla. Mercurial también es muy popular en el mundo del desarrollo de software.
- Perforce (P4): es un sistema de control de versiones centralizado utilizado en entornos empresariales, especialmente para proyectos grandes y complejos.
- CVS (Concurrent Versions System): es un sistema de control de versiones centralizado y uno de los más antiguos. Aunque ha sido superado por sistemas más modernos como Git y SVN, todavía se utiliza en algunos proyectos de código abierto.

Cada sistema de control de versiones tiene sus propias características y ventajas, y la elección de uno u otro dependerá de las necesidades específicas de cada proyecto.

QUE ES GITHUB?

GitHub es una plataforma web para alojar y compartir código fuente de proyectos de software utilizando el sistema de control de versiones Git. Fundada en 2008, GitHub es una de las plataformas más populares para alojar y colaborar en proyectos de software de código abierto y privados.

Además de ser una plataforma de alojamiento de código, GitHub también ofrece herramientas para la colaboración y gestión de proyectos. Algunas de las características de GitHub incluyen:

- **Control de versiones:** GitHub utiliza Git para el control de versiones de los proyectos alojados en su plataforma, lo que permite a los desarrolladores rastrear y colaborar en los cambios realizados en el código fuente.
- **Comunicación y colaboración:** GitHub permite a los desarrolladores colaborar en proyectos utilizando herramientas como pull requests, issues y comentarios en línea.
- **Integración con otras herramientas:** GitHub se integra con una amplia variedad de herramientas de desarrollo, como editores de código, sistemas de integración continua, herramientas de pruebas y más.
- **Comunidad:** GitHub cuenta con una gran comunidad de desarrolladores y proyectos de código abierto. Los desarrolladores pueden contribuir a proyectos de código abierto, aprender de otros desarrolladores y compartir su propio trabajo.

En resumen, GitHub es una plataforma esencial para la colaboración y el desarrollo de software de código abierto y privado. Proporciona una forma fácil y efectiva para alojar, gestionar y colaborar en proyectos de software, lo que facilita el trabajo en equipo y la creación de software de alta calidad.

QUE ES GITHUB DESKTOP?

GitHub Desktop es una aplicación de escritorio que facilita el uso de Git y el control de versiones en GitHub. Es una herramienta gratuita que permite a los desarrolladores manejar sus repositorios

de GitHub directamente desde su computadora de escritorio, sin necesidad de usar la línea de comandos.

Entre las funciones de GitHub Desktop se encuentran:

- Crear y clonar repositorios de GitHub.
- Ver los cambios realizados en los archivos y hacer seguimiento de los cambios.
- Hacer commits y push de los cambios al repositorio de GitHub.
- Crear y trabajar en ramas de manera sencilla.
- Resolver conflictos de manera intuitiva.
- Hacer pull y merge de cambios de otros colaboradores.
- Visualizar el historial de cambios del repositorio.

GitHub Desktop es una herramienta muy útil para aquellos que están empezando a usar Git y GitHub, así como para aquellos que prefieren una interfaz gráfica para gestionar sus repositorios en lugar de usar la línea de comandos. Además, permite trabajar de manera más eficiente y colaborativa en proyectos con otros desarrolladores.

Para utilizar GitHub Desktop, necesitarás los siguientes requisitos:

1. **Sistema operativo compatible:** GitHub Desktop es compatible con Windows 10 (64 bits), macOS 10.14 o posterior, y distribuciones de Linux basadas en Debian o Red Hat.
2. **Cuenta de GitHub:** Deberás tener una cuenta de GitHub para usar GitHub Desktop. Si aún no tienes una, puedes crear una cuenta gratuita en la página de registro de GitHub.
<https://github.com/>
3. **Git:** GitHub Desktop utiliza Git como su sistema de control de versiones. Si aún no tienes Git instalado en tu computadora, puedes descargarlo desde la página de descarga de Git.
<https://git-scm.com/downloads>

Una vez que tengas todos los requisitos previos, puedes descargar GitHub Desktop desde el sitio web oficial de GitHub Desktop. Luego, podrás instalar GitHub Desktop en tu computadora y comenzar a usarlo para administrar tus repositorios de GitHub.

<https://desktop.github.com/>

Algunos conceptos previos:

Para poder usar Git de manera efectiva, es importante comprender algunos conceptos clave, entre ellos:

- **Repositorio:** Un repositorio es un lugar donde se almacena el código fuente de un proyecto. En Git, un repositorio es una carpeta que contiene todos los archivos y subcarpetas del proyecto, así como el historial de cambios realizados.
- **Commit:** Un commit es un registro de los cambios realizados en el código fuente del proyecto en un momento determinado. Cada commit incluye una descripción de los cambios realizados y un identificador único.
- **Branch:** Un branch o rama es una copia del código fuente del proyecto que se ha separado del flujo principal del desarrollo. Los branches permiten experimentar con cambios sin afectar la rama principal del proyecto y facilitan la integración de cambios realizados por diferentes desarrolladores.
- **Merge:** Un merge es el proceso de combinar dos ramas diferentes de un proyecto en una sola. Este proceso permite integrar los cambios realizados en diferentes ramas y mantener una versión única y actualizada del código fuente.
- **Push:** El comando push se utiliza para enviar los cambios realizados en el repositorio local al repositorio remoto en GitHub.
- **Pull:** El comando pull se utiliza para descargar los cambios realizados en el repositorio remoto y fusionarlos con el repositorio local.
- **Conflictos:** Los conflictos son situaciones en las que Git no puede fusionar automáticamente dos versiones diferentes de un archivo. En estos casos, los desarrolladores deben resolver manualmente los conflictos para poder continuar con la fusión.

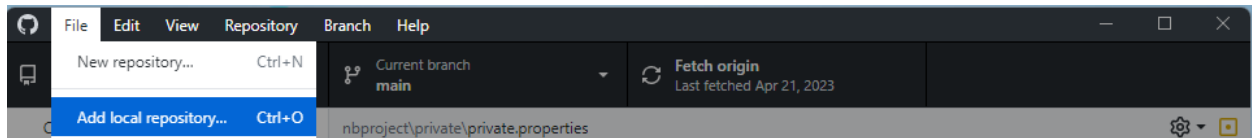
Estos son algunos de los conceptos más importantes que se deben conocer para usar Git de manera efectiva.

Creando un repositorio:

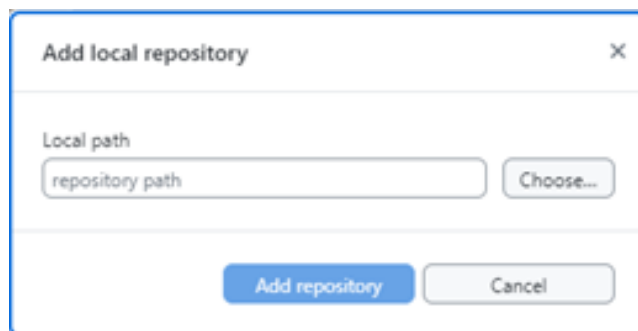
Una de las formas de trabajar dentro del equipo, quizás no la mejor manera, pero para dar los primeros pasos con esta herramienta será suficiente, ya que todos los colaboradores trabajarán sobre la rama principal (main); es la siguiente:

Primero: Uno de los miembros del equipo, creará en Netbeans un nuevo proyecto.

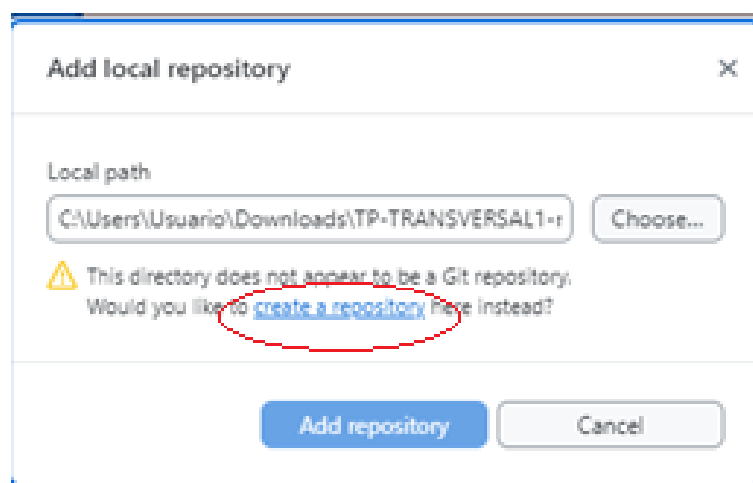
Segundo: Una vez creado el proyecto en netbeans, desde GitHub Desktop, irá a File → Add local repository...



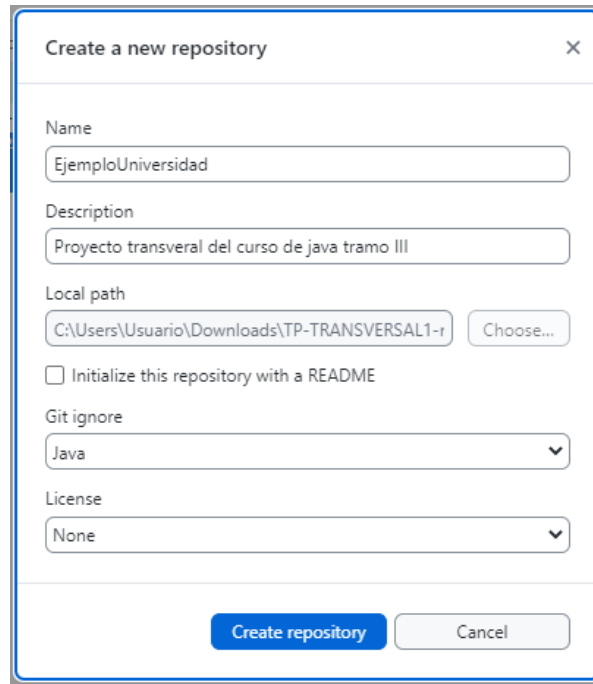
Desde el diálogo que aparecerá a continuación, elegimos la carpeta en donde se encuentra localmente nuestro proyecto Java, haciendo clic sobre el botón “Choose”.



Una vez seleccionado nuestro proyecto local, haremos clic sobre el link “create a repository”



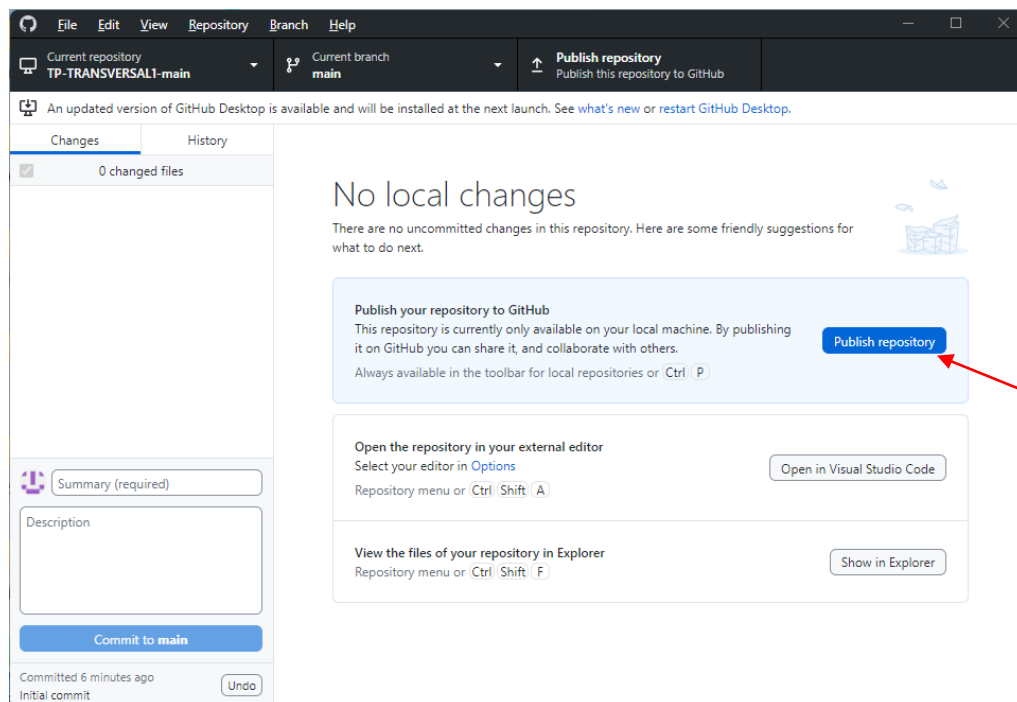
Y en la siguiente pantalla:



The screenshot shows a 'Create a new repository' dialog box. It has a title bar with a close button. The form contains the following fields and options:

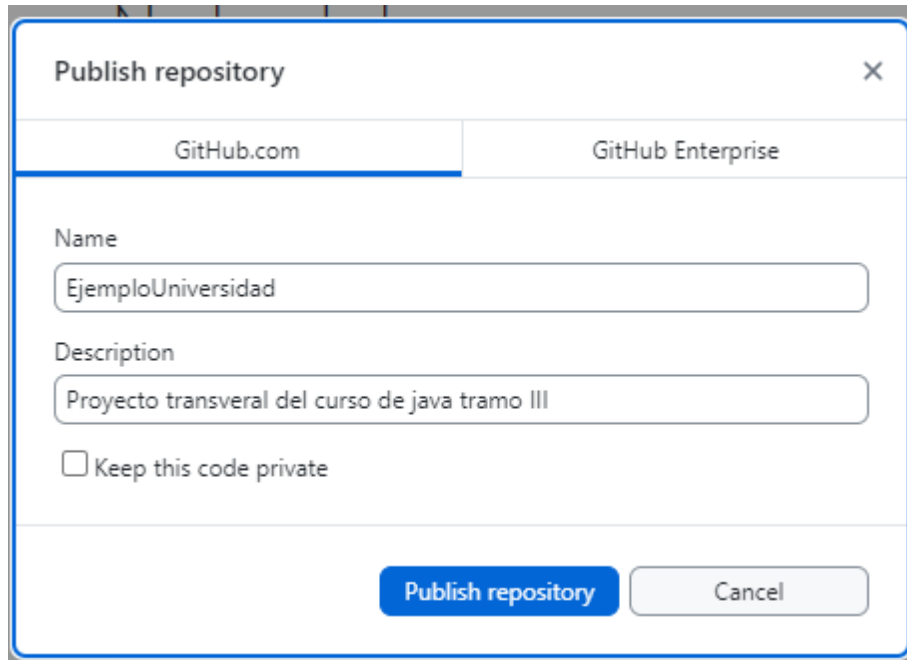
- Name:** EjemploUniversidad
- Description:** Proyecto transversal del curso de java tramo III
- Local path:** C:\Users\Usuario\Downloads\TP-TRANSVERSAL1-r (with a 'Choose...' button)
- ☐ Initialize this repository with a README
- Git ignore:** Java (dropdown menu)
- License:** None (dropdown menu)
- Buttons:** 'Create repository' (blue) and 'Cancel' (grey)

Completamos con el nombre que vamos a usar para identificar a nuestro repositorio, una breve descripción y muy importante en opción “Git ignore”¹ indicar que es un proyecto creado utilizando el lenguaje Java; al finalizar hacemos clic sobre el botón “Create repository”; nos volverá a la vista anterior que cerraremos; y ya estaremos listos para publicar nuestro repositorio en la web de GitHub, y para ello bastará con hacer clic sobre el botón “Publish repository”



¹ “.gitignore” Es un archivo especial utilizado en los repositorios de Git para especificar qué archivos o directorios deben ser ignorados y no deben ser rastreados por Git.”

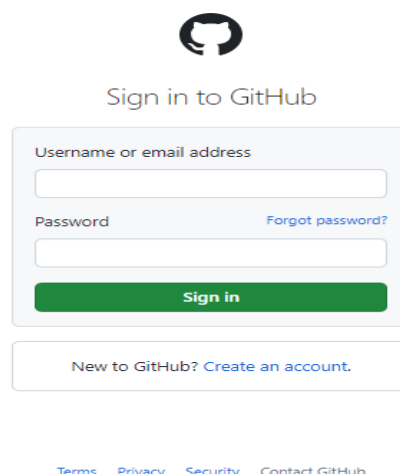
En la vista siguiente nos vamos a asegurar que en la web de GitHub el repositorio tenga el mismo nombre que le dimos localmente y decido si quiero que se publique como “public” o “private”. Si lo hacemos privado, solo podrá ser accedido por el propietario del repositorio y sus colaboradores; pero en este curso, los vamos a crear como públicos a nuestros repositorios, para que puedan ser accedidos por su tutor del curso sin necesidad de tener que agregarlo como colaborador, y a eso lo haremos **no seleccionando** la opción “Keep this code private” (Mantener este código privado); al finalizar haremos clic sobre el botón “Publish repository” de esta ventana activa.



The screenshot shows a 'Publish repository' dialog box with a close button (X) in the top right corner. It has two tabs: 'GitHub.com' (selected) and 'GitHub Enterprise'. Below the tabs, there are two text input fields: 'Name' with the value 'EjemploUniversidad' and 'Description' with the value 'Proyecto transversal del curso de java tramo III'. Below these fields is a checkbox labeled 'Keep this code private' which is unchecked. At the bottom right, there are two buttons: 'Publish repository' (highlighted in blue) and 'Cancel'.

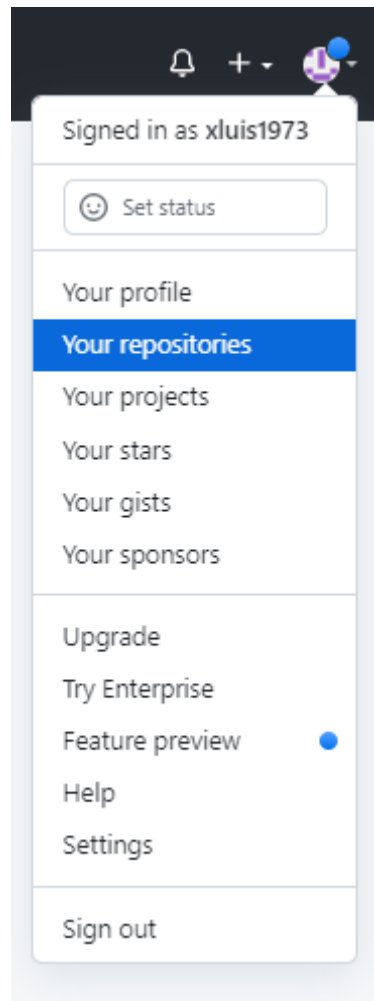
AGREGANDO COLABORADORES A MI REPOSITORIO

Para que todos los miembros de mi equipo de trabajo, no sólo puedan acceder a mi repositorio, sino también poder hacer cambios sobre los archivos que se encuentran en él, los debo agregar como colaboradores del mismo. Para ello, debo acceder a la web de [GitHub](https://github.com) y por supuesto ingresar como mi usuario y contraseña.

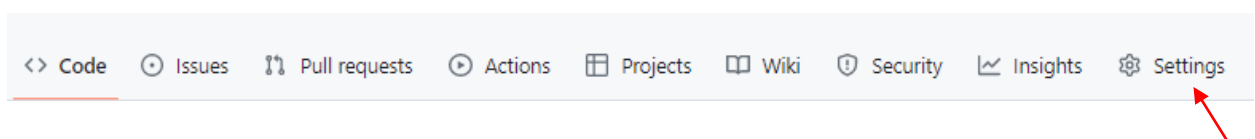


The screenshot shows the GitHub sign-in page. At the top is the GitHub logo. Below it is the text 'Sign in to GitHub'. There is a form with two input fields: 'Username or email address' and 'Password'. To the right of the password field is a link that says 'Forgot password?'. Below the input fields is a green 'Sign in' button. At the bottom of the form is a link that says 'New to GitHub? Create an account.'. At the very bottom of the page, there are links for 'Terms', 'Privacy', 'Security', and 'Contact GitHub'.

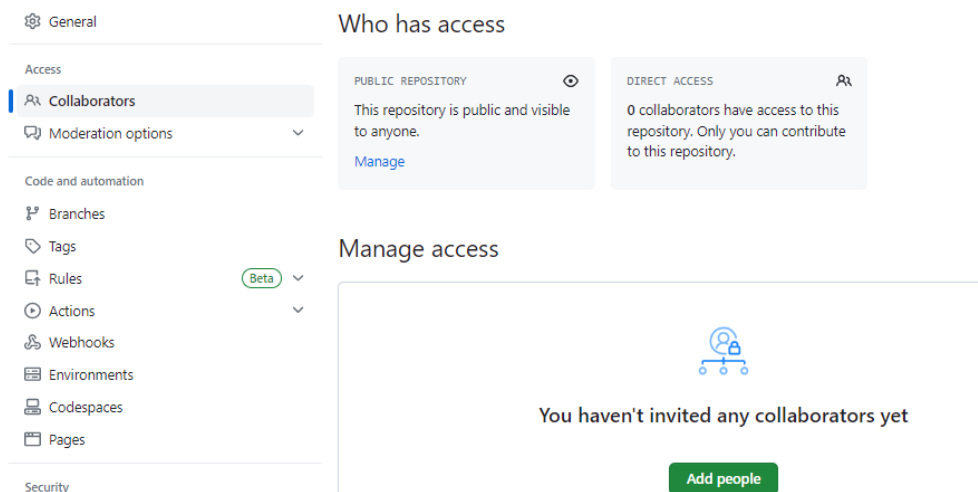
Una vez en el sitio web, navegamos a nuestros repositorios.



Accedemos al repositorio que queremos compartir y hacemos clic sobre el botón “Setting” de la barra de herramientas.



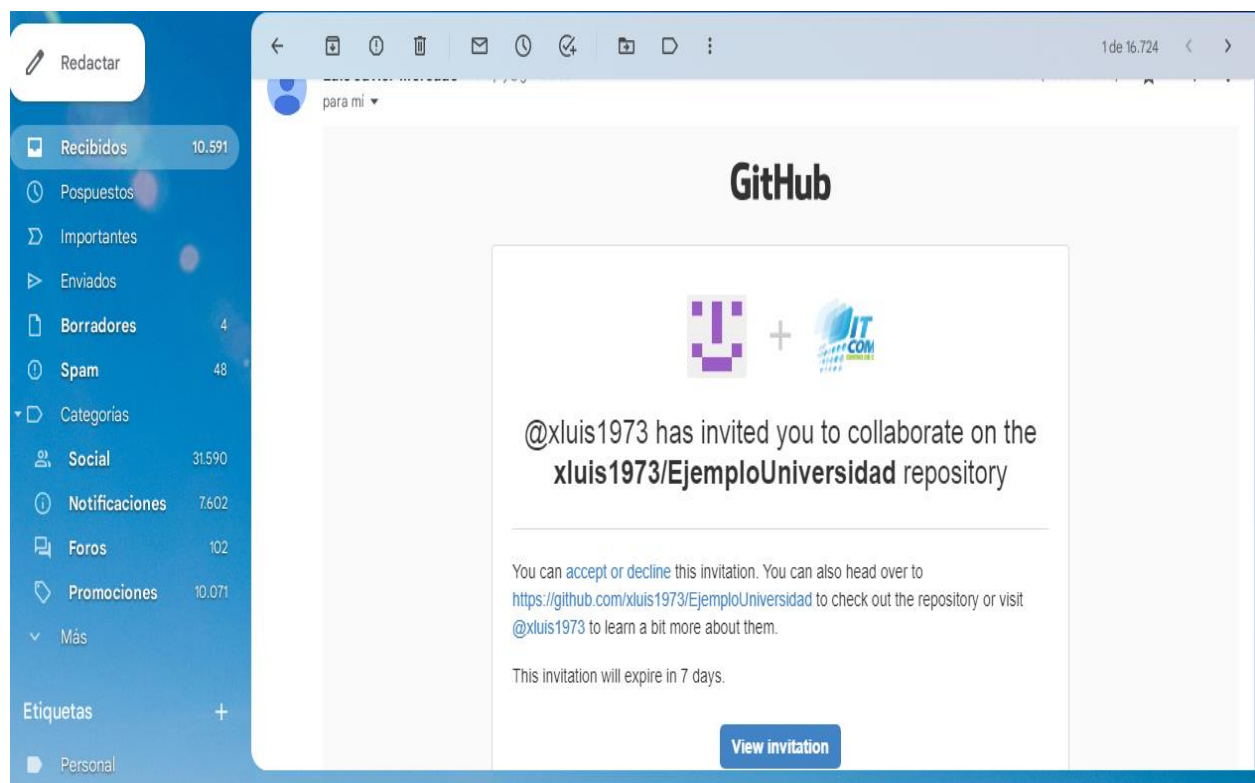
Una vez allí, seleccionaremos la opción “Collaborators” y luego haciendo clic en el botón “Add people” iremos incorporando de a uno a nuestros colaboradores colocando el nombre de usuario de ellos o el mail con el que están registrados en GitHub.



Nuestros colaboradores recibirán un mail para que confirmen la invitación para colaborar en nuestro proyecto, que deberán aceptar.

COLABORADORES CLONAN REPOSITORIO REMOTO EN SUS ORDENADORES.

La responsabilidad de los colaboradores es, una vez recibido el mail con la invitación del creador del repositorio, deben aceptarla.





xluis1973 invited you to collaborate

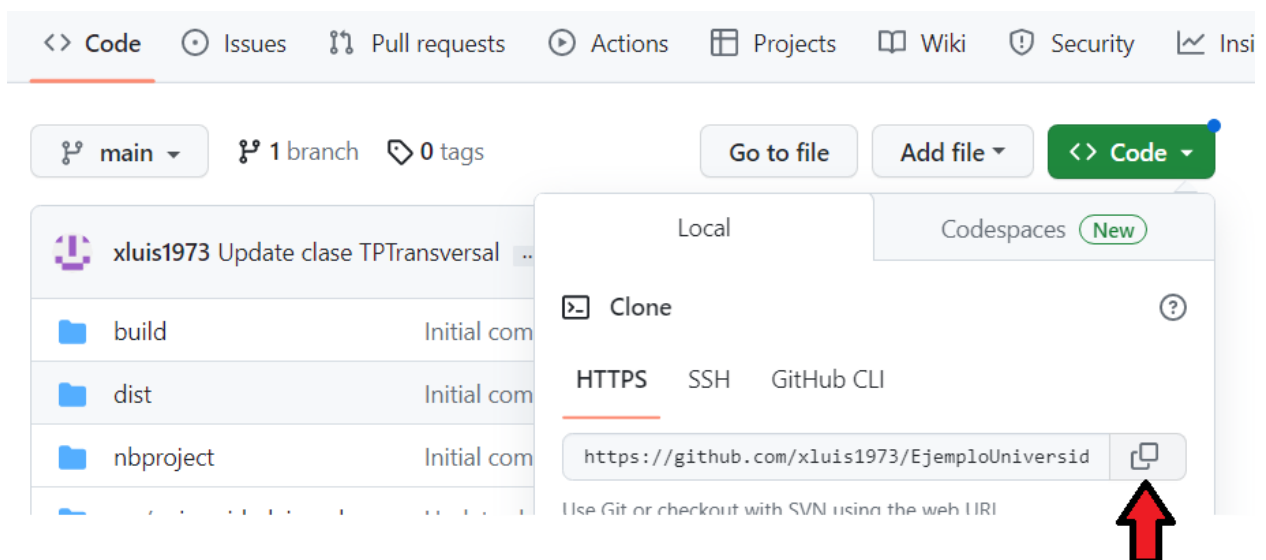
Accept invitation

Decline

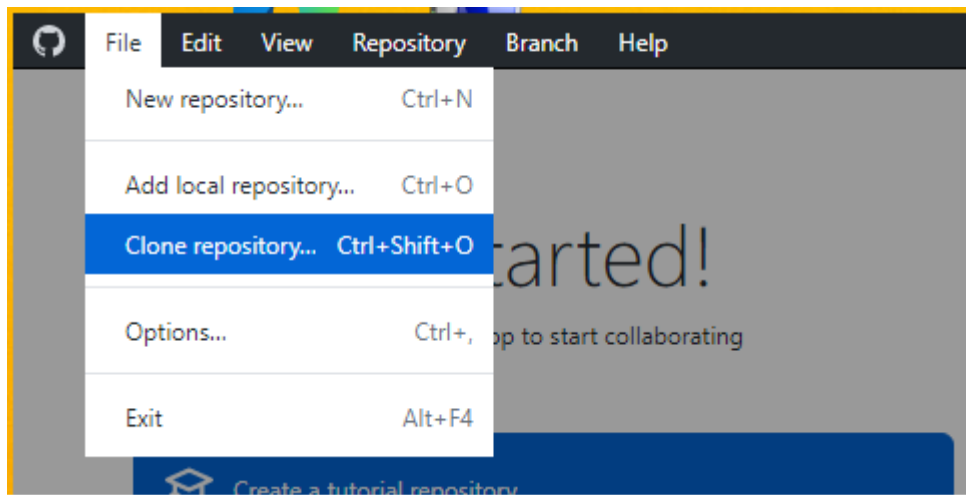
🔒 Owners of EjemploUniversidad will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

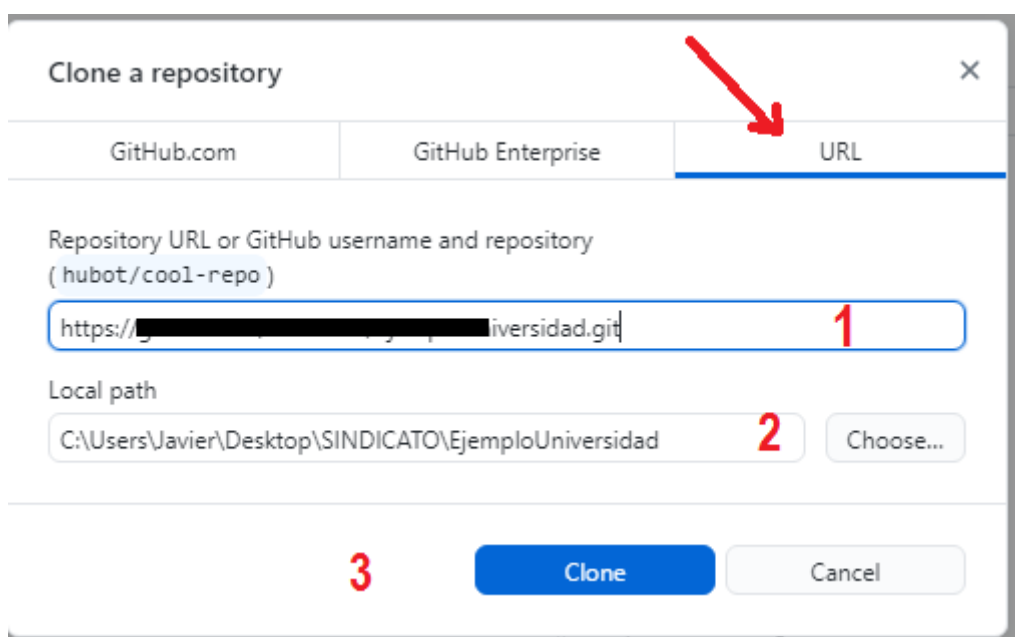
Una vez aceptada la invitación, tendrán no sólo acceso al repositorio, sino que también podrán hacer cambios en el mismo; y para ello, deben, desde el repositorio remoto (el que está alojado en la web de GitHub), copiar el link.



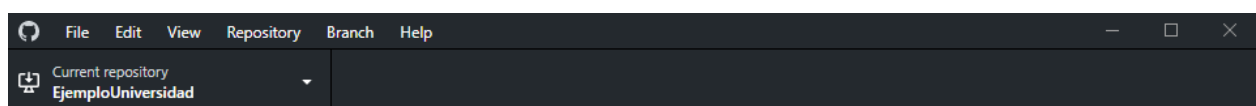
Una vez hecho esto, deben abrir la aplicación que previamente han instalado en su ordenador, GitHub Desktop, y procederán con el clonado, yendo al menú File y elegir la opción “Clone repository”



Desde la ventana que aparecerá a continuación deberán seleccionar la pestaña “URL”, para hacer lo siguiente:



1. Pego el link del repositorio remoto.
2. Puedo elegir en donde quiero que quede la copia en mi ordenador.
3. Hago clic en el botón “Clone” y comenzará con la descarga a mi ordenador.



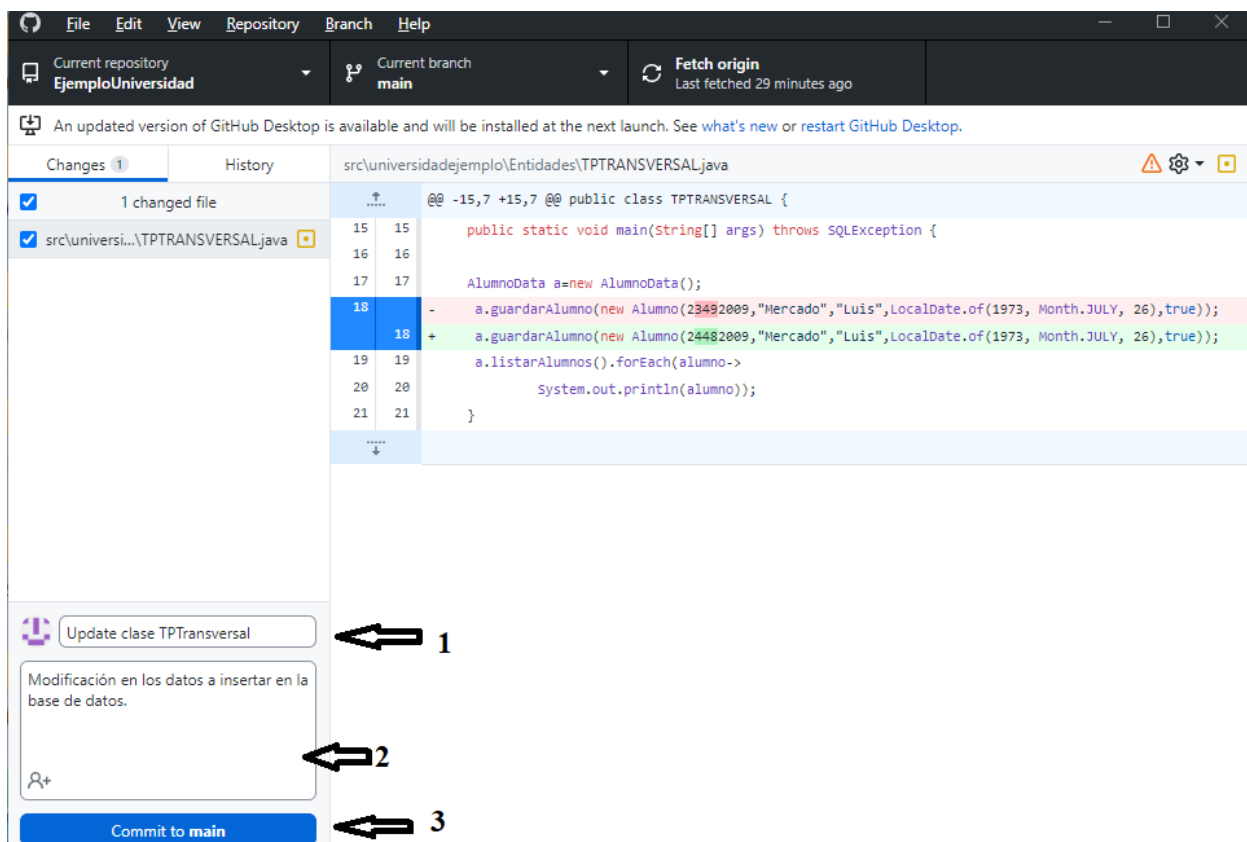
 Cloning EjemploUniversidad

Resolving deltas: 100% (15/15), done.

Una vez hecho esto último, tendrás en tu ordenador una réplica exacta de lo que está alojado en el repositorio remoto (web de GitHub).

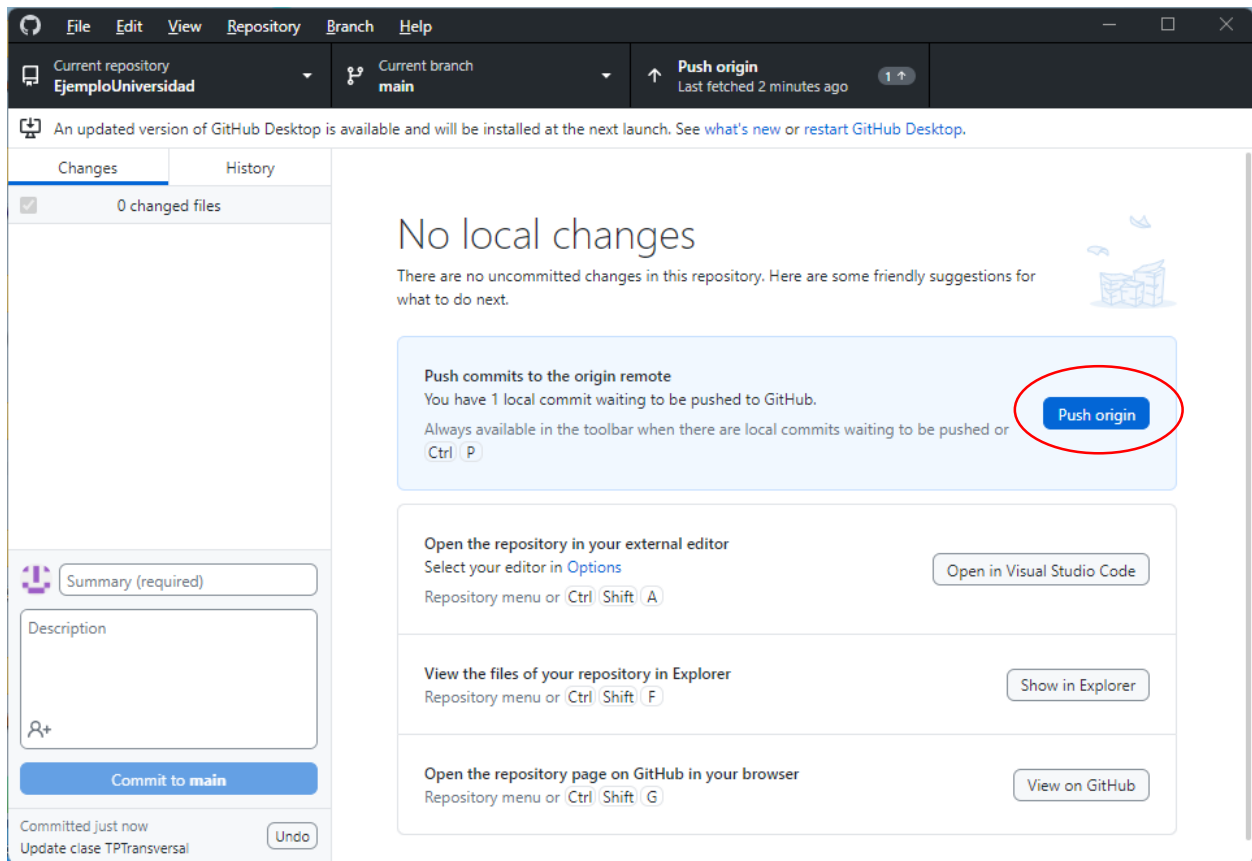
SUBIENDO CAMBIOS AL REPOSITORIO

Como habíamos aclarado desde el comienzo, que a todos los cambios los vamos a subir sobre la rama principal de nuestro repositorio remoto, es conveniente que cada colaborador trabaje sobre archivos diferentes en sus repositorios locales. Por lo tanto, cuando uno de los integrantes del equipo necesita subir un cambio hecho sobre alguno de los archivos de su repositorio local (PC) al repositorio remoto (el alojado en la web), la acción a realizar será hacer un COMMIT y PUSH. A estas operaciones obviamente las realizaremos desde GitHub Desktop.



- 1- Le colocamos un título al commit para poderlo identificar.
- 2- Podemos escribir una breve descripción acerca del cambio realizado.
- 3- Pulsamos el botón “Commit to main”

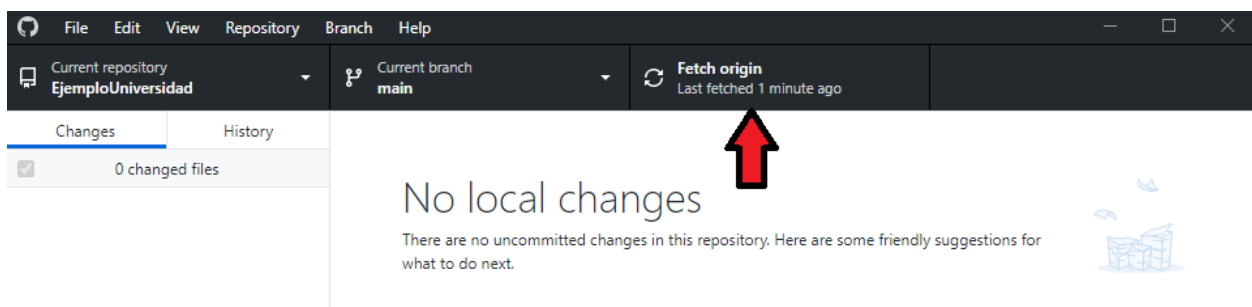
Y Luego, para que este cambio se refleje finalmente en el repositorio remoto, haremos clic en el botón “Push origin”



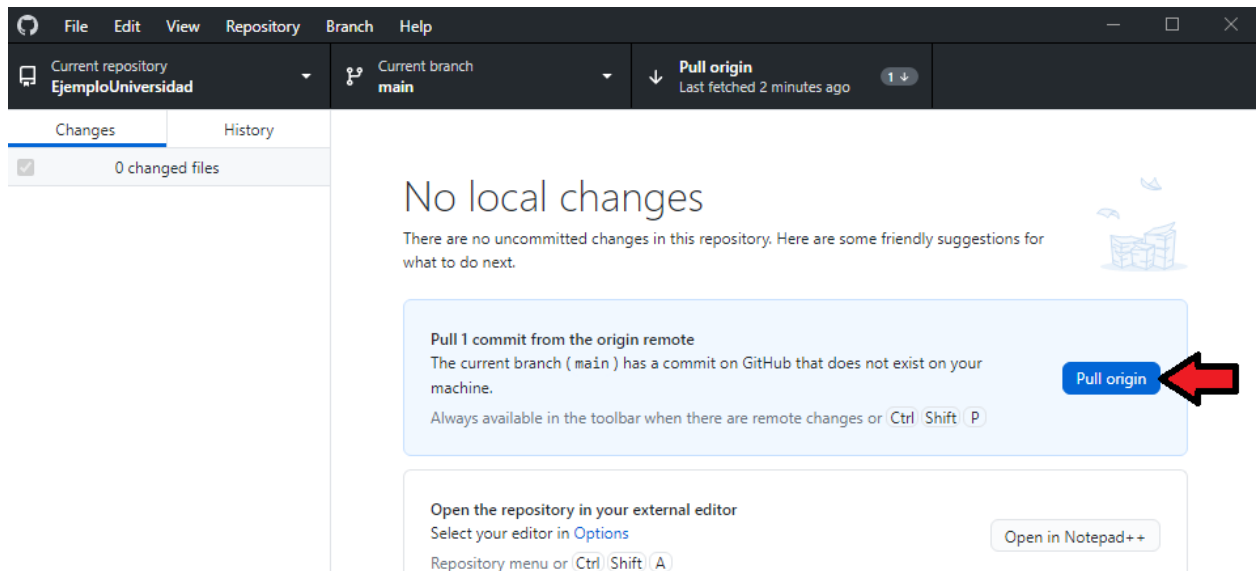
Una vez subido el cambio al repositorio remoto, los demás miembros del equipo, deberán bajar esa actualización a sus repositorios locales.

BAJAR CAMBIOS A LOS REPOSITORIOS LOCALES.

Consiste básicamente en descargar desde el repositorio remoto (repositorio alojado en la web de GitHub) los cambios hechos por algún miembro del equipo a nuestro ordenador (repositorio local). Para ello, la acción a realizar será un PULL. Puede suceder que si tienes abierto GitHub Desktop, no te informe que hay cambios hechos en el repositorio remoto, entonces procederemos a realizar una operación FETCH que forzará una lectura sobre el repositorio remoto en busca de cambios.



Una vez detectados los cambios en el repositorio remoto, podremos bajarlos a nuestro ordenador utilizando el comando PULL.



Ahora, si abres desde Netbeans el proyecto, verás que está actualizado con los cambios que se encuentran en la web (repositorio remoto).

Para más información sobre el uso de **GitHub**, puede acceder a la [documentación](#).

Para más información sobre el uso de **GitHub Desktop** puede acceder a la [documentación](#).