

Sistemas de control de versións – Git

Introdución a Git

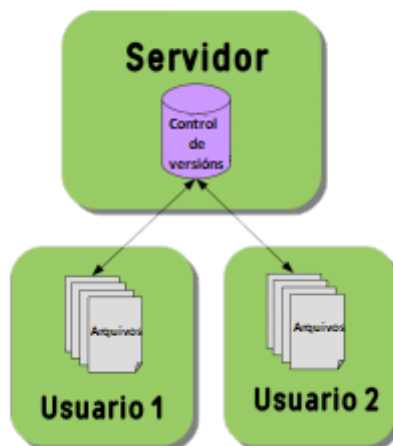
Un sistema de control de versións permite levar un **rexistro** das modificacións realizadas sobre un conxunto de arquivos e directorios en canto a creación, modificación ou eliminación.

O obxectivo do rexistro é poder reverter en calquera momento as modificacións realizadas tempo atrás, comparalas coas versión actual ou incluso manter varias versións do proxecto simultaneamente.

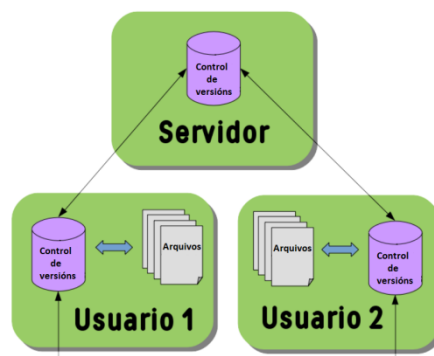
Ademais, é posible clonar un proxecto completo noutra localización ou sincronizar o repositorio local con outro remoto mediante ferramentas como **GitHub** ou **GitLab**.

Git é un sistema de control de versións **distribuído**. Nos anos 90 a maioría de sistemas de control de versións eran sistemas **centralizados** nos que un servidor central garda toda a información e os clientes conéctanse ao servidor.

Modelo centralizado



O modelo distribuído é o máis utilizado actualmente. Neste caso, cada usuario ten un control de versións propio que á súa vez é manexado polo servidor. Desta maneira, como cada usuario ten a súa copia completa do repositorio, pode traballar sen estar conectado ao servidor.



Un **repositorio** contén todos os arquivos dun proxecto xunto co historial de revisións de cada un deles.

Git permite comparar diferentes versións dun arquivo, é dicir, pode saberse o que se cambiou e quen o cambiou en calquera momento.

Se nalgún momento durante a codificación temos un erro fatal e non sabemos a causa, sempre podemos volver a un estado estable anterior.

Git tamén axuda a **sincronizar** código entre varias persoas.

Case o 50% do código de Git está escrito en **linguaxe C**.

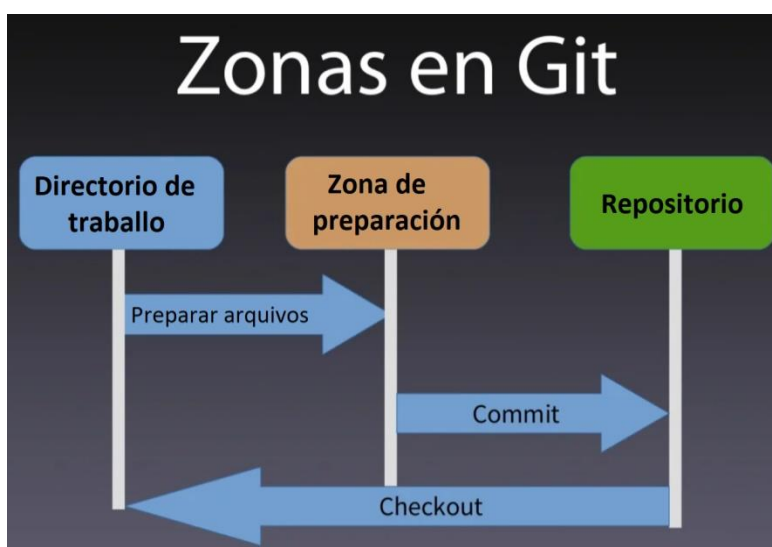
Git deseñouse tendo en conta o **rendemento**, a **seguridade** e a **flexibilidade**.

Hoxe en día, Git é, con diferenza, o sistema de control de versións moderno máis utilizado do mundo. Git é un proxecto de **código aberto** maduro e cun mantemento activo que desenvolveu orixinalmente en 2005 **Linus Torvalds**, o creador do kernel do sistema operativo Linux. Esta ferramenta foi inicialmente desenvolvida para que varios desenvolvedores puideran traballar simultaneamente no núcleo de Linux.

Fundamentos de Git

O código dun **repositorio local** Git pasa por **tres zonas** diferentes:

1. Directorio de traballo
2. Área de preparación
3. Repositorio Git



O **directorio de traballo** é onde podemos facer calquera cambio sen afectar ao repositorio en absoluto. En canto modificamos algo nun código, este estará no estado **modificado**.

Unha vez temos feito os cambios necesarios no código, pasamos os arquivos á **área de preparación** (tecnicamente chamada "index" en linguaxe Git).

A **zona de preparación** é un arquivo ubicado no directorio `.git` que garda información sobre arquivos que van pasar ao directorio `git`. Ao pasar os arquivos do directorio de traballo á zona de preparación, o código cambia de estado de modificado a **preparado**.

Se confirmamos os cambios, os arquivos pasan da área de preparación ao **directorio git**. O código cambia de estado preparado a **confirmado**.

O **HEAD** refírese ao **commit** no que está un repositorio en cada momento. Por regra xeral, HEAD soe coincidir co último commit da rama na que se estea traballando, xa que normalmente estamos traballando no último. Pero se nos movemos cara calquera outro commit anterior, entón o HEAD estará máis atrás.

Para **crear un novo repositorio** usamos o comando **git init**. `git init` é un comando que se utiliza só durante a configuración inicial dun novo repositorio. Ao executar este comando, creárase un novo subdirectorio `.git` no directorio de traballo actual. `git init` pode usarse para converter un proxecto existente e sen versión nun repositorio de Git ou para inicializar un novo repositorio baleiro. Ao executar este comando tamén se crea unha nova rama principal.

Tras inicializar un repositorio, pode confirmarse a versión dos arquivos.

Existen moitas formas de usar Git. Por un lado temos as ferramentas orixinais de **liña de comandos** e, por outro, temos unha grande variedade de interfaces de usuario con distintas capacidades.

A continuación, imos comentar algúns dos **comandos** máis **básicos** de **Git** para traballar en **local** que todo desenvolvedor de software debe coñecer.

O comando **git config** pode usarse para establecer unha configuración específica de usuario, como o correo electrónico, nome de usuario, etc.

O comando **git status** mostra o estado do directorio de traballo e da área de preparación.

O comando **git add** é un comando que se usa para agregar un arquivo que está no directorio de traballo á área de preparación.

O comando **git reset** serve para sacar arquivos da área de preparación ou do repositorio local.

O comando **git commit** é un comando que se usa para agregar ao repositorio todos os arquivos que están na **área de preparación**.

O comando **git diff** serve para observar as diferenzas entre orixes de datos de Git.

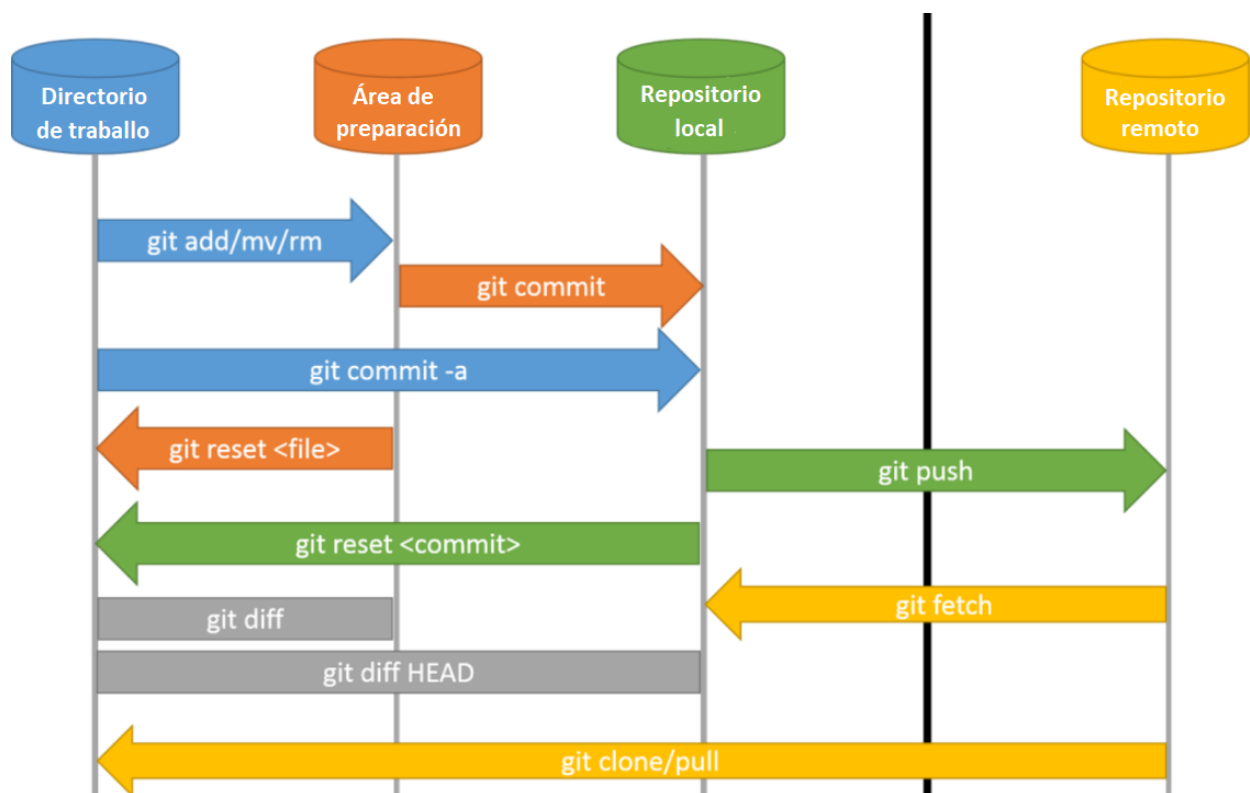
O comando **git branch** serve para crear ramas a partir de onde estamos e co estado actual pero non nos leva a elas.

O comando **git checkout** lévanos á rama indicada.

O comando **git merge** fusiona os cambios dunha rama á rama na que nos atopamos.

Para poder colaborar en calquera proxecto Git, necesitamos saber como xestionar **repositorios remotos**. Os repositorios remotos son versións dun proxectos que están aloxadas en Internet ou en calquera outra rede. Podemos ter varios repositorios remotos e, en cada un deles teremos uns permisos determinados (só lectura ou lectura e escritura). Colaborar con outras persoas implica xestionar estes repositorios remotos enviando e traendo datos deles cada vez que necesitamos compartir o noso traballo. Xestionar un repositorio remoto inclúe sabe como engadir un repositorio remoto, eliminar os remotos que xa non son válidos, xestionar varias ramas remotas, etc.

Na seguinte imaxe amósanse os **comandos básicos** para traballar con **Git** tanto en local como en remoto.



O comando **git clone** serve para descargar un proxecto dun repositorio remoto para traballar en local.

O comando **git fetch** descarga o historial do repositorio remoto.

O comando **git pull** descarga o historial do repositorio remoto e incorpora os cambios.

O comando **git push** sube as confirmacións (commits) que haxa na rama local á rama co mesmo nome do repositorio remoto.

Utilización de Git en Apache NetBeans

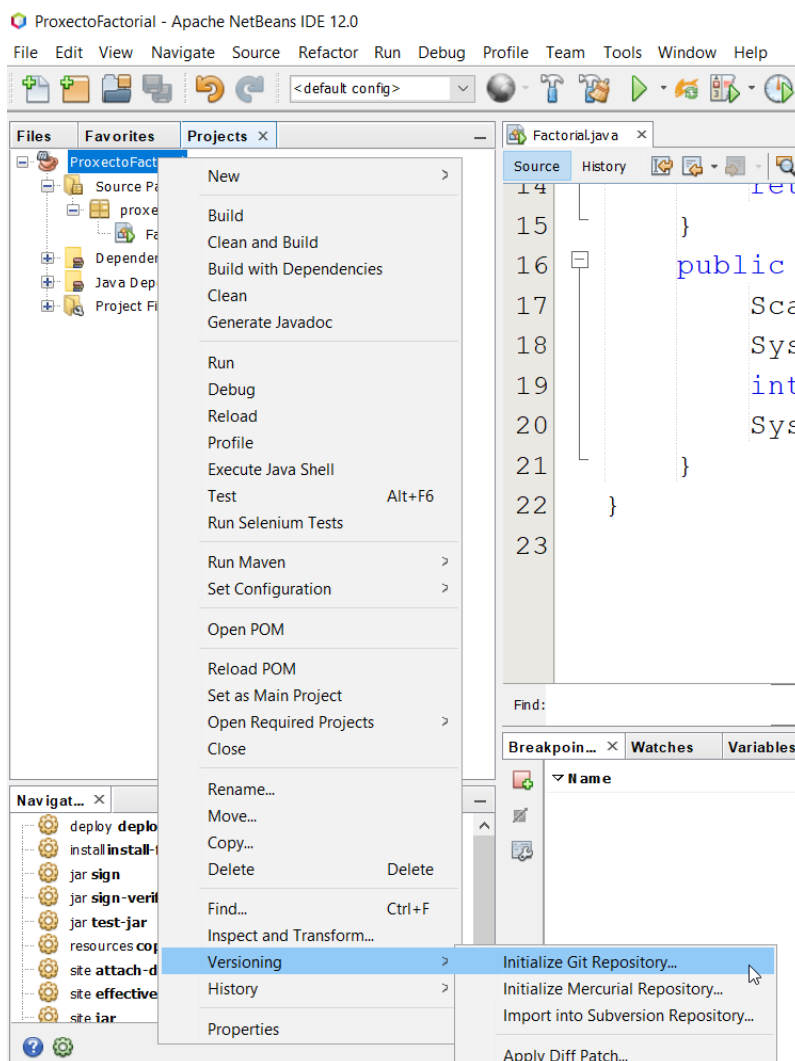
O IDE Apache NetBeans ofrece soporte para o sistema de control de versións Git. Polo tanto, pódense realizar as tarefas de Git desde os proxectos e código dentro do IDE.

Git está deseñado para xestionar desde proxectos pequenos ata proxectos moi grandes con velocidade e eficiencia. Cada clon de Git é un repositorio completo con historial completo e capacidades completas de seguimentos de versións, que non dependen do acceso á rede nin dun servidor central. A creación e o fusionado de ramas faise con rapidez e é doado de facer.

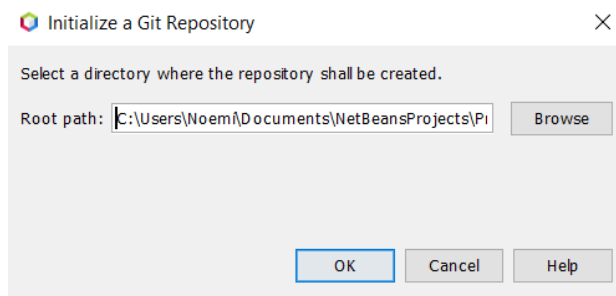
Inicialización dun repositorio Git

Para **inicializar** un **repositorio** de **Git** a partir de ficheiros existentes que aínda non están no sistema de control de versións, cómpre completar os seguintes pasos:

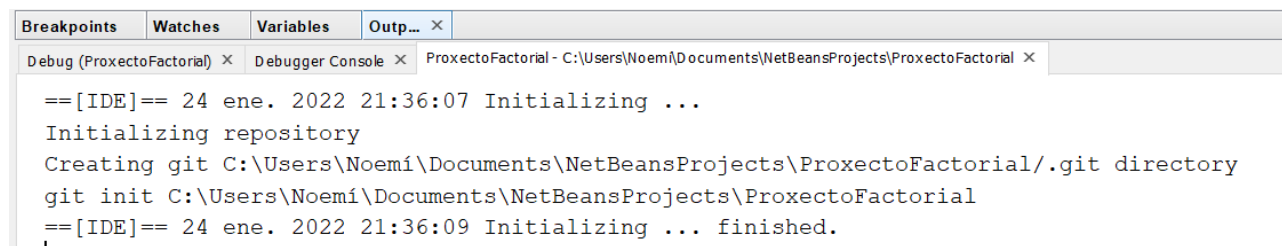
1. Na xanela Proxectos, seleccionamos un proxecto sen versión e prememos co botón dereito do rato no nome do proxecto.
2. No menú contextual, escollemos **Versioning > Initialize Git Repository...** (como alternativa, no menú principal, escollemos **Team > Git > Initialize Repository...**).



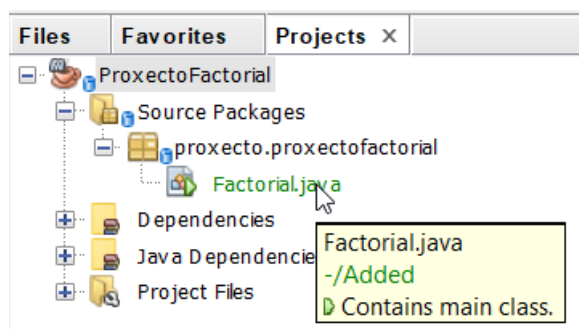
A continuación, debemos especificar la ruta al repositorio en el que queremos guardar los archivos versionados.



Tras indicar la ruta al repositorio, pulsamos en el botón OK.



Crease un **subdirectorio .git** en el cartel especificado en los pasos anteriores que es el **repositorio Git** donde se almacenará todo el **historial del proyecto**. En la ventana **Output** puede verse un informe del progreso de la creación del repositorio en el directorio de trabajo local.

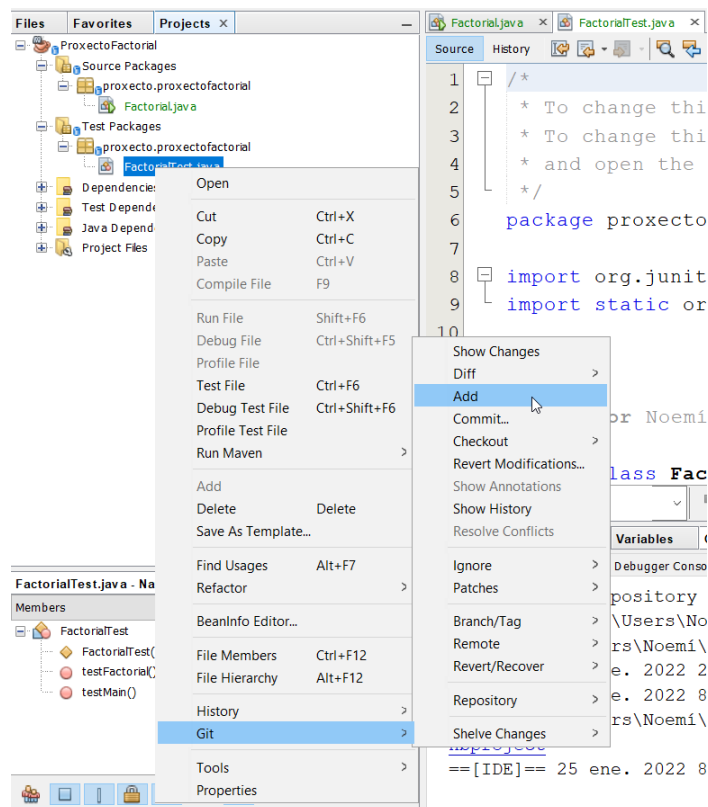


Todos los archivos del proyecto marcanse como **Added** en el **directorio de trabajo**. Para ver el **status** de un archivo, situamos el cursor sobre el nombre del archivo en la ventana **Projects**. El **status** del archivo amósase en **verde** en el directorio de trabajo tal e como se puede ver en la imagen anterior.

Después de inicializar un repositorio Git, pueden engadirse (**add**) archivos o directamente confirmarlos (**commit**) al repositorio Git.

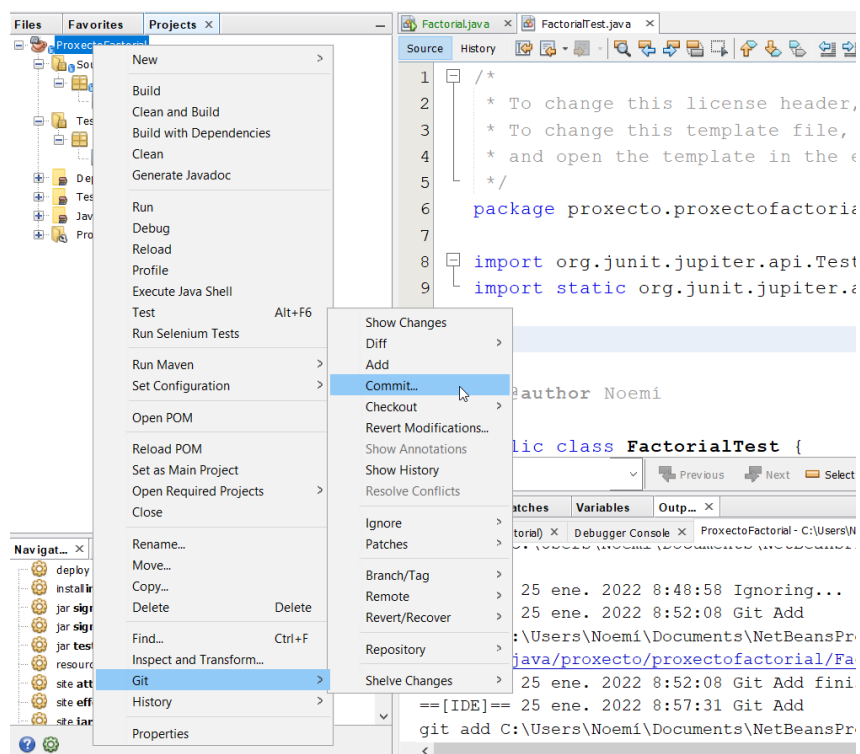
Engadir archivos a un repositorio Git

Para comenzar a hacer el seguimiento de un nuevo archivo y también para realizar cambios en un archivo que ya está en un repositorio Git, es necesario pasarlo a la zona de preparación y después confirmarlos. En NetBeans situámonos en la ventana **Projects** y pinchamos el botón derecho del ratón sobre el archivo que queremos engadir y elegimos **Git > Add** en el menú contextual que aparece.

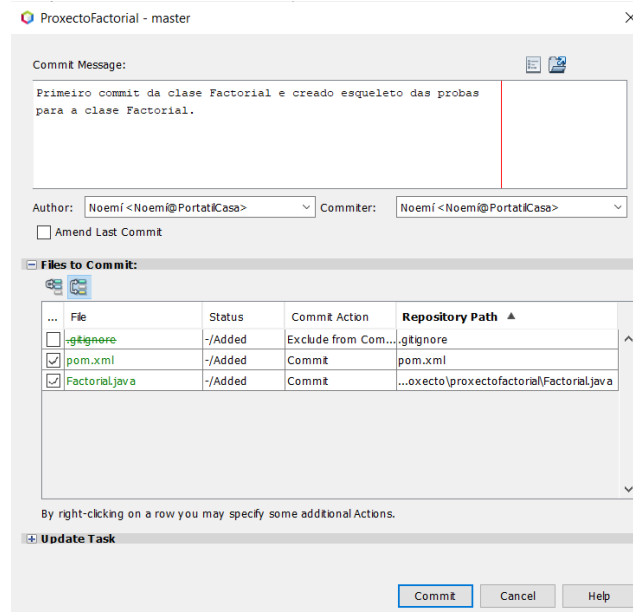


Desta maneira engádesse o contido do arquivo á **área de preparación** antes de confirmalo. Vemos como na ventá Projects a cor do nome do arquivo engadido á área de preparación pasa a ser verde. Tamén pasa a **cor verde** o nome do arquivo na ventá de edición.

Para confirmar os cambios e pasalos ao directorio Git prememos na ventá **Projects** no directorio ou arquivo que queremos confirmar (commit). Eliximos **Git > Commit...** no menú contextual que nos aparece.



Aparece a pantalla do **Commit** onde debemos seleccionar os arquivos que queremos confirmar.



A mensaxe de commit é un texto curto que debe servir para comunicar o contexto sobre un cambio ao resto de desenvolvedores nun proxecto (e a un mesmo no futuro). Por este motivo, e moi importante pensar a mensaxe que se vai utilizar e non deixar nunca sen texto esta mensaxe.

Finalmente, debemos pinchar no botón **Commit** se realmente desexamos confirmar os cambios.

Tras confirmar (commit) os cambios dun arquivo, o **nome do arquivo** pasa a **cor negra**.

En NetBeans é posible facer a confirmación (commit) sen ter engadido previamente os arquivos a confirmar á área de preparación utilizando directamente **Git > Commit...** na ventá **Projects**.

Edición de arquivos

Unha vez temos un proxecto versionado con Git no IDE, podemos comezar a facer cambios no código. Podemos abrir o editor de código fonte facendo dobre clic no nodo da ventá **Projects** co seu nome.

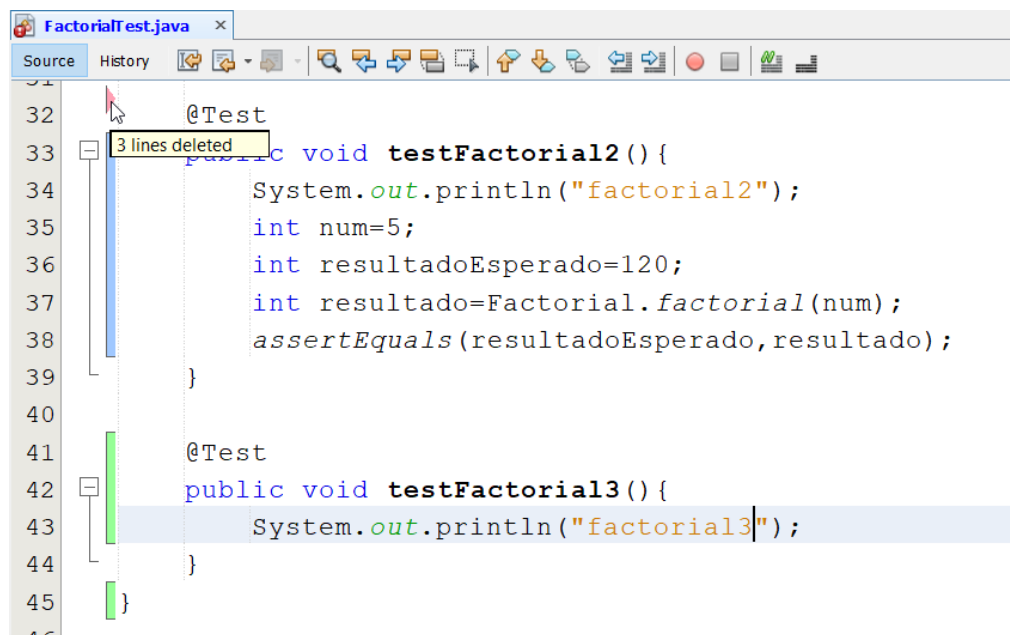
Mentres se traballa cos ficheiros fonte no IDE, hai varios compoñentes da interface de usuario á nosa disposición para axudarnos a traballar cos comandos de control de versións.

Ver os cambios no editor de código fonte

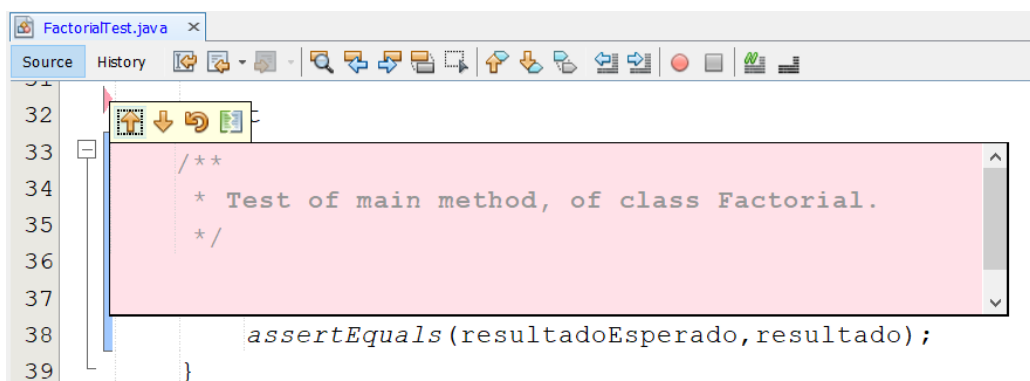
Cando abrimos un ficheiro versionado no editor do IDE, podemos ver os cambios que se producen no ficheiro en tempo real mentres modificamos o código con respecto á versión base do repositorio Git. Mentres traballamos, o IDE usa códigos de cores nas marxes do editor de código fonte para transmitir a seguinte información:

- **Azul.** Indica as liñas que foron modificadas desde a versión anterior.
- **Verde.** Indica as liñas que se engadiron desde a versión anterior.
- **Vermello.** Indica as liñas que foron eliminadas desde a revisión anterior.

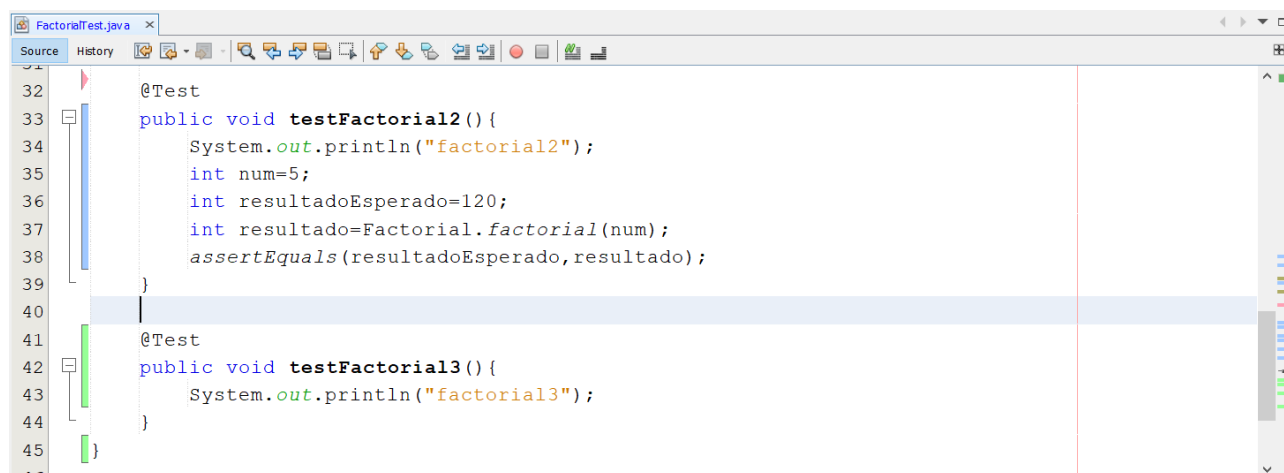
O editor amosa os cambios realizados en cada liña inmediatamente na marxe esquerda.



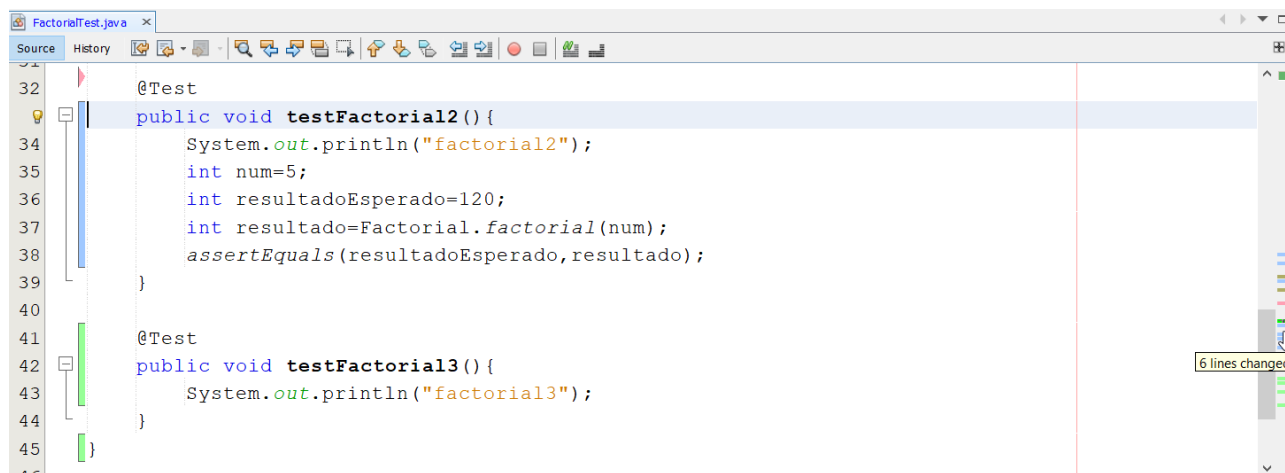
Podemos facer clic sobre un grupo de cor na marxe esquerda para chamar aos comandos de Git. Por exemplo, a imaxe inferior mostra o que se visualiza cando pinchamos na icona vermella que indica que se borraron liñas da copia local:



A marxe dereita do editor de código fonte ofrece unha visión xeral dos cambios realizados no ficheiro en conxunto, de arriba a abaixo. A codificación de cores xérase mentres se fan os cambios no ficheiro.

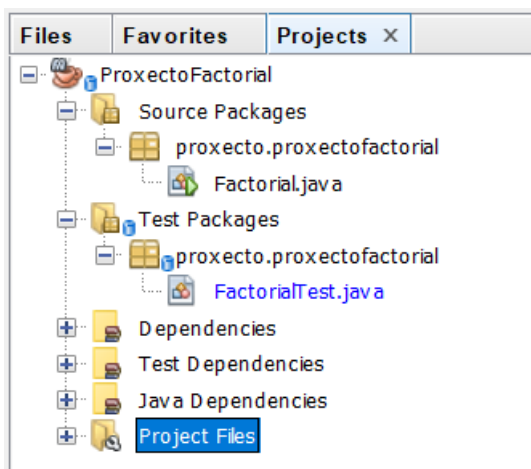


Podemos facer clic nun punto específico da marxe dereita para levar o cursor á liña do ficheiro representada por ese punto. Para ver o número de liñas afectadas podemos pasar o rato sobre as iconas de cores da marxe dereita.



Ver a información de estado dos ficheiros



Cando traballamos na ventá **Projects**, (Ctrl+1), na ventá **Files** (Ctrl+2), na ventá **Favorites** (Ctrl+3), ou na vista **Versioning** o IDE ofrece varias axudas visuais para ver a información de estado dos ficheiros. Na imaxe inferior podemos ver como o distintivo azul e a cor do ficheiro coinciden para ofrecer de xeito sinxelo pero eficaz información de versión dos ficheiros.



Os distintivos, a codificación de cores e o visualizador de Git Diff axudan a xestionar de forma eficaz a información de versións no IDE.

Distintivos e codificación de cores

Os distintivos aplícanse aos nodos do proxecto, cartafol e paquete e informa do estado dos ficheiros contidos nese nodo.

Distintivos	
Distintivo azul 	Distintivo vermello 

O **distintivo azul** indica a presenza de **ficheiros** que foron modificados, **engadidos** ou **eliminados** nun directorio de traballo. Para os paquetes, este distintivo aplícase só ao paquete en si e non aos subpaquetes. No caso de proxectos ou cartafoles, o distintivo indica os cambios dentro dese elemento ou de calquera dos subcartafoles contidos.

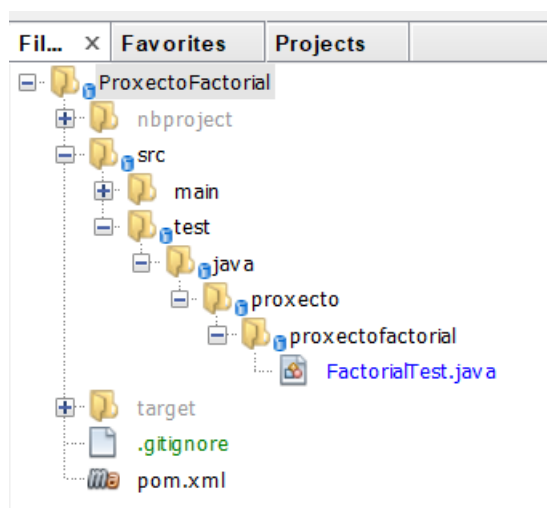
O **distintivo vermello** marca proxectos, cartafoles ou paquetes que conteñen ficheiros en **conflito**. Para os paquetes, este distintivo aplícase só ao paquete en si e non aos subpaquetes. Para proxectos ou cartafoles, o distintivo indica conflitos dentro dese elemento ou calquera dos subcartafoles contidos.

Etiquetas de estado dos ficheiros

A codificación de cores aplícase aos nomes dos ficheiros para indicar o seu estado actual en referencia ao repositorio. As cores son:

- **Negra**. Indica que o ficheiro non ten cambios.
- **Azul**. Indica que o ficheiro foi modificado localmente.
- **Verde**. Indica que o ficheiro foi engadido.
- **Vermella**. Indica que o ficheiro está nun conflito de combinación (merge).
- **Gris**. Indica que Git ignora o ficheiro.

Na seguinte imaxe amósase a ventá **Files** dun proxecto no que aparecen case todas as cores para os nomes dos ficheiros indicadas anteriormente:

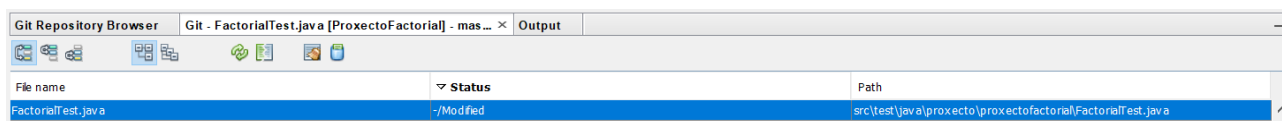


Vista Versioning

A vista **Versioning** ofrece unha listaxe de todos os cambios feitos nos ficheiros dun determinado cartafol do directorio de traballo local.

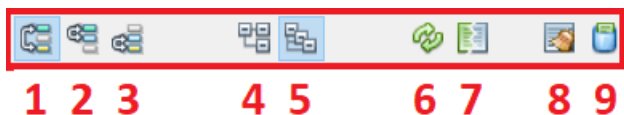
Para abrir a vista **Versioning**, seleccionamos un ficheiro ou carpeta versionados (das ventás Projects, Files ou Favorites) e eliximos **Git > Show Changes** do menú contextual que aparece ao

pinchar no botón dereito do rato ou eliximos **Team > Show Changes** do menú principal. Aparece a seguinte ventá no IDE:



Por defecto a vista **Versioning** inclúe tamén unha **barra de ferramentas** con botóns que permiten invocar as tarefas máis comúns de Git para todos os ficheiros amosados na lista.

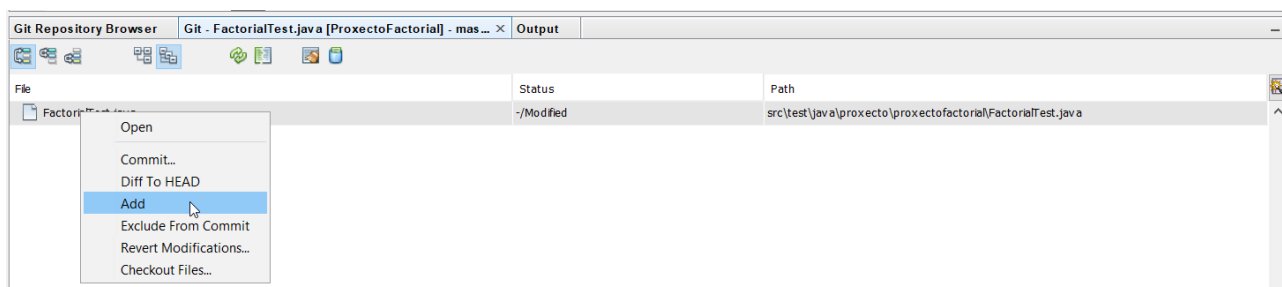
A barra de ferramentas representámola de novo na seguinte imaxe:



A utilidade de cada un dos botóns da barra de ferramentas é a seguinte:

1. Cambios entre HEAD e o directorio de traballo. Mostra unha lista de ficheiros que xa están na área de preparación ou só modificados/creados e non preparados.
2. Cambios entre HEAD e Index. Mostra unha lista cos ficheiros que está preparados.
3. Cambios entre Index e o directorio de traballo. Mostra os ficheiros nos que hai diferencias entre os ficheiros preparados e os ficheiros do directorio de traballo.
4. Mostra os resultados con formato de táboa.
5. Mostra os resultados con formato de árbore.
6. Actualiza os estados. Actualiza os estados dos ficheiros e cartafoles seleccionados. Os ficheiros da vista Versioning poden actualizarse para reflexar calquera cambio que se teña feito externamente.
7. Abrir o visor Diff. Abre o visor Diff para poder comparar copias locais coas versións almacenadas no repositorio.
8. Desfacer modificacións. Mostra a ventá de diálogo Revert Modifications.
9. Confirmar cambios. Mostra a ventá de diálogo Commit.

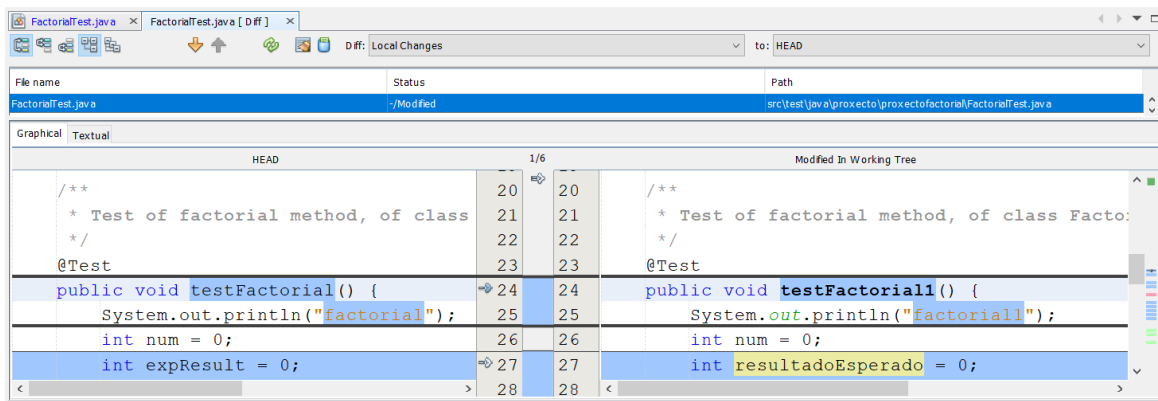
Podemos acceder a outros comandos Git na vista Versioning seleccionando unha fila da táboa que se corresponda cun ficheiro modificado, e elixindo un comando do menú contextual que aparece tras pinchar no botón dereito do rato:



Comparar versións de ficheiros

A comparación de versións de ficheiros é unha tarefa habitual cando se traballa con proxectos versionados. O IDE permite comparar versións usando o comando **Diff**.

Para utilizar Diff seleccionamos un ficheiro ou cartafol versionado (por exemplo dende a ventá Projects, Files ou Favorites) e escollemos **Team > Diff > Diff to HEAD** no menú principal. Abrirase un visor gráfico de diferenzas na xanela principal do IDE. O visor mostra a copia máis actual á dereita e a versión do HEAD á esquerda.



O visor de Diff fai uso da codificación de cores usada noutros lugares para amosar os cambios de control de versión.

A barra de ferramentas do visor Diff tamén inclúe botóns que permiten invocar as tarefas de Git máis comúns en todos os ficheiros que aparecen na lista.

A barra de ferramentas representámola de novo na seguinte imaxe:






A utilidade de cada un dos botóns da barra de ferramentas é a seguinte:

1. Cambios entre HEAD e o directorio de traballo. Mostra unha lista de ficheiros que xa están na área de preparación ou só modificados/creados e non preparados.
2. Cambios entre HEAD e Index. Mostra unha lista cos ficheiros que está preparados.
3. Cambios entre Index e o directorio de traballo. Mostra os ficheiros nos que hai diferenzas entre os ficheiros preparados e os ficheiros do directorio de traballo.
4. Mostra os resultados con formato de táboa.
5. Mostra os resultados con formato de árbore.
6. Ir á seguinte diferenza. Mostra a seguinte diferenza entre ficheiros.
7. Ir á anterior diferenza. Mostra a anterior diferenza entre ficheiros.

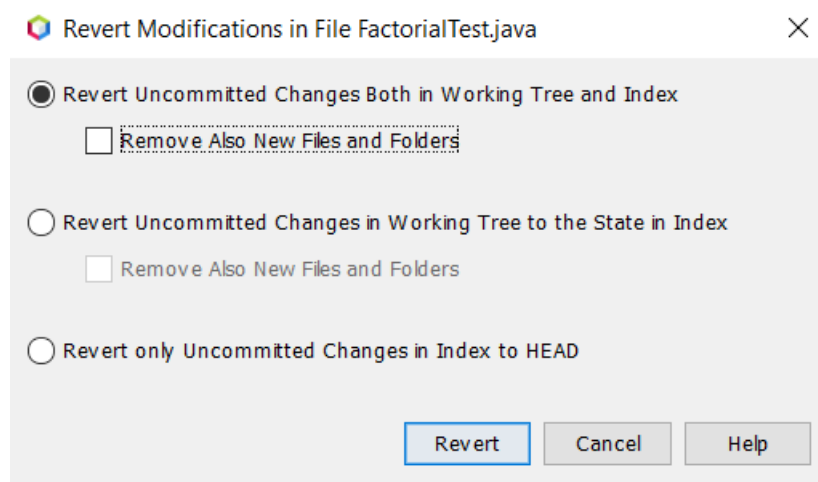
8. Actualiza os estados. Actualiza os estados dos ficheiros e cartafolse seleccionados. Os ficheiros da vista Versioning poden actualizarse para reflexar calquera cambio que se teña feito externamente.
9. Desfacer modificacións. Mostra a ventá de diálogo Revert Modifications.
10. Confirmar cambios. Mostra a ventá de diálogo Commit.

Cando se está realizando un diff con respecto ao directorio de traballo local, o IDE permite facer cambios directamente desde o visor Diff. Para facelo, podemos colocar o cursor no panel dereito do visor Diff e modificar os ficheiros ou pode facerse uso das iconas que se visualizan ao lado de cada cambio.

Icona	Nome	Función
	Replace (reempazar)	Insire o texto remarcado na copia do directorio de traballo
	Move All (mover todas)	Desfai todos os cambios feitos na copia local
	Remove (borrar)	Borra o texto remarcado da copia do directorio de traballo

Desfacer cambios

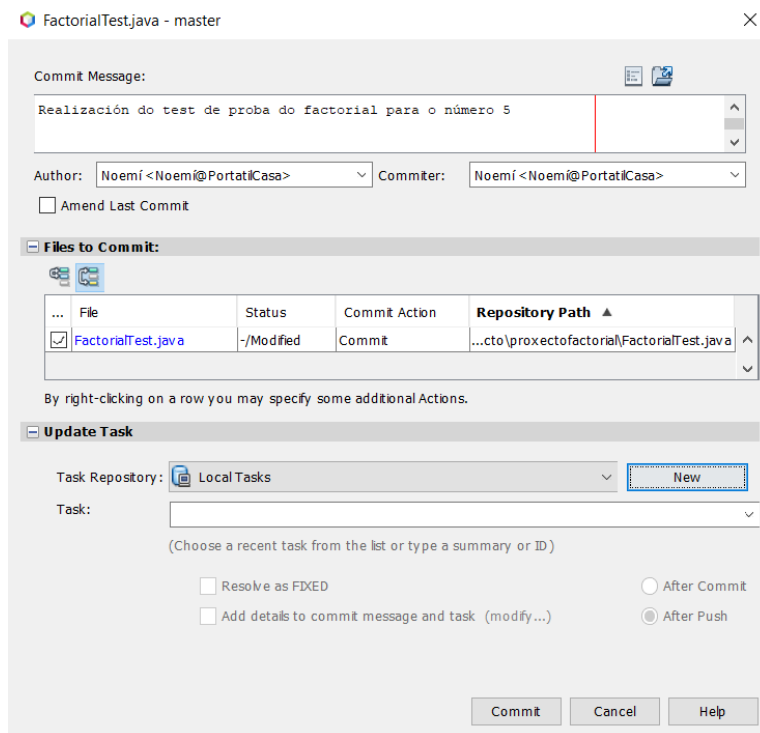
Para eliminar calquera cambio local feito a ficheiros do directorio de traballo e reempazalos cos do Index ou HEAD, debemos, en primeiro lugar seleccionar o ficheiro versionado ou o cartafol e ir no menú principal a **Team > Revert Modifications**. Aparecerá o cadro de diálogo Revert Modifications.



Debemos indicar o tipo de cambio que queremos facer e, premer o botón **Revert** e o IDE fará o reempazamento que se lle teña indicado.

Gardar o código fonte no repositorio Git


Para gardar unha instantánea do código fonte no repositorio Git temos que elixir previamente na vista Projects o ficheiro ou ficheiros que queremos gardar. Unha vez seleccionados, prememos no botón dereito do rato para que nos apareza o menú contextual no que eliximos **Git > Commit** para que apareza a caixa de diálogo **Commit**.




A **caixa de diálogo Commit** ten os seguintes compoñentes:

- A caixa de texto para a mensaxe de Commit para describir o cambio que se está gardando.
- **autor** e a **persoa** que **garda o cambio** para distinguir entre quen fixo o cambio e quen gardou o cambio no repositorio.
- Unha sección que lista os ficheiros a gardar:
 - Todos os ficheiros modificados.
 - Todos os ficheiros borrados do directorio de traballo local.
 - Todos os ficheiros novos e que aínda non existen no repositorio Git.
 - Todos os ficheiros que se renomearon.



Hai dispoñibles tamén dous botóns que cambian o modo no que se realizará a confirmación do cambio.

Compoñente da IU	Nome	Descrición
	Cambios entre HEAD e Index	Mostra unha lista dos ficheiros que están preparados.

	Cambios entre HEAD e o directorio de traballo	Mostra unha lista de ficheiros que están preparados ou só modificados/creados e, polo tanto, aínda non están preparados.
---	--	--

A sección **Update Task** permite traballar con ferramentas de seguimento de proxectos.

Podemos escribir unha mensaxe de confirmación na caixa de texto **Commit Message** ou podemos:

- Facer clic na icona **Recent Messages** () situada na esquina superior dereita para ver e seleccionar dunha lista de mensaxes utilizadas anteriormente.
- Premer na icona **Load Template** () situada na esquina superior dereita para seleccionar un modelo de mensaxe.

Unha vez configurada a confirmación que se quere realizar, temos que pinchar no botón Commit para que o IDE execute o commit que almacena as instantáneas no repositorio.

Tras un commit exitoso, todos os distintivos de versión desaparecen das ventás Projects, Files e Favorites e a codificación de cores dos ficheiros confirmados volve ao negro.

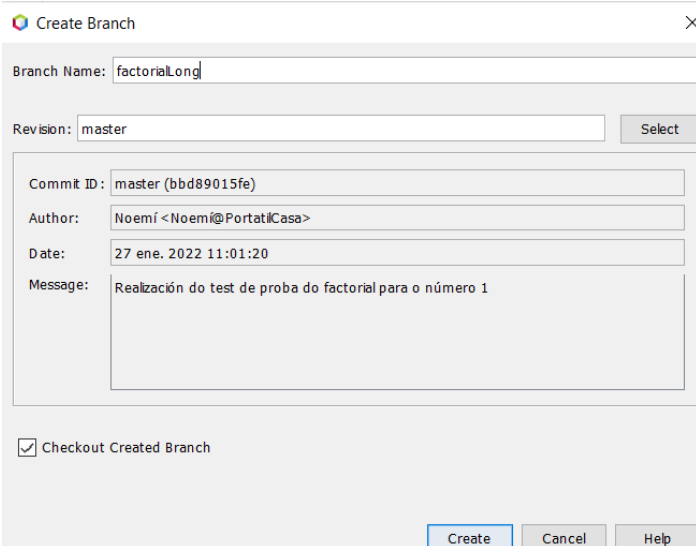
Traballar con ramas

O IDE NetBeans permite traballar con ramas de Git. Git permite crear novas ramas dun proxecto para probar ideas, illar novas características ou experimentar sen impactar ao proxecto principal.

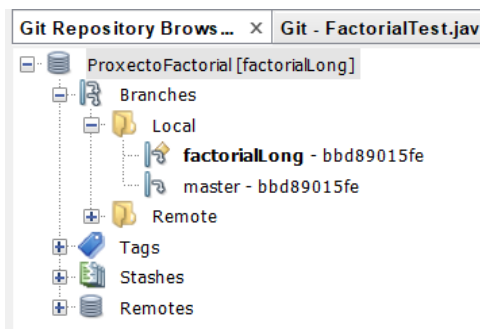
Crear unha rama

Para crear unha rama en NetBeans debemos elixir o proxecto ou cartafol do repositorio no que queremos crear a rama. No menú principal eliximos **Team > Branch/Tag > Create Branch**. Tamén podemos pinchar co botón dereito de rato sobre o proxecto ou cartafol versionado e elixir no menú contextual **Git > Branch/Tag > Create Branch**.

A continuación, debe aparecer a caixa de diálogo **Create Branch**.



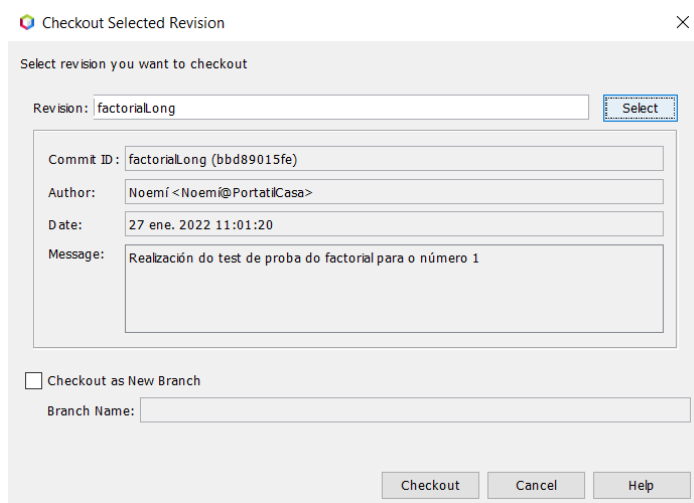
Na caixa de texto Branch Name debemos indicar o nome da rama que estamos a crear. Podemos indicar a partir de que commit queremos crear a rama indicando o identificador de commit, seleccionando de entre as versións que hai no repositorio, indicando unha rama que xa existe... Unha vez temos configurado correctamente a rama que queremos crear, prememos no botón **Create** e engádese unha nova rama no cartafol Branches/Local do repositorio Git.



Traballar cunha rama

Se queremos editar ficheiros nunha rama que xa existe podemos movernos (check out) a esa rama para copiar os ficheiros ao directorio de traballo.

Para movernos a unha versión temos que elixir **Team > Checkout > Checkout Revision** desde o menú principal.

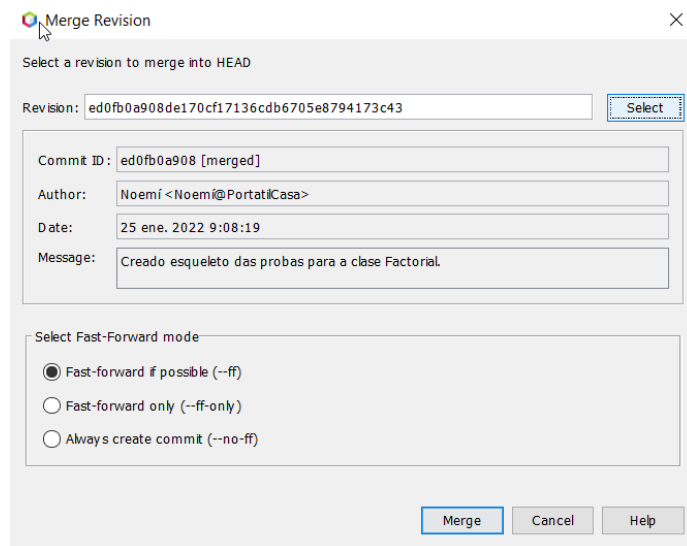


Temos que especificar a revisión á que queremos ir indicando un identificador de commit, unha rama existente ou o nome dunha etiqueta no campo Revision ou presionar o botón Select para ver a lista de versións que hai no repositorio.

Presionamos no botón **Checkout** para movernos á versión requirida.

Fusión de ramas

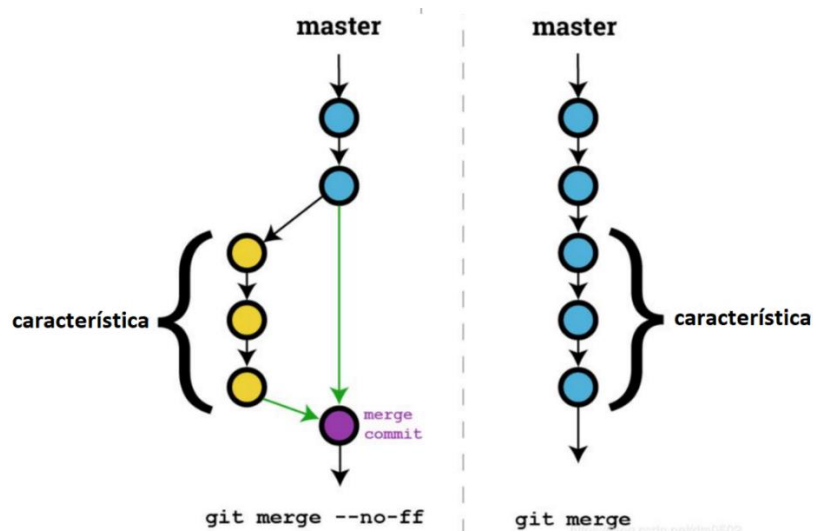
Para levar as modificacións dunha versión do repositorio ao directorio de traballo temos, en primeiro lugar, que escoller **Team > Branch/Tag > Merge Revision** no menú principal. Aparece a caixa de diálogo **Merge Revision**.



Pódense elixir varias formas de fusionar as ramas que non cambian o resultado final véndose afectada só a forma en que veremos o historial das ramas. Estas formas podemos velas na caixa de diálogo **Merge Revision** na zona onde se fai **Select Fast-Forward mode**. Por defecto Git utiliza o modo fast forward.

Se utilizamos fast-forward, farase a fusión das ramas e levaranse todos os cambios da rama que se indique no comando á rama na que estea Git nese momento.

Na seguinte imaxe pode verse unha comparativa entre un merge con fast-forward e un merge sen fast-forward.



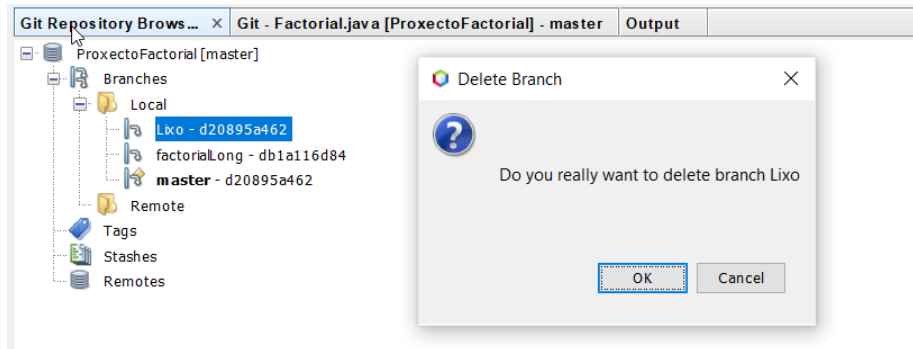
Se Git non detecta que o seu HEAD actual é un antepasado da rama que está tentando fusionar, non poderá facerse unha fusión con fast-forward.

Unha vez elixida a forma de fusionar, prememos no botón **Merge** e farase, se é posible, unha fusión entre a rama actual e a rama especificada.

Se se produce algún conflito durante a fusión, os ficheiros conflitivos márcanse cun distintivo vermello para indicalo.

Borrar unha rama

Para borrar unha rama que xa non se necesite podemos ir a **Team > Repository Browser** no menú principal. No **Repository Browser** eliximos a rama que queremos borrar. Facemos clic no botón dereito do rato e eliximos **Delete Branch** no menú contextual.



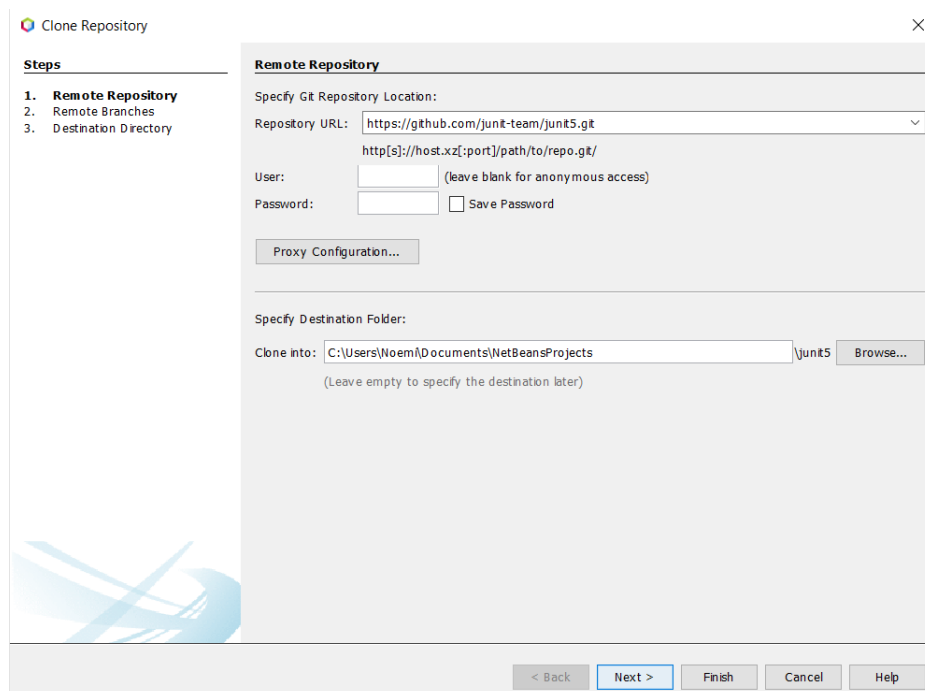
Premendo no botón **OK** confirmamos que queremos borrar a rama indicada. Tras isto, a rama queda borrada do repositorio local de Git e xa non se visualiza no Visor de Repositorios (Repository Browser) en NetBeans.

Clonar un repositorio remoto

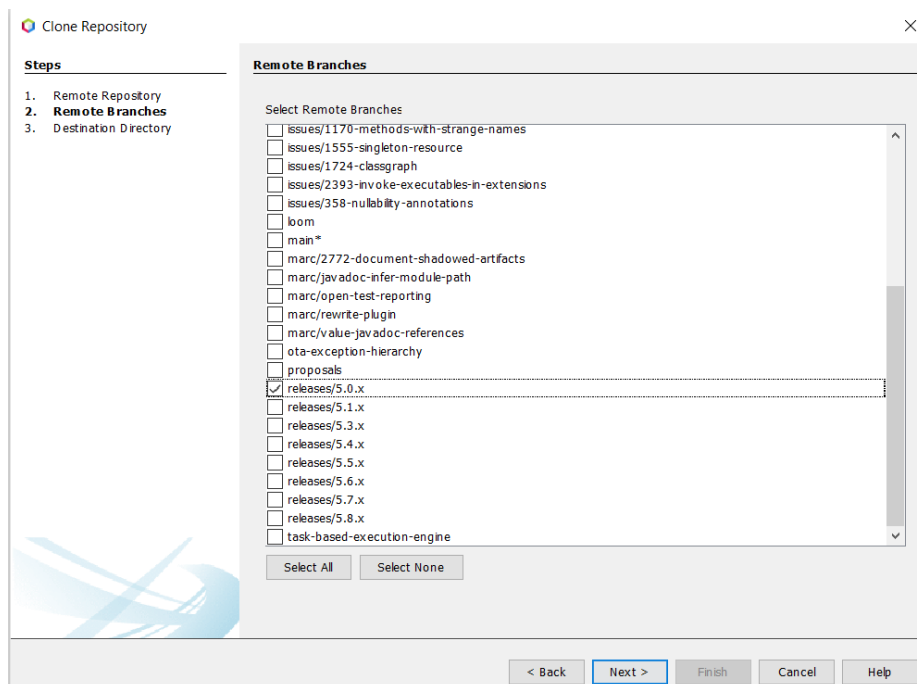
Para obter unha copia dun repositorio Git que xa existe, é necesario clonalo. Debemos saber a URL do repositorio antes de comezar a traballar co asistente de clonación.

Eliximos **Team > Git > Clone** no menú principal para abrir o asistente **Clone Repository**.

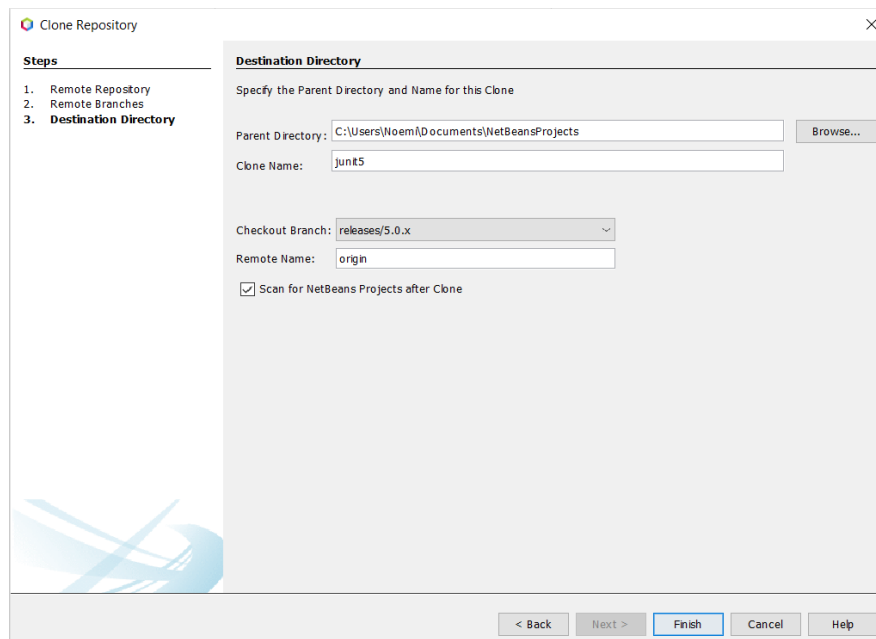
Debemos especificar a URL do repositorio remoto, o usuario e o contrasinal (PAT) e opcionalmente indicar a configuración do Proxy. Unha vez teñamos especificados os datos necesarios para acceder ao repositorio remoto, prememos no botón **Next>** do asistente.



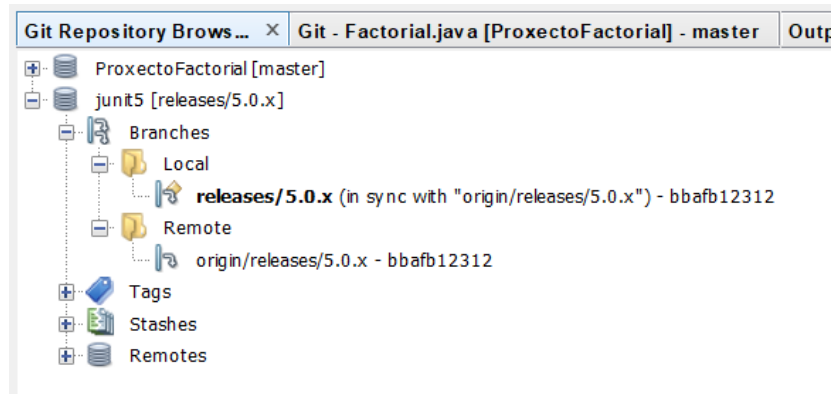
Na pantalla de ramas remotas (Remote Branches) debemos seleccionar as ramas do repositorio que queremos descargar ao repositorio local. Tras a selección, pinchamos no botón **Next>**.



Na pantalla do directorio de destino (Destination Directory) especificamos o directorio do disco duro onde queremos clonar o repositorio. No nome do clon (Clone Name) indicamos o nome do cartafol local onde o proxecto orixinal vai ser clonado. Por defecto o nome do clon échese co nome do repositorio Git remoto. Selecciónase a rama e o nome do repositorio a clonar. O alias por defecto do repositorio a clonar é origin. Recoméndase usar este valor.



Unha vez rematada a configuración do directorio de destino, prememos en **Finish**. Cando o repositorio se teña clonado, o cartafol con metadatos .git crease no cartafol elixido no asistente. No **Repository Browser** veranse entón as ramas local e remota dos repositorios.



Traballar con repositorios remotos

Cando traballamos con outros desenvolvedores necesitamos compartir o noso traballo. Polo tanto debemos comprobar, enviar e recibir datos cara e desde servidores remotos xa sexa na rede local ou en Internet.

Imos traballar cun repositorio remoto, por exemplo GitHub.

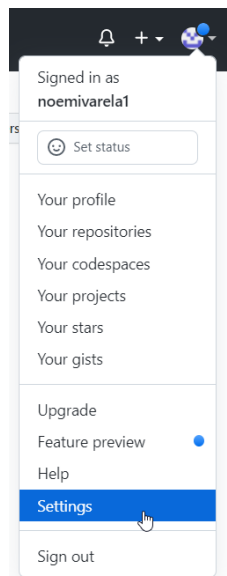
Despois de crear unha conta, creamos un repositorio.

Para crear un repositorio podemos ir simplemente a <https://github.com/new>. O repositorio debe estar limpo totalmente se queremos sincronizalo cun repositorio local.

En GitHub declararon como obsoleta a autenticación por medio de usuario e contrasinal e agora os desenvolvedores deben usar desde o IDE chamado **Personal Access Token (PAT)**.

Este PAT debe usarse en todo lugar onde se necesite un contrasinal para poder usar GitHub desde fora do sitio web. Por exemplo, clonar un repositorio privado de GitHub en local, enviar cambios a GitHub con Push, recibir cambios de GitHub con Pull, etc.

O **PAT** créase no sitio web de **GitHub**. Tras estar logueados co noso usuario e contrasinal no sitio web de GitHub, accedemos á opción **Settings** e despois a **Developer settings** e por último a **Personal access tokens**. A opción **Settings** atópase no menú desplegable do noso avatar.



As seguintes opcións accédese mediante o navegador vertical da esquerda.

Finalmente accédese a unha páxina onde podemos administrar os nosos **Personal access tokens**.

Creamos un novo token co botón **Generate new token**.



Aparece entón un formulario que hai que encher, indicando o nome do token e outros detalles como a expiración do token e os “scopes”.

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Token para usar en local

What's this token for?

Expiration *

30 days

The token will expire on Fri, Aug 27 2021

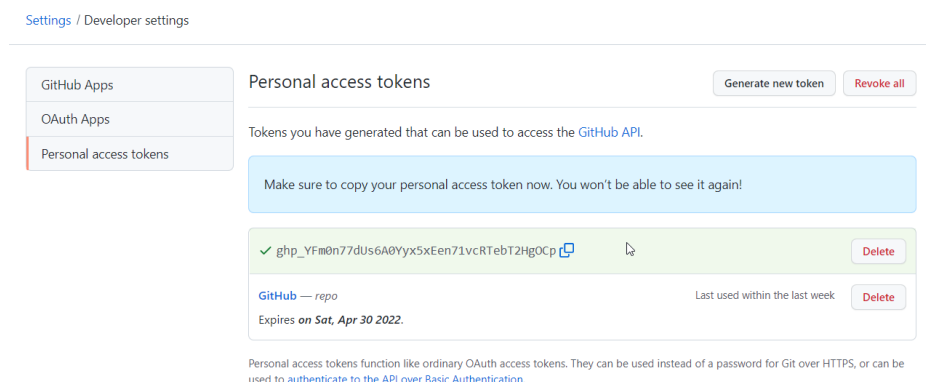
Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- | | |
|--|--------------------------------------|
| <input type="checkbox"/> repo | Full control of private repositories |
| <input type="checkbox"/> repo:status | Access commit status |
| <input type="checkbox"/> repo_deployment | Access deployment status |
| <input type="checkbox"/> public_repo | Access public repositories |
| <input type="checkbox"/> repo:invite | Access repository invitations |
| <input type="checkbox"/> security_events | Read and write security events |

Os scopes son as operación que estarán ou non permitidas para ese token. O máis común é dar permisos para acceder aos repositorios remotos aloxados en GitHub.

Unha vez xerado o token, aparecerá na páxina de GitHub. Podemos copialo e poñelo nun lugar seguro. Temos que ter en conta que é como se fose un contrasinal de usuario.



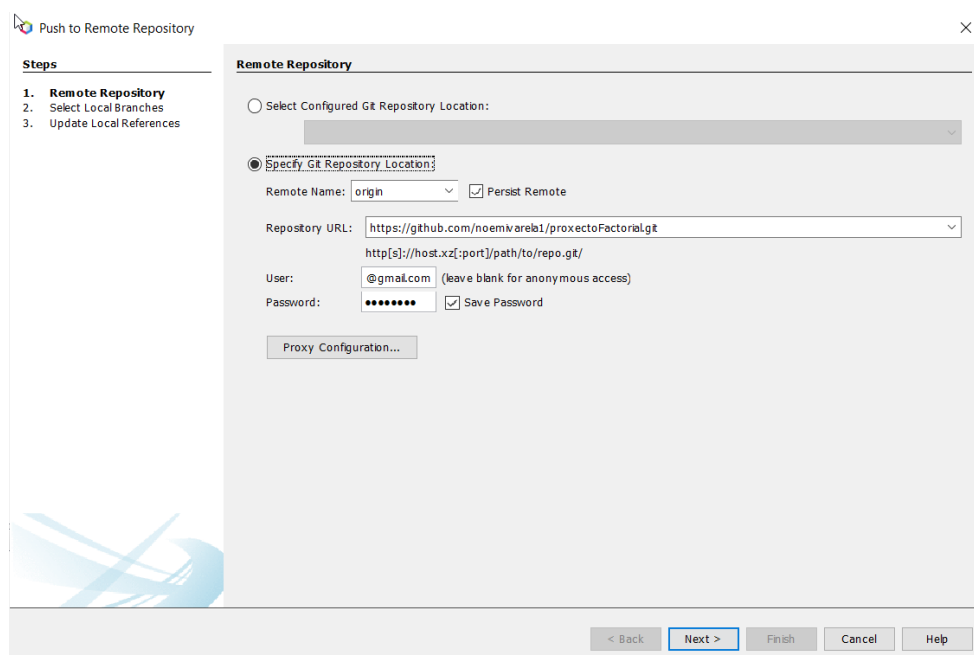
O PAT non debemos pegalo no código de ningunha aplicación para non deixalo visible a outras persoas ou o noso acceso a GitHub podería verse comprometido.

Temos que ter en conta, tamén, que este token só aparecerá unha vez na páxina de GitHub. Se necesitamos recuperalo por telo perdido, non poderemos facelo. Teremos que xerar un novo.

Para usar o PAT, cando unha operación de GitHub pida o contrasinal de usuario, en lugar do contrasinal colocaremos o token.

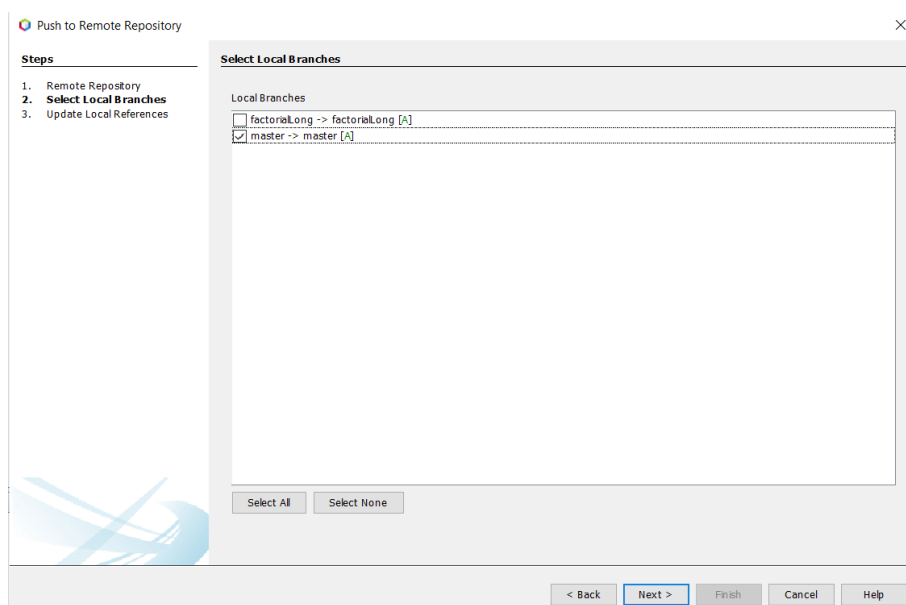
Enviar datos a un repositorio remoto

Para enviar cambios desde un repositorio local Git a un repositorio público Git temos que elixir **Team > Remote > Push**. Aparece entón o asistente **Push to Remote Repository**.

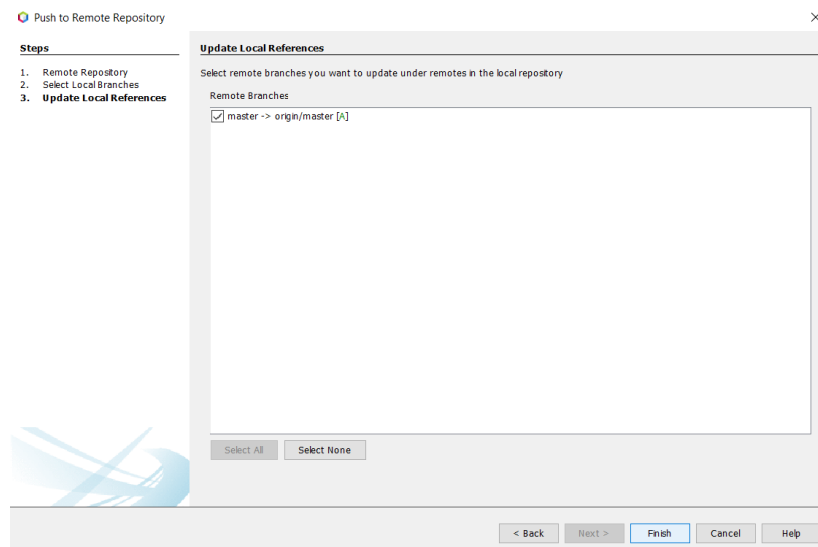


Indicamos o nome do repositorio remoto e o usuario e contrasinal (PAT) da conta de GitHub.

A continuación, indicamos as ramas locais que queremos enviar ao repositorio remoto.



Na páxina **Update Local References** eliximos as ramas a actualizar no directorio **Remotes** do repositorio local.



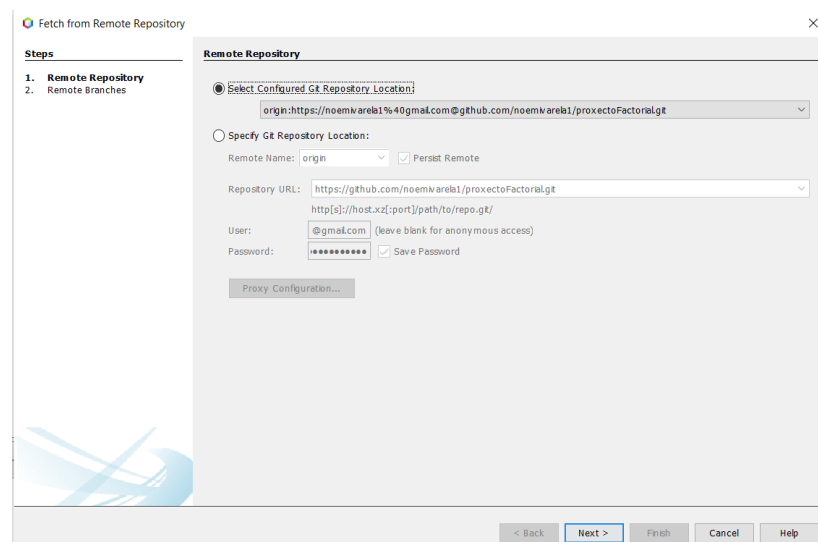
Pinchamos no botón **Finish** e a rama do repositorio remoto actualízase co último estado da rama local.

Recibir datos dun repositorio remoto con fetch

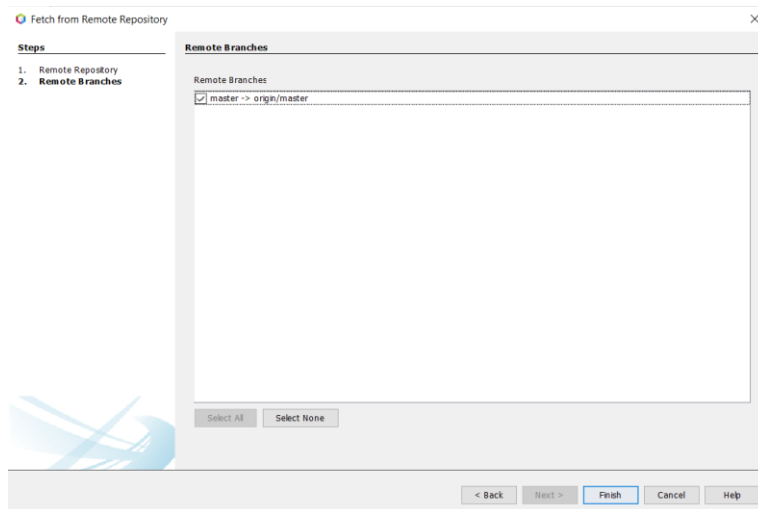
Podemos obter os cambios do repositorio remoto usando o comando de Git **fetch**. Con **fetch** obtéñense todos os cambios do repositorio remoto que aínda non temos pero sen cambiar nada nas ramas locais.

Para utilizar **fetch** temos que elixir **Team > Remote > Fetch**. Aparece o asistente **Fetch from Remote Repository**.

Na páxina **Remote Repository** do asistente seleccionamos o repositorio configurado ou seleccionamos especificar a localización do repositorio Git para definir a ruta a un repositorio remoto ao que aínda non temos accedido. Unha vez configurado o repositorio remoto, pinchamos en **Next >**.



Na páxina **Remote Branches** do asistente eliximos as ramas das que queremos obter os cambios e, pinchamos no botón **Finish**. Créase unha copia local da rama remota.



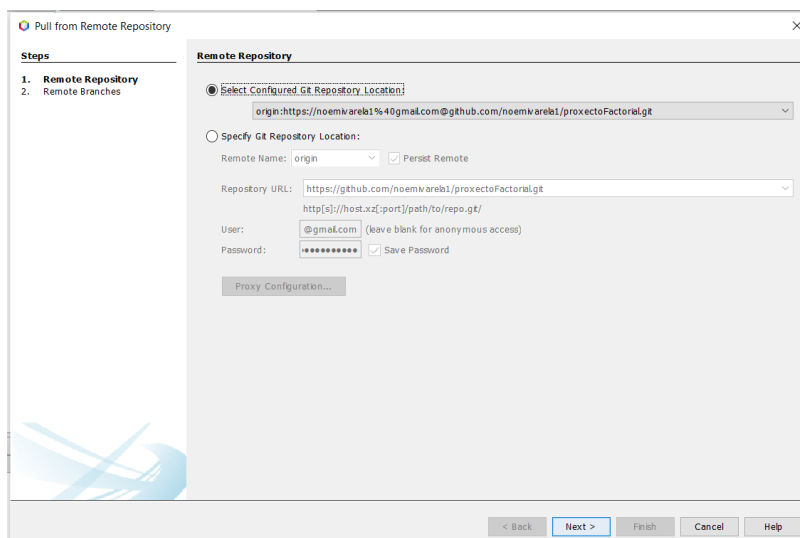
A rama seleccionada actualízase no directorio **Branches > Remote** do **Git Repository Browser**. Posteriormente as actualizacións obtidas poden fusionarse coa rama local.



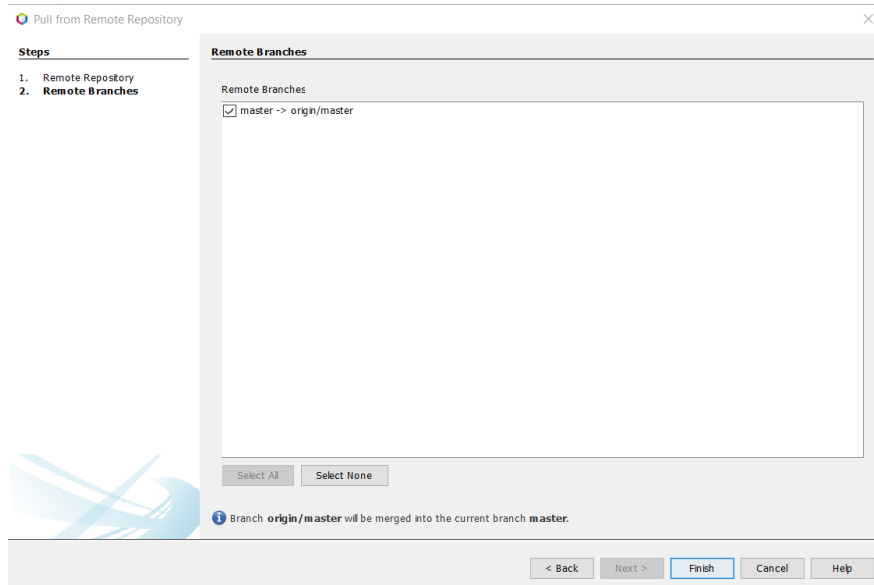
Recibir datos dun repositorio remoto con pull

Cando recibimos con **pull** actualizacións dun repositorio Git remoto, os cambios obtéñense dese repositorio e fusiónanse na HEAD actual do repositorio local.

Para recibir as actualizacións eliximos **Team > Remote > Pull**. Aparecerá o asistente **Pull from Remote Repository**.



Na páxina **Remote Repository** do asistente seleccionamos o repositorio configurado ou seleccionamos especificar a localización do repositorio Git para definir a ruta a un repositorio remoto ao que aínda non temos accedido. Unha vez configurado o repositorio remoto, pinchamos en **Next >**.



Na páxina **Remote Branches** do asistente eliximos as ramas das que queremos obter os cambios e pulsamos no botón **Finish**. O repositorio local sincronízase co repositorio remoto tal e como podemos ver no Git Repository Browser.

