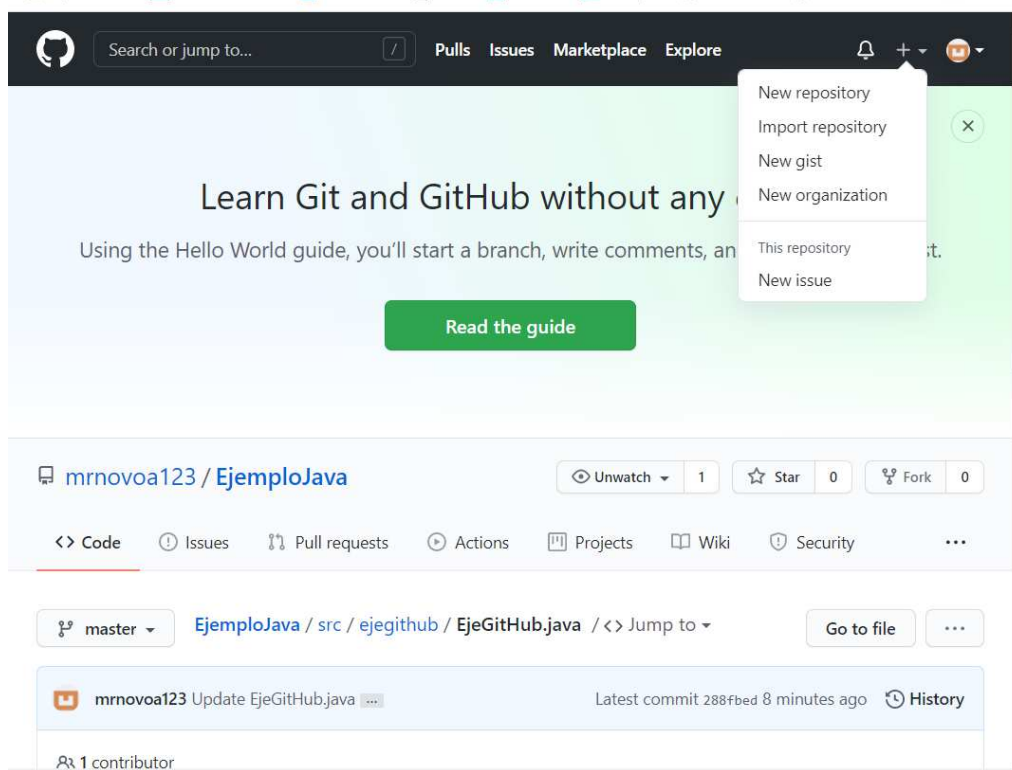


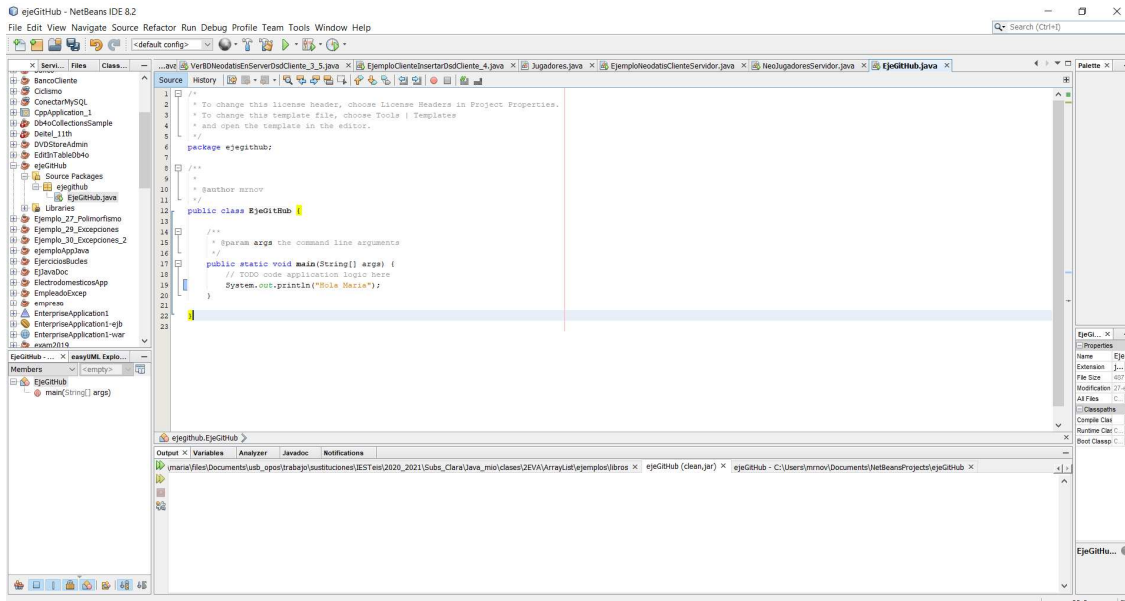
GITHUB

- En primer lugar, tener o crear Cta. GitHub
- Acceder al menú desplegable y seleccionar *Nuevo repositorio*

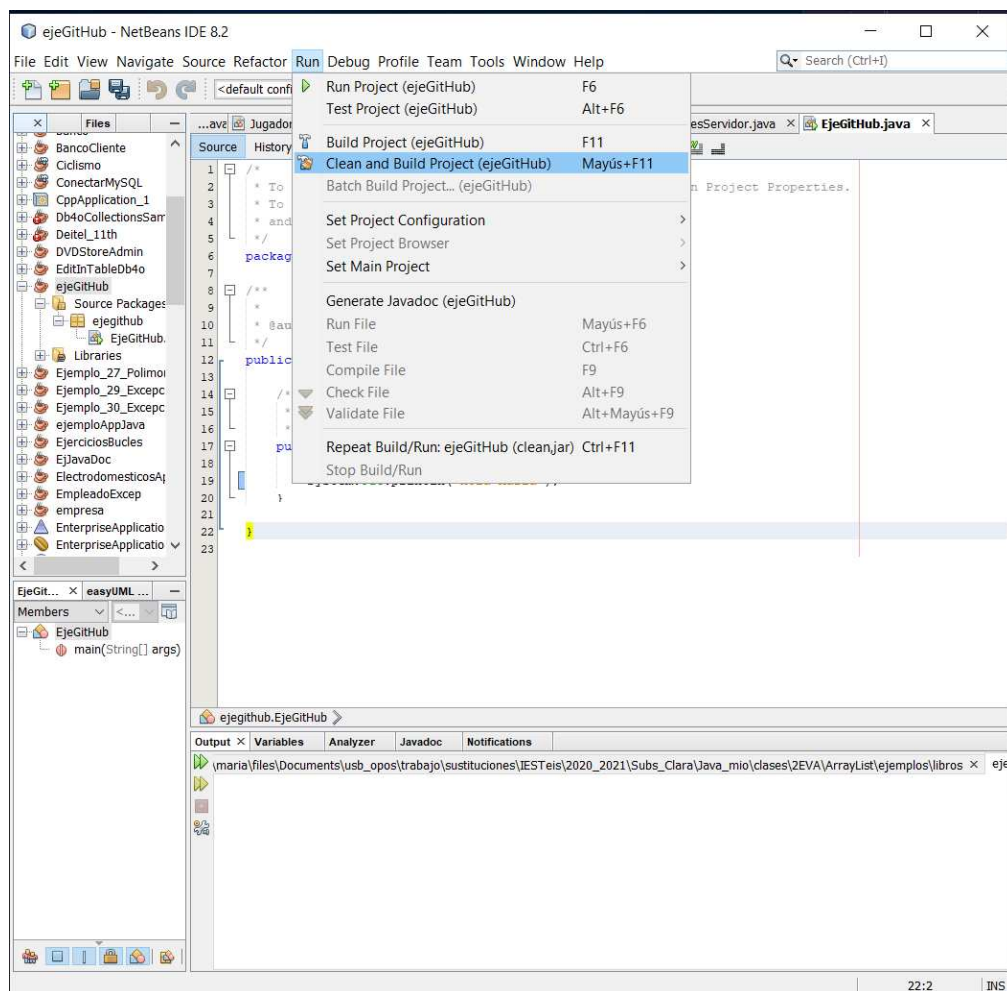


NETBEANS

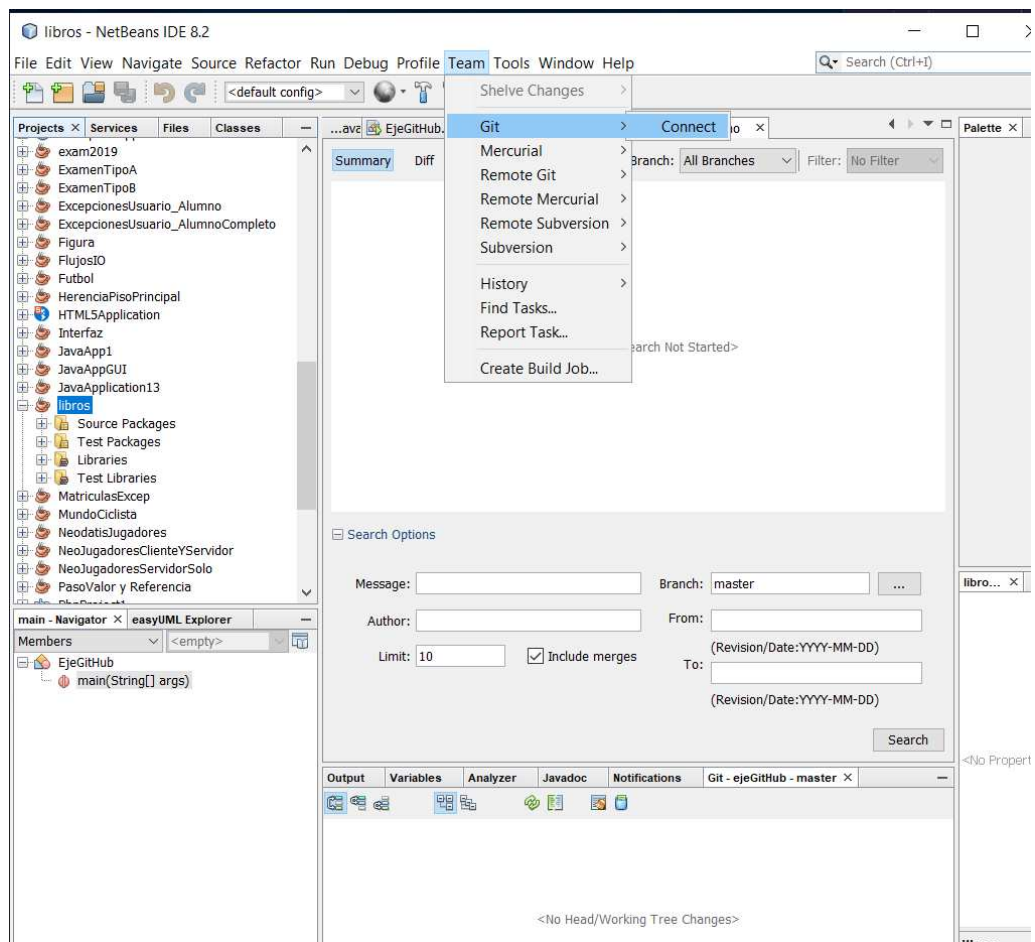
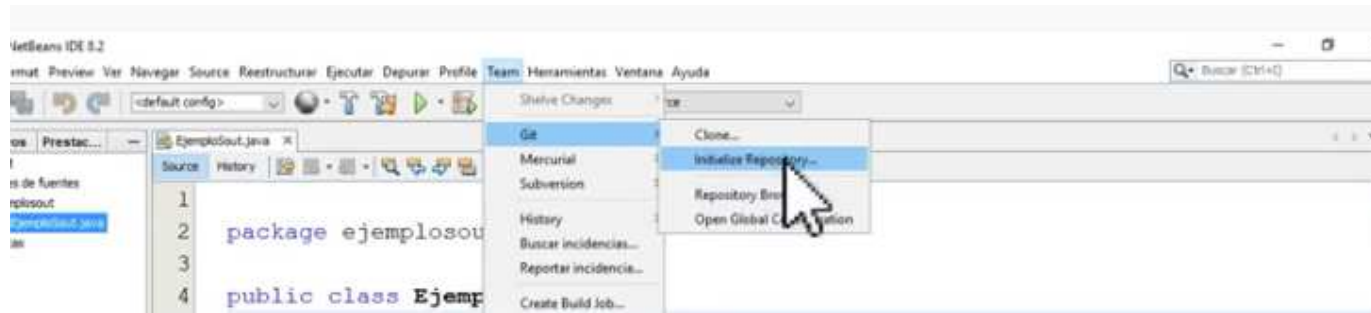
- Crear app en Java



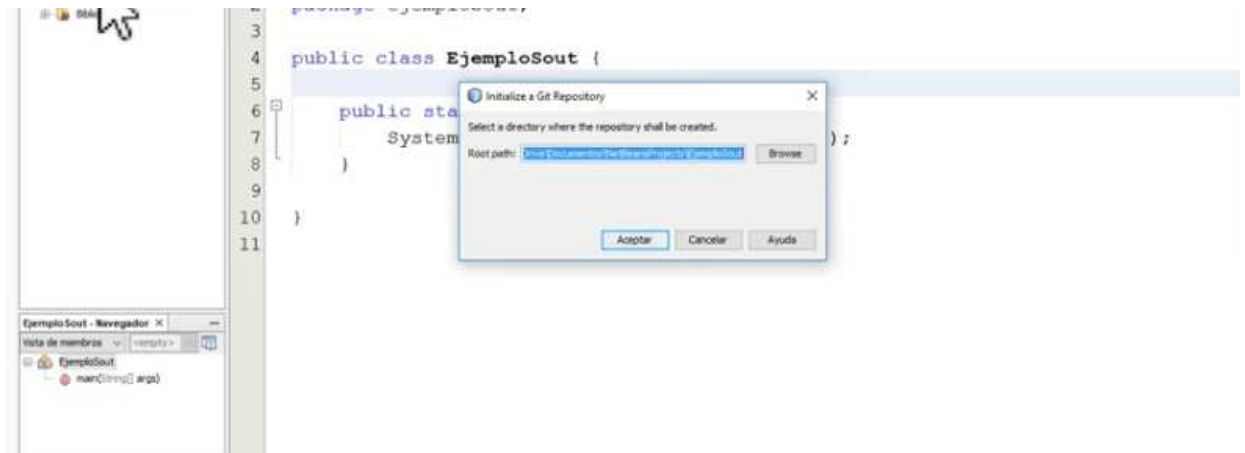
Generar .jar



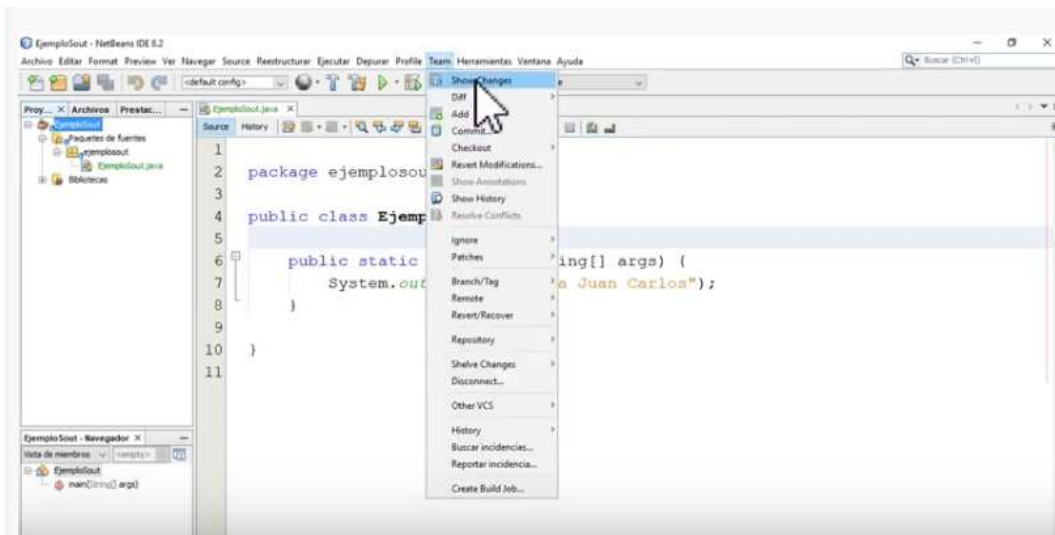
- Inicializar repositorio



Indicar la ruta correcta en la *ruta local*

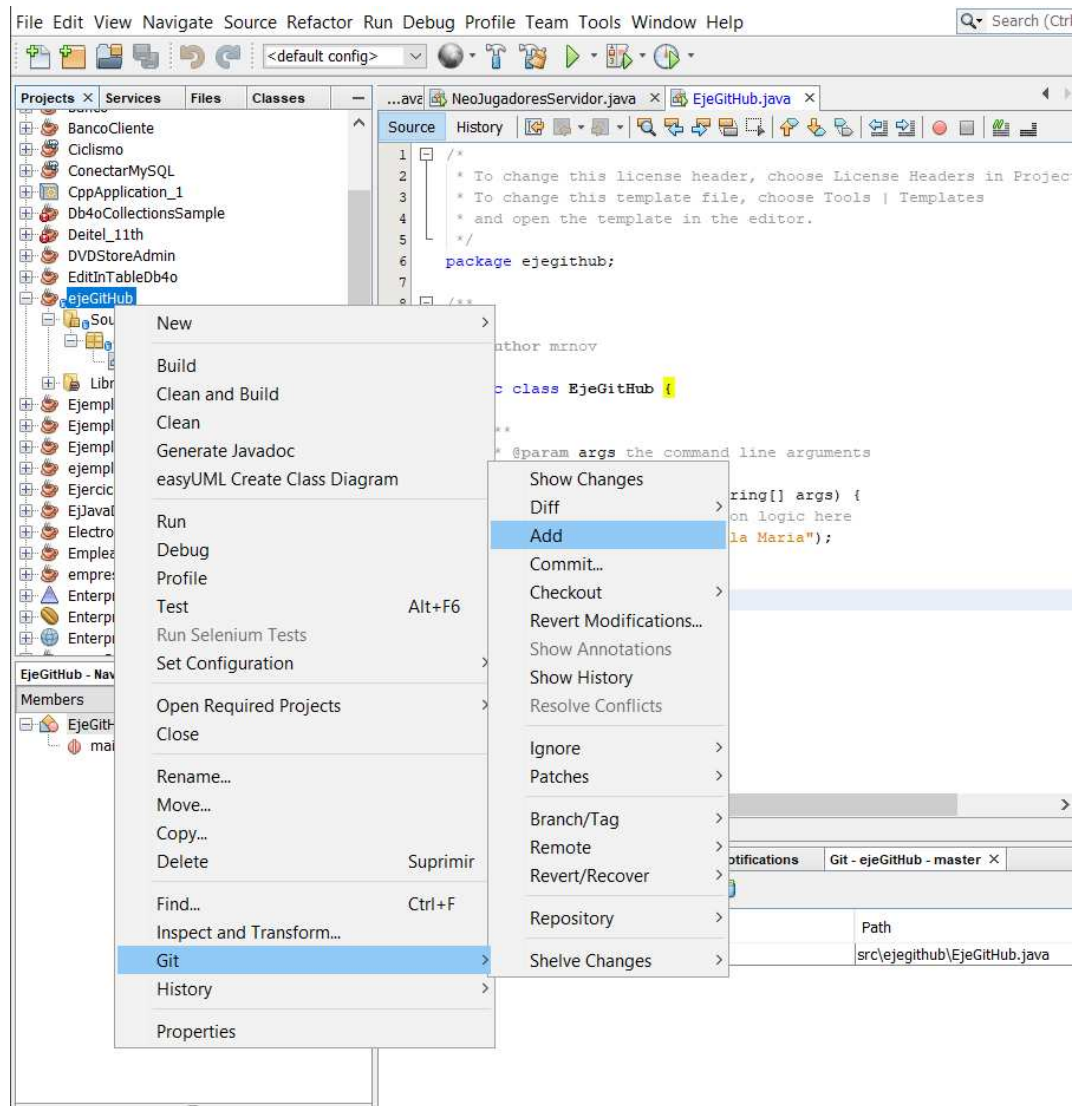


Ya se está conectado al repositorio

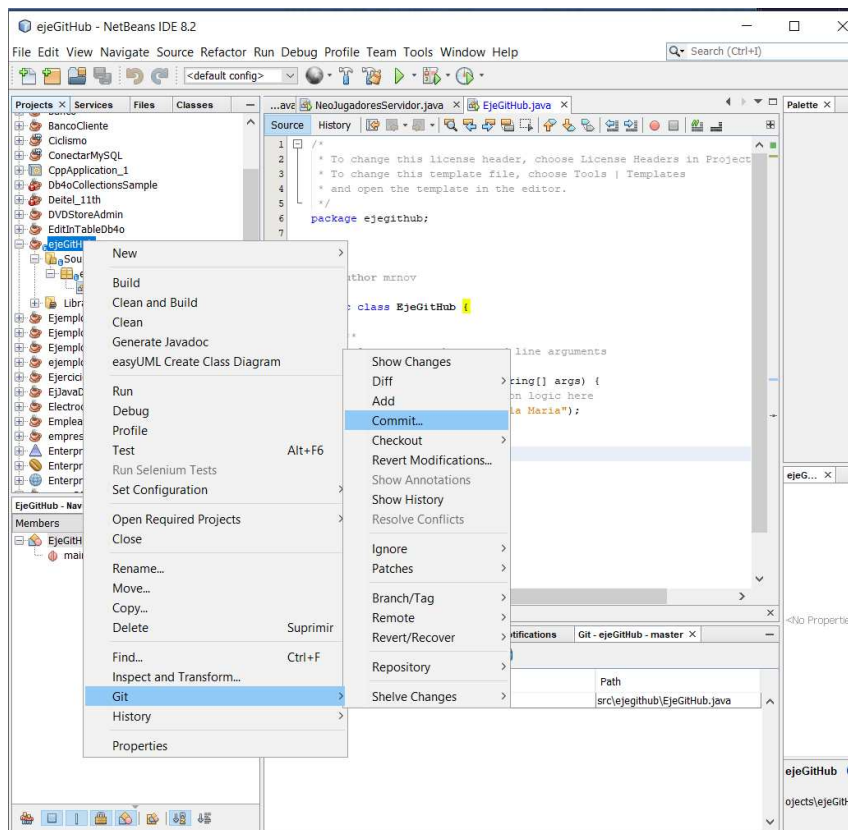


- Enganchar con GitHub online:

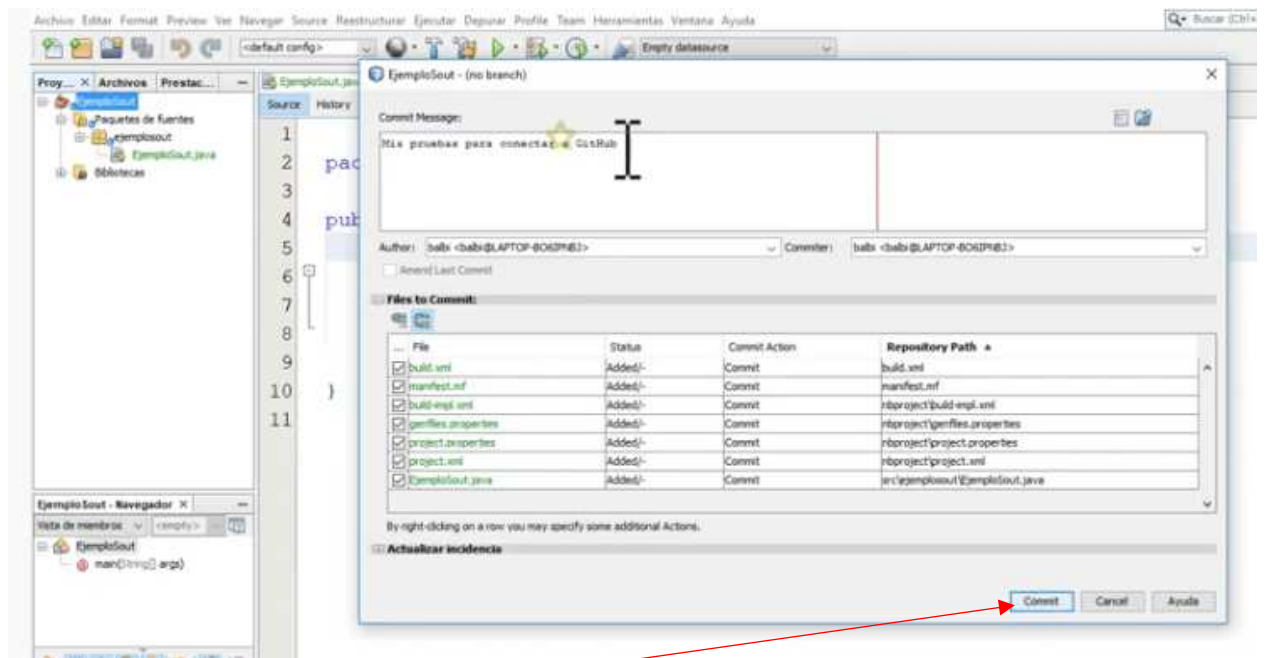
Botón derecho ratón sobre el proyecto. Opciones *Git/Add*



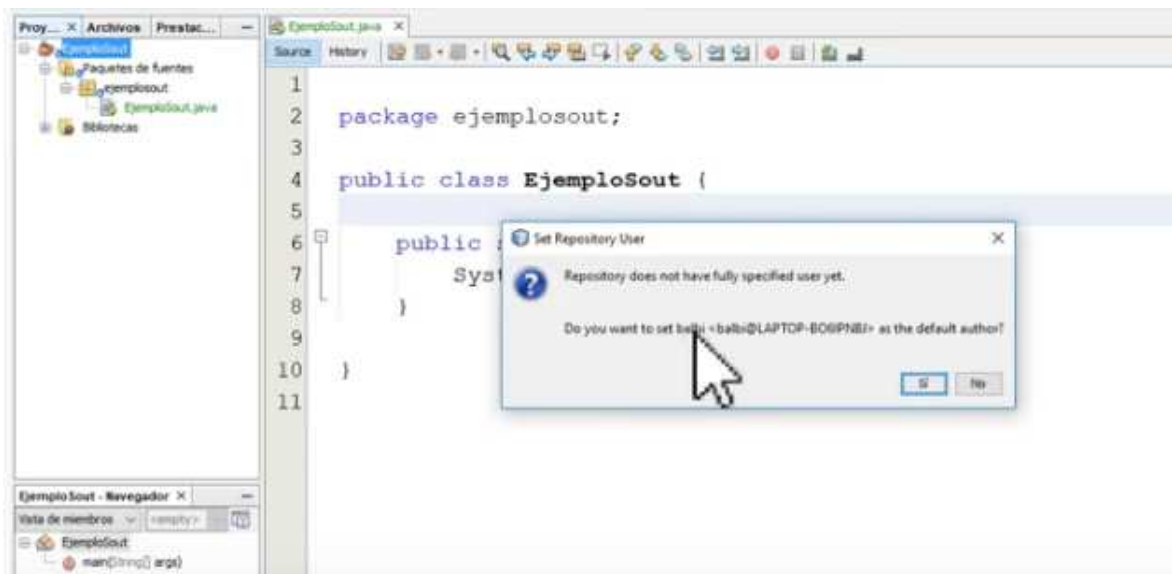
Git/Commit



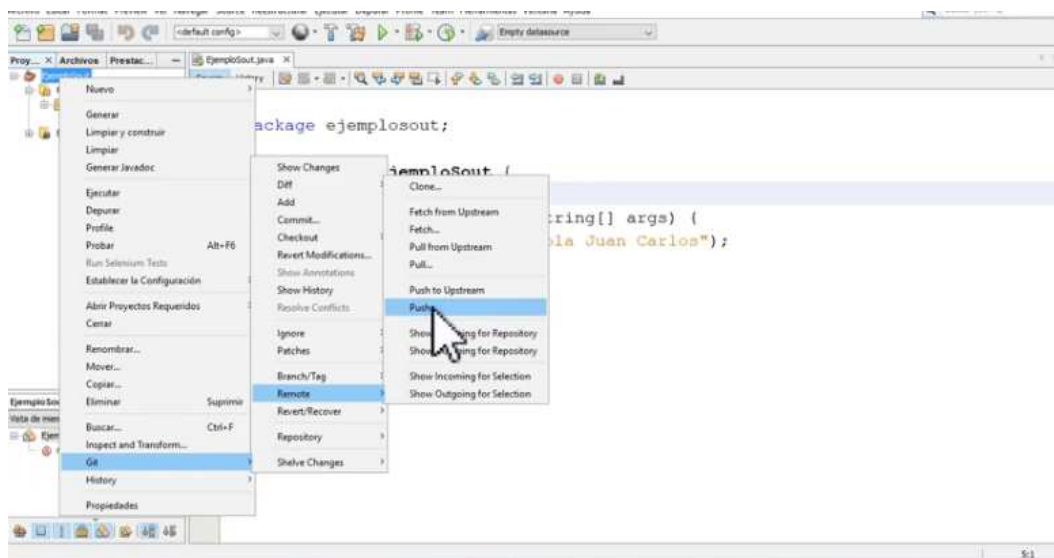
La primera subida

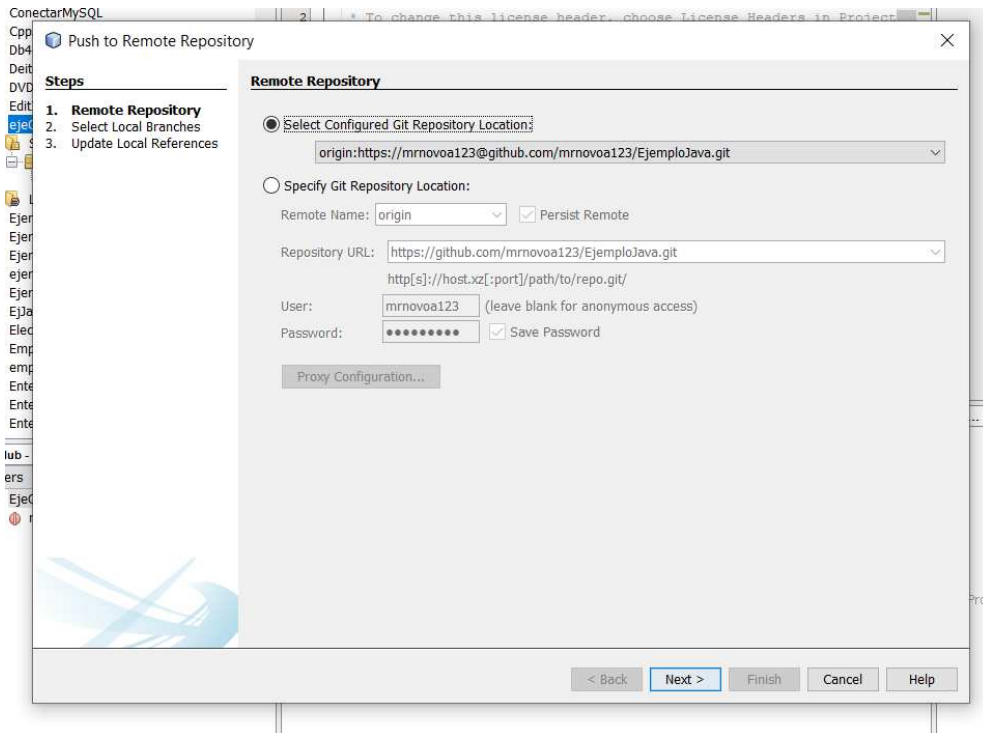


Commit- confirmar Primera subida



- Hacer efectiva subida del proyecto-Git/Remote/Push



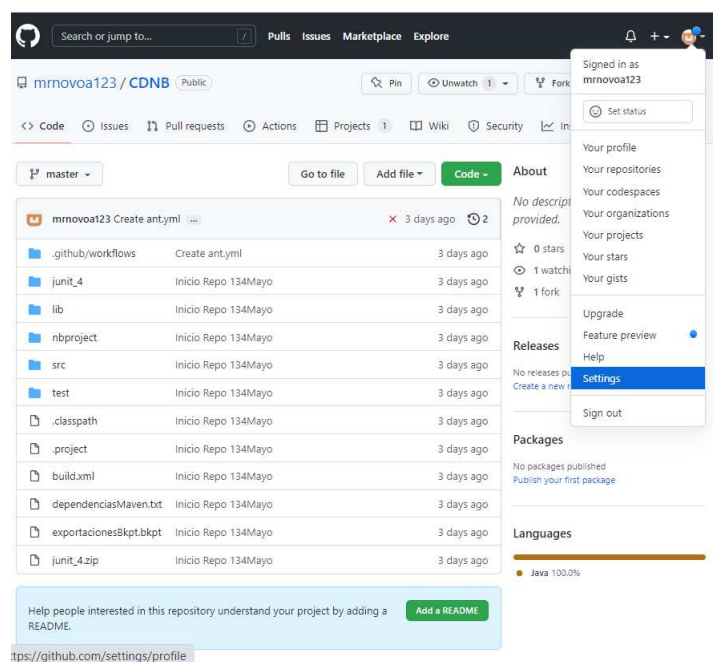


Misma url indicada al crear el repositorio en GitHub

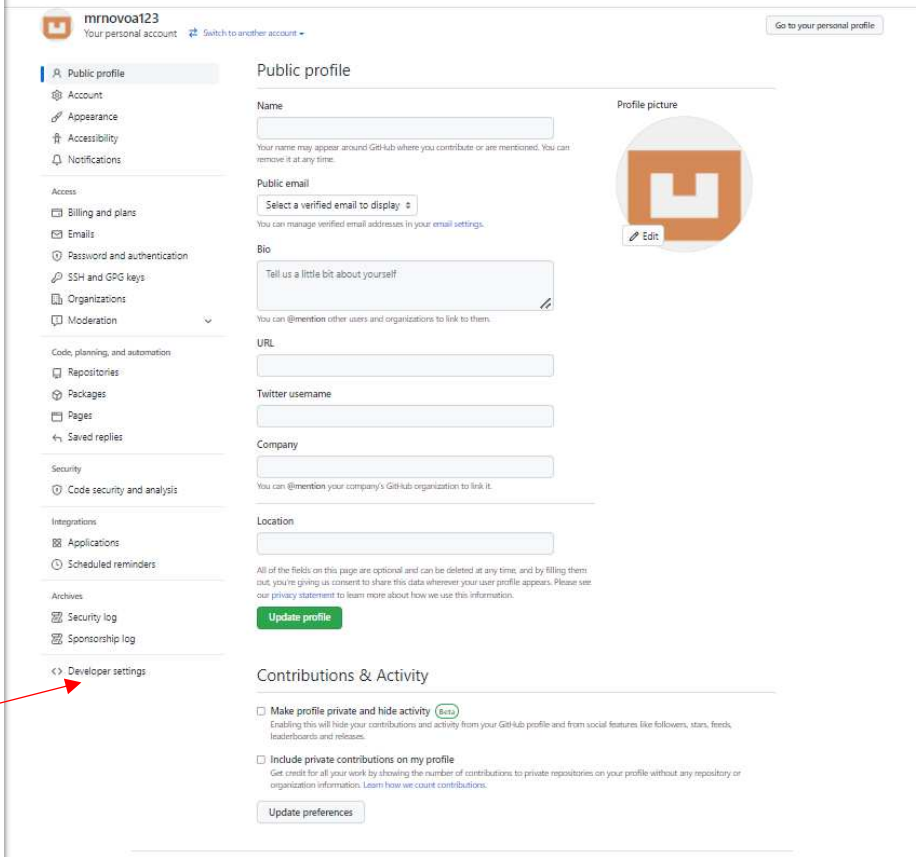
OJO. Uso de PAT (Personal Access Token)

- clonar un repositorio en *GitHub* era usando la URL que usa el protocolo *HTTPS*. Estas URLs tienen el formato:
<https://github.com/<nombre-usuario>/<nombrerepositorio>.git> y para clonarlo solamente necesitábamos identificarnos con nuestro nombre de usuario y la contraseña.

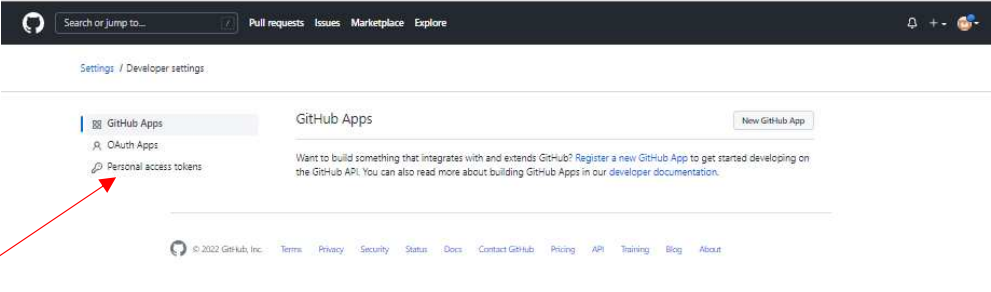
El equipo de Github anunció en julio del 2020 la intención de eliminar la autenticación de *Git* usando contraseñas, y en su lugar podemos identificarnos usando **Personal Access Token**, claves *SSH*, *GitHub App*, etc. Desde el 13 de agosto del 2021 ha dejado de estar operativo, por lo que habrá las diferentes alternativas que tenemos usando por ejemplo el sistema operativo *Windows*.



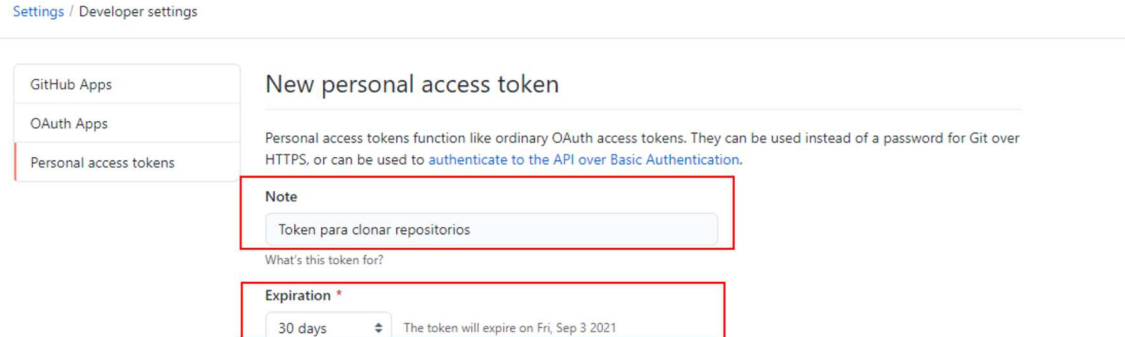
A continuación, vamos a crear un **Personal Access Token** el cuál contendrá los permisos necesarios, para ello, vamos a *Nuestro perfil > Settings > Developer settings > Personal access tokens* para generar uno nuevo:



The screenshot shows the GitHub profile page for user 'mrnovoa123'. The left sidebar contains a list of settings categories. A red arrow points to the 'Developer settings' option at the bottom of this list.



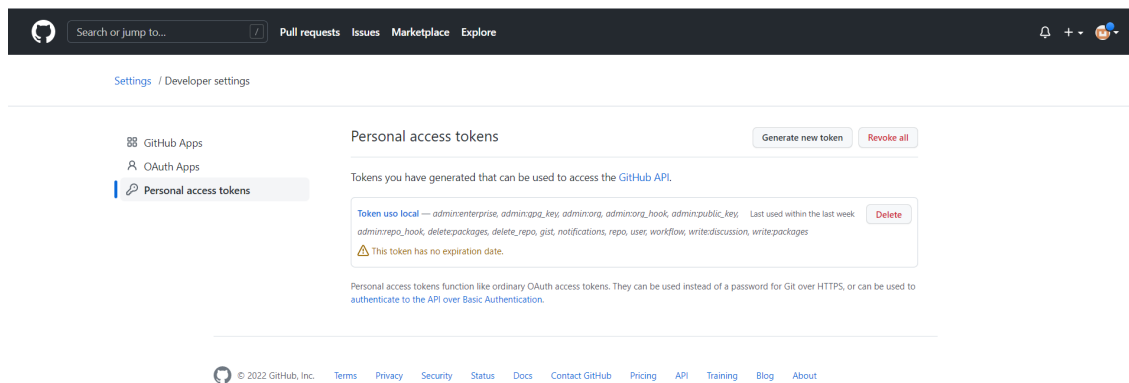
The screenshot shows the 'Developer settings' page. The left sidebar lists 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens'. A red arrow points to the 'Personal access tokens' option.



The screenshot shows the 'New personal access token' form. The left sidebar lists 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens'. The main content area has the title 'New personal access token' and a description: '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](#).' There are two red boxes highlighting the 'Note' field (containing 'Token para clonar repositorios') and the 'Expiration' field (set to '30 days', with a note that the token will expire on 'Fri, Sep 3 2021').

Se podría hacer que no expirase, pero esta opción no se recomienda por temas de seguridad, es

bueno ir renovando de vez en cuando las credenciales.



Por último, nos quedaría darle los permisos necesarios para poder trabajar con el repositorio, como hacer un clone, realizar commits y push para el desarrollo de nuestra aplicación:

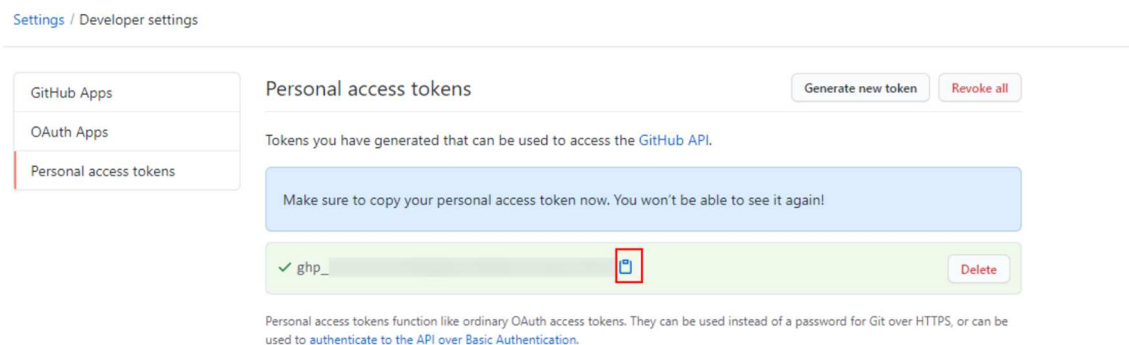
Select scopes

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

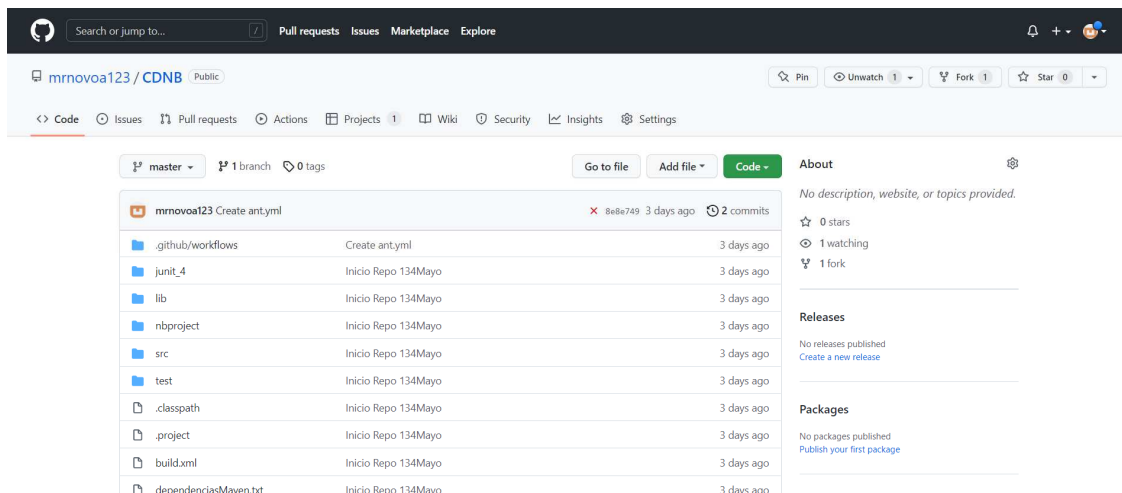
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input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

Y generamos el token pulsando el botón *Generate token* abajo del todo.

Ahora, hay que copiar el token generado y guardarlo a buen recaudo, ya que esta será la única vez que podamos verlo. Si por algún motivo lo hemos perdido, simplemente tendríamos que generar uno nuevo repitiendo el proceso:

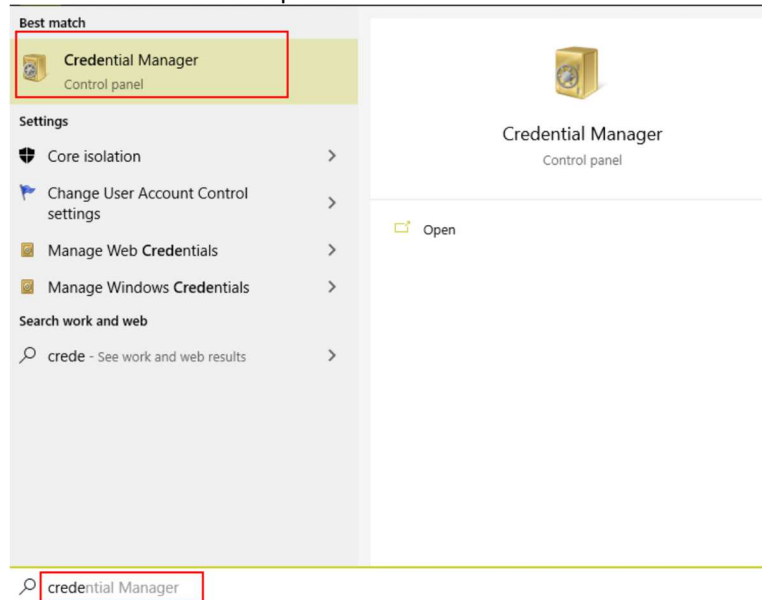


Ya estamos listos para volver a la pantalla principal del repositorio que hemos creado anteriormente, y copiar la URL usando el protocolo *HTTPS*

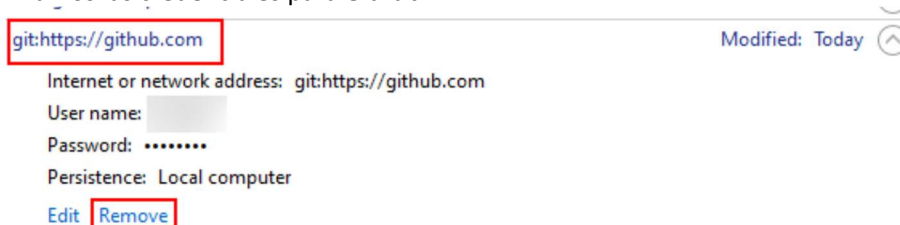


- **Gestión Credenciales Windows**

Podemos abrir cualquier terminal en nuestra máquina local donde podamos ejecutar los comandos de *git* y antes de nada nos aseguramos que borramos todas las credenciales que tenemos configurados, en *Windows* tendríamos que utilizar el *Administrador de credenciales*:



Y eliminamos las credenciales para *Github*:

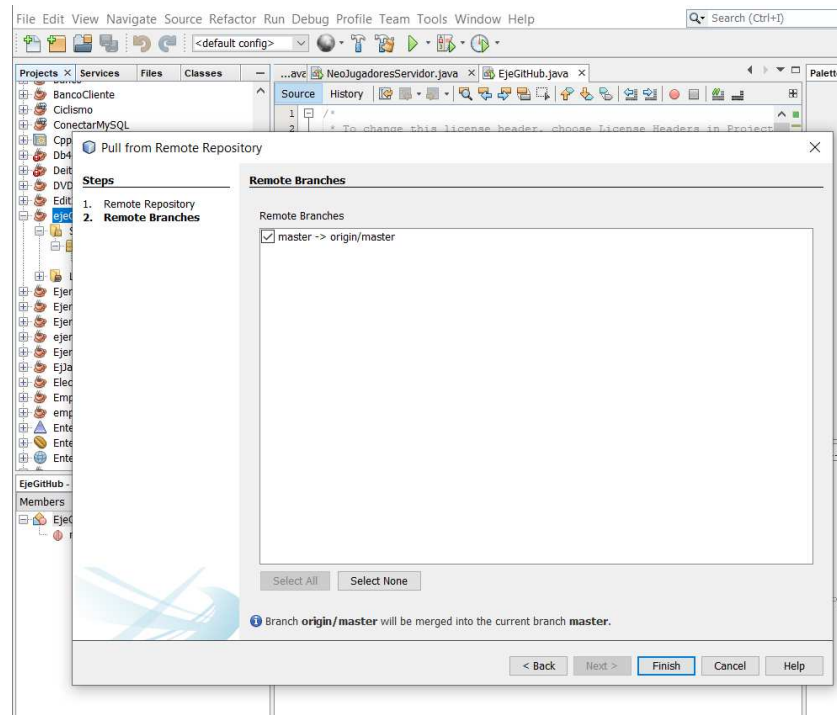


Ahora en el terminal, clonamos el repositorio usando nuestro usuario y el *Personal Access Token* como contraseña:

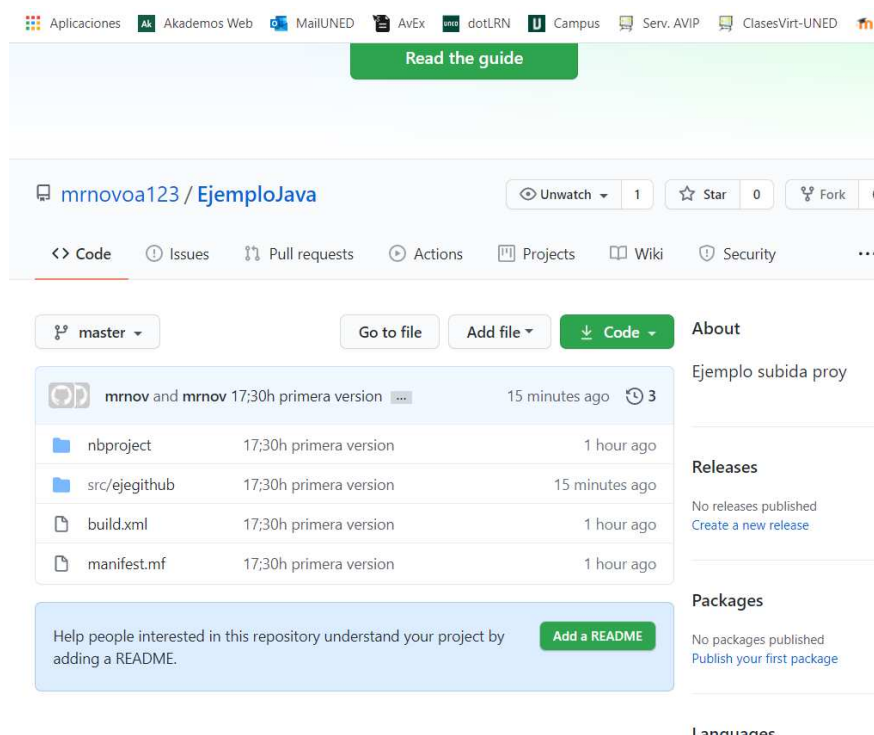
```
git clone https://github.com/<nombre-usuario>/<nombre-repositorio>.git .
> Username for 'https://github.com': <nuestro-usuario>
> Password for 'https://...@github.com': <personal-access-token>
```

Así, tenemos disponibles estas credenciales mientras el token sea válido (recuerda que cuando expiré tendrás que renovarlo desde la propia web de *GitHub*).

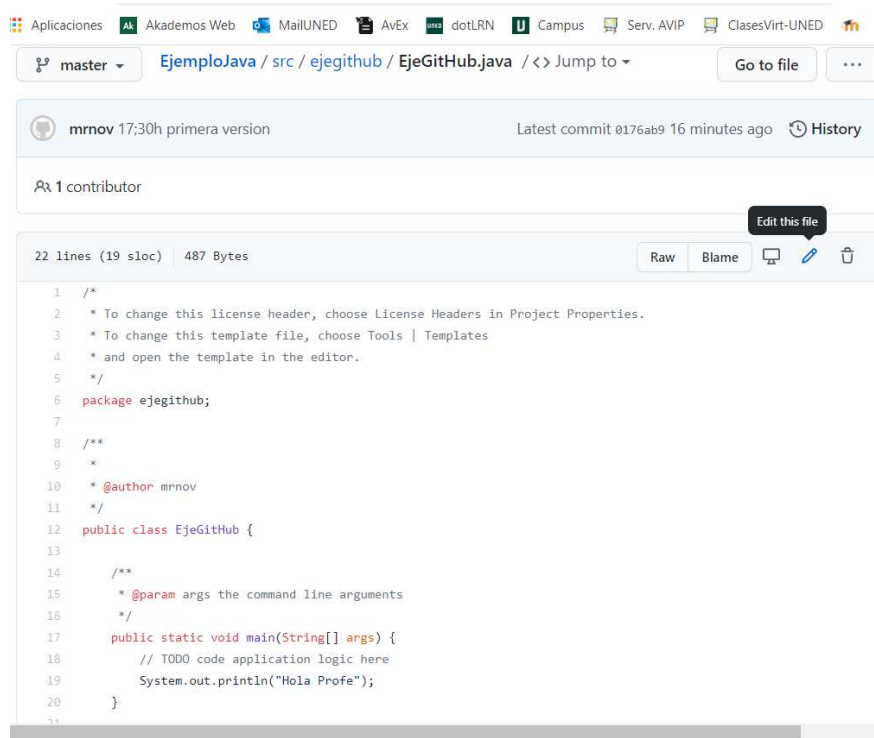
- Siguiendo con NetBeans



Comprobación online:



Quiero hacer un cambio online en el código.



Aplicaciones Ak Akademus Web MailUNED AvEx dotLRN Campus Serv. AVIP ClasesVirt-UNED

master EjemploJava / src / ejegithub / EjeGitHub.java / <> Jump to Go to file ...

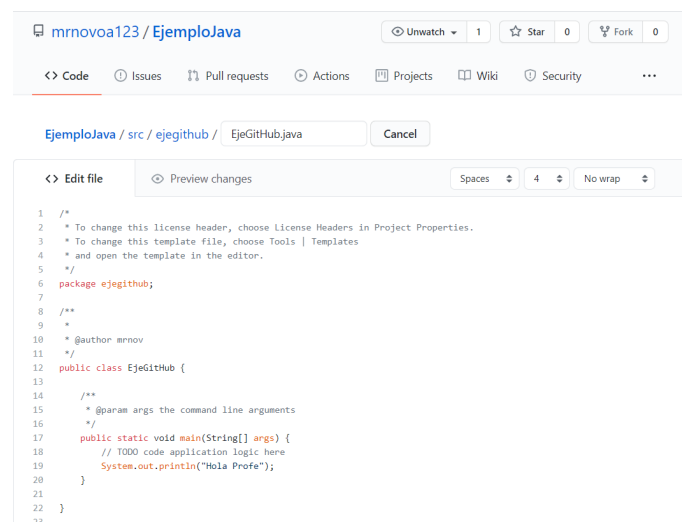
mrnov 17:30h primera version Latest commit 0176ab9 16 minutes ago History

1 contributor

22 lines (19 sloc) 487 Bytes Raw Blame Edit this file

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package ejegithub;
7
8  /**
9   *
10  * @author mrnov
11  */
12  public class EjeGitHub {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19          System.out.println("Hola Profe");
20      }
21  }
```

- Cambio el código online



mrnovoa123 / EjemploJava Unwatch 1 Star 0 Fork 0

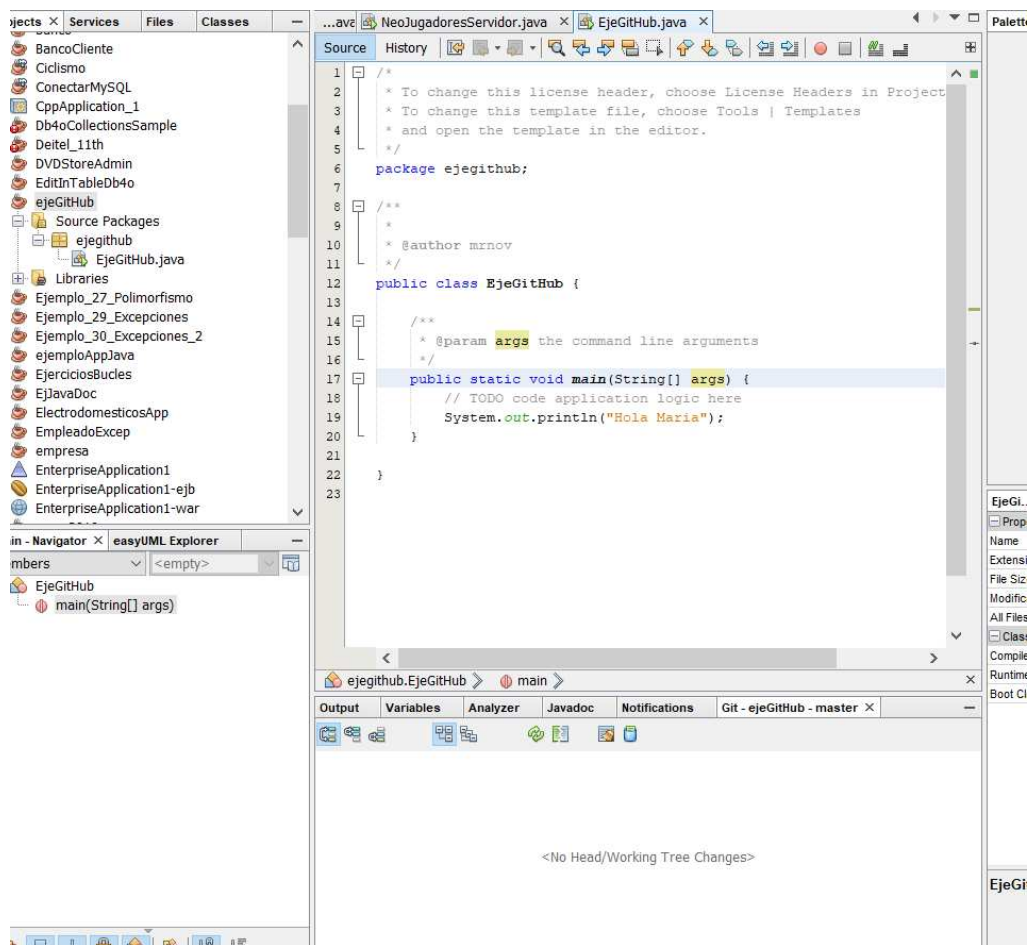
<> Code Issues Pull requests Actions Projects Wiki Security ...

EjemploJava / src / ejegithub / EjeGitHub.java Cancel

<> Edit file Preview changes Spaces 4 No wrap

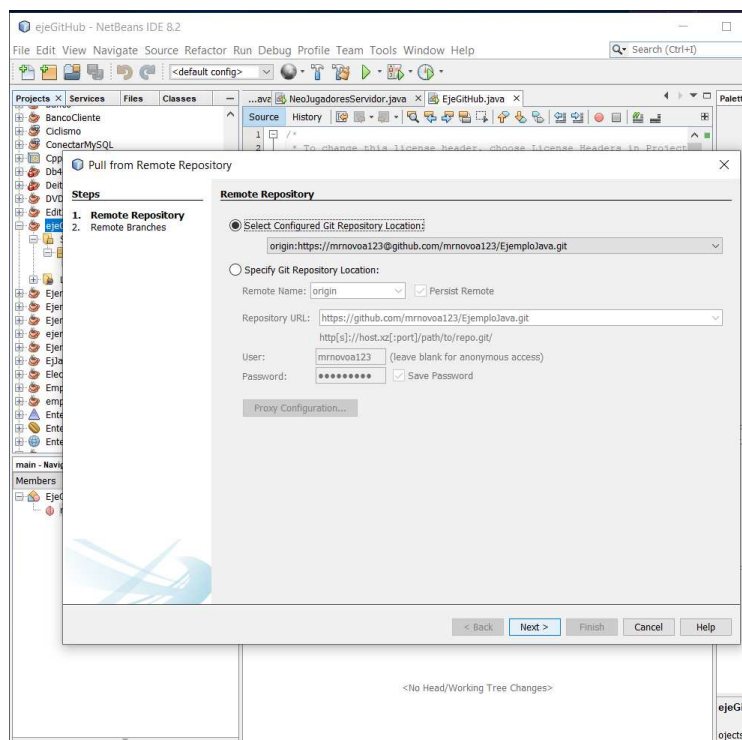
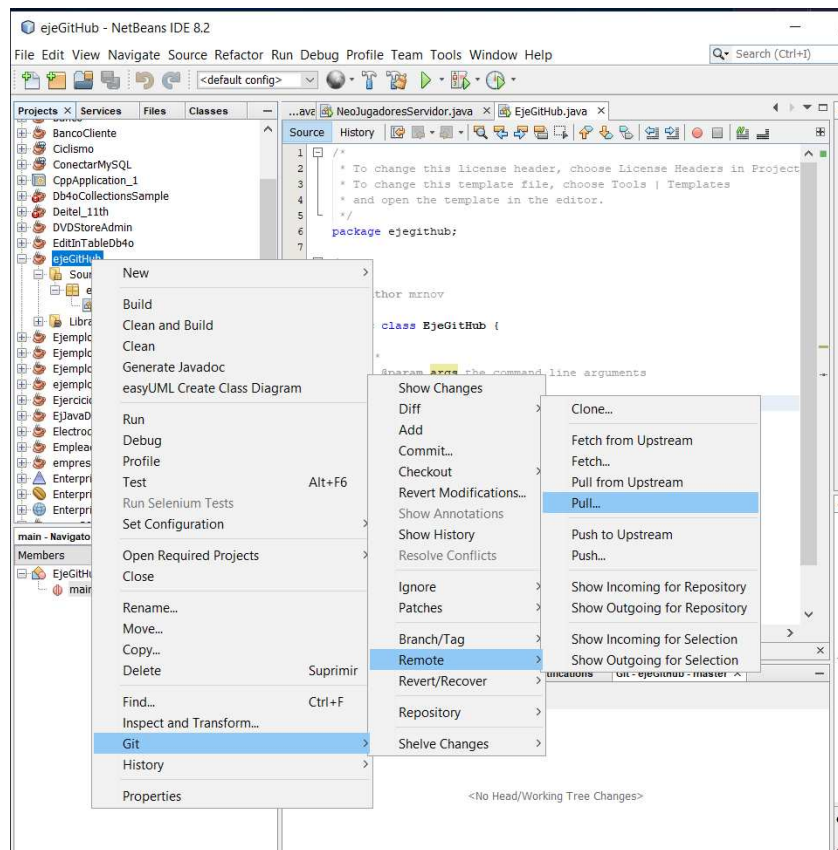
```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package ejegithub;
7
8  /**
9   *
10  * @author mrnov
11  */
12  public class EjeGitHub {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19          System.out.println("Hola Profe");
20      }
21  }
```

Antes de actualizar

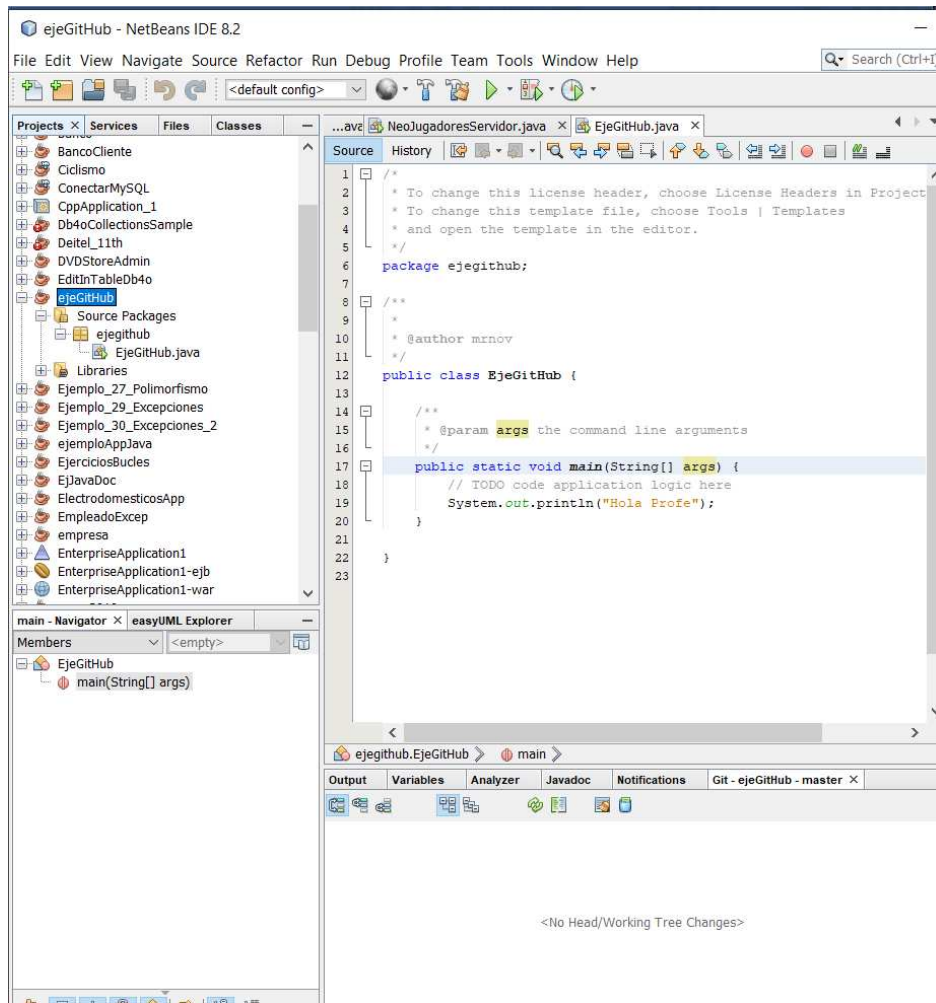


Actualizar con la versión modificada online:

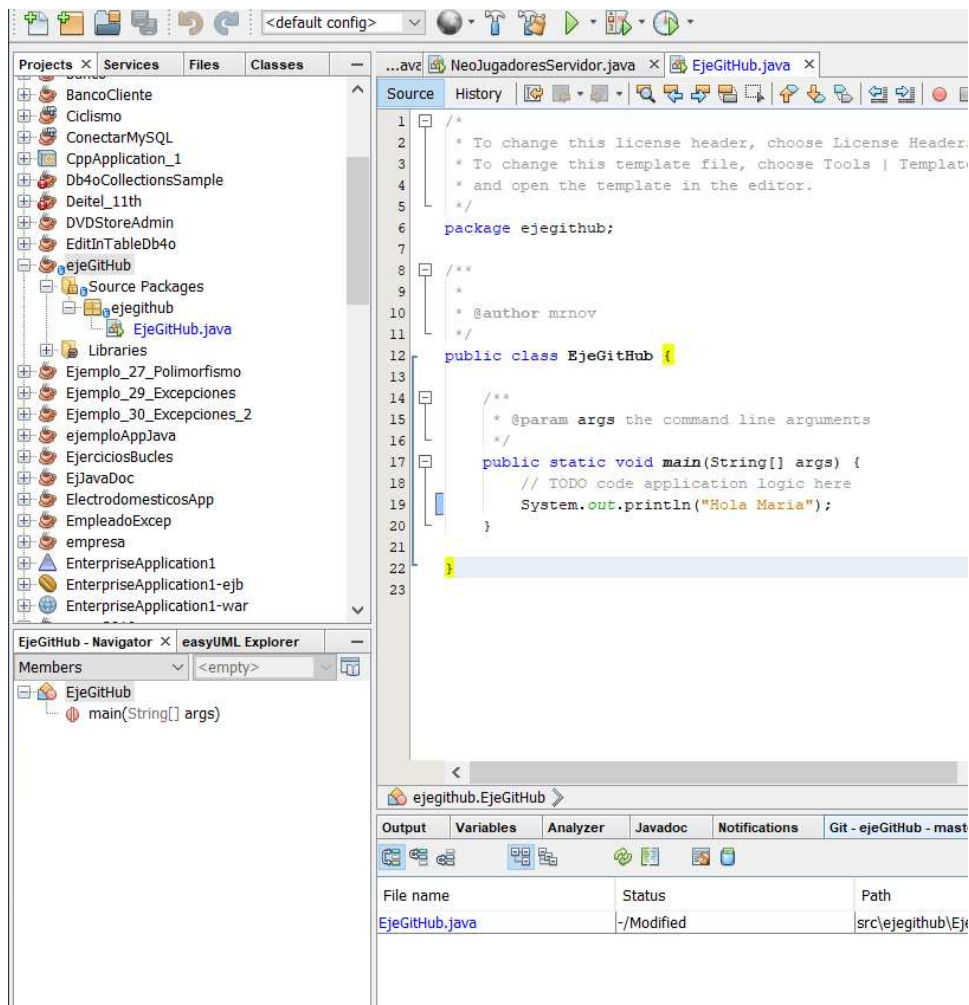
Git/Remote/**Pull**



Y aparece en el código la modificación

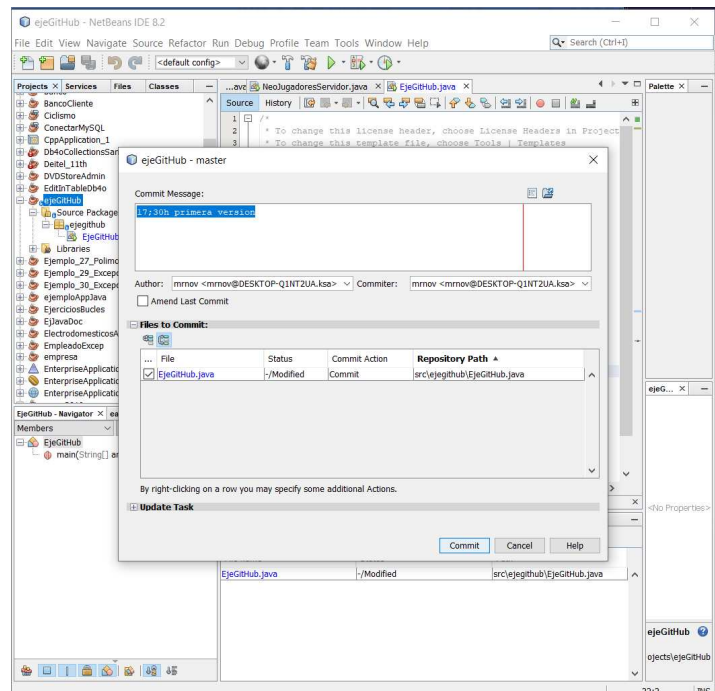


- Si hubiese alguna modificación en NetBeans, la denominación del fichero estará en **letras azules**.

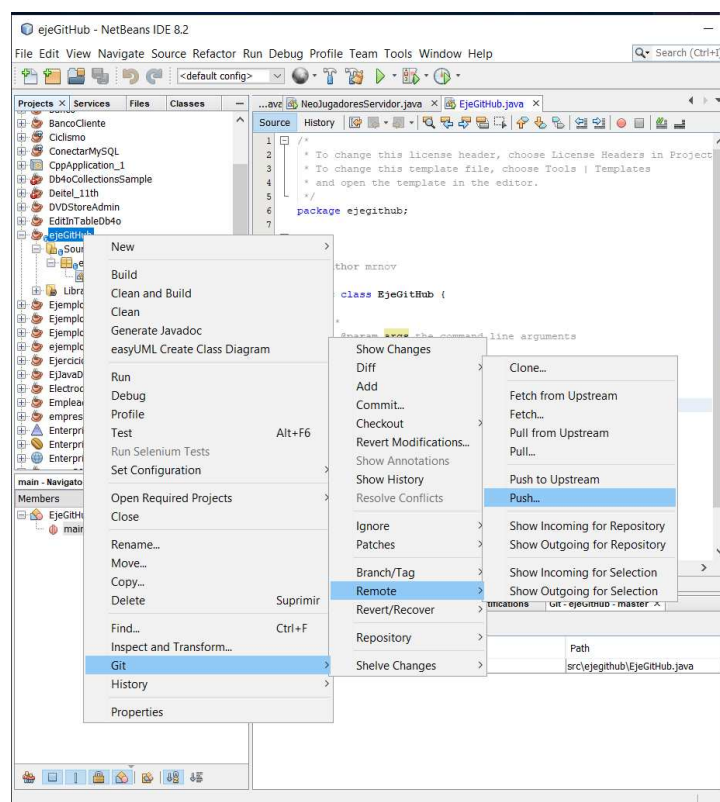


Git/Commit

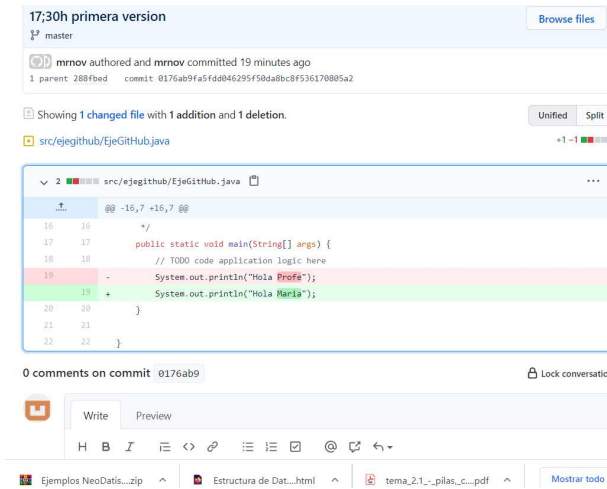
Cuando ya se han realizado más subidas con anterioridad el aspecto de la ventana sería el sigte:



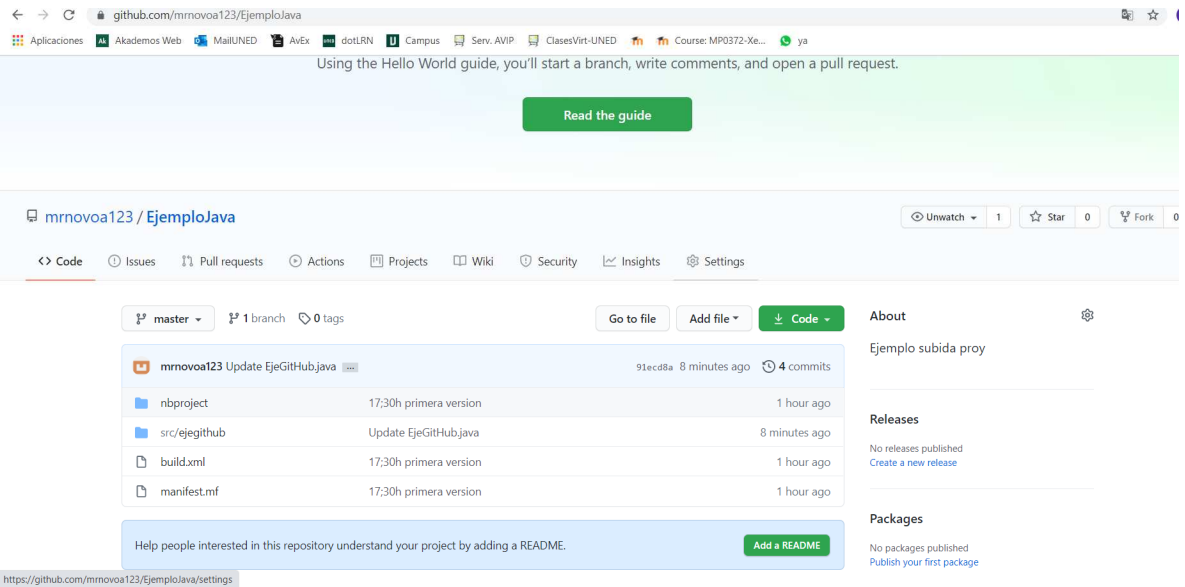
Para subir los cambios a la red, de nuevo sería:



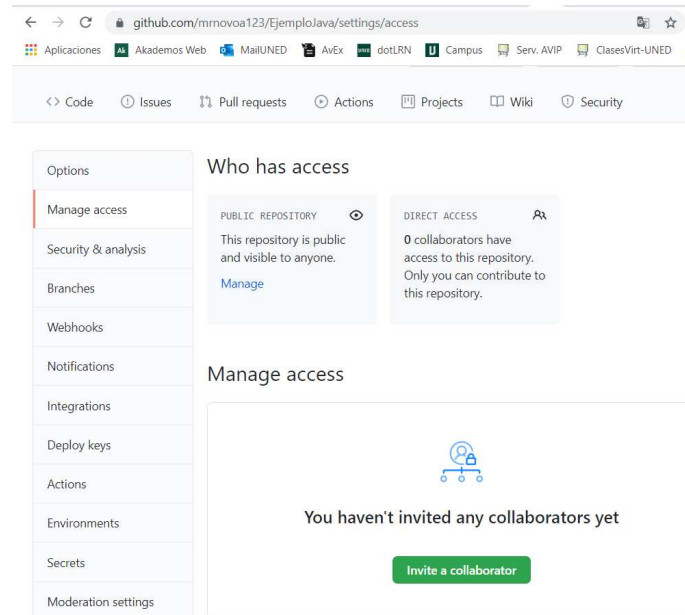
Y podemos ver online que aparecen los cambios señalados



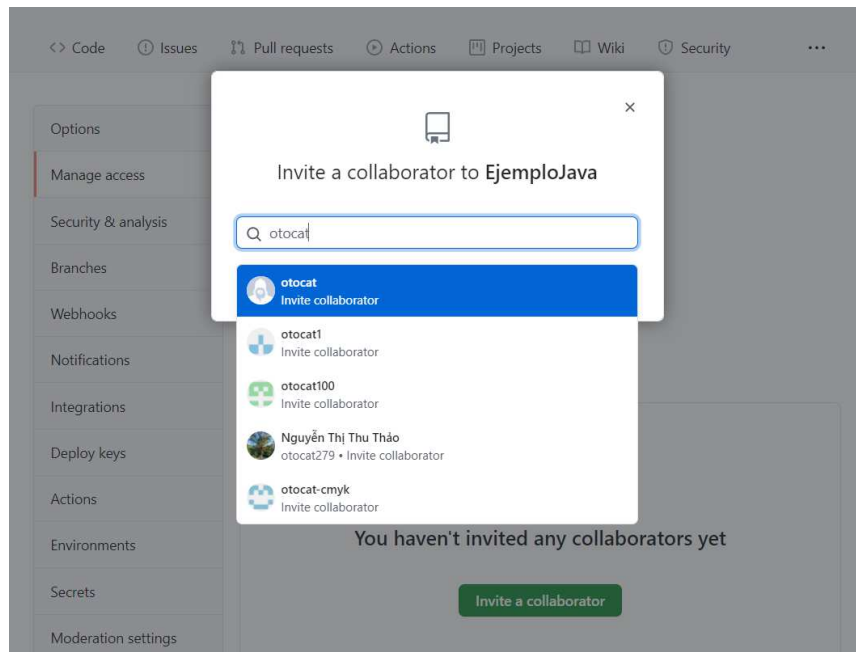
- Añadir colaboradores

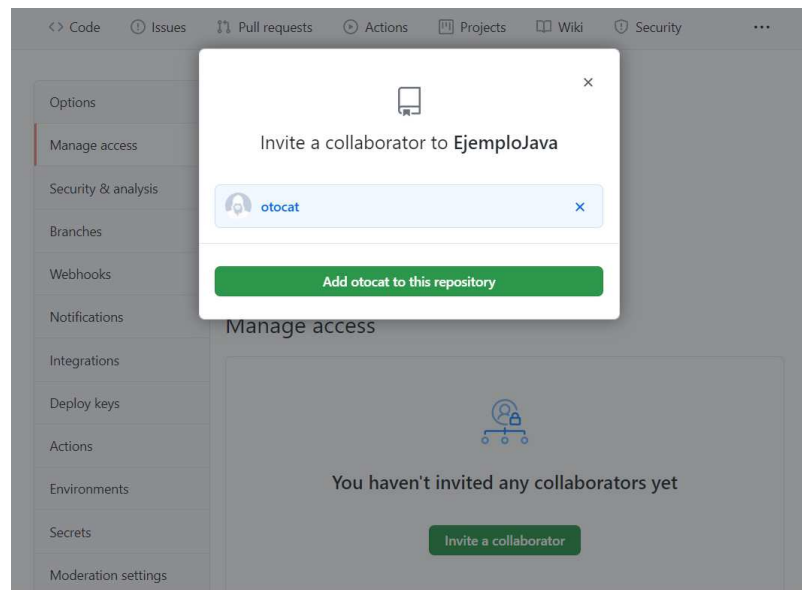


Opción **Settings/Manage access**



Se puede buscar el usuario que interese.

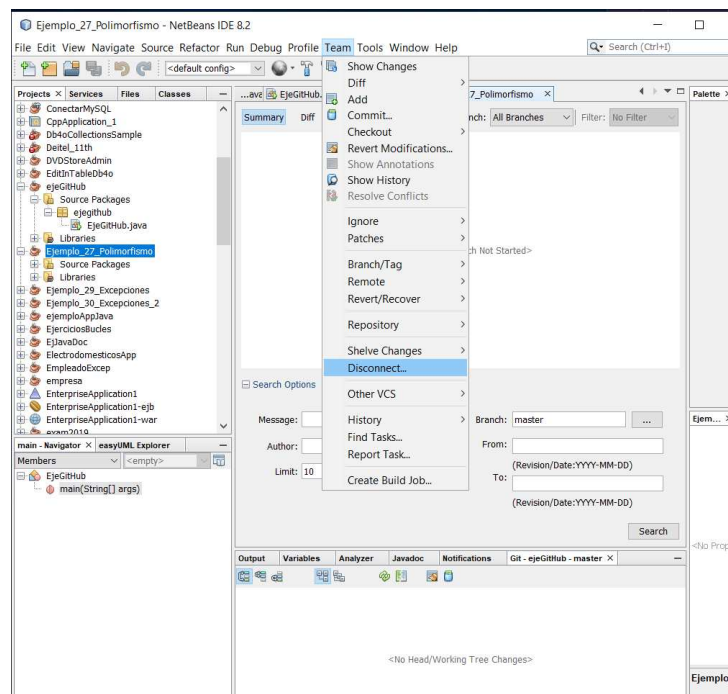


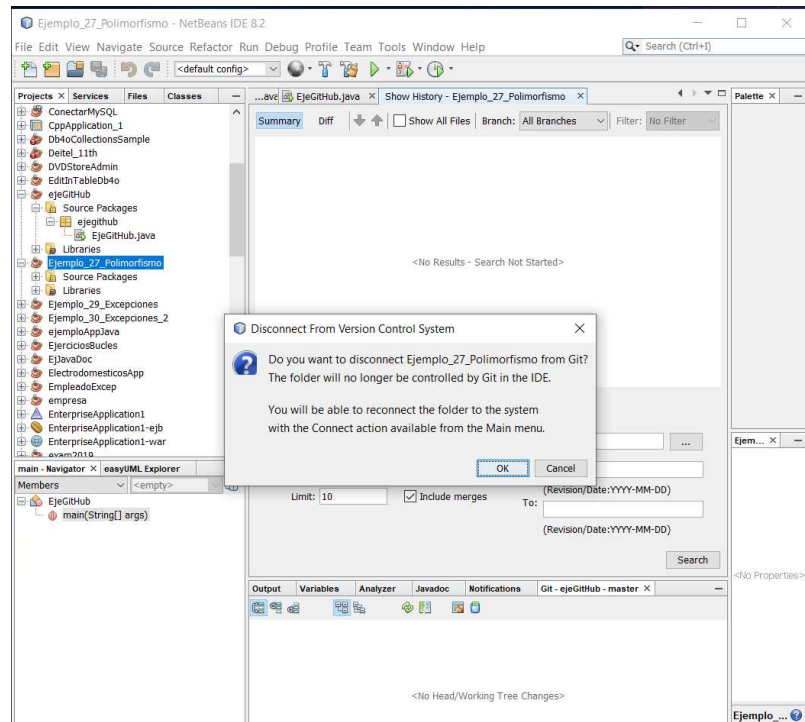


Invitar colaborador

Buscar el usuario que se quiere invitar y una vez seleccionado, este recibirá un mail invitándolos al repositorio. Cuando acepte la invitación, tendrá acceso de colaborador al repositorio

- Desconectar proyecto del repositorio





CONSOLA BASH

Empezando a trabajar con *git* en local, lo primero que hacemos es inicializar nuestro repo en local, para ello vamos al terminal:

```
git init
```

Ahora queremos guardar en la base de datos de *git* nuestro código, en Git trabajamos en dos fases, primero pasamos los ficheros a un paso que llaman "*staging*" y después de "*staging*" lo comiteamos a la base de datos en local:

Añadimos todos los ficheros a fase *stage* (esto lo hacemos con el comodín .)

```
git add .
```

Y ahora vamos a *comitearlo* en nuestra base de datos en local, añadimos también un mensaje de *commit*:

```
git commit -m "hola git"
```

Genial ya tenemos el código subido, ahora faltaría cómo subirlo a un repositorio en la nube (usaremos github como ejemplo). Ya se ha visto anteriormente en este documento cómo crear el repositorio.

- Interactuar con el repositorio

Partimos de un repositorio en local, vamos a conectarlo con el repositorio a la nube y enviar cambios...

Tenemos nuestro repositorio creado en local, vamos ahora a conectarlo con el que hemos creado en la nube, para ello nos vamos al terminal (en la carpeta en la que lo hemos creado), y le añadimos como *origin* remoto la *url* del repo en la nube.

```
git remote add origin git@....
```

Ahora que lo hemos añadido sólo tenemos que subir la rama local que tenemos al repo

Nota ver si es master o main

```
git push -u origin master
```