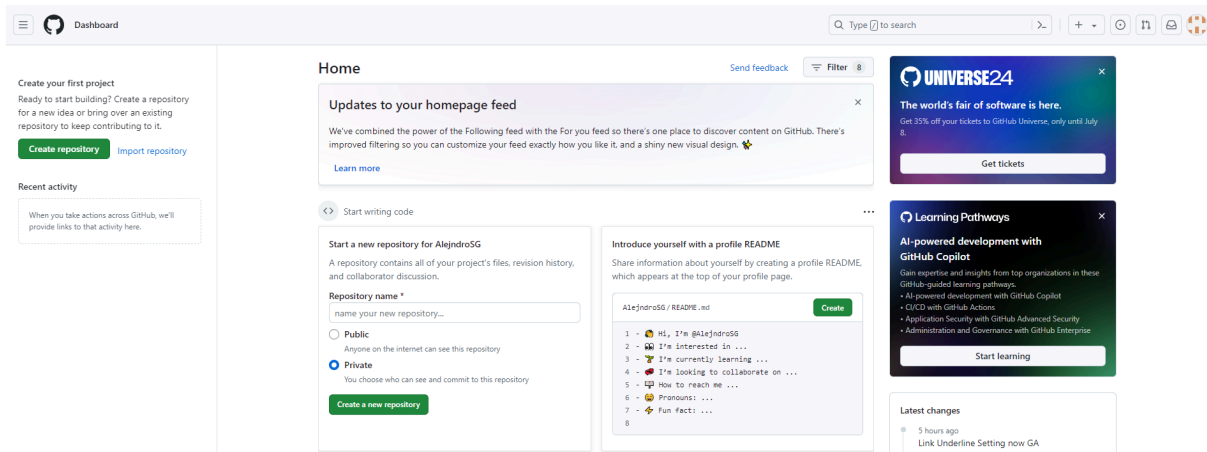


# Tema 8: Control de Versiones



## Ejercicio 1: Si no tienes cuenta en GitHub <https://github.com/> ábrete una.



No tenía cuenta, por lo que me he abierto una

## Ejercicio 2: ¿Qué tipo de programa es CVS? Haz un resumen del mismo. Realiza una comparación con Git

CVS es un Sistema Concurrente de Versiones, una forma de trabajo habitualmente utilizada para almacenar el código fuente de grandes proyectos de software. CVS almacena todas las versiones de todos los ficheros de tal forma que nada puede perderse, y su utilización por varias personas es registrada

A rasgos generales, GIT es un sistema de control de versiones distribuido y bastante más popular que CVS. Permite más flexibilidad a la hora de gestionar repositorios y tiene la capacidad de manejar operaciones atómicas. Por otro lado, CVS es un sistema de control de versiones centralizado que requiere la configuración de CVSROOT.

Además, importar fuentes existentes en CVS es más complicado en comparación con GIT y las operaciones en CVS no son atómicas.

## Ejercicio 3: Encuentra otros SCV que no sean ni Git ni CVS. Haz un breve resumen de cada uno de ellos.

1. **Mercurial:** Cada desarrollador tiene una copia completa del proyecto en su repositorio local. Mercurial es conocido por su facilidad de uso y su enfoque en la simplicidad. Es utilizado por proyectos como Openbravo
2. **Subversion:** En este sistema todos los archivos se almacenan en un repositorio central y todos los desarrolladores pueden acceder desde un servidor central. Permite trabajar en equipo y el seguimiento del código
3. **Perforce:** Es centralizado y está diseñado para manejar grandes cantidades de datos y archivos binarios. Se conoce por su velocidad y capacidad para gestionar proyectos de gran escala. Se suele usar en videojuegos y en proyectos con archivos muy grandes

#### **Ejercicio 4: ¿Qué es GitLab? Haz una comparación entre GitLab y GitHub**

GitLab es una plataforma que incluye control de versiones, seguimiento de problemas, integración continua... Tiene versiones de código abierto y versiones de pago, y tiene tanto servidor local como nube de GitLab.

Sin embargo, GitHub se centra en alojar los repositorios y facilitar la colaboración en los proyectos. Ofrece un servicio en la nube con opciones gratuitas y planes de pago para empresas y equipos.

Hablando de funcionalidades GitLab es más completa a la hora de herramientas de desarrollo software dentro de una plataforma, mientras que GitHub tiene algunas funcionalidades integradas pero hay muchas otras que vienen por servicios externos o integraciones.

#### **Ejercicio 5: ¿Qué significa el término 'DevOps'? ¿Cómo crees que se relaciona el término con los SCV?**

DevOps: Significa Development Operations, son un conjunto de prácticas que buscan integrar de manera más estrecha los equipos de desarrollo de software (dev) con los equipos de operaciones de sistemas (ops) en una organización.

Se relaciona bastante en una serie de casos:

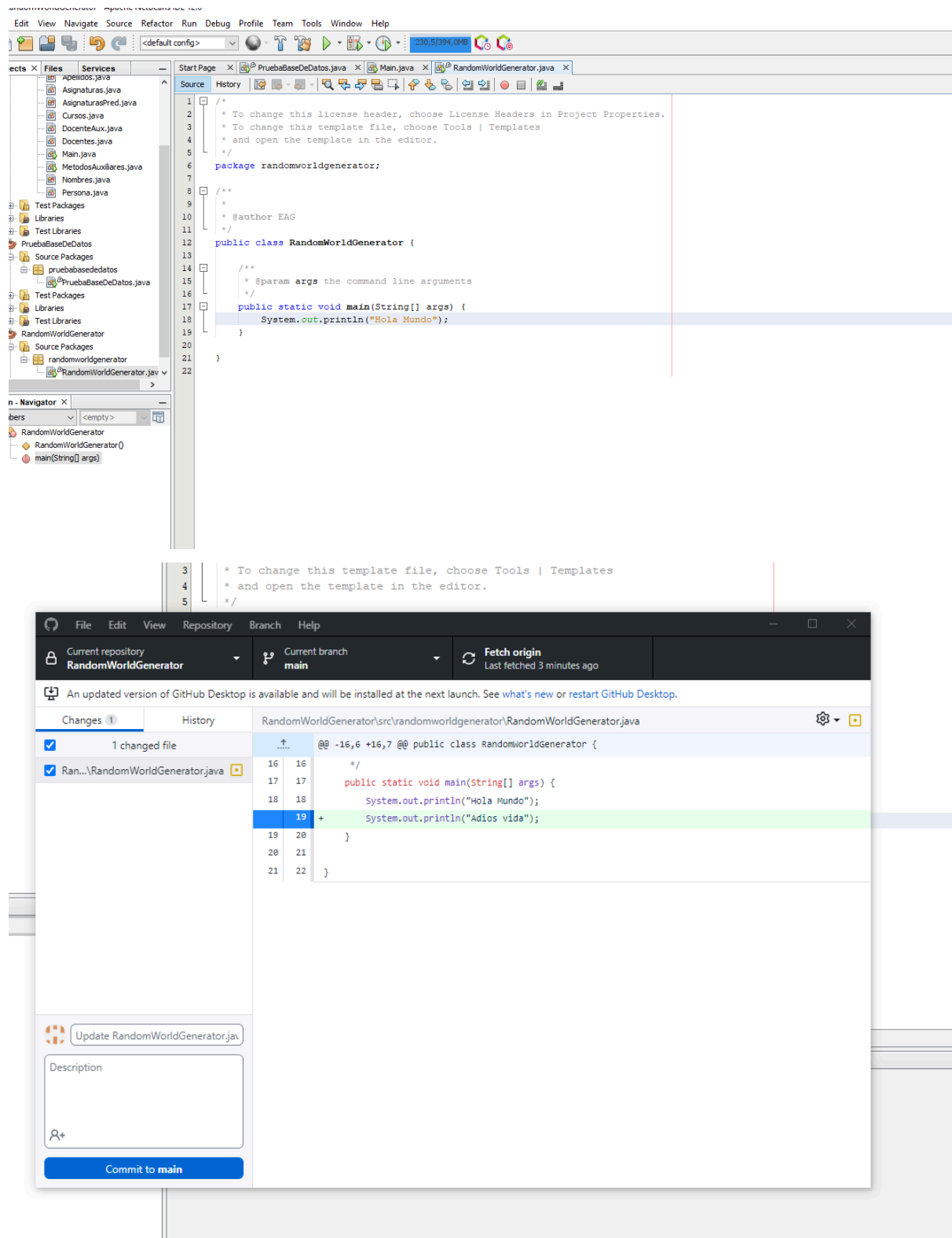
- **Colaboración y seguimiento de cambios:** Es una de los principales casos en los que están relacionados, ya que el control de versiones permite que se pueda realizar un seguimiento del proyecto en equipo y sin haber conflicto, cosa que apoyan por completo los DevOps.
- **Integración continua:** Además los DevOps apoyan que se pruebe el código con cada cambio y realizar varias pruebas diarias para comprobar el correcto funcionamiento de este. Esto es muy sencillo gracias a los SCV

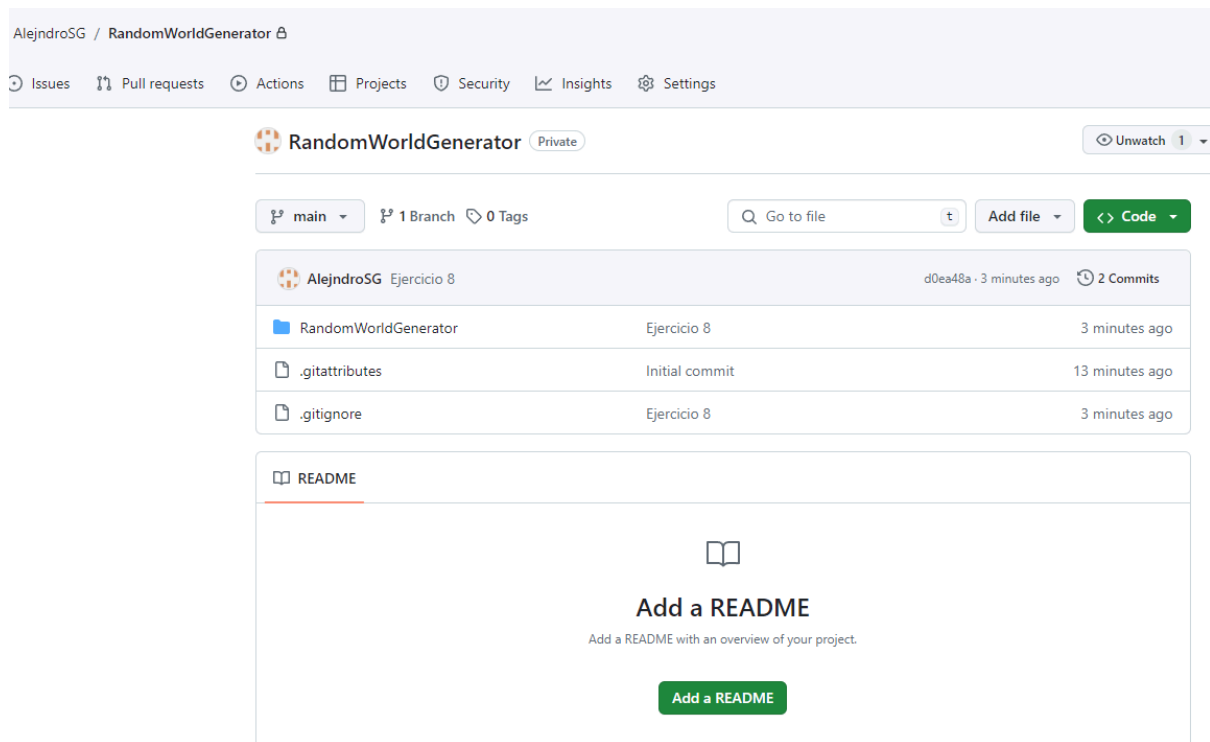
Básicamente los SCV son fundamentales en la estructura de los DevOps, ya que proporcionan lo necesario para facilitar la colaboración, la continuidad, la automatización y el correcto funcionamiento del software de alta calidad de manera rápida y eficiente

The screenshot shows the 'Options' dialog box in GitHub Desktop. The 'Accounts' tab is active, displaying the user 'AlejandroSG' (@AlejandroSG) is signed in to GitHub.com. A 'Sign out' button is present. Below this, the 'GitHub Enterprise' section is visible, with a 'Sign in' button. The background shows the GitHub Desktop interface with a 'Let's get started' tutorial.

The screenshot shows a web browser window with the URL `github.com/AlejandroSG/RandomWorldGenerator`. The page displays the GitHub repository interface for the project 'RandomWorldGenerator' by user 'AlejandroSG'. The repository is marked as 'Private'. Key statistics shown are 1 commit, 0 stars, and 0 forks. A list of commits is visible, with the most recent being 'Initial commit' by 'AlejandroSG' 1 minute ago. Below the commits, there is a section for adding a README file, which includes a large text area and a button labeled 'Add a README'. The right sidebar contains sections for 'About' (no description provided), 'Activity' (0 stars, 1 watching, 0 forks), and 'Releases' (no releases published, with a link to 'Create a new release').

**Ejercicio 8: Utilizando Netbeans, crea un proyecto Java para programar el proyecto RandomWorldGenerator, un potente generador procedural de mapas para un juego estilo Minecraft... Evidentemente solo debes crear el proyecto con ese nombre, no hace falta que lo programes. De momento. Eso sí, haz al menos un 'Hola Mundo' que podamos comprobar que el código se está subiendo.**

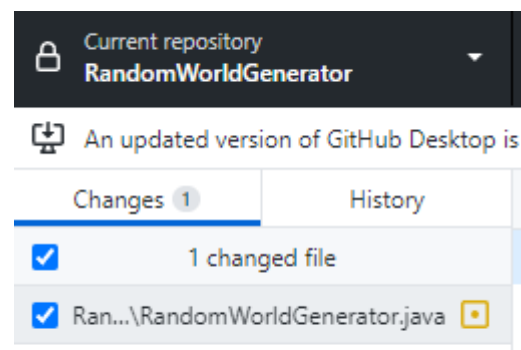




## Ejercicio 9: Comprueba que en local NetBeans ha generado los ficheros del proyecto dentro de la carpeta. ¿Están subidos a la web? ¿Por qué?

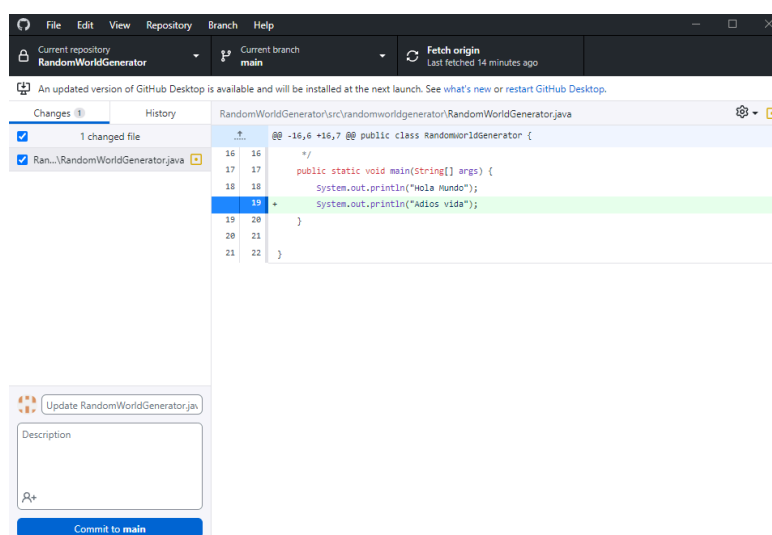
Se verían más archivos pero a la primera versión le hice commit y push sin querer, pero este es el repositorio local.

No, no está subido a la web ya que no he hecho un commit y un push, por lo que ahora mismo solo están en local, y sin los cambios guardados

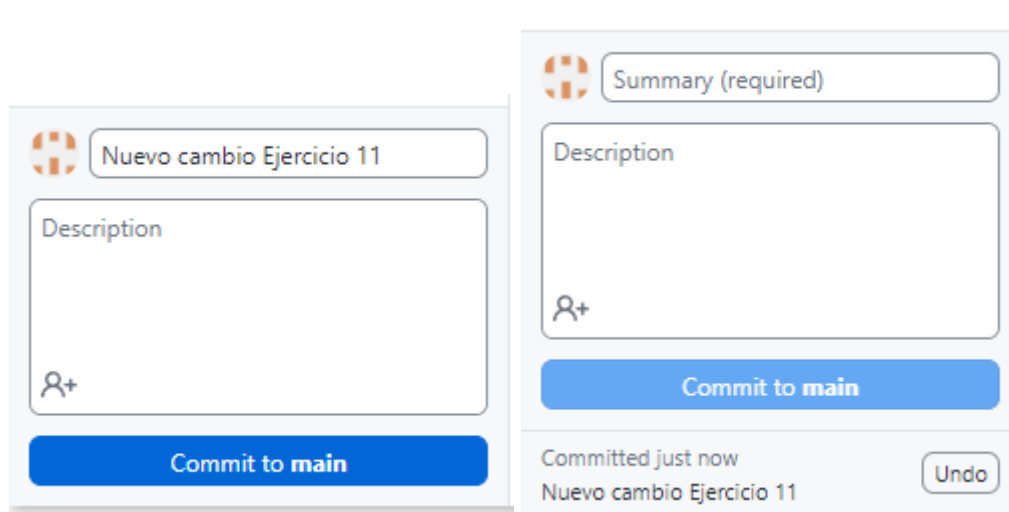


## Ejercicio 10: Abre GitHub Desktop ¿Qué te está mostrando?

Me está mostrando los cambios que he hecho en el archivo, y todos los archivos que están en mi repositorio local



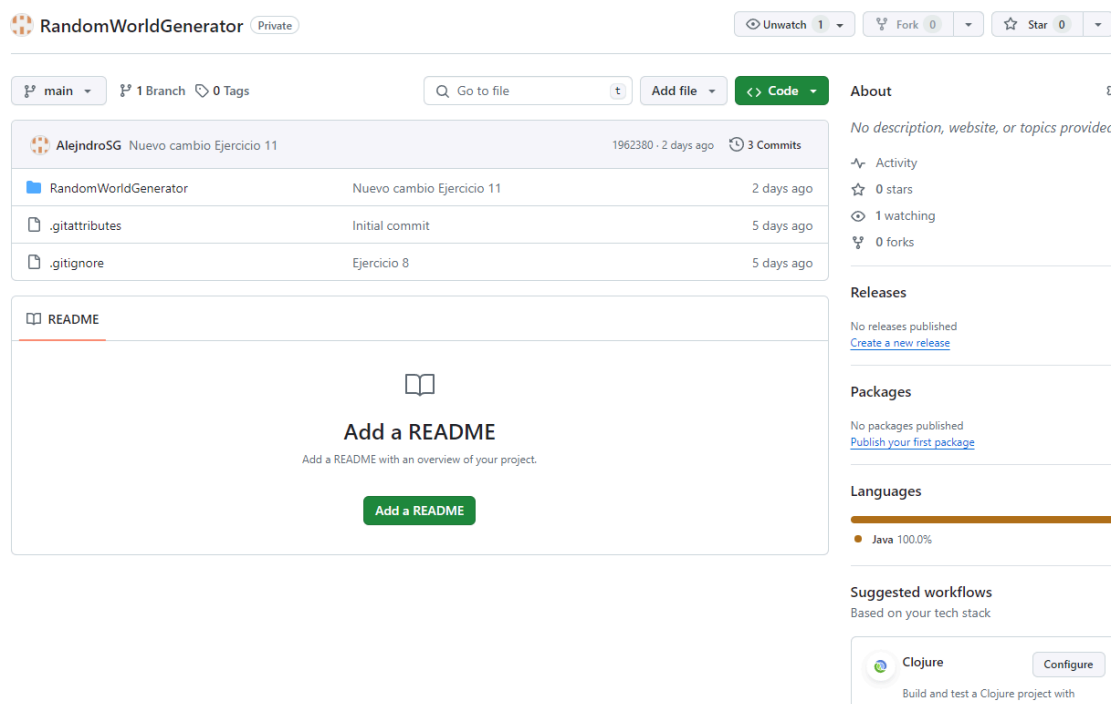
## Ejercicio 11: Haz un commit. ¿Se ha subido el fichero a la web? ¿Por qué?



No, no se ha subido a la web ya que aún no he hecho el push

## Ejercicio 12: Realiza un push. Comprueba que el código se ha subido a la web.

Hice un push y ya sale directamente mi repositorio en la web



## Ejercicio 13: Cuando llegues a casa bájtate el repositorio en tu PC ¿qué comando debes realizar, clone o pull? ¿Por qué?

Necesitas el clone, ya que no tenemos el repositorio en nuestro repositorio local de la casa. Si lo tuviéramos en nuestra casa, simplemente tendríamos que hacer un pull.

**Ejercicio 14: Una vez hayas terminado de realizar cambios sobre el código ¿qué comandos se deben realizar?**

Una vez hayas terminado los cambios, si queremos llevar una buena práctica, debemos hacer un commit para guardar los cambios y, antes de hacer el push, primero debemos hacer un pull y luego un push (no es necesario si estamos trabajando solo nosotros)

**Ejercicio 15: Define con tus propias palabras los términos: Git, GitHub, repositorio, 'clone', 'commit', 'push' y 'pull'.**

- **Git:** Es un lenguaje de control de versiones muy común y muy usado que sirve, como su nombre indica, para controlar las versiones de los archivos que tenemos en nuestros repositorios, entre otras cosas
- **GitHub:** Es una web en la que puedes subir tus repositorios y tus proyectos, además de poder hacer proyectos en conjunto con varias personas. Su uso principal es compartir códigos y poder tenerlos almacenados en una nube.
- **Repositorio:** Un repositorio no es más que un espacio en el que tenemos guardados nuestros proyectos y todo lo que queramos tener
- **Clone:** Es el comando que se usa para "clonar" o tener una copia de nuestro repositorio que está en la nube en nuestro repositorio local.
- **Commit:** El commit es un comando que se usa para confirmar los cambios realizados, y se requiere si o si un mensaje para poder guardarlo
- **Push:** Este comando permite que subas a tu repositorio en la nube los cambios realizados o lo que tienes en tu repositorio local
- **Pull:** Se usa para poder traer los datos que hay en tu repositorio en la nube, ya que así los actualizas con tu repositorio local

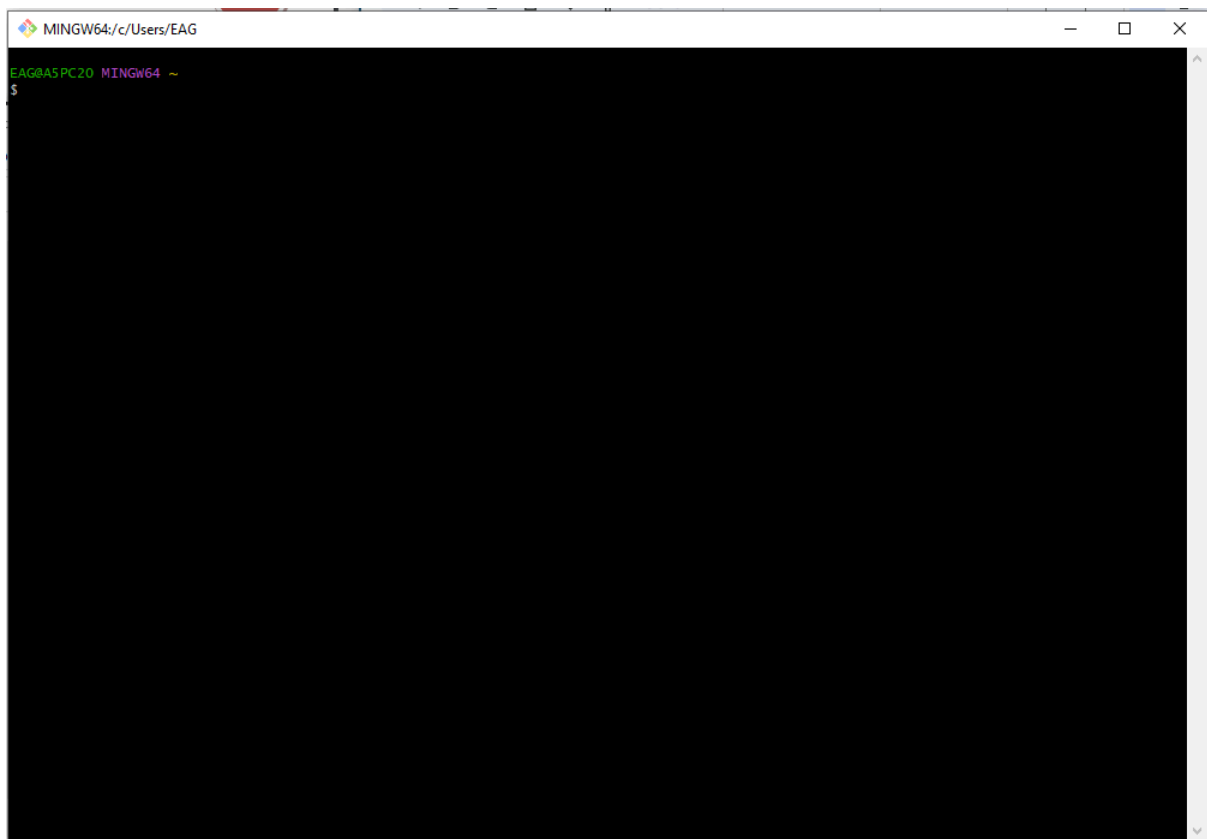
**Ejercicio 16: A partir de lo visto en tu clase, escribe un breve texto en el que intentes convencer a un amigo que está empezando a programar para que use Git. Explícale cuáles son las ventajas de usar Git.**

Se lo recomendaría a cualquier persona que está empezando a programar y, sobre todo, a trabajar en equipo. Git da muchas ventajas, pero las ventajas principales es que puedes trabajar con los compañeros simultáneamente, y que puedes guardar tus proyectos sin peligro de pérdida, además de introducirte en una bonita comunidad de programadores en la que todos se ayudan con todos.

Yo personalmente no sabía que era tan bueno, pero desde que lo uso la verdad que veo la comodidad y todas las ventajas que ofrece, y eso que no uso todas las funcionalidades.



## Ejercicio 17: Comprueba que Git Bash está instalada en el PC de clase.



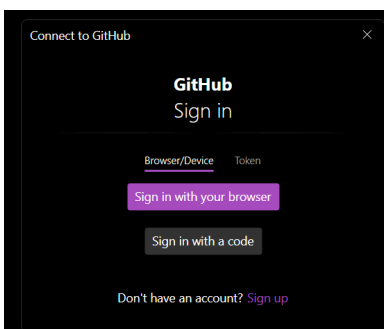
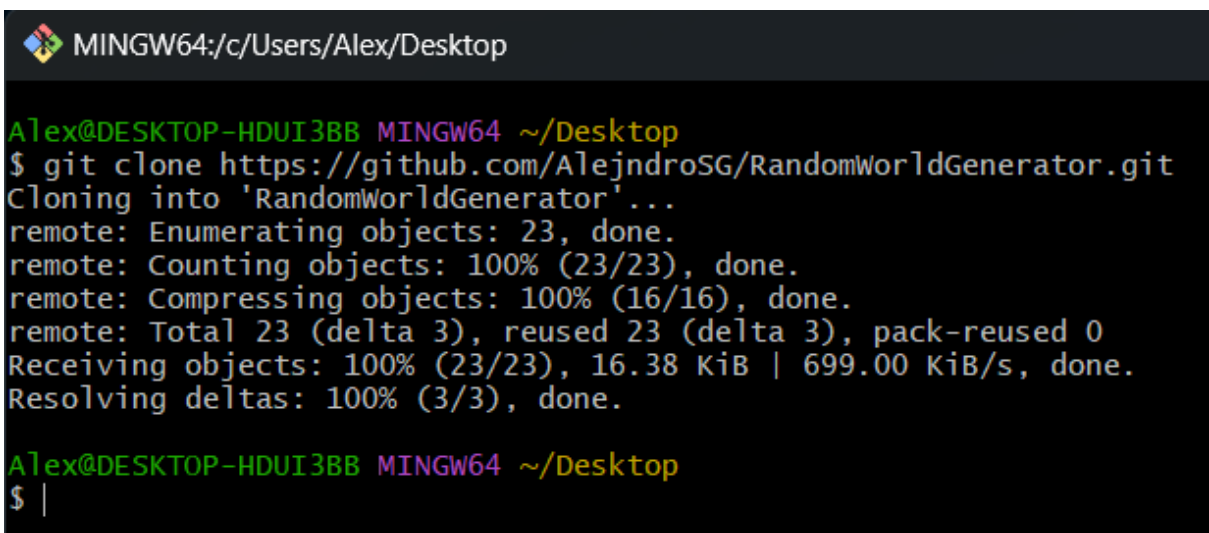
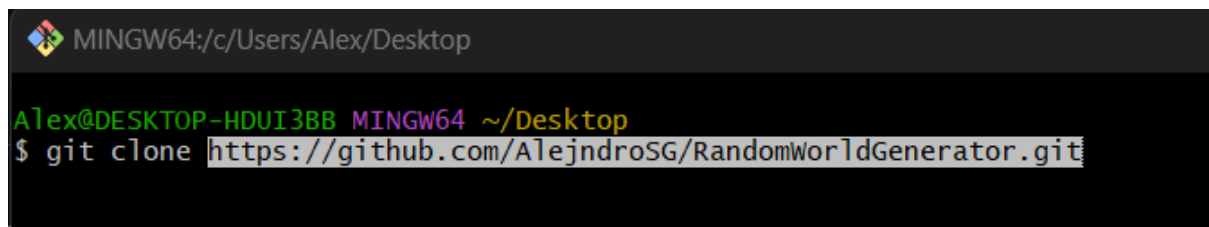
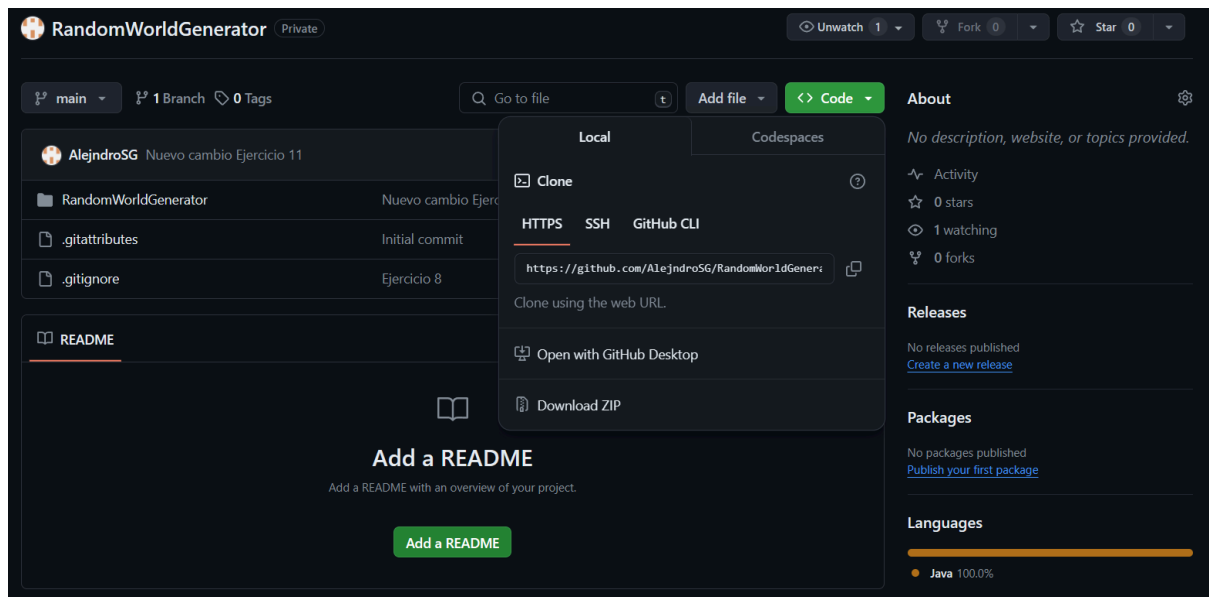
Sí, está instalada en el PC de clase

## Ejercicio 18: Instala Git en la MV del tema anterior.

Lo he hecho en la OVA del exámen, ya que es la que tenía disponible.

```
root@eag:/home/alumno# apt-get install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver libaac3 libaom3 libass9 libavcodec58
 libavformat58 libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1 libcodec2-1.0 libdavid5 libflashrom1
 libflite1 libftdi1-2 libgme0 libgsm1 libgstreamer-plugins-bad1.0-0 libigdgmm12 libilv-0-0 libllvm13 libmfx1 libmysofa1
 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2 libserd-0-0 libshine3 libsnappy1v5 libsord-0-0
 libsratom-0-0 libsrtp1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0 libva-drm2 libva-wayland2 libva-x11-2
 libva2 libvdpau libvidstab1.1 libx265-199 libxvidcore4 libzimg2 libzmq5 libzvti-common libzvti mesa-va-drivers
 mesa-vdpau-drivers pocketsphinx-en-us va-driver-all vdpau-driver-all
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
 git-man liberror-perl
Paquetes sugeridos:
 git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
 git git-man liberror-perl
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 450 no actualizados.
Se necesita descargar 4.120 kB/4.147 kB de archivos.
Se utilizarán 21,0 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3.166 kB]
Descargados 4.120 kB en 3s (1.416 kB/s)
Seleccionando el paquete liberror-perl previamente no seleccionado.
(Leyendo la base de datos ... 207920 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../liberror-perl_0.17029-1_all.deb ...
Desempaquetando liberror-perl (0.17029-1) ...
Seleccionando el paquete git-man previamente no seleccionado.
Preparando para desempaquetar .../git-man_1%3a2.34.1-1ubuntu1.10_all.deb ...
Desempaquetando git-man (1:2.34.1-1ubuntu1.10) ...
Seleccionando el paquete git previamente no seleccionado.
Preparando para desempaquetar .../git_1%3a2.34.1-1ubuntu1.10_amd64.deb ...
Desempaquetando git (1:2.34.1-1ubuntu1.10) ...
Configurando liberror-perl (0.17029-1) ...
```

**Ejercicio 19:** Comprueba que el código del repositorio se ha descargado en tu PC tras hacer el comando clone.



**Ejercicio 20: Prueba algunos de los comandos vistos en el tema anterior en la terminal. ¿Funcionan? ¿por qué si estamos en Windows?**

```
Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop
$ ls
Adobe/                      Edición de Video y TikTok/      Papeleo y Documentos/          desktop.ini
Blender/                   Github/                          PruebaSliderconJS/             images.png
Contraseñas.txt             Github Desktop.lnk              RandomWorldGenerator/          sqldeveloper - Acceso directo.lnk
Copia USB/                 Ideas Para Hacer y Enfocar a futuro.txt  Regalos mi novia su cumple.txt
Cosas Bachiller/           Juegos/                          Visual Studio Code.lnk
DAW/                       Mi Nuevo Gran Proyecto/         bravofinal2[1].pdf

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop
$ ls DAW/
1TDAW_PR_SANCHEZ_GONZALEZ_ALEJANDRO_TIENDA_V2.zip  Examen.ova  'Proyecto ER Base Datos'/  'Repasar HTML para el examen'/  correccion_ex_bucles/
Entrega/                                           NetBeans/   'Pruebas HTML'/          RepasoTema2/                   correccion_ex_bucles.zip
Entrega.zip                                         ORACLE/     'Recursos para Entornos'/  Ubuntu-SSII.ova                mavenproject2/

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop
$ cd DAW/

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/DAW
$ ls -l
total 14865184
-rw-r--r-- 1 Alex 197121      21211 Nov 14 17:45 1TDAW_PR_SANCHEZ_GONZALEZ_ALEJANDRO_TIENDA_V2.zip
drwxr-xr-x 1 Alex 197121      0 Dec  2 03:16 Entrega/
-rw-r--r-- 1 Alex 197121    18828391 Dec  2 03:15 Entrega.zip
-rw-r--r-- 1 Alex 197121    8100345856 Apr  9 17:18 Examen.ova
drwxr-xr-x 1 Alex 197121      0 Jan 15 23:19 NetBeans/
drwxr-xr-x 1 Alex 197121      0 Jan 28 11:47 ORACLE/
drwxr-xr-x 1 Alex 197121      0 Dec  3 17:33 'Proyecto ER Base Datos'/
drwxr-xr-x 1 Alex 197121      0 Oct 17 2023 'Pruebas HTML'/
drwxr-xr-x 1 Alex 197121      0 Nov 11 2023 'Recursos para Entornos'/
drwxr-xr-x 1 Alex 197121      0 Dec 11 18:48 'Repasar HTML para el examen'/
drwxr-xr-x 1 Alex 197121      0 Oct 23 2023 RepasoTema2/
-rw-r--r-- 1 Alex 197121    7102723584 Apr  1 22:48 Ubuntu-SSII.ova
drwxr-xr-x 1 Alex 197121      0 Dec 17 16:17 correccion_ex_bucles/
-rw-r--r-- 1 Alex 197121    1620 Dec 17 16:17 correccion_ex_bucles.zip
drwxr-xr-x 1 Alex 197121      0 Oct  5 2023 mavenproject2/

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/DAW
$
```

Sí, sí que funcionan ya que bash se encarga de emular un entorno de línea de comandos similar a la de Linux en el entorno de Windows. Por ende, los comandos de la terminal de Linux funcionan en Git Bash de Windows.

**Ejercicio 21: Repite el ejercicio 19, ahora en la MV con Linux ¿cambia la forma de realizar el clone por estar en Linux?**

```
alumno@eag: ~/Escritorio/RandomWorldGenerator
alumno@eag: ~/Escritorio$ git clone https://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@github.com/AlejndroSG/RandomWorldGenerator.git
Clonando en 'RandomWorldGenerator'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 32 (delta 7), reused 31 (delta 6), pack-reused 0
Recibiendo objetos: 100% (32/32), 17.23 KiB | 345.00 KiB/s, listo.
Resolviendo deltas: 100% (7/7), listo.
alumno@eag: ~/Escritorio$ ls
Escritorio  Examen_2  nombres.txt  prueba  RandomWorldGenerator
alumno@eag: ~/Escritorio$ cd RandomWorldGenerator/
alumno@eag: ~/Escritorio/RandomWorldGenerator$ ls
RandomWorldGenerator
alumno@eag: ~/Escritorio/RandomWorldGenerator$
```

**Ejercicio 22: Una vez tengas el repositorio cargando tanto en Windows como en Linux ábrelo con Netbeans. Modifica el código en Windows, por ejemplo añade un nuevo 'sout'**

```
public static void main(String[] args) {
    System.out.println(x: "Hola Mundo");
    System.out.println(x: "Adios vida");
    System.out.println(x: "He vuelto a modificarlo, aquí estamos de nuevo");
}
```

**Ejercicio 23:** En Windows, realiza el commit del cambio realizado.

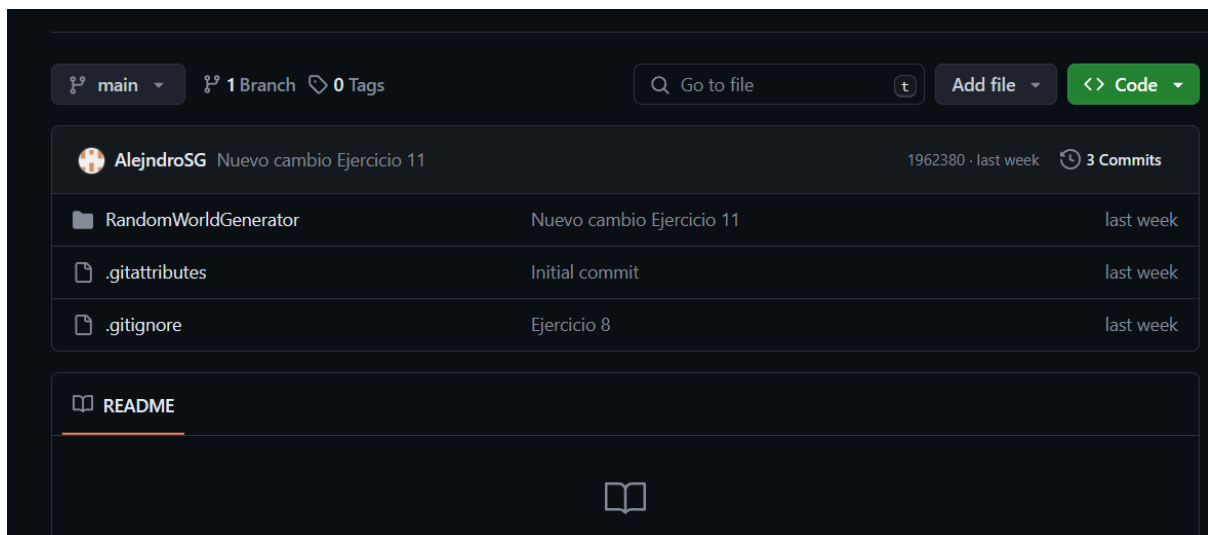
```
MINGW64:/c/Users/Alex/Desktop/RandomWorldGenerator

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop
$ cd RandomWorldGenerator/

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git add .

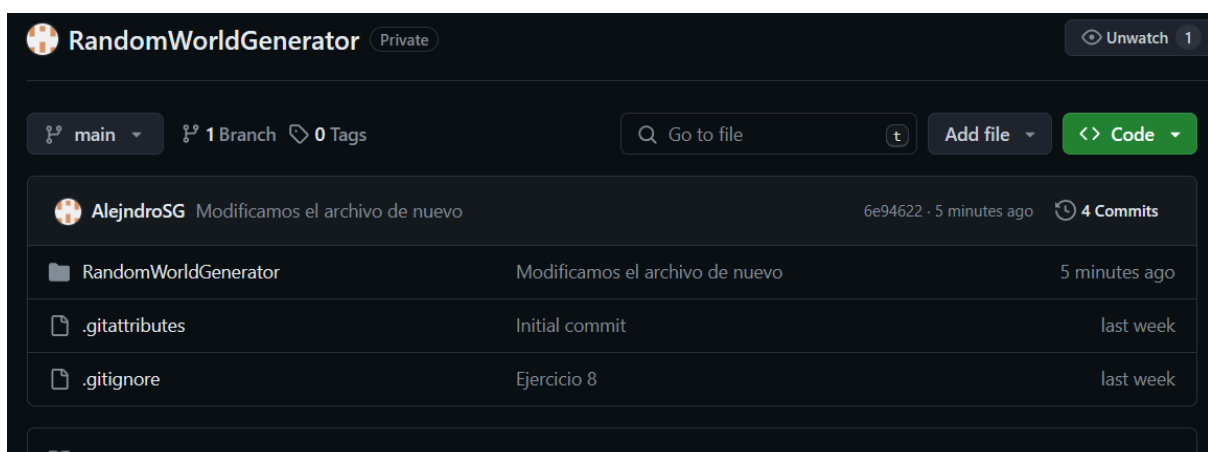
Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git commit -m "Modificamos el archivo de nuevo"
[main 6e94622] Modificamos el archivo de nuevo
3 files changed, 6 insertions(+), 4 deletions(-)
```

**Ejercicio 24.** Visita la web de tu repositorio, navega hasta el fichero .java en el que hayas realizado el cambio ¿se ve reflejado? ¿por qué?



No, no se ve reflejado ya que hemos confirmado los cambios en nuestro repositorio local, pero no los hemos realizado en nuestro repositorio en la nube. Tenemos que hacer un git push.

**Ejercicio 25.** Haz un push al servidor y comprueba desde la web que los cambios se han realizado.



```
MINGW64:/c/Users/Alex/Desktop/RandomWorldGenerator

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git push origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 968 bytes | 968.00 KiB/s, done.
Total 9 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/AlejndroSG/RandomWorldGenerator.git
1962380..6e94622 main -> main

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ |
```

**Ejercicio 26. ¿El código del proyecto en Linux está actualizado? ¿por qué? ¿Qué comando debemos realizar?**

No, no está actualizado ya que debemos de hacer un pull. Si no hacemos el pull, tenemos el cambio realizado en Windows y en la nube pero no en el repositorio local de Linux.

**Ejercicio 27. Realiza un pull de proyecto desde Linux. Comprueba que el código pasa a estar actualizado.**

```
alumno@eag: ~/Escritorio/RandomWorldGenerator/RandomWorldGenerator/src/randomworldgenerator
alumno@eag:~/Escritorio/RandomWorldGenerator/RandomWorldGenerator/src/randomworldgenerator$ cat RandomWorldGenerator.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package randomworldgenerator;

/**
 *
 * @author EAG
 */
public class RandomWorldGenerator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hola Mundo");
        System.out.println("Adios vida");
        System.out.println("He vuelto a modificarlo, aquí estamos de nuevo");
    }

}
```

Aquí aún no se ha modificado

```
alumno@eag: ~/Escritorio/RandomWorldGenerator/RandomWorldGenerator/src/randomworldgenerator

alumno@eag:~/Escritorio/RandomWorldGenerator/RandomWorldGenerator$ git pull
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 7 (delta 4), reused 7 (delta 4), pack-reused 0
Desempaquetando objetos: 100% (7/7), 663 bytes | 221.00 KiB/s, listo.
Desde https://github.com/AlejandroSG/RandomWorldGenerator
   6e94622..e0ecdcd  main    -> origin/main
Actualizando 6e94622..e0ecdcd
Fast-forward
   RandomWorldGenerator/src/randomworldgenerator/RandomWorldGenerator.java | 1 +
   1 file changed, 1 insertion(+)
alumno@eag:~/Escritorio/RandomWorldGenerator/RandomWorldGenerator$ cd
nbproject/ src/
alumno@eag:~/Escritorio/RandomWorldGenerator/RandomWorldGenerator$ cd src/randomworldgenerator/
alumno@eag:~/Escritorio/RandomWorldGenerator/RandomWorldGenerator/src/randomworldgenerator$ cat RandomWorldGenerator.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package randomworldgenerator;

/**
 *
 * @author EAG
 */
public class RandomWorldGenerator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hola Mundo");
        System.out.println("Adios vida");
        System.out.println("He vuelto a modificarlo, aqui estamos de nuevo");
        System.out.println("A esto quiero hacerle un pull en Linux");
    }
}
```

Y aquí sí que se han realizado los cambios en mi repositorio local de Linux

## Ejercicio 29. ¿Con qué comando podemos volver a versiones anteriores de Git? ¿Cuál es su sintaxis? Intenta volver a la primera versión de tu proyecto

Tenemos que poner git log, copiamos el log y luego ponemos git reset con el log.

```
MINGW64/c:/Users/Alex/Desktop/RandomWorldGenerator

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git log .
commit 6e946220537a84fc04e3e7420b6b2a024740d343 (HEAD -> main, origin/main, origin/HEAD)
Author: AlejandroSG <alex2005sg@gmail.com>
Date: Mon May 13 17:36:41 2024 +0200

    Modificamos el archivo de nuevo

commit 19623809a9270c6a64b6e6a450b73928ae8560ae
Author: AlejandroSG <alex2005sg@gmail.com>
Date: Mon May 6 14:12:50 2024 +0200

    Nuevo cambio Ejercicio 11

commit d0ea48abdba7358ef8699c57ea67166caffdb5f1
Author: AlejandroSG <alex2005sg@gmail.com>
Date: Fri May 3 14:15:16 2024 +0200

    Ejercicio 8

commit 4877756e66b13308f291ec64f7b1f165b2d6adbe
Author: AlejandroSG <alex2005sg@gmail.com>
Date: Fri May 3 14:05:01 2024 +0200

    Initial commit

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git reset 4877756e66b13308f291ec64f7b1f165b2d6adbe

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
$ git log .
commit 4877756e66b13308f291ec64f7b1f165b2d6adbe (HEAD -> main)
Author: AlejandroSG <alex2005sg@gmail.com>
Date: Fri May 3 14:05:01 2024 +0200

    Initial commit

Alex@DESKTOP-HDUI3BB MINGW64 ~/Desktop/RandomWorldGenerator (main)
```

Así, luego si hacemos el git log, vemos como solo hay una versión, la primera de todas

**Ejercicio 30: En Git, ¿qué es un rama?, ¿cómo crees que pueden beneficiarte a la hora de organizar el código de tu proyecto?**

En Git, una rama es, por así decirlo, una línea independiente a la creación del proyecto. Su función es poder hacer esa parte del código sin tener que estar constantemente haciendo push y pull en la rama principal junto a todos los compañeros. La idea es que, una vez tengas tu parte terminada, hagas un push a la rama principal, destruyendo la rama que habías creado.

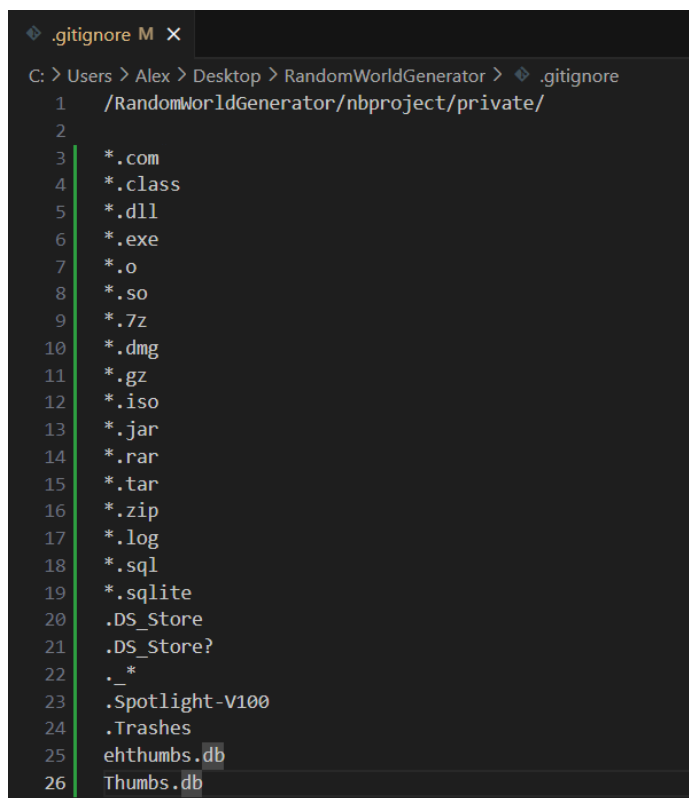
Fomenta mucho el trabajo en equipo y el arreglo de errores sin tener que alterar todo el código.

**Ejercicio 31: Explica con tus propias palabras que tipos de ficheros ignora el fichero <https://gist.github.com/octocat/9257657> ¿Por qué crees que se ignoran dichos ficheros?**

Se ignoran:

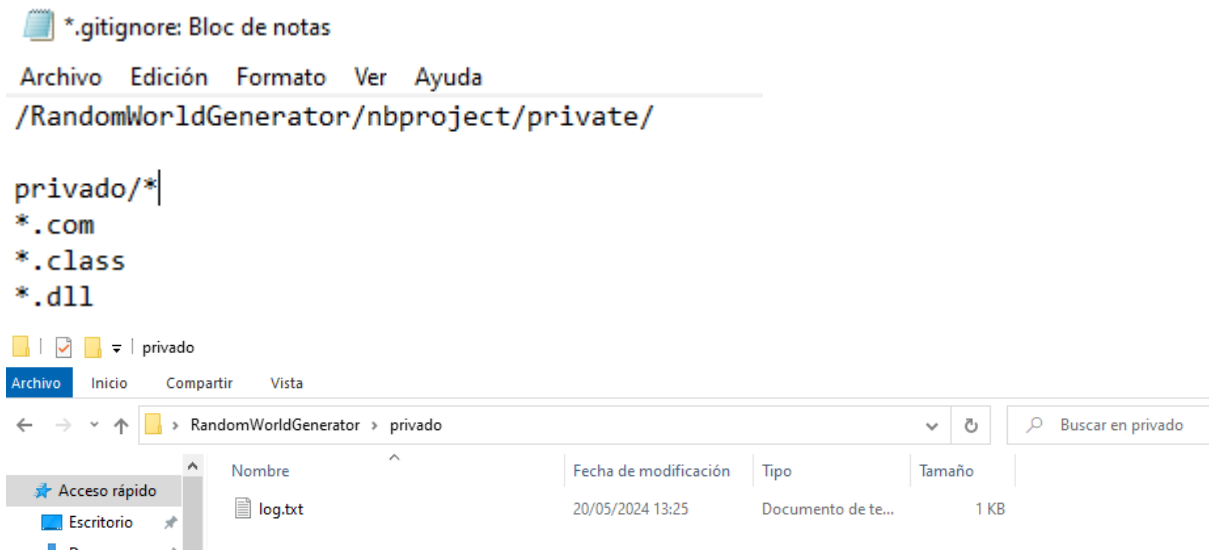
- Ejecutables: No queremos tener un ejecutable en nuestro repositorio, ya que ese lo tendremos nosotros
- Paquetes: Cosas como .zip o .rar no nos interesan tenerlas en la nube cuando podemos hacerlo nosotros mismos en nuestro repositorio local
- Logs y bases de datos: Pueden contener información sensible, por lo que no es recomendable que se suban a la nube
- Archivos de SO: Son archivos importantes para nuestro SO que contiene también información sensible, por lo que no debemos subirlos

**Ejercicio 32. Crea un fichero .gitignore, o si ya está creado modifícalo, con el contenido de <https://gist.github.com/octocat/9257657> dentro de tu proyecto**

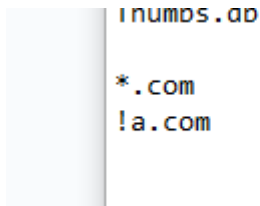


```
.gitignore M X
C: > Users > Alex > Desktop > RandomWorldGenerator > .gitignore
1 /RandomWorldGenerator/nbproject/private/
2
3 *.com
4 *.class
5 *.dll
6 *.exe
7 *.o
8 *.so
9 *.7z
10 *.dmg
11 *.gz
12 *.iso
13 *.jar
14 *.rar
15 *.tar
16 *.zip
17 *.log
18 *.sql
19 *.sqlite
20 .DS_Store
21 .DS_Store?
22 *
23 __
24 .Spotlight-V100
25 .Trashes
26 ehthumbs.db
Thumbs.db
```

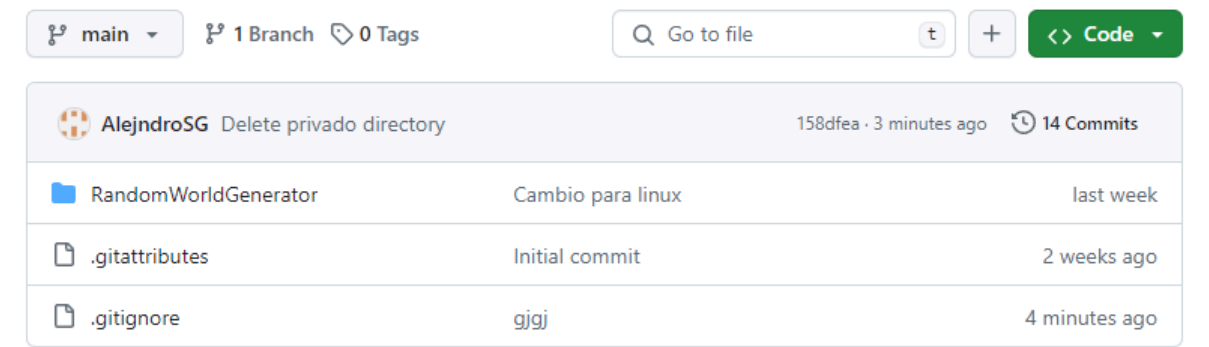
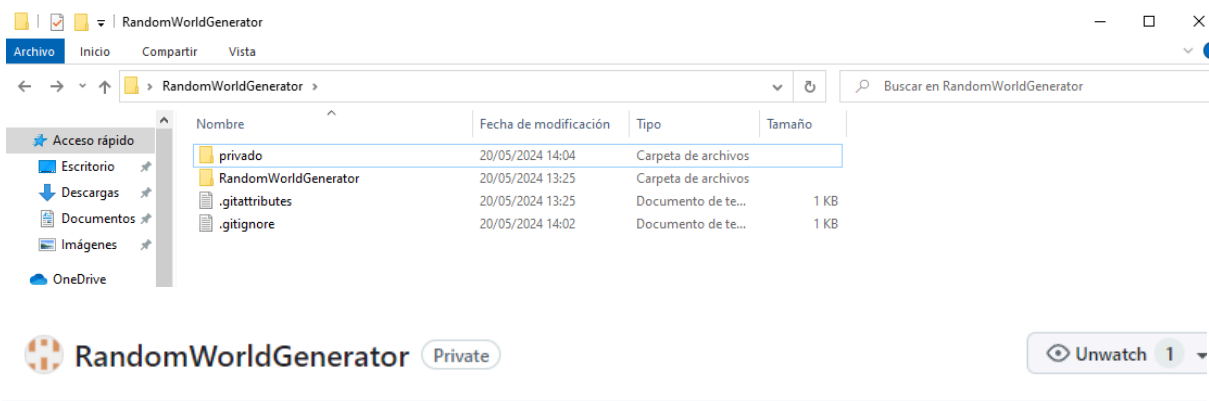
**Ejercicio 33:** Añade una carpeta con el nombre ‘privado’ al proyecto y dentro un fichero con el nombre ‘log.txt’. Haz que todo lo que hay dentro de dicha carpeta no se sincronice con el servidor.



**Ejercicio 34:** Ignora todos los ficheros .com excepto a.com



**Ejercicio 35:** Haz que los cambios realizados en el fichero .gitignore tengan efecto sobre el repositorio. ¿Se ha subido la carpeta ‘private’? ¿Por qué?





No se ha subido la carpeta privado ya que le hemos dicho que la ignore directamente en el archivo .gitignore

Ejercicio 36: Añade los ficheros a.com y b.com en local. Sincroniza la BD remota ¿se han subido los dos ficheros? ¿Por qué?

Nombre	Fecha de modificación	Tipo	Tamaño
fgdfgdfg	20/05/2024 17:02	Carpeta de archivos	
RandomWorldGenerator	20/05/2024 17:02	Carpeta de archivos	
.gitattributes	20/05/2024 17:02	Archivo de origen ...	1 KB
.gitignore	20/05/2024 17:02	Archivo de origen ...	1 KB
a	20/05/2024 17:23	Aplicación MS-DOS	0 KB
b	20/05/2024 17:23	Aplicación MS-DOS	0 KB

FileEditViewRepositoryBranchHelp

Current repository  
RandomWorldGenerator

Current branch  
main

Fetch origin  
Last fetched 25 minutes ago

Changes 1

History

a.com

1 changed file

a.com

RandomWorldGeneratorPrivate

Unwatch1

main1 Branch0 Tags

Go to file

Add file

Code

AlejandroSG crear el com534d7d5 · now16 Commits

RandomWorldGeneratorCambio para linuxlast week

fgdfgdfghola3 hours ago

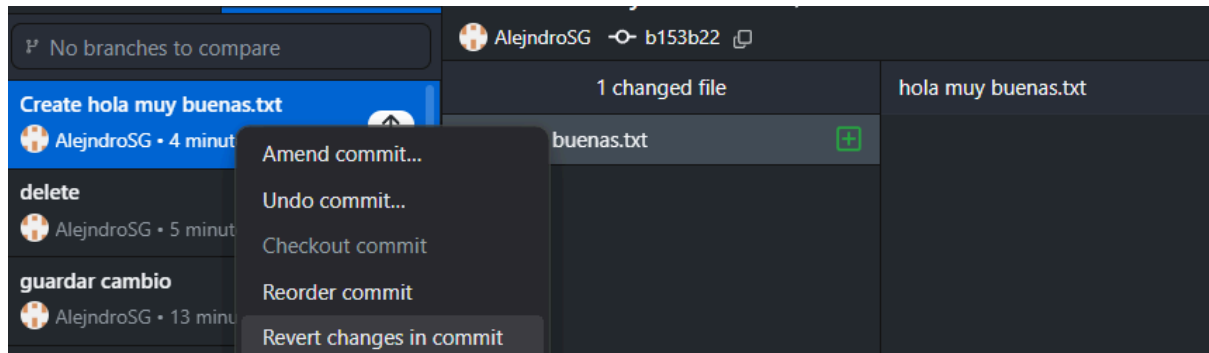
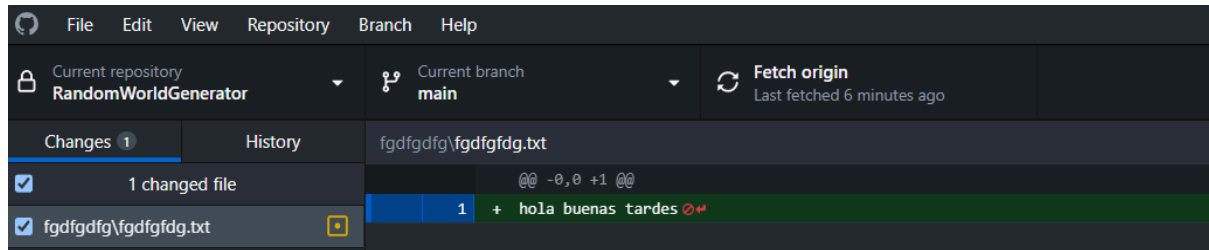
.gitattributesInitial commit2 weeks ago

.gitignoregjj3 hours ago

a.comcrear el comnow

Se ha subido el a.com ya que hemos metido todos los archivos .com en las excepciones de git, menos a.com, ya que es la excepción que hemos puesto anteriormente.

**Ejercicio 37: Modifica un fichero del proyecto y guarda los cambios. Cierra el fichero y haz un restore del repositorio. ¿Qué ha pasado?**



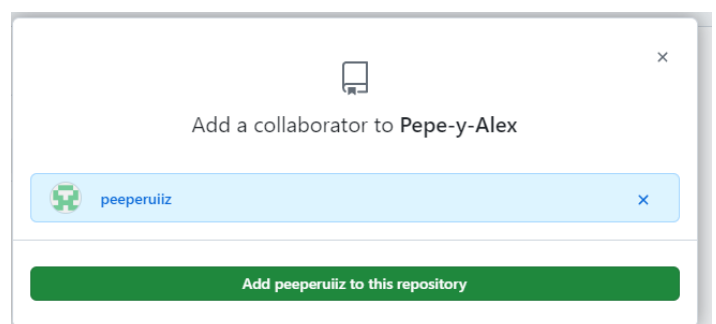
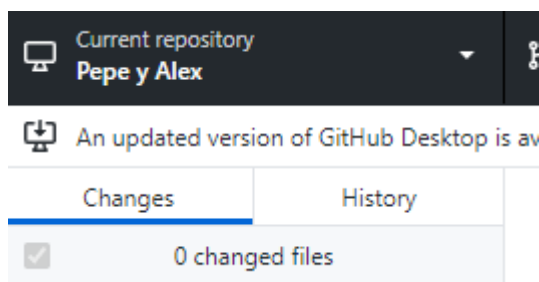
Lo que ha pasado es que se han revertido los cambios que hemos hecho hasta el último commit que teníamos anteriormente.

**Ejercicio 38: Hemos visto cómo volver a versiones concretas en local pero... ¿cómo podríamos volver a dichas versiones en remoto?**

Deberemos encontrar la versión del commit al que queremos volver, encontrando el hash de la dicha. Para ello usamos git log, y una vez encontremos la versión del commit al que queremos volver, haremos git reset --hard (y la versión del commit)

**Ejercicio 39: El alumno A deberá crear un nuevo proyecto, descargarlo en su pc y dar acceso a su proyecto al alumno B. El alumno B deberá aceptar la invitación y descargarse el repositorio en una carpeta en su PC.**

**SOY EL ALUMNO A Y PEPE EL B**



## Ejercicio 40:

### SOY EL ALUMNO A Y PEPE EL B

IntelliJ IDEA interface showing the initial commit of a repository named "Pepe-y-Alex" on the "main" branch. The commit was made by AlejandroSG and includes 10 changed files, including .gitattributes, build.xml, manifest.mf, and project.properties files. The .gitattributes file is highlighted, showing settings for LF normalization and text encoding.

IntelliJ IDEA interface showing the "Pull origin" button in the toolbar, indicating that the local branch is up-to-date with the remote repository.

IntelliJ IDEA interface showing the "Push origin" button in the toolbar, indicating that the local branch is ready to be pushed to the remote repository.

IntelliJ IDEA interface showing the "No local changes" dialog box, suggesting to pull the latest changes from the origin remote. The dialog includes a "Pull origin" button and a message indicating that the current branch (main) has a commit on GitHub that does not exist on the machine.

```
public class PepeYAlex {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        System.out.println("Hola Alex");  
        System.out.println("Buenas Pepe");  
        System.out.println("Que tal");  
        System.out.println("Sigo trabajando");  
    }  
}
```

## Ejercicio 41:

YO SOY EL A Y PEPE ES EL B

```
public class PepeYAlex {
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hola Alex");
        System.out.println("Buenas Pepe");
        System.out.println("Que tal");
        System.out.println("Sigo trabajando");
        System.out.println("Añado linea para provocar error");
    }
}
```

File Edit View Repository Branch Help

Current repository: Pepe-y-Alex | Current branch: main | Fetch origin: Last fetched 5 minutes ago

Changes: 1 | History

Pepe y Alex\src\pepe\y\alex\PepeYAlex.java

File	Changes	Diff
Pepe y Alex\src\p...\PepeYAlex.java	1 changed file	@@ -19,6 +19,7 @@ public class PepeYAlex { 19 19 System.out.println("Buenas Pepe"); 20 20 System.out.println("Que tal"); 21 21 System.out.println("Sigo trabajando"); 22 22 + System.out.println("Añado linea para provocar error"); 22 23 } 23 24 24 25 }

Error

Resolve conflicts before Merge

1 conflicted file

Pepe y Alex\src\pepe\y\alex\PepeYAlex.java  
1 conflict

Open in Sublime Text

[Open in command line](#), your tool of choice, or close to resolve manually.

Continue merge

Abort merge

```
System.out.println("Que tal");
System.out.println("Sigo trabajando");
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
System.out.println("Añado linea para provocar error");
=====
System.out.println("Soy un chico conflictivo");
>>>>>> 2a3a568fe276a0dcf2b5eb720d737d38eb7ed100 (Incoming Change)
}
```

```
public static void main(String[] args) {
    System.out.println("Hola Alex");
    System.out.println("Buenas Pepe");
    System.out.println("Que tal");
    System.out.println("Sigo trabajando");
    System.out.println("Añado linea para provocar error");
    System.out.println("Soy un chico conflictivo");
}
```

#### Ejercicio 42: Realiza un resumen de cómo se debe utilizar Git cuando estás trabajando en un proyecto junto a más personas.

Cuando estemos trabajando, tenemos que tener en cuenta varias cosas:

1. Todos tenemos que tener un repositorio central en el que tendremos todo nuestro proyecto, y todos tenemos que tener acceso al repositorio.
2. Además, cada uno tiene que tener clonado el repositorio en local, para poder hacer pull y luego push, y no perder los cambios realizados (además de evitar conflictos)
3. Es muy conveniente el uso de ramas, ya que es muy útil a la hora de organizar el trabajo, y siempre tienen que estar creado a partir del repositorio general, al que llamaremos (como normal general) **main**.
4. Tenemos que llevar a cabo buenas prácticas para evitar conflictos, como por ejemplo lo dicho anteriormente: cuando tengamos un cambio pendiente por subir, no hagamos directamente el push, porque sino nos puede dar fallo y perderemos datos. Lo suyo es hacer: **COMMIT, PULL y PUSH**.

#### Ejercicio 43: ¿Qué es un fork? Encuentra ejemplos de forks famosos en la vida real.

Un fork no es más que una copia del proyecto en el que podemos realizar cambios en el código, sin afectar al proyecto original. Dichos cambios luego sí que se puede pedir permiso al creador para poder efectuar los cambios en el código original con un pull request.

Esto permite y fomenta la colaboración entre personas y la integridad de la comunidad, entre otras cosas.

Un ejemplo claro de fork en la vida real es, por ejemplo, LibreOffice. Querían que creciera más, por ende dejaron que personas actuaran y modificaran el código del programa, convirtiéndose en uno de los fork más populares del mundo.

También tenemos MariaDB que se creó como un fork de MySQL, firefox...

#### Ejercicio 44: ¿Qué es una pull request? ¿Puedo hacer una pull request en mi propio código? ¿Pueden otras personas hacer una pull request en mi código?

Una pull request es una solicitud para subir al archivo original un fork, y por ende, aplicar los cambios realizados en dicho fork. El dueño verá los cambios realizados en el código, y tendrá la opción de añadir los cambios o no. Si los aceptas quedarán registrados como una aportación de tu perfil al proyecto, siendo de mucha ayuda a la hora de encontrar trabajo (debido a que las empresas lo tienen muy en cuenta).

Sí, sí que puedes hacer una pull request en tu propio código, aunque si estás trabajando solo no tiene mucho sentido (a no ser que vayas a darle a otra persona el fork, en este caso sí que merece la pena)

Sí, otras personas sí que pueden hacer un pull request de mi código, de hecho es fundamental para la colaboración del código y la integración de la comunidad en muchos proyectos. Si es público, podrás hacer directamente un fork y un pull request, pero si es privada tienes que esperar a que el creador te dé permiso

#### Ejercicio 45: Haz un fork del proyecto de un compañero y clónalo en tu PC de forma que se descarguen todos los ficheros del proyecto.

The screenshot shows the GitHub interface for a repository named 'Nuevo-Proyecto-Fork' by user 'peeperuiiz'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows an initial commit by 'peeperuiiz' 3 minutes ago. The file list includes a folder named 'Fork' and a file named '.gitattributes', both committed 3 minutes ago.

File	Commit	Time
Fork	Initial commit	3 minutes ago
.gitattributes	Initial commit	3 minutes ago

Owner \* AlejandroSG / Repository name \* Nuevo Fork

✓ Your new repository will be created as **Nuevo-Fork**.  
The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

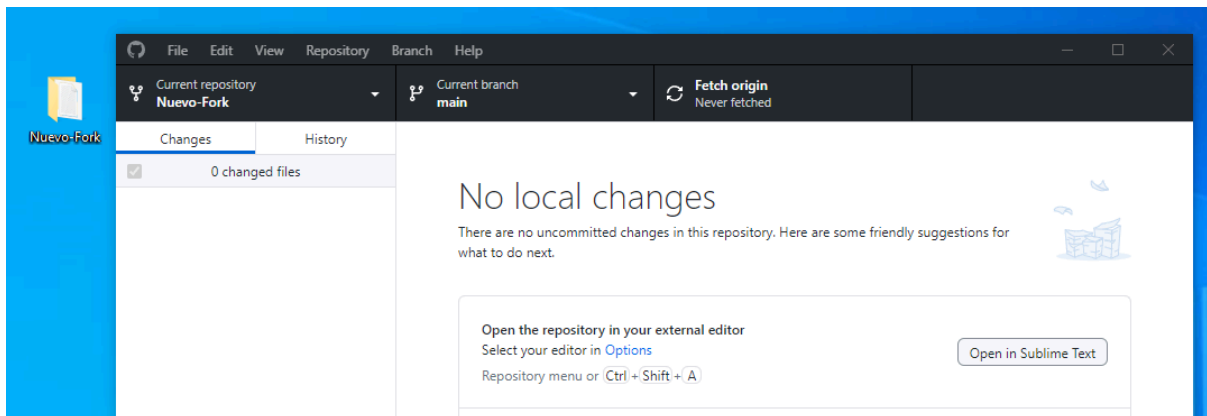
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

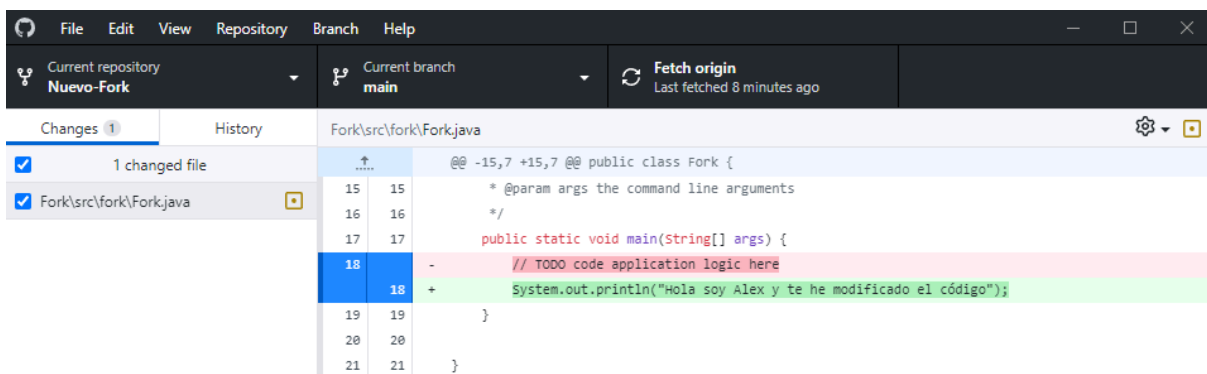
☒ Copy the main branch only  
Contribute back to peeperuiiz/Nuevo-Proyecto-Fork by adding your own branch. [Learn more.](#)



*i* You are creating a fork in your personal account.

**Create fork**



## Ejercicio 46: Haz una pull request del proyecto de tu compañero del que hiciste fork



 base repository: peeperuiiz/Nuevo-Proyecto-Fork base: main  head repository: AlejandroSG/Nuevo-Fork compare: main

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)


Create pull request



1 commit

1 file changed

1 contributor




Commits on May 21, 2024

V1  
 AlejandroSG committed 2 minutes ago

 3984242 


Showing 1 changed file with 1 addition and 1 deletion.


Split Unified

 2  Fork/src/fork/Fork.java 

↑	@@ -15,7 +15,7 @@	public class Fork {
15	15	* @param args the command line arguments
16	16	*/
17	17	public static void main(String[] args) {
18	-	// TODO code application logic here
18	+	System.out.println("Hola soy Alex y te he modificado el código");
19	19	}
20	20	
21	21	}

## V1 #1

 Open

AlejandroSG wants to merge 1 commit into peeperuiiz:main from AlejandroSG:main 

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



AlejandroSG commented now

...




No description provided.



  V1

3984242

  AlejandroSG / Nuevo-Proyecto-Fork


   Pull requests 1     

Label issues and pull requests for new contributors

[Dismiss](#)


Now, GitHub will help potential first-time contributors [discover issues](#) labeled with good first issue

Filters

 Labels 9

 Milestones 0

New pull request

☐  1 Open ☒ 0 Closed

Author

Label


Projects

Milestones

Reviews

Assignee

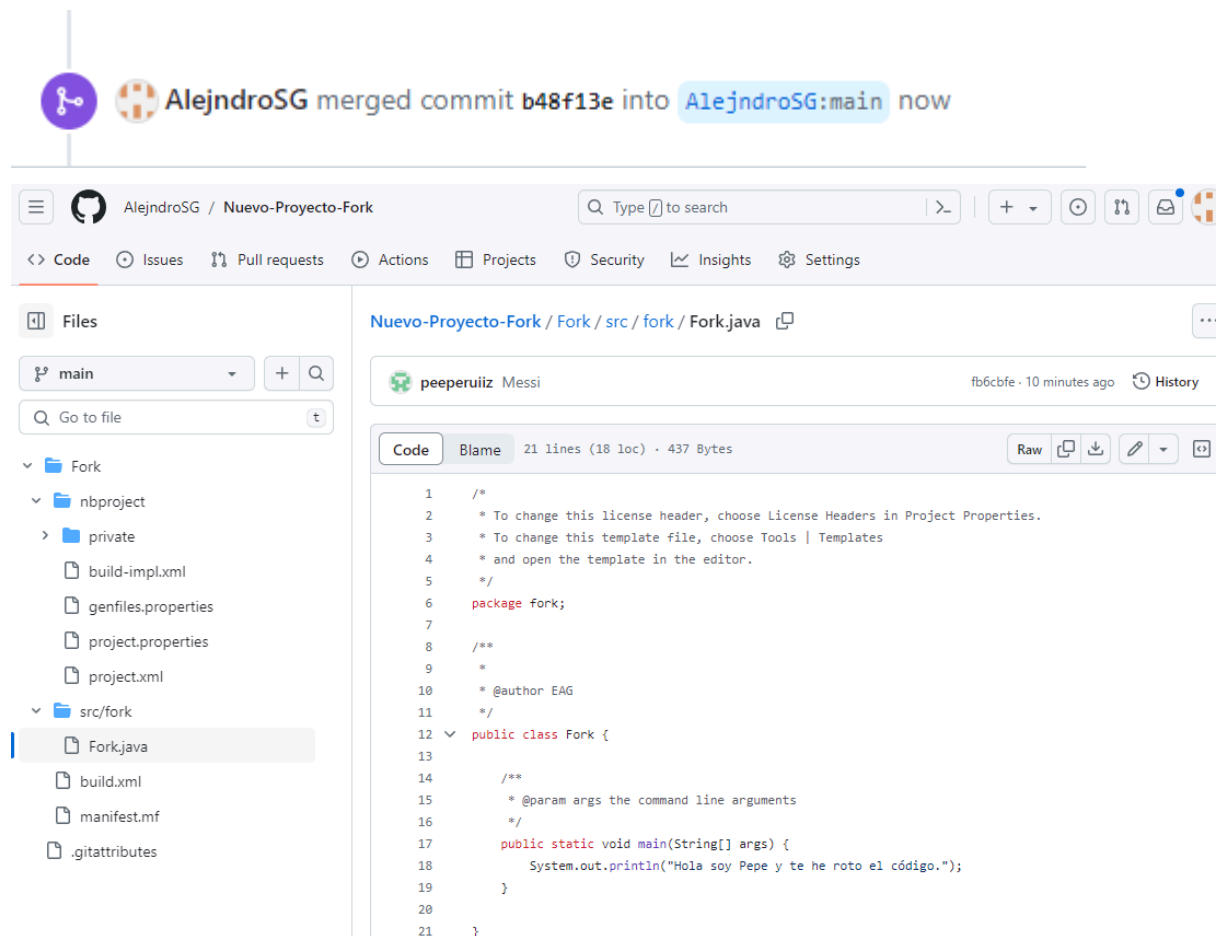
Sort

☐  Messi

#1 opened 6 minutes ago by peeperuiiz



## Ejercicio 47: Acepta la pull request que te haya hecho tu compañero.



The screenshot shows a GitHub pull request interface. At the top, a notification bar states: "AlejandroSG merged commit b48f13e into AlejandroSG:main now". Below this, the repository name "AlejandroSG / Nuevo-Proyecto-Fork" is visible. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The left sidebar shows the file explorer with the following structure:

- main
- Fork
  - nbproject
    - private
      - build-impl.xml
      - genfiles.properties
      - project.properties
      - project.xml
    - src/fork
      - Fork.java
      - build.xml
      - manifest.mf
      - .gitattributes

The main content area displays the file "Nuevo-Proyecto-Fork / Fork / src / fork / Fork.java" by user "peeperuiz" (Messi) with commit hash "fb6cbfe" from 10 minutes ago. The code editor shows the following Java code:

```
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 fork;
7
8  /**
9   *
10   * @author EAG
11   */
12  public class Fork {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          System.out.println("Hola soy Pepe y te he roto el código.");
19      }
20  }
21  }
```

## Ejercicio 48: Crea tu página personal en GitHub.

Junto a Jose, el profesor, seguimos los pasos y tengo la página ya creada, con todos los pasos seguidos de uno a uno desde cero, y está totalmente aplicada.

Solo puedes tener una página personal para cada perfil de GitHub, ya que el repositorio tiene que tener tu nombre (entre otras cosas) y no puede haber dos repositorios que se llamen igual (sobre todo por el tema del nombre del enlace).

Mi web es la siguiente: <https://alejandrosgh.github.io>