

Repaso de conceptos vistos en clase

Sergio Steeven Moreno Forero

Facultad De Ingeniería, Universidad De Cundinamarca

Programación II

William Alexander Matallana Porras

20 de febrero del 2025

Tabla de contenido

Introducción.....	3
Objetivos.....	4
Desarrollo.....	5
Conclusion.....	28
Referencias.....	29

Introducción

En el desarrollo de software, llevar el control de versiones es esencial para gestionar cambios en el código y facilitar la colaboración entre desarrolladores. Git se ha convertido en uno de los sistemas de control de versiones más utilizados debido a su eficiencia, flexibilidad y capacidad de trabajo distribuido.

IntelliJ IDEA, es un entorno de desarrollo integrado IDE que es popular por su integración avanzada con Git, permite a los desarrolladores ejecutar la mayoría de las operaciones de control de versiones de forma fácil e intuitiva. A través de su interfaz visual y comandos integrados, IntelliJ permite a los usuarios clonar repositorios, crear ramas, fusionar cambios, solucionar conflictos y muchas otras funciones. En este trabajo, explicaremos los principales comandos de Git dentro de IntelliJ IDEA.

Objetivos

Objetivo General:

- Comprender el uso de Git dentro de IntelliJ IDEA para gestionar el control de versiones de manera eficiente en proyectos.

Objetivos específicos:

- Identificar y aplicar los comandos esenciales de Git dentro de IntelliJ.
- Demostrar mediante un ejemplo práctico cómo realizar operaciones con Git en IntelliJ.

Desarrollo

Configuración básica de Git

1. Git config -- list

Este comando muestra todas las configuraciones establecidas en Git, incluyendo las configuraciones a nivel de sistema, global y local. La salida incluye parámetros como el nombre y correo electrónico del usuario, el editor predeterminado, la rama principal por defecto, alias personalizados y otras opciones relevantes.

The screenshot shows an IDE interface with a terminal window open. The terminal is running on a Windows system (PS C:\) and executing the command `git config --list`. The output lists various global and local Git configuration settings, including http.proxy, core.autocrlf, core.fscache, core.symlinks, pull.rebase, credential.helper, init.defaultbranch, user.name, and user.email. The user email listed is `ssmoreno@ucundinamarca.edu.co`.

```

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=channel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.name=SergioMoreno12
user.email=ssmoreno@ucundinamarca.edu.co

```

2. Git config --global user.email

Con este comando, Git asigna un correo electrónico al usuario a nivel global, lo que significa que se aplicará a todos los repositorios en el sistema. Este correo electrónico es utilizado en los metadatos de cada commit, permitiendo identificar quién realizó cada cambio en el historial del repositorio. Es importante asegurarse de que el correo electrónico utilizado coincida con el registrado en GitHub para evitar problemas con la autoría de los commits. Después se puede utilizar un `git config --list` para confirmar los cambios.

```

Project: EjercicioDeReposo
Main.java
public class Main {
    public static void main(String[] args) {
        double suma = numero1 + numero2;
        // resta
        double resta = numero1 - numero2;
        // resultado
        System.out.println("la suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

Terminal: Local (2)
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --global user.email ssmoreno@ucundinamarca.edu.co
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=ssllambda
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.name=SergioMoreno12
user.email:ssmoreno@ucundinamarca.edu.co
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true

```

3. Git config --global user.name

Este comando configura el nombre del usuario a nivel global en Git, de manera similar a la configuración del correo electrónico. El nombre proporcionado aparecerá en cada commit realizado por el usuario en cualquier repositorio. Es una práctica recomendada configurar tanto el nombre como el correo electrónico antes de comenzar a trabajar con Git, ya que estos datos permiten que los cambios sean correctamente atribuidos a su autor en el historial del proyecto. Después se puede utilizar un git config --list para confirmar los cambios.

```

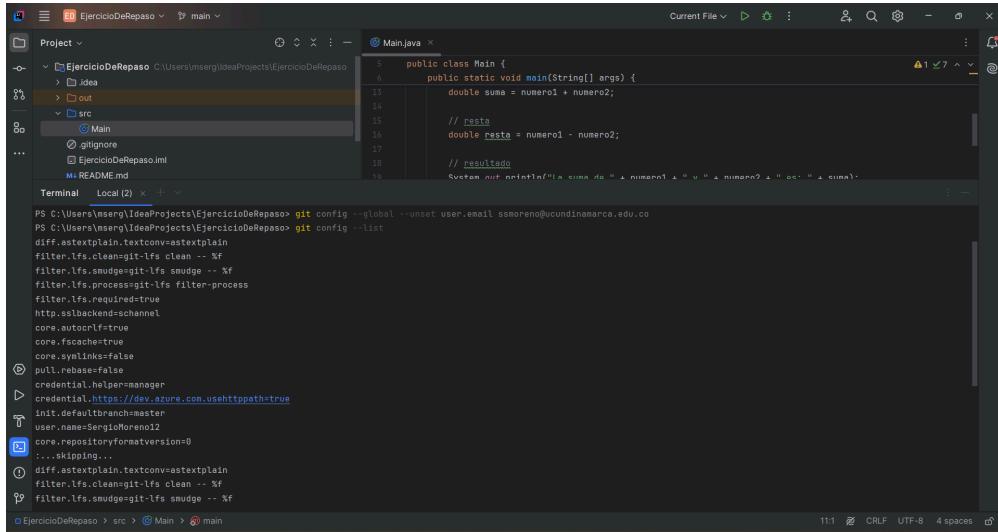
Project: EjercicioDeReposo
Main.java
public class Main {
    public static void main(String[] args) {
        double suma = numero1 + numero2;
        // resta
        double resta = numero1 - numero2;
        // resultado
        System.out.println("la suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

Terminal: Local (2)
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --global user.name SergioMoreno12
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=ssllambda
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.name=SergioMoreno12
user.email:ssmoreno@ucundinamarca.edu.co
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true

```

4. Git config --global - -unset user.email

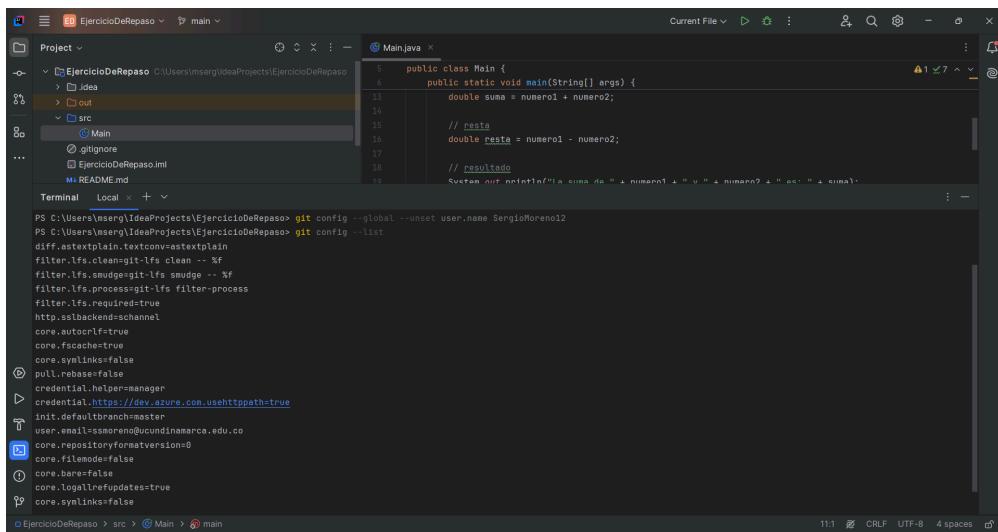
Con este comando, se elimina la configuración del correo electrónico del usuario a nivel global, es decir, ya no habrá un correo electrónico predeterminado en todos los repositorios. Después se puede utilizar un git config --list para confirmar los cambios.



```
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --global --unset user.email sserg@cundinamarca.edu.co
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.cleangit-lfs clean -- %f
filter.lfs.smudgegit-lfs smudge -- %f
filter.lfs.processgit-lfs filter-process
filter.lfs.required=true
http.sslbackend=sshchannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.name=SergioMoreno12
core.repositoryformatversion=0
... skipping...
diff.astextplain.textconv=astextplain
filter.lfs.cleangit-lfs clean -- %f
filter.lfs.smudgegit-lfs smudge -- %f
EjercicioDeReposo > src > Main > main
```

5. Git config --global - -unset user.name

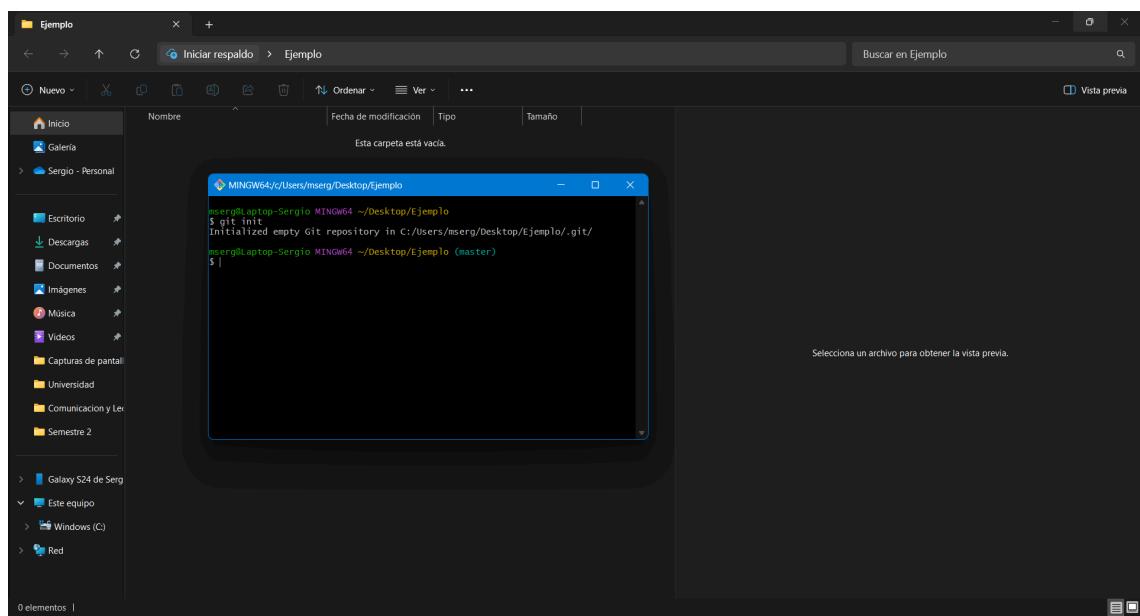
Este comando elimina la configuración global del nombre del usuario en Git. Después de ejecutarlo, Git ya no tendrá un nombre definido para los commits en todos los repositorios. Después se puede utilizar un git config --list para confirmar los cambios.



```
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --global --unset user.name SergioMoreno12
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.cleangit-lfs clean -- %f
filter.lfs.smudgegit-lfs smudge -- %f
filter.lfs.processgit-lfs filter-process
filter.lfs.required=true
http.sslbackend=sshchannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.email=sserg@cundinamarca.edu.co
core.repositoryformatversion=0
core.filmedefalse
core.bare=false
core.logallrefupdates=true
core.symlinks=false
EjercicioDeReposo > src > Main > main
```

6. Git init

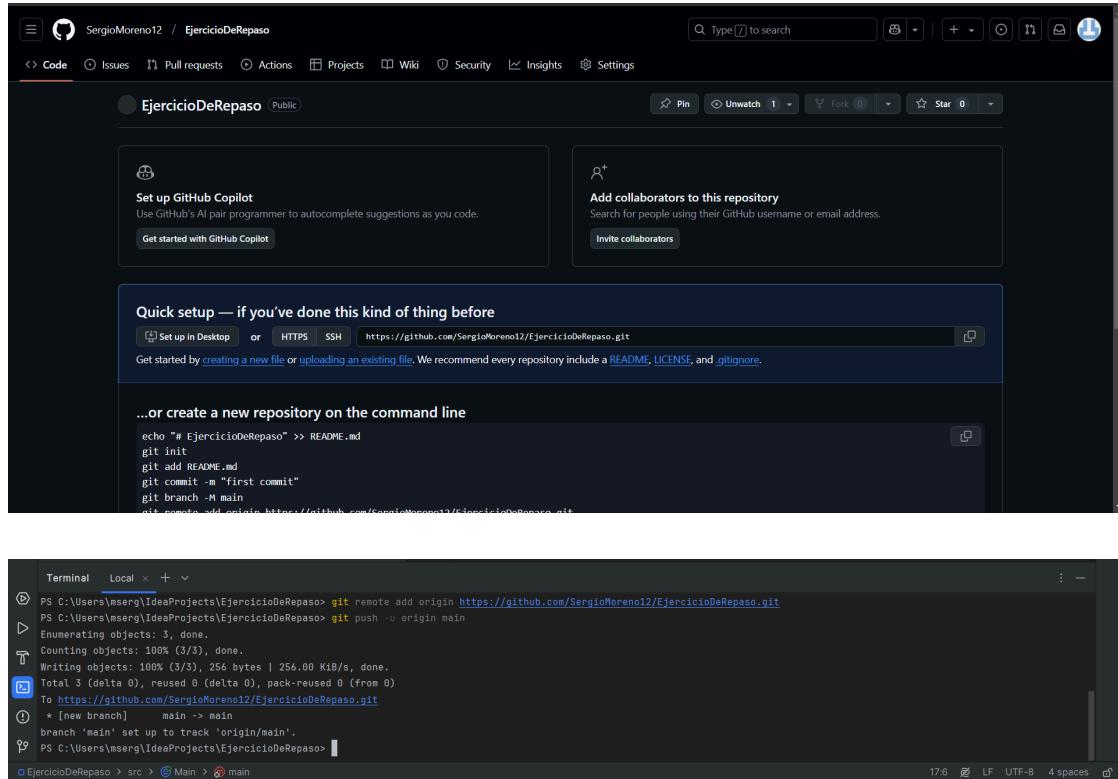
Este comando se utiliza para inicializar un nuevo repositorio de Git en un directorio específico. Al ejecutarlo, se crea un subdirectorio oculto llamado `.git`, donde Git almacenará todos los archivos necesarios para rastrear los cambios en el proyecto, incluyendo el historial de commits, ramas, configuraciones y otros metadatos.



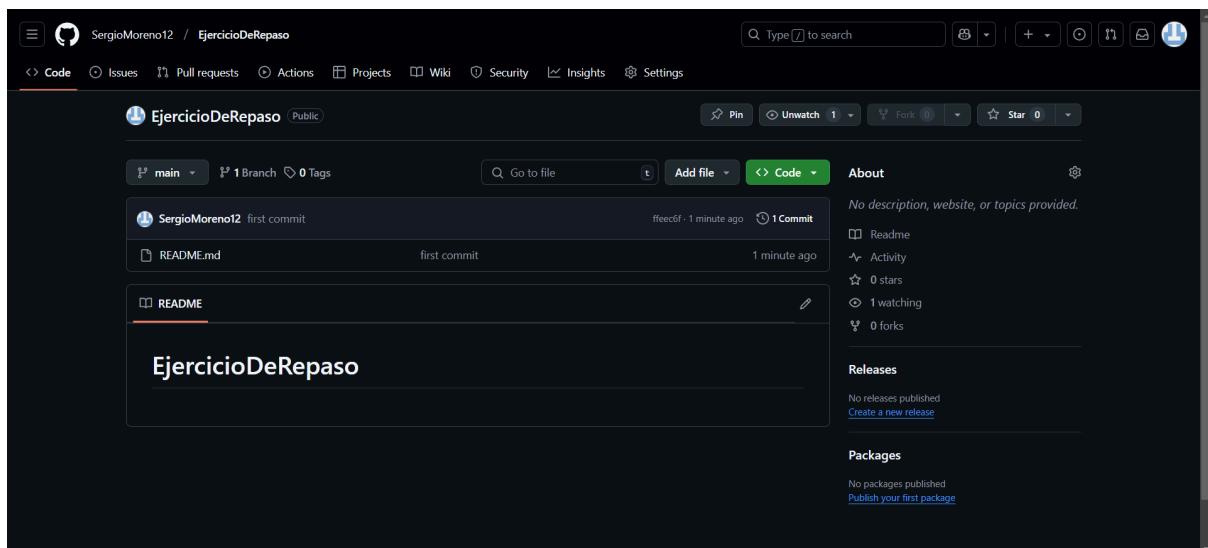
Comandos esenciales para gestionar un repositorio

Para empezar con la explicación del uso de comandos git, creamos un proyecto en intelliJ y un repositorio en GitHub.

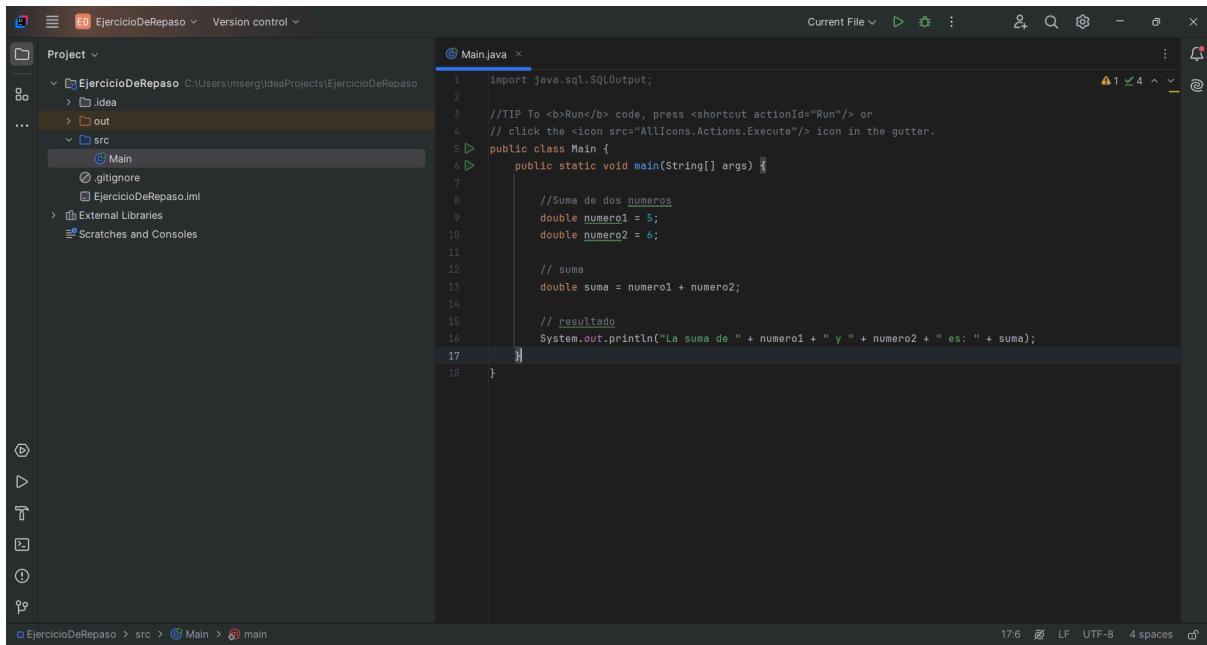
Para crear la conexión entre el repositorio que creamos en GitHub y el proyecto que tenemos en local, copiamos el código que nos proporciona GitHub y lo pegamos en la terminal del proyecto en IntelliJ.



Actualizamos el repositorio en GitHub para confirmar la conexión entre ellos.



Ya una vez que tenemos la conexión entre el repositorio y el proyecto en local, procedemos a realizar el ejercicio de suma de dos números.



```

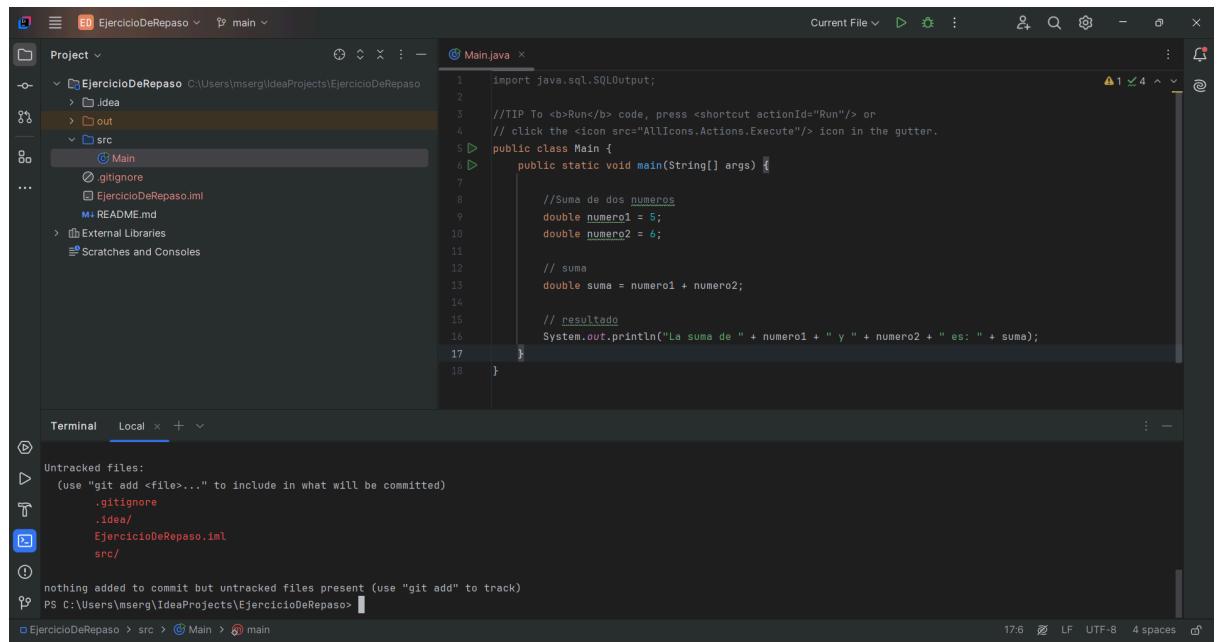
1 import java.sql.SQLOutput;
2
3 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
4 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7
8         //Suma de dos numeros
9         double numero1 = 5;
10        double numero2 = 6;
11
12        // suma
13        double suma = numero1 + numero2;
14
15        // resultado
16        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
17    }
18 }

```

Ahora procedemos a usar comandos de Git para guardar los cambios en el repositorio.

1. Git status

Permite ver el estado actual del repositorio, mostrando los archivos modificados, nuevos archivos sin seguimiento y aquellos que están en el área de preparación (staging). Es útil para verificar qué cambios se han realizado antes de hacer un commit.



```

main | Ejer EjercicioDeRepaso v
Project ▾ EjercicioDeRepaso C:\Users\mserg\IdeaProjects\EjercicioDeRepaso
  > .idea
  > out
  > src
    > Main
      > .gitignore
      > EjercicioDeRepaso.iml
      > README.md
      > External Libraries
      > Scratches and Consoles

Main.java x
1 import java.sql.SQLOutput;
2
3 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
4 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7
8         //Suma de dos numeros
9         double numero1 = 5;
10        double numero2 = 6;
11
12        // suma
13        double suma = numero1 + numero2;
14
15        // resultado
16        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
17    }
18 }

Terminal Local + ▾
Untracked files:
(use "git add <file>..." to include in what will be committed)
  .gitignore
  .idea/
  EjercicioDeRepaso.iml
  src/
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\mserg\IdeaProjects\EjercicioDeRepaso>

```

2. Git add .

Agrega todos los archivos modificados y nuevos al área de preparación, lo que significa que estarán listos para ser confirmados en el próximo commit. Se usa cuando queremos incluir todos los cambios en un solo paso, sin tener que añadir archivos individualmente.

The screenshot shows the IntelliJ IDEA IDE. In the top navigation bar, the project name 'EjercicioDeReposo' is selected. The left sidebar displays the project structure under 'src'. The main editor window shows the code for 'Main.java':

```

import java.sql.SQLOutput;
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        //Suma de dos numeros
        double numero1 = 5;
        double numero2 = 6;
        // suma
        double suma = numero1 + numero2;
        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

```

Below the editor is a terminal window titled 'Local' showing the command line output:

```

.idea/
EjercicioDeReposo.iml
src/
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.idea/vcs.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

3. Git commit -m "Actualización"

Guarda los cambios agregados al área de preparación en el historial del repositorio. El mensaje entre comillas ("") debe describir brevemente los cambios realizados. Se usa después de git add para confirmar los cambios.

```

import java.sql.SQLOutput;
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        //Suma de dos numeros
        double numero1 = 5;
        double numero2 = 6;

        // suma
        double suma = numero1 + numero2;

        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git commit -m "Suma de dos numeros"
[main 02cabla] Suma de dos numeros
7 files changed, 81 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 EjercicioDeReposo.iml
create mode 100644 src/Main.java
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>
EjercicioDeReposo > src > Main > main

```

4. Git push origin (Nombre de la rama)

Envía los commits confirmados en la rama actual al repositorio remoto, específicamente a la rama main.

```

import java.sql.SQLOutput;
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        //Suma de dos numeros
        double numero1 = 5;
        double numero2 = 6;

        // suma
        double suma = numero1 + numero2;

        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

create mode 100644 EjercicioDeReposo.iml
create mode 100644 src/Main.java
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git push origin main
Enumerating objects: 12, done.
Counting objects: 100%, done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.88 KiB | 213.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
   ffeec6f..02cabla  main -> main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>
EjercicioDeReposo > src > Main > main

```

Podemos verificar los cambios refrescando la página de GitHub

The screenshot shows a GitHub repository named 'EjercicioDeReposo' owned by 'SergioMoreno12'. The repository has 2 commits and 1 branch. The README file contains the text 'EjercicioDeReposo'.

```

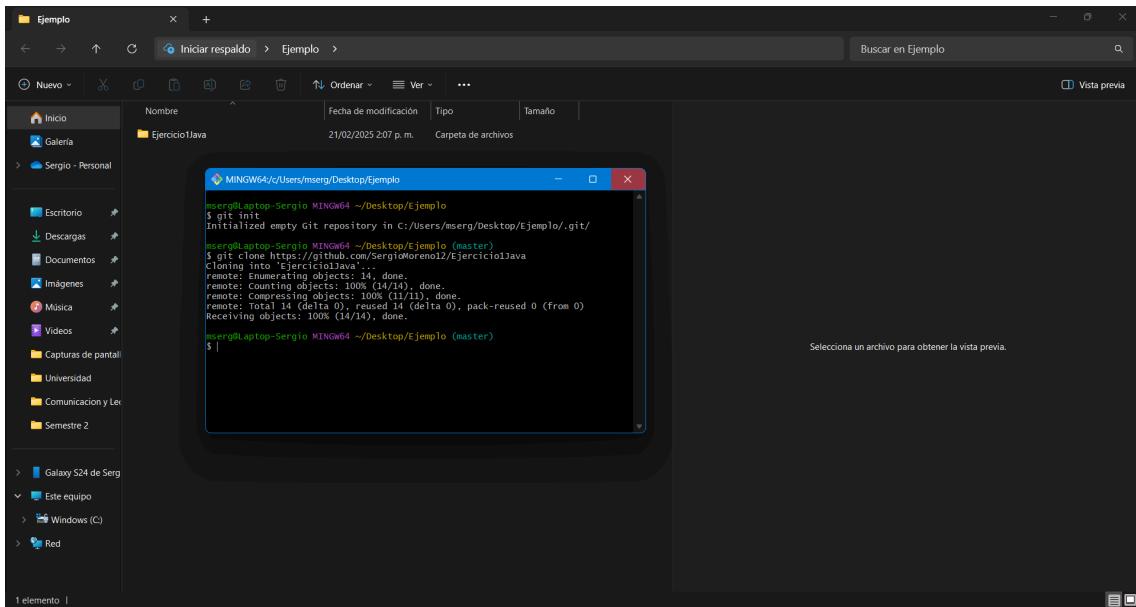
import java.sql.SQLOutput;
public class Main {
    public static void main(String[] args) {
        //Suma de dos numeros
        double numero1 = 5;
        double numero2 = 6;
        // SUMA
        double suma = numero1 + numero2;
        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

```

Como clonar un repositorio

El comando **git clone** se utiliza para copiar un repositorio remoto en el equipo local.

Descarga todos los archivos del repositorio, así como su historial completo de cambios y configuraciones de Git. El comando también configura automáticamente la conexión con el repositorio remoto, lo que permite a los usuarios realizar futuras actualizaciones o contribuciones sin necesidad de volver a configurarlo.



Como hacer un revert

Para hacer un git revert debemos saber el ID de los commits, para esto podemos usar los siguientes comandos:

1. Git log

Muestra el historial de commits con detalles como el autor, la fecha y el mensaje de cada uno. Es útil para rastrear cambios y entender la evolución del proyecto. Se puede complementar con opciones para filtrar o resumir la información.

The screenshot shows an IntelliJ IDEA interface. On the left, the project structure for 'EjercicioDeReposo' is visible, showing a 'src' directory containing a 'Main' package with a 'Main.java' file. The code in Main.java is:

```
public class Main {
    public static void main(String[] args) {
        double numero2 = 4;
        // suma
        double suma = numero1 + numero2;
        // resta
        double resta = numero1 - numero2;
    }
}
```

In the bottom-left corner, there's a terminal window with the following output:

```
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git log
commit 493022ff0b19395775d9d4d7e1236ba6ff24d (HEAD --> main, origin/main)
Merge: 02cab1a05699a
Author: SergioMoreno12 <ssmoreno@ucundinamarca.edu.co>
Date:   Wed Feb 19 23:13:05 2025 -0500

Merge pull request #1 from SergioMoreno12/RamaPrueba

Se agrego la resta

commit a65699a4ed22c686d62c090b29c9eb574deddf8a (origin/RamaPrueba)
Author: SergioMoreno12 <ssmoreno@ucundinamarca.edu.co>
Date:   Wed Feb 19 23:04:21 2025 -0500

Se agrego la resta

commit 02cab1a3f777eb1c6b205ea2c67da280bf143d20
Author: SergioMoreno12 <ssmoreno@ucundinamarca.edu.co>
Date:   Wed Feb 19 22:46:29 2025 -0500

Suma de dos numeros
```

2. Git reflog

Registra todas las acciones realizadas en el repositorio local, incluyendo cambios de rama y commits eliminados. Es útil para recuperar un commit perdido o revertir cambios accidentales. A diferencia de git log, incluye referencias a operaciones no visibles en la historia normal.

```

Main.java
5  public class Main {
6      public static void main(String[] args) {
7          double numero1 = 5;
8          double numero2 = 6;
9          // suma
10         double suma = numero1 + numero2;
11         // resta
12         double resta = numero1 - numero2;
}

Terminal Local (2) x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git reflog
493022f (HEAD -> main, origin/main) HEAD@{0}: pull origin main: Fast-forward
02cab1a HEAD@{1}: checkout: moving from RamaPrueba to main
805699a (origin/RamaPrueba) HEAD@{2}: commit: Se agrego la resta
02cab1a HEAD@{3}: checkout: moving from main to RamaPrueba
02cab1a HEAD@{4}: checkout: moving from RamaPrueba to main
02cab1a HEAD@{5}: checkout: moving from main to RamaPrueba
02cab1a HEAD@{6}: commit: Suma de dos numeros
ffeedcf HEAD@{7}: Branch: renamed refs/heads/master to refs/heads/main
ffeedcf HEAD@{8}: commit (initial): first commit
T PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

3. Git log --oneline

Muestra un historial de commits en un formato resumido, con un solo commit por línea. Usa el hash abreviado y el mensaje correspondiente, facilitando la visualización rápida de la evolución del proyecto. Se usa comúnmente para obtener una vista compacta del historial.

```

5  public class Main {
6      public static void main(String[] args) {
7          double numero2 = 6;
8
9          // suma
10         double suma = numero1 + numero2;
11
12         // resta
13         double resta = numero1 - numero2;
14
15     }
16 }

```

Windows PowerShell
Copyright © Microsoft Corporation. Todos los derechos reservados.
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. <https://aka.ms/PSWindows>

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> Git log --oneline
493022f (HEAD -> main, origin/main) Merge pull request #1 from SergioMoreno12/RamaPrueba
a05699a (origin/RamaPrueba) Se agrego la resta
02cab1a Suma de dos numeros
ffec6cf first commit
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

4. Git revert

Deshace un commit específico creando un nuevo commit que invierte sus cambios sin alterar el historial. Es útil cuando se quiere eliminar un cambio sin modificar el estado previo del repositorio.

Escribimos git revert y el ID del commit que averiguamos con los comandos explicados anteriormente. En seguida aparece este editor.

```

5  public class Main {
6      public static void main(String[] args) {
7          // resultado
8          System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
9          System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
10     }
11 }

```

Terminal Local (2) x Local x Local (3) x + ~

Revert "Ejemplo revert"
This reverts commit 4601a5666e132d5c7fb8e2dcdfcd045acd6861d6.

Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.

On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
new file: .gitignore
new file: .idea/.gitignore
modified: .idea/misc.xml
new file: .idea/modules.xml
new file: .idea/vcs.xml
deleted: .idea/workspace.xml
.git/COMMIT_EDITMSG [unix] (14:33 21/02/2025)
"/IdeaProjects/EjercicioDeReposo/.git/COMMIT_EDITMSG" [unix] 21L, 622B

Si realizamos un comentario escribimos :wq si no realizamos comentarios solamente

:q

```

public class Main {
    public static void main(String[] args) {
        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
        System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
    }
}

Terminal Local (2) < Local < Local (3) < +
escapeRevert "Ejemplo revert"
This reverts commit 4601a566e132d5c7fb8e2ddcf0d45acd6861d6.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#       new file: .gitignore
#       new file: .idea/.gitignore
#       modified: .idea/misc.xml
#       new file: .idea/modules.xml
#       new file: .idea/vcs.xml
#       deleted: .idea/workspace.xml
.git/COMMIT_EDITMSG[+]
:cq
1,6 Top

```

Escribimos :wq porque realizamos un comentario y enseguida damos enter.

```

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git log --oneline
4601a56 (HEAD -> main, origin/main) Ejemplo revert
493022f Merge pull request #1 from SergioLoMorenO12/RamaPrueba
a05699a (origin/RamaPrueba) Se agrego la resta
02ca0ba Suma de dos numeros
Freebsd first commit
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git revert 4601a56
[main 341854a] escapeRevert "Ejemplo revert"
 9 files changed, 58 insertions(+), 47 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 delete mode 100644 .idea/workspace.xml
 create mode 100644 EjercicioDeReposo.iml
 delete mode 100644 out/production/EjercicioDeReposo/Main.class
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

Creación de Ramas

La creación de ramas en GitHub permite desarrollar nuevas funcionalidades, corregir errores o probar cambios sin afectar la rama principal del proyecto. Cada rama actúa como una versión independiente del código, facilitando el trabajo en equipo y la gestión de actualizaciones.

1. Git branch

Se utiliza para listar todas las ramas disponibles en el repositorio y mostrar en cuál estamos actualmente.

```

Project: EjercicioDeReposo
Main.java
1 import java.sql.SQLOutput;
2
3 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
4 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7
8         //Suma de dos números
9         double numero1 = 5;
10        double numero2 = 6;
11
12        // suma
13        double suma = numero1 + numero2;
14
15        // resultado
16        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
17    }
18 }

Terminal Local + v
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.88 KiB | 213.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
  ffec0cf..02cabb1  main -> main
① PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* main
② PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>
③ EjercicioDeReposo > src > Main > main

```

2. Git switch -c (Nombre de la rama que vamos a crear)

Crea y cambia a una nueva rama en un solo paso. Es una alternativa más moderna a git checkout -b Nombre. Se usa cuando se quiere empezar a trabajar en una nueva rama sin cambiar manualmente después.

```

Project: EjercicioDeReposo
Main.java
1 import java.sql.SQLOutput;
2
3 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
4 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7
8         //Suma de dos números
9         double numero1 = 5;
10        double numero2 = 6;
11
12        // suma
13        double suma = numero1 + numero2;
14
15        // resultado
16        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
17    }
18 }

Terminal Local + v
Writing objects: 100% (11/11), 1.88 KiB | 213.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
  ffec0cf..02cabb1  main -> main
④ PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* main
⑤ PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git switch -c RamaPrueba
Switched to a new branch 'RamaPrueba'
⑥ PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* RamaPrueba
⑦ PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>
⑧ EjercicioDeReposo > src > Main > main

```

3. Git switch (Nombre de la Rama)

Cambia a una rama existente sin crear una nueva. Se usa cuando necesitamos movernos entre diferentes ramas en un repositorio.

```

EjercicioDeReposo main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git switch -c RamaPrueba
Switched to a new branch 'RamaPrueba'
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* RamaPrueba
  main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* RamaPrueba
  main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

4. Git push origin (Nombre de la rama)

Envía una rama específica al repositorio remoto. Se usa cuando hemos creado una nueva rama y queremos compartirla con otros colaboradores o simplemente mantener una copia en el servidor.

```

EjercicioDeReposo RamaPrueba
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* RamaPrueba
  main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git push origin RamaPrueba
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'RamaPrueba' on GitHub by visiting:
remote:   https://github.com/SergioMoreno12/EjercicioDeReposo/pull/new/RamaPrueba
remote:
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
 * [new branch]  RamaPrueba -> RamaPrueba
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

Verificamos la creación de la rama en GitHub

The screenshot shows a GitHub repository named 'EjercicioDeReposo'. The repository is public and contains two branches: 'main' and 'RamaPrueba'. The 'main' branch has several commits, all of which are related to summing numbers. The 'RamaPrueba' branch has one commit. The README file is present but contains no content.

Procedemos a realizar cambios en el código en la rama creada, en este caso agregamos la resta de dos números.

The screenshot shows an IDE interface with the 'Main.java' file open. The code has been modified to include subtraction logic. A terminal window at the bottom shows the command 'git push origin RamaPrueba' being run and completed successfully.

```

// suma
double suma = numero1 + numero2;

// resta
double resta = numero1 - numero2;
System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
}

```

Aplicamos los comandos de Git ya explicados anteriormente para guardar los cambios en la rama creada.

The screenshot shows the IntelliJ IDEA interface. On the left is the Project tool window with a tree view of files: EjercicioDeReposo (idea, out, src), .gitignore, EjercicioDeReposo.iml, README.md, External Libraries, and Scratches and Consoles. The src folder is expanded, showing Main.java. The code in Main.java is:

```

4 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7         //numeros
8         double numero1 = 5;
9         double numero2 = 6;
10
11        // suma
12        double suma = numero1 + numero2;
13
14        // resta
15        double resta = numero1 - numero2;
16
17        // resultado
18        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
19

```

The right side of the interface shows the Main.java editor with line numbers 4 to 19. Below the editor is a Terminal window with the following output:

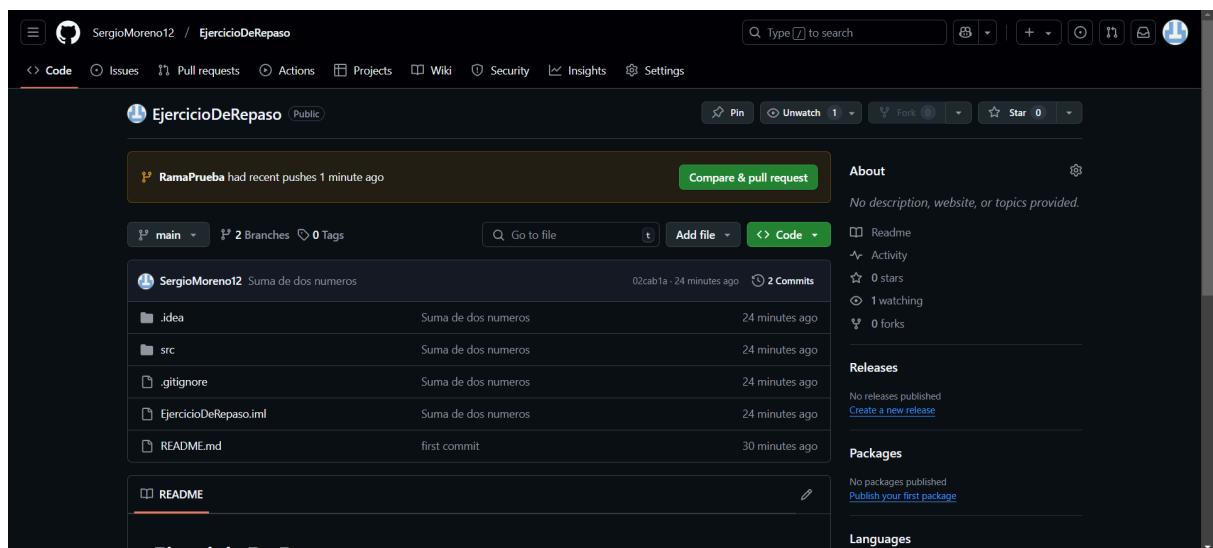
```

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git commit -m "Se agrego la resta"
[RamaPrueba a05699a] Se agrego la resta
 1 file changed, 5 insertions(+), 1 deletion(-)
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git push origin RamaPrueba
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 404 bytes | 404.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
 * [new branch]      RamaPrueba -> RamaPrueba
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

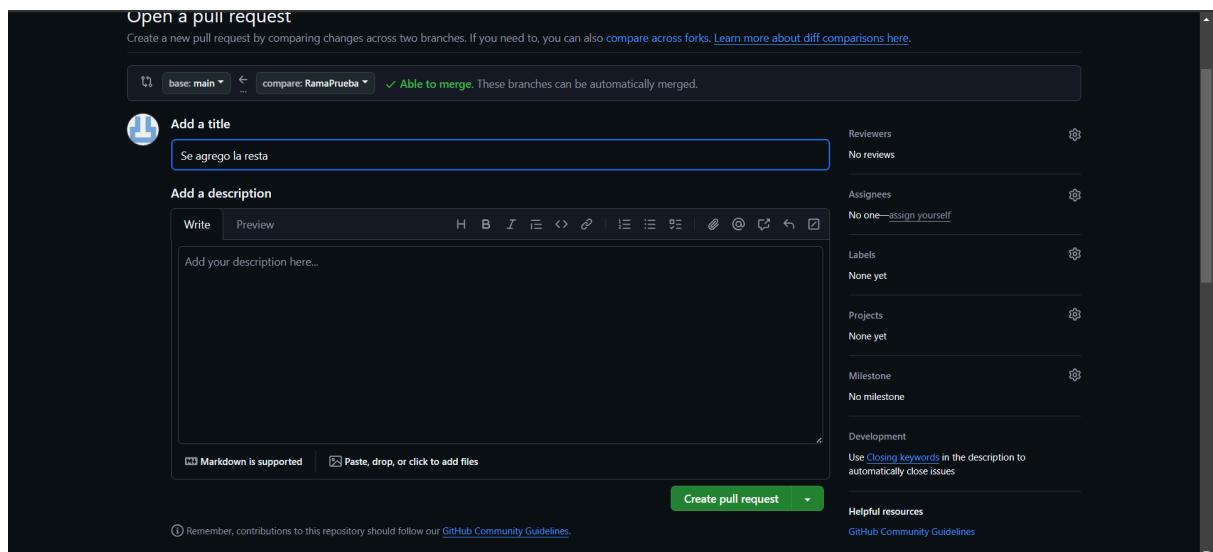
```

At the bottom right of the IDE, it says 11:1 LF UTF-8 4 spaces.

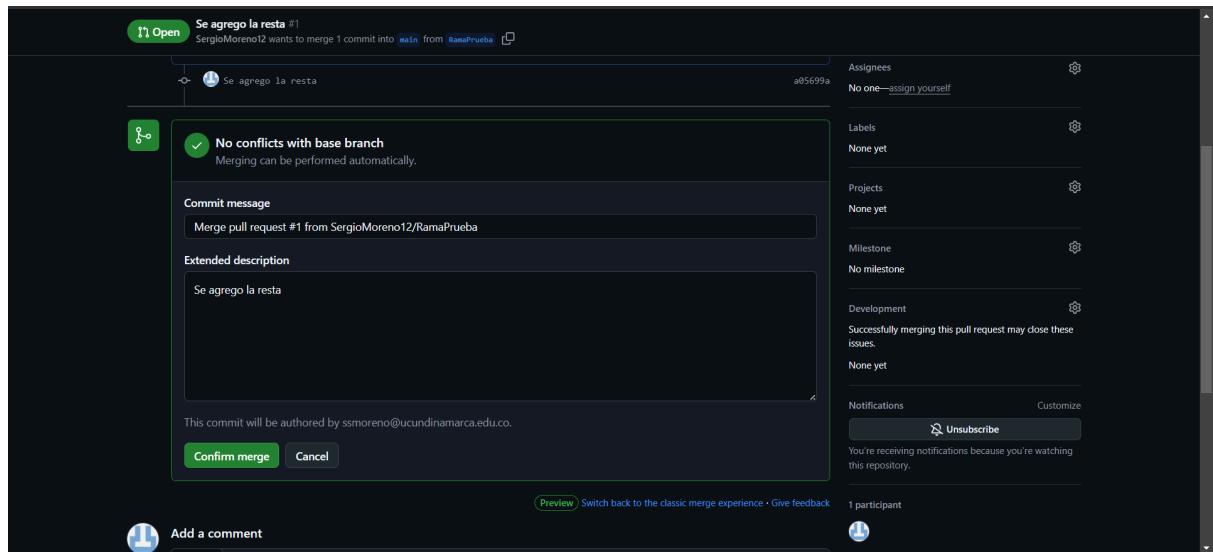
Al realizar esto, en GitHub nos va a aparecer la solicitud de un Pull Request (PR). En GitHub es una solicitud para fusionar cambios de una rama a otra, permitiendo revisión y discusión antes de integrarlos al código principal. Facilita la colaboración y asegura un desarrollo organizado.



Creamos la solicitud del Pull Request para que sea revisado por nuestros compañeros.



Cuando ya haya sido revisado, procedemos a hacer el merge. El merge en Git es el proceso de combinar los cambios de una rama con otra, integrando su historial de modificaciones en una única versión. Se usa comúnmente para fusionar una rama de desarrollo con la principal (main).



Verificamos la fusión de las ramas abriendo la rama main.

```

1 import java.sql.SQLOutput;
2
3 //TIP To <b>Run/b> code, press <shortcuts actionId="Run"/> or
4 // click the icon src="AllIcons.Actions.Execute"/> icon in the gutter.
5 public class Main {
6     public static void main(String[] args) {
7
8         //numeros
9         double numero1 = 5;
10        double numero2 = 6;
11
12        // suma
13        double suma = numero1 + numero2;
14
15        // resta
16        double resta = numero1 - numero2;
17
18        // resultado
19        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
20        System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
21    }
22 }

```

Para hacer el merge desde la consola se utiliza el siguiente comando:

5. Git merge

Permite fusionar los cambios de una rama en otra, integrando su historial. Se debe ejecutar desde la rama destino, asegurándose de que esté actualizada antes de la fusión.

```

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git merge main
Already up to date.
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

Ahora eliminamos la rama secundaria que habíamos creado con los siguientes comandos debido a que ya no es necesaria. Tanto de forma local como remota.

6. Git branch -D Nombre

Borra una rama local. Se usa cuando una rama ya no es necesaria y queremos eliminarla sin advertencias.

```

Project: EjercicioDeReposo
Main.java
Public class Main {
    public static void main(String[] args) {
        double numero2 = 6;
        // suma
        double suma = numero1 + numero2;

        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
    }
}

Terminal Local x + v
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
* RamaPrueba
  main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch
  RamaPrueba
* main
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch -d RamaPrueba
error: the branch 'RamaPrueba' is not fully merged
hint: If you are sure you want to delete it, run 'git branch -D RamaPrueba'
hint: Disable this message with "git config set advice.forceDeleteBranch false"
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git branch -D RamaPrueba
EjercicioDeReposo > src > Main > main
11:12 CRLF UTF-8 4 spaces

```

7. Git push origin --delete (nombre de la rama)

Se utiliza para eliminar una rama en el repositorio remoto. Esto es útil cuando una rama ya no es necesaria, como después de fusionar sus cambios en la rama principal.

```

Project: EjercicioDeReposo
Main.java
Public class Main {
    public static void main(String[] args) {
        // resultado
        System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
        System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
    }
}

Terminal Local (2) x Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git push origin --delete RamaEjemplo
To https://github.com/SergioMoreno12/EjercicioDeReposo.git
 - [deleted]   RamaEjemplo
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

8. Git pull origin main

Descarga y fusiona los cambios más recientes de la rama main desde el repositorio remoto a nuestro repositorio local. Se usa para mantener el código actualizado con los cambios hechos por otros.

```

Deleted branch RamaPrueba (was a05699a).
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 903 bytes | 129.00 KiB/s, done.
From https://github.com/SergioMoreno12/EjercicioDeReposo
 * branch      main      -> FETCH_HEAD
  02cab1a..493022f  main      -> origin/main
Updating 02cab1a..493022f
Fast-Forward
①  src/Main.java | 6 +++++-
 1 file changed, 5 insertions(+), 1 deletion(-)
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

```

9. Git fetch --all

Descarga las actualizaciones de todas las ramas remotas sin fusionarlas con las locales. Es útil para obtener cambios recientes del repositorio antes de hacer un merge o rebase. De esta manera, se puede revisar el estado del proyecto sin alterar la rama actual.

```

5  public class Main {
6      public static void main(String[] args) {
18         // resultado
19         System.out.println("La suma de " + numero1 + " y " + numero2 + " es: " + suma);
20         System.out.println("La resta de " + numero1 + " y " + numero2 + " es: " + resta);
21
22         System.out.println("Hola mundo");
23     }
24 }

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. <https://aka.ms/PSWindows>
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo> git fetch --all
PS C:\Users\mserg\IdeaProjects\EjercicioDeReposo>

10. Git Branch -r

Muestra todas las ramas remotas del repositorio, lo que permite conocer qué ramas existen en el servidor. Es útil para verificar el trabajo de otros colaboradores.

Además, facilita la creación de una rama local basada en una remota.

```

origin/HEAD -> origin/main
origin/RamaEjemplo
origin/RamaPrueba
origin/main

```

Enlace de GitHub para acceder al ejercicio:

<https://github.com/SergioMoreno12/EjercicioDeReparo>

Porcentaje de IA utilizada: 30%

Conclusión

En conclusión, Git es una herramienta muy importante para el control de versiones y la gestión eficiente del código en proyectos de desarrollo de software. Cada uno de los comandos utilizados son importantes para la organización del trabajo, le permite a los desarrolladores gestionar cambios, crear ramas para trabajo y sincronizar el código con repositorios remotos. Estos comandos nos facilitan la organización de los archivos dentro del repositorio, el registro de cambios así como compartirlos de manera efectiva. Además, la creación y gestión de ramas permiten un flujo de trabajo estructurado, evitando conflictos y asegurando un desarrollo ordenado. En general, dominar estos comandos básicos de Git es fundamental para optimizar el trabajo en equipo y garantizar el funcionamiento del código fuente en cualquier proyecto de desarrollo.

Referencias

HubSpot. (s. f.). Lista de comandos de GitHub más utilizados. HubSpot.

<https://blog.hubspot.es/website/comandos-github>

Atlassian. (s. f.). *Glosario de Git: comandos y terminología clave*. Atlassian.

<https://www.atlassian.com/es/git/glossary#commands>

Apiumhub. (s. f.). 20 comandos básicos de Git que todo ingeniero QA debe conocer.

Apiumhub.

<https://apiumhub.com/es/tech-blog-barcelona/20-comandos-basicos-de-git-que-todo-ingenierto-qa-debe-conocer/>