

Manual de Uso de Git en IntelliJ

Ingeniería de sistemas y computación

Integrantes

Samuel Castellanos Rivera

Profesor

William Javier Matallana Porras

Universidad de Cundinamarca – UDEC

Chía

2025

Tabla de contenido

Introducción	3
Objetivos.....	3
Desarrollo.....	4
Git Config.....	4
Repositorio.....	5
Git init.....	6
Git Status	6
Git Add.....	7
Git Commit	7
Git Push.....	8
Git Clone	8
Git Branch	8
Git Pull.....	11
Git Fetch.....	11
Git Revert	11
Git Rebase	11
Git Reflog	12
Como asociar un archivo a un repositorio existente.....	12
Conclusiones	14
Uso de IA:	14
Link repositorio:	14
Referencias.....	15

Introducción

En el desarrollo de software, el control de versiones es una herramienta fundamental para ejecutar cambios en el código y facilitar el trabajo colaborativo, Git es uno de los sistemas más utilizados de control de versiones, y su incorporación en entornos de desarrollo como IntelliJ nos permite optimizar la circulación de trabajo. IntelliJ ofrece una interfaz gráfica sutil que simplifica el uso de comandos de Git, como clonar repositorios, hacer commits, crear ramas y resolver conflictos.

El empleo de Git en IntelliJ es clave para los desarrolladores, ya que permite mantener un historial de cambios, trabajar en equipo de manera eficiente y evitar el riesgo de pérdida de código. Además, adecua herramientas visuales para comparar versiones, hacer merges (fusionar) y gestionar repositorios sin necesidad de recurrir a la línea de comandos. En el documento, se explorarán los principales comandos de Git en IntelliJ y como estos nos posibilitan la organización y el control del código en proyectos de programación.

Objetivos

Entender la integración de Git en IntelliJ para gestionar versiones de código de manera eficiente dentro del entorno de desarrollo.

Investigar los principales comandos Git en IntelliJ, incluyendo la clonación, commits, creación de ramas y resolución de conflictos.

Impulsar buenas prácticas en el uso de Git dentro de proyectos de programación, mejorando el trabajo colaborativo y la organización del código fuente.

Desarrollo

Git Config

Primero debemos hacer una configuración básica en IntelliJ, iniciando por un “git config –global user.name “dadada””, para configurar el nombre del usuario.

```
Terminal 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.name "Samuel Catellanos Rivera"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Ahora, usaremos el “git config –global user.email dadada@mail.com” para configurar el email del usuario.

```
Terminal 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.name "Samuel Catellanos Rivera"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.email castellanosriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Para comprobar que nos haya quedado bien hecho, debemos hacer un “git config --list” para asegurarnos que todo esté bien.

```
Terminal 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.name "Samuel Catellanos Rivera"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.email castellanosriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global --list
user.name=Samuel Catellanos Rivera
user.email=castellanosriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Como dato extra tenemos el “git config --global --unset user.name” y también tenemos el “git config --global --unset user.email” que los usamos para desasociar el nombre y el correo del usuario.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global --unset user.name
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global --unset user.email
```

Así mismo, los podemos reconfigurar todo y asegurarnos que todo allá quedado bien.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.name "Samuel Castellanos Rivera"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.email castellanosriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global --list
user.name=Samuel Castellanos Rivera
user.email=castellanosriverasamuel@gmail.com
```

Por último, tenemos el “git config –list”, en el cual nos muestra la configuración global de Git en el sistema, que en este podemos observar ajustes del manejo de archivos con Git, la gestión de credenciales, la conversión de finales de línea, el uso de “schannel” para conexiones seguras y la estrategia de pull. También define por defecto la rama “master” y muestra el nombre y correo del usuario configurado para los commits.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> 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=schannel
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=Samuel Castellanos Rivera
user.email=castellanosriverasamuel@gmail.com
```

Repositorio

Luego de esto, podemos iniciar con el repositorio, aquí primero crearemos el repositorio en Github, donde nos darán una serie de líneas de códigos que usaremos en el terminal, en ellas esta “git init” que nos sirve para iniciar un repositorio Git en el directorio actual, también un “git add README.md” para que añada el archivo “README.md” al área de preparación, además de eso un “git commit -m “first commit”” para realizar el primer commit con el mensaje dado, asimismo un “git branch -M main” para cambiar el nombre de la rama principal a “main”, también un “git remote add origin <https://github.com/samucr27/Ejemplo-PDF.git>” para enlazar el repositorio local con el repositorio remoto, y por ultimo un “git push -u origin main” que sube el contenido de la rama main en el repositorio remoto.

```
Terminal Local < + v
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.name "Samuel Castellanos Rivera"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global user.email castellanoriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git config --global --list
user.name=Samuel Castellanos Rivera
user.email=castellanoriverasamuel@gmail.com
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> echo "e Ejemplo-PDF" >> README.md
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git init
Reinitialized existing Git repository in C:/Users/caste/Programacion II/Ejercicios/Ejemplo PDF/.git/
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add README.md
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git commit -m "first commit"
[master d6156aa] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch -M main
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git remote add origin https://github.com/samucr27/Ejemplo-PDF.git
error: remote origin already exists.
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 146.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/samucr27/Ejemplo-PDF.git
fc97cdd..d6156aa main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git init

Usamos el “git init” para inicializar un nuevo repositorio en un directorio, eso transforma una carpeta local en un repositorio de Git.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git init
Reinitialized existing Git repository in C:/Users/caste/Programacion II/Ejercicios/Ejemplo PDF/.git/
```

Git Status

Ahora vamos a revisar si todo quedo subido en el repositorio con un “git status” que sirve para identificar si hay archivos modificados, añadidos o eliminados.

```
Terminal Local < + v
1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch -M main
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git remote add origin https://github.com/samucr27/Ejemplo-PDF.git
error: remote origin already exists.
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 146.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/samucr27/Ejemplo-PDF.git
fc97cdd..d6156aa main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .idea/
        Ejemplo PDF.iml
        src/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Add

Con el “git add .” se añadirán los archivos y modificaciones nuevas en el área de preparación, para que estén listos para el commit.

```
Terminal Local x 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   .idea/.gitignore
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/vcs.xml
    new file:   Ejemplo PDF.iml
    new file:   src/Main.java

PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Commit

Después de esto, haremos un “git commit -m “Se agrega clase””, que nos sirve para registrar los cambios añadidos en el área de preparación en el historial del repositorio reuniéndolo con un mensaje descriptivo.

```
Terminal Local x Local (2) x + v

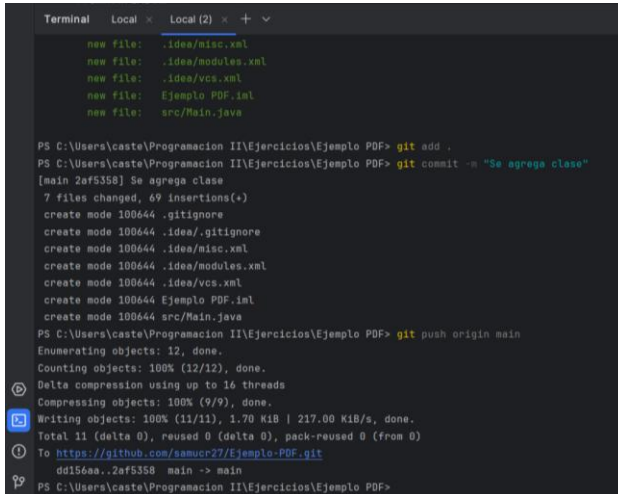
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   .idea/.gitignore
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/vcs.xml
    new file:   Ejemplo PDF.iml
    new file:   src/Main.java

PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git commit -m "Se agrega clase"
[main 2ef0d58] Se agrega clase
 7 files changed, 69 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 Ejemplo PDF.iml
 create mode 100644 src/Main.java
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Push

Por último, utilizaremos un “git push origin main” para que los cambios en el repositorio local se suban al repositorio remoto, después de eso podemos revisar el repositorio remoto para observar que ya todo quedo actualizado.

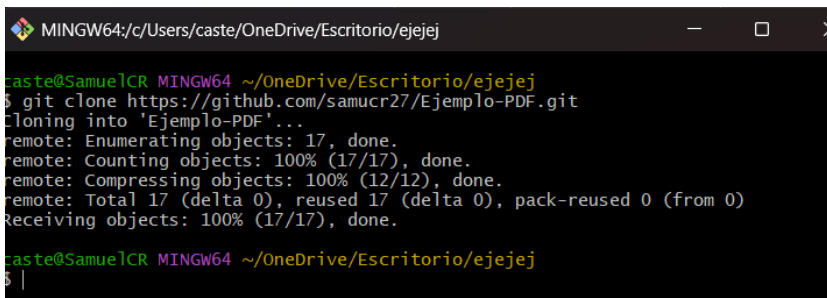


```
Terminal Local x Local (2) x + v
new file:   .idea/misc.xml
new file:   .idea/modules.xml
new file:   .idea/vcs.xml
new file:   Ejemplo PDF.iml
new file:   src/Main.java

PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git commit -m "Se agrega clase"
[main 2af5358] Se agrega clase
7 files changed, 69 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 Ejemplo PDF.iml
create mode 100644 src/Main.java
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.70 KiB | 217.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/samucr27/Ejemplo-PDF.git
 dd156aa..2af5358  main -> main
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Clone

Después de eso si queremos clonar un repositorio de algún compañero, abriremos la carpeta donde queremos descargar el archivo, después de eso abriremos un Git Bash, ya después de eso pegaremos el enlace y ya quedaría descargado.

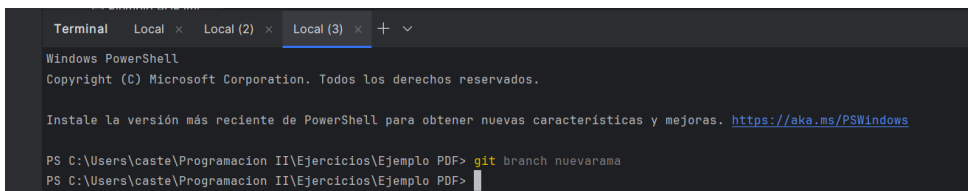


```
MINGW64/c:/Users/caste/OneDrive/Escritorio/ejejej
caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/ejejej
$ git clone https://github.com/samucr27/Ejemplo-PDF.git
Cloning into 'Ejemplo-PDF'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 17 (delta 0), reused 17 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (17/17), done.

caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/ejejej
$ |
```

Git Branch

Si queremos crear una nueva rama para cuando hay más de 2 personas trabajando en un proyecto, haremos un “git branch nueva rama” para crear una nueva rama.



```
Terminal Local x Local (2) x Local (3) 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```


Para asegurarnos que quedo creada usaremos un “git branch” para verificar las ramas.

```
Terminal Local x Local (2) x Local (3) 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch
* main
  nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Ya después de eso, usaremos un “git switch nuevarama” que se usa para ir cambiando entre las distintas ramas que estén creadas en el proyecto.

```
Terminal Local x Local (2) x Local (3) 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch
* main
  nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git switch nuevarama
Switched to branch 'nuevarama'
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Así mismo, tenemos el “git switch -c nuevarama” que nos sirve para crear y cambiar una nueva rama en un solo paso, ya que es una alternativa más moderna.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git switch -c nuevarama
Switched to a new branch 'nuevarama'
```

Para subir la nueva información que tengamos en la rama, usaremos un “git push origin nuevarama”, que nos sirve para subir un repositorio local a uno remoto.

```
Terminal Local x + v
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git status
On branch nuevarama
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/Main.java

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git commit -m "nuevarama"
[nuevarama 8ad7755] nuevarama
1 file changed, 1 insertion(+)
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push origin nuevarama
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 373 bytes | 186.00 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/samucc27/Ejemplo-PDF.git
2af5358..8ad7755 nuevarama -> nuevarama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Después de eso podemos hacer un merge & PR (pull request) en GitHub para unificar las dos ramas, ahí mismo nos dirá si queremos eliminar la rama “nuevarama”, pero al eliminarla la eliminaremos de forma remota y no local.

Por eso de forma local usaremos “git branch -D newerama” para eliminar el repositorio local, una vez ya este fusionado con el main.

```
Terminal Local x + v
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git add .
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git commit -m "nuevarama"
[nuevarama 8ad7755] newerama
1 file changed, 1 insertion(+)
```

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push origin newerama
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 373 bytes | 186.00 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/samucr27/Ejemplo-PDF.git
2af5358..8ad7755 newerama -> newerama
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch -D newerama
error: the branch 'nuevarama' is not fully merged
hint: If you are sure you want to delete it, run 'git branch -D newerama'
hint: Disable this message with "git config set advice.forceDeleteBranch false"
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch -D newerama
Deleted branch newerama (was 8ad7755).
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

También, tenemos un “git branch -r” para listar todas las ramas remotas que existen en el repositorio en el que se está conectado.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git branch -r
origin/HEAD -> origin/main
origin/ejemplo1
origin/ejemplo2
origin/main
origin/nuevarama
```

De igual manera, también tenemos el “git push origin --delete ejemplo1” que nos sirve para eliminar una rama remota en un repositorio. Después de ingresarla en el terminal vamos a GitHub y comprobamos que quede eliminada la rama.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push origin --delete ejemplo1
To https://github.com/samucr27/Ejemplo-PDF.git
- [deleted]          ejemplo1
```

Git Pull

Por último, para descargar los cambios más recientes en el repositorio remoto, usaremos “git pull origin main” para fusionarlos con el repositorio local.

```
Terminal Local x + v
Deleted branch nuevarama (was 8ad7755).
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> 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), 893 bytes | 178.00 KiB/s, done.
From https://github.com/samucr27/Ejemplo-PDF
* branch main -> FETCH_HEAD
2af5358..e47adde main -> origin/main
Updating 2af5358..e47adde
Fast-forward
 src/Main.java | 1 +
 1 file changed, 1 insertion(+)
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Fetch

También tenemos el “git fetch --all” que nos sirve para descargar otras ramas del repositorio sin llegar a fusionarlas con la rama principal.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git fetch --all
From https://github.com/samucr27/Ejemplo-PDF
* [new branch] ejemplo1 -> origin/ejemplo1
* [new branch] ejemplo2 -> origin/ejemplo2
```

Git Revert

Para añadir, también tenemos otros dos comandos que usaremos en Git, que son “git log --oneline” para ver el historial de commits, después de eso usaremos un “git revert fc97cdd” para deshacer un commit, creando uno nuevo que revierte los cambios, manteniendo el historial.

```
Terminal 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\caste\Programacion II\Ejercicios\Ejemplo PDF> git log --oneline
7d2d51d (HEAD -> main, origin/main) Guardando cambios antes del revert
e47adde Merge pull request #1 from samucr27/nuevarama
8ad7755 (origin/nuevarama) nuevarama
2af5358 Se agrega clase
e47adde first commit
fc97cdd first commit
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git revert fc97cdd
Revert fc97cdd
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git push origin main
Everything up-to-date
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF>
```

Git Rebase

Usamos el “git rebase” para reescribir el historial de commits, transportando una rama sobre otra como si se hubieran creado en la misma rama principal.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git rebase
Current branch main is up to date.
```

Git Reflog

Acá usaremos un “git reflog” para que se nos muestre un historial de todos los cambios que se han hecho, esto nos podría servir mucho para recuperar commits o volver a un estado anterior.

```
PS C:\Users\caste\Programacion II\Ejercicios\Ejemplo PDF> git reflog
7e2a51d (HEAD -> main, origin/main, origin/ejemplo2, origin/HEAD) HEAD@{0}: checkout: moving from nuevarama to main
7e2a51d (HEAD -> main, origin/main, origin/ejemplo2, origin/HEAD) HEAD@{1}: checkout: moving from main to nuevarama
7e2a51d (HEAD -> main, origin/main, origin/ejemplo2, origin/HEAD) HEAD@{2}: commit: Guardando cambios antes del revert
e47adde HEAD@{3}: pull origin main: Fast-forward
2af5358 HEAD@{4}: checkout: moving from nuevarama to main
8ad7755 (origin/nuevarama) HEAD@{5}: commit: nuevarama
2af5358 HEAD@{6}: checkout: moving from main to nuevarama
2af5358 HEAD@{7}: checkout: moving from nuevarama to main
2af5358 HEAD@{8}: checkout: moving from main to nuevarama
2af5358 HEAD@{9}: commit: Se agrega clase
dd156aa HEAD@{10}: Branch: renamed refs/heads/main to refs/heads/main
dd156aa HEAD@{12}: commit: first commit
fc97cdd HEAD@{13}: Branch: renamed refs/heads/master to refs/heads/main
fc97cdd HEAD@{15}: commit (initial): first commit
```

Como asociar un archivo a un repositorio existente

Para subir un archivo a un repositorio existente, primero debemos asociar el repositorio con la carpeta donde tenemos el archivo, abriendo un Bash en la carpeta para pegar el código que hay en el repositorio.

```
$ echo "# ManualUsuarioGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/samucr27/ManualUsuarioGit.git
git push -u origin main
Reinitialized existing Git repository in C:/Users/caste/OneDrive/Escritorio/archivo a subir/.git/
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
[main 966fd5e] first commit
 1 file changed, 1 insertion(+)
error: remote origin already exists.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 430 bytes | 107.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/samucr27/ManualUsuarioGit.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Después de esto, haremos un “git init” para revisar si el archivo esta en la carpeta, después de eso haremos un “git add .” para añadir las modificaciones que tenga el archivo, ahora podemos hacer un “git commit -m ”Archivo Pdf”” para que se agregue el commit, ya por último haremos un “git push origin main” para que el archivo ya se suba al repositorio remoto.

```
caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/archivo a subir (main)
$ git init
Reinitialized existing Git repository in C:/Users/caste/OneDrive/Escritorio/archivo a subir/.git/

caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/archivo a subir (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Manual de Uso de Git en IntelliJ_.pdf

nothing added to commit but untracked files present (use "git add" to track)

caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/archivo a subir (main)
$ git add .

caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/archivo a subir (main)
$ git commit -m "Archivo Pdf"
[main 07a78a0] Archivo Pdf
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Manual de Uso de Git en IntelliJ_.pdf

caste@SamuelCR MINGW64 ~/OneDrive/Escritorio/archivo a subir (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 948.84 KiB | 13.36 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/samucr27/ManualUsuarioGit.git
 966fd5e..07a78a0  main -> main
```

Conclusiones

Como pudimos observar, la integración de Git en IntelliJ facilita la gestión del código fuente, permitiendo a los desarrolladores mantener un historial de cambios y trabajar en equipo de manera eficiente.

La interfaz gráfica de IntelliJ optimiza el uso de Git al proporcionar herramientas visuales que simplifican tareas como la clonación de repositorios, la creación de ramas y la solución de conflictos, reduciendo la necesidad de utilizar la línea de comandos.

La sincronización constante del código con el código remoto asegura que todos los miembros del proyecto trabajen con la versión más actualizada del repositorio, evitando conflictos y pérdida de información.

Uso de IA:

En este trabajo, utilice un 25% de IA para guiarme como hacer la introducción, como redactar las referencias y para investigar el uso de algunos comandos.

Link repositorio:

<https://github.com/samucr27/Ejemplo-PDF.git>

Referencias

Git SCM. (s.f.). *Apéndice C: Comandos de Git - Seguimiento Básico.* <https://git-scm.com/book/es/v2/Ap%C3%A9ndice-C:-Comandos-de-Git-Seguimiento-B%C3%A1sico>

Dasdo. (s.f.). *Comandos básicos de Git.* GitHub Gist.
<https://gist.github.com/dasdo/9ff71c5c0efa037441b6>

Hostinger. (s.f.). *Comandos de Git: Lista de los más útiles con ejemplos.*
<https://www.hostinger.co/tutoriales/comandos-de-git>

Atlassian. (s.f.). *Glosario de Git: Comandos esenciales.*
<https://www.atlassian.com/es/git/glossary#commands>