

- #### Principales conceptos de Git
- **Repository**: Almacén de archivos e historial
 - **Commit**: Conjunto de cambios
 - **Rama (Branch)**: Línea de desarrollo independiente
 - **Fusión (Merge)**: Combinación de ramas
 - **Clonación (Clone)**: Copia exacta de un repositorio remoto

Comandos básicos de Git

```
```bash
git init # Inicializa un repositorio
git clone [URL] # Clona remoto
git add [archivo] # Prepara archivo
git commit -m "msg" # Commit
git status # Estado
git log # Historial
git push # Envía al remoto
git pull # Trae cambios remotos
git branch # Muestra ramas
git merge [rama] # Fusiona rama
```
[
```

Comandos básicos de Git

```
```bash
git init # Inicializa un repositorio
git clone [URL] # Clona remoto
git add [archivo] # Prepara archivo
git commit -m "msg" # Commit
git status # Estado
git log # Historial
git push # Envía al remoto
git pull # Trae cambios remotos
git branch # Muestra ramas
git merge [rama] # Fusiona rama
```
---
```

Creación de Repos 1: Crearlo en GitHub, clonarlo localmente y subir cambios

1. Crear una cuenta en <https://github.com>
2. Crear un nuevo repositorio en dicha página con el Readme.md por defecto
3. Clonar el repo remoto en un nuevo directorio local

```
```bash
git clone URL
git status
```
```

```

4. Editar archivo Readme.md, agregar algunas líneas con texto a dicho archivo.

```
```bash
git status
```
```

```

5. Editar (crearlo si no existe) el archivo .gitignore

- Agregar *.bak

6. Crear un commit y proveer un mensaje descriptivo.

```
```bash
git add .
git commit -m "Texto"
git status
```
```

```

7. Hacer un push al repositorio remoto.

```
```bash
git push origin main
```
```

```

Creación de Repos 2: Crearlo localmente y subirlo a GitHub

1. Crear un repo local en un nuevo directorio

```
```bash
mkdir demo2tp01
cd demo2tp01
git init
````
```

```
C:\Users\arnon>cd Downloads

C:\Users\arnon\Downloads>mkdir demo2tp01

C:\Users\arnon\Downloads>cd demo2tp01

C:\Users\arnon\Downloads\demo2tp01>git init
Initialized empty Git repository in C:/Users/arnon/Downloads/demo2tp01/.git/

C:\Users\arnon\Downloads\demo2tp01>
```

2. Agregar archivo Readme.md, agregar algunas líneas con texto a dicho archivo.

```
C:\Users\arnon\Downloads\demo2tp01>echo # Mi Proyecto > README.md
```

3. Crear archivo .gitignore

```
C:\Users\arnon\Downloads\demo2tp01>type nul > .gitignore
```

4. Crear un commit y proveer un mensaje descriptivo.

```
```bash
git add .
git commit -m "Texto"
```

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
README.md

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

C:\Users\arnon\Downloads\demo2tp01>git add .

C:\Users\arnon\Downloads\demo2tp01>git commit -m "Agregamos un git ignore y
un readme"
[master (root-commit) 14fefc2] Agregamos un git ignore y un readme
 2 files changed, 1 insertion(+)
 create mode 100644 .gitignore
 create mode 100644 README.md

C:\Users\arnon\Downloads\demo2tp01>git status
On branch master
nothing to commit, working tree clean
```

5. Crear repo en GitHub y asociar

```
```bash
git remote add origin <URL>
git branch -M main
git push -u origin main

C:\Users\arnon\Downloads\demo2tp01>git remote add origin https://github.com/
ArnonNahmias/demo2tp01.git

C:\Users\arnon\Downloads\demo2tp01>git branch -M main

C:\Users\arnon\Downloads\demo2tp01>git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 287 bytes | 143.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ArnonNahmias/demo2tp01.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
```

The screenshot shows a GitHub repository named 'demo2tp01' owned by 'AmonNahmias'. The repository is public and has 0 stars, 0 forks, and 0 releases. The README file contains the text 'Mi Proyecto'.

## ## Ramas

Una rama en Git es una línea independiente de desarrollo que permite a los desarrolladores crear, editar y probar cambios sin afectar directamente la rama principal o "master". Cada rama representa una versión del repositorio con su propio conjunto de cambios.

### - Ver ramas:

```
```bash
git branch -a
```

```

### - Crear rama:

```
```bash
git branch newFeature
```

```

### - Cambiarnos a la nueva rama

```
```bash
git checkout newFeature
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git branch -a
fatal: a branch named '-a' already exists

C:\Users\arnon\Downloads\demo2tp01>git branch newFeature
fatal: a branch named 'newFeature' already exists

C:\Users\arnon\Downloads\demo2tp01> git checkout newFeature
Switched to branch 'newFeature'
```

- Hacemos un cambio en un archivo, comiteamos y vemos la diferencia

```
```bash
git add .
git commit -m "Cambio en el Branch"
git diff main newFeature
```
```

```
C:\Users\arnon\Downloads\demo2tp01>git status
On branch newFeature
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: README.md

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

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\arnon\Downloads\demo2tp01>git add .
warning: in the working copy of '.idea/encodings.xml', LF will be replaced by CRLF the next time Git touches it
```

```
C:\Users\arnon\Downloads\demo2tp01>git commit -m "Cambio en el branch new feature y modificamos el readme"
[newFeature d0b77f4] Cambio en el branch new feature y modificamos el readme
 5 files changed, 33 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/encodings.xml
 create mode 100644 .idea/indexLayout.xml
 create mode 100644 .idea/vcs.xml

C:\Users\arnon\Downloads\demo2tp01>git status
On branch newFeature
nothing to commit, working tree clean
```

```
C:\Users\arnon\Downloads\demo2tp01>git dif main newFeature
git: 'dif' is not a git command. See 'git --help'.

The most similar commands are
 diff
 config
 difftool
 init
 refs
```

```
C:\Users\arnon\Downloads\demo2tp01>git diff main newFeature
diff --git a/.idea/.gitignore b/.idea/.gitignore
new file mode 100644
index 000000..2177ffc
--- /dev/null
+++ b/.idea/.gitignore
@@ -0,0 +1,13 @@
+ # Default ignored files
+/shelf/
+/workspace.xml
+## Rider ignored files
+/projectSettingsUpdater.xml
+/modules.xml
+/contentModel.xml
+/.idea.demo2tp01.iml
+## Editor-based HTTP Client requests
+/httpRequests/
+## Datasource local storage ignored files
+/dataSources/
+/dataSources.local.xml
diff --git a/.idea/encodings.xml b/.idea/encodings.xml
new file mode 100644
index 000000..df87cf9
--- /dev/null
+++ b/.idea/encodings.xml
@@ -0,0 +1,4 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<project version="4">
+ <component name="Encoding" addBOMForNewFiles="with BOM under Windows, with no BOM otherwise" />
+</project>
\ No newline at end of file
diff --git a/.idea/indexLayout.xml b/.idea/indexLayout.xml
new file mode 100644
index 000000..7b08163
--- /dev/null
+++ b/.idea/indexLayout.xml
@@ -0,0 +1,8 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<project version="4">
+ <component name="UserContentModel">
+....skipping...
diff --git a/.idea/.gitignore b/.idea/.gitignore
new file mode 100644
index 000000..2177ffc
```

## Merges

### Fast-Forward Merge (FF)

Este tipo de fusión ocurre cuando no ha habido cambios en la rama principal (master) desde que se creó la rama secundaria (feature).

En este caso, Git simplemente mueve el puntero de la rama principal hacia adelante para incluir los commits de la rama secundaria.

- Hacemos el merge:

```
```bash
git checkout main
git merge newFeature
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\arnon\Downloads\demo2tp01>git merge newFeature
Updating 14fefc2..d0b77f4
Fast-forward
 .idea/.gitignore | 13 ++++++++=====
 .idea/encodings.xml | 4 +++
 .idea/indexLayout.xml | 8 ++++++++
 .idea/vcs.xml | 6 ++++++
 README.md | 2 ++
 5 files changed, 33 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/encodings.xml
 create mode 100644 .idea/indexLayout.xml
 create mode 100644 .idea/vcs.xml

C:\Users\arnon\Downloads\demo2tp01>
```

- Pusheamos

```
```bash
git push origin main
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.13 KiB | 288.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ArnonNahmias/demo2tp01.git
 14fefc2..d0b77f4 main -> main
```

- Borramos la rama y vemos el log

```
```bash
git branch -d newFeature
git log --oneline
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git branch -d newFeature
C:\Users\arnon\Downloads\demo2tp01>git log --oneline
d0b77f4 (HEAD -> main, origin/main, -d, newFeature) Cambio en el branch new
feature y modificamos el readme
14fefc2 (-a) Agregamos un git ignore y un readme
```

### No Fast-Forward Merge (No-FF)

Este tipo de fusión ocurre cuando se han realizado commits tanto en la rama principal como en la rama secundaria desde que se bifurcaron. Git crea un nuevo commit de fusión que combina los cambios de ambas ramas.

1. Creamos nueva rama

```
C:\Users\arnon\Downloads\demo2tp01>git checkout -b newFeature2
Switched to a new branch 'newFeature2'
```

2. Hacemos un cambio en archivo en rama main, hacemos commit

```
C:\Users\arnon\Downloads\demo2tp01>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

```
C:\Users\arnon\Downloads\demo2tp01>git add .
C:\Users\arnon\Downloads\demo2tp01>git commit -m "cambio en main"
[main 5835704] cambio en main
 1 file changed, 3 insertions(+), 1 deletion(-)
```

3. Modificamos un archivo y hacemos commit

```
C:\Users\arnon\Downloads\demo2tp01>git checkout newFeature2
Switched to branch 'newFeature2'
C:\Users\arnon\Downloads\demo2tp01>git add .
C:\Users\arnon\Downloads\demo2tp01>git commit -m "cambio en new feature 2"
[newFeature2 caa64ac] cambio en new feature 2
 1 file changed, 3 insertions(+), 1 deletion(-)
```

#### 4. Vemos la diferencia entre ramas

```
C:\Users\arnon\Downloads\demo2tp01>git diff main newFeature2
diff --git a/README.md b/README.md
index 4be2cdc..77b1628 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,4 @@

 # Este es el proyecto de Arnon Y felipe
-
-# Cambio en la rama main
-\ No newline at end of file
+# Modificamos en la rama newfeature2
-\ No newline at end of file
C:\Users\arnon\Downloads\demo2tp01>
```

#### 5. Hacemos el merge:

```
```bash
git checkout main
git merge newFeature2 --no-ff -m "merge con no ff"
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
 (use "git push" to publish your local commits)

C:\Users\arnon\Downloads\demo2tp01>git merge newFeature2 --no-ff -m "Merge con no-ff"
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

```
① README.md ! X
① README.md > # Modificamos en la rama newfeature2
1 # Mi Proyecto
2
3 # Este es el proyecto de Arnon Y felipe
4
5 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
6 <<<< HEAD (Current Change)
7 =====
8 # Modificamos en la rama newfeature2
9 >>>> newFeature2 (Incoming Change)
10
```

```
④ README.md □ Modificamos en la rama newfeature2
1 # Mi Proyecto
2
3 # Este es el proyecto de Arnon Y felipe
4
5 # Cambio en la rama main
6 # Modificamos en la rama newfeature2
7 |
```

```
C:\Users\arnon\Downloads\demo2tp01>git add .
C:\Users\arnon\Downloads\demo2tp01>git commit -m "conflicto resuelto"
[main 88b1026] conflicto resuelto
```

6. Borramos la rama y vemos el log

```
```bash
git branch -d newFeature2
git log --oneline --graph --decorate
```

```

```
C:\Users\arnon\Downloads\demo2tp01>git branch -d newFeature2
Deleted branch newFeature2 (was caa64ac).
```

```
C:\Users\arnon\Downloads\demo2tp01>git log --oneline --graph --decorate
* 88b1026 (HEAD -> main) conflicto resuelto
|\ \
| * caa64ac cambio en new feature 2
* | 5835704 cambio en main
|\ /
* d0b77f4 (origin/main, -d, newFeature) Cambio en el branch new feature y modifcamos el readme
* 14fefc2 (-a) Agregamos un git ignore y un readme
```

### AutoMerge

Git puede auto-fusionar cambios sin conflictos.

## Resolución de Conflictos

En casos en los que no hay conflictos entre los cambios en las ramas, Git realiza un "auto-merge". Esto significa que Git es capaz de combinar automáticamente los cambios sin intervención del usuario.

Supongamos que dos usuarios trabajan en la misma rama "feature" y realizan cambios diferentes en archivos no conflictivos.

Después de que ambos usuarios envían sus cambios y uno de ellos realiza un merge con `git merge feature`, Git realizará un auto-merge sin problemas.

1. Creamos una nueva rama conflictBranch
2. Realizamos una modificación en la linea 1 del Readme.md desde main y commiteamos
3. En la conflictBranch modificamos la misma línea del Readme.md y commiteamos
4. Vemos las diferencias con `git difftool main conflictBranch`
5. Me cambio a main e intento un `git merge conflictBranch`
6. Me indicará que no pudo hacer el automerge
7. Veo como quedó el Readme.md
8. Veo que estoy en un estado MERGING dentro de main
9. Resuelvo el conflicto con `git mergetool`
10. Agrego .orig al .gitignore
11. Hago commit
12. Hago push

### **Ya lo solucionamos más arriba**

#### **## Pull Request**

En el contexto del desarrollo de software y los sistemas de control de versiones distribuidos, un Pull Request (PR) o solicitud de extracción es una funcionalidad clave que permite a los desarrolladores colaborar y revisar los cambios antes de fusionarlos en la rama principal del repositorio. Es una característica fundamental en plataformas como GitHub y GitLab.

El Pull Request se utiliza principalmente cuando un desarrollador ha trabajado en una rama secundaria (por ejemplo, una rama de funcionalidad) y desea incorporar esos cambios en la rama principal del proyecto (generalmente llamada "master" o "main"). En lugar de realizar una fusión directa, el desarrollador crea un Pull Request para notificar a otros miembros del equipo sobre los cambios y solicitar su revisión antes de la fusión.

1. Creo una rama local, creo un nuevo archivo y la subo al repo remoto:

```
```bash
git branch pullReqBranch
git push origin pullReqBranch
```
```

2. Desde github hago un PullRequest

3. Ver opciones de seguridad de la rama

```
C:\Users\arnon\Downloads\demo2tp01>git switch -c pullReqBranch
Switched to a new branch 'pullReqBranch'

C:\Users\arnon\Downloads\demo2tp01>echo "Archivo para PR" > notas-pr.txt

C:\Users\arnon\Downloads\demo2tp01>git add notas-pr.txt

C:\Users\arnon\Downloads\demo2tp01>git commit -m "feat: agregar notas-pr.txt"
[pushReqBranch 50818e8] feat: agregar notas-pr.txt
 1 file changed, 1 insertion(+)
 create mode 100644 notas-pr.txt

C:\Users\arnon\Downloads\demo2tp01>git push -u origin pullReqBranch
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.06 KiB | 361.00 KiB/s, done.
Total 12 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 1 local object.
remote:
remote: Create a pull request for 'pullReqBranch' on GitHub by visiting:
remote: https://github.com/ArnonNahmias/demo2tp01/pull/new/pullReqBranch
remote:
remote: To https://github.com/ArnonNahmias/demo2tp01.git
 * [new branch] pullReqBranch -> pullReqBranch
branch 'pullReqBranch' set up to track 'origin/pullReqBranch'.

C:\Users\arnon\Downloads\demo2tp01>
```

### Beneficios de PRs:

Los PRs ofrecen una serie de beneficios que mejoran la calidad del código, la colaboración del equipo y la gestión del proyecto:

1. **Facilita la revisión del código**: Proporcionan una forma estructurada para que los miembros del equipo revisen y discutan los cambios propuestos antes de que se fusionen en la rama principal. Esto ayuda a identificar errores, mejorar el diseño y asegurar que el código cumpla con los estándares de calidad del proyecto.

2. **Fomenta la colaboración**: Al utilizar PRs, los miembros del equipo pueden compartir sus conocimientos y experiencia, ofrecer sugerencias y trabajar juntos para mejorar el código

3. **Mejora la transparencia**: Proporcionan un registro claro y completo de los cambios propuestos, las discusiones y las revisiones realizadas por el equipo.
4. **Permite pruebas y validaciones**: Antes de que los cambios se fusionen en la rama principal, los PRs pueden someterse a pruebas automatizadas, integración continua y validaciones por parte de otros miembros del equipo.
5. **Evita conflictos y errores en la rama principal**: Al utilizar PRs, se minimiza el riesgo de conflictos y errores en la rama principal del repositorio. Los cambios se revisan cuidadosamente antes de la fusión, lo que reduce la probabilidad de introducir problemas en el código base.
6. **Flexibilidad y control**: Los PRs ofrecen flexibilidad al desarrollador, ya que permiten continuar trabajando en la rama de funcionalidad sin necesidad de fusionarla de inmediato. El desarrollador puede recibir comentarios y realizar mejoras antes de completar la fusión.

Es una práctica altamente beneficiosa en el desarrollo de software colaborativo, ya que mejora la calidad del código, fomenta la colaboración y brinda un mayor control sobre los cambios que se incorporan en el proyecto.

---

## # Trabajo Práctico 01 – Git Básico (2025)

### ## 🎯 Objetivo

Aplicar y demostrar el uso práctico de Git mediante un caso simulado de trabajo en equipo. Este trabajo se aprueba \*\*solo si podés explicar qué hiciste, por qué lo hiciste y cómo lo resolviste\*\*.

---

### ## 🌱 Escenario

Recibiste tres tareas clave como parte de un equipo de desarrollo:

1. Agregar una nueva funcionalidad.
2. Corregir un error en producción.
3. Preparar una versión estable del sistema.

Debés organizarte con Git para realizar estas tareas de forma \*\*ordenada, trazable y profesional\*\*.

---

### ## 📋 Tareas que debés cumplir

#### ### 1. Configurar tu entorno y preparar tu repositorio

- Cloná o forkeá el repositorio base [https://github.com/ingsoft3ucc/2025\\_TP01\\_RepoBase](https://github.com/ingsoft3ucc/2025_TP01_RepoBase)

```
D:\FACU 2025\ING SOFT 3>git clone https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
Cloning into '2025_TP01_RepoBase'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 3 (from 1)
Receiving objects: 100% (8/8), done.

D:\FACU 2025\ING SOFT 3>cd 2025_TP01_RepoBase

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git remote -v
origin https://github.com/ArnonNahmias/2025_TP01_RepoBase.git (fetch)
origin https://github.com/ArnonNahmias/2025_TP01_RepoBase.git (push)
```

Con git remote -v ----->verifica que 'origin' apunta a TU fork

- Configurá tu identidad y dejá constancia en el archivo `decisiones.md` de cómo lo hiciste.

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git config --global user.name "ArnonNahmias"
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git config --global user.email "2222270@ucc.edu.ar"
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git config --global init.defaultBranch main
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git config --list --show-origin
file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true
file:C:/Program Files/Git/etc/gitconfig http.sslbackend=schannel
file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true
file:C:/Program Files/Git/etc/gitconfig core.fscache=true
file:C:/Program Files/Git/etc/gitconfig core.symlinks=false
file:C:/Program Files/Git/etc/gitconfig pull.rebase=false
file:C:/Program Files/Git/etc/gitconfig credential.helper=manager
file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com/usehttppath=true
file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/aronn/.gitconfig user.email=2222270@ucc.edu.ar
file:C:/Users/aronn/.gitconfig user.name=ArnonNahmias
file:C:/Users/aronn/.gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Users/aronn/.gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Users/aronn/.gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Users/aronn/.gitconfig filter.lfs.required=true
file:C:/Users/aronn/.gitconfig init.defaultbranch=main
file:.git/config core.repositoryformatversion=0
file:.git/config core.filemode=false
file:.git/config core.bare=false
file:.git/config core.logallrefupdates=true
file:.git/config core.symlinks=false
file:.git/config core.ignorecase=true
file:.git/config remote.origin.url=https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
file:.git/config remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
file:.git/config branch.main.remote=origin
file:.git/config branch.main.merge=refs/heads/main
```

|                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>✓ 2025_TP01_REPOBASE   ✓ data     └ info.txt   ✓ src     JS app.js     decisiones.md  U     README.md</pre> | <pre>decisiones.md &gt; # Decisiones del TP01 &gt; ## Configuración de identidad 1  # Decisiones del TP01 2  ## Configuración de identidad 3  Para este trabajo práctico configuré mi identidad de Git de la siguiente forma: 4  ```bash 5  git config --global user.name "ArnonNahmias" 6  git config --global user.email "2222270@ucc.edu.ar" 7  git config --global init.defaultBranch main 8   </pre> |
|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

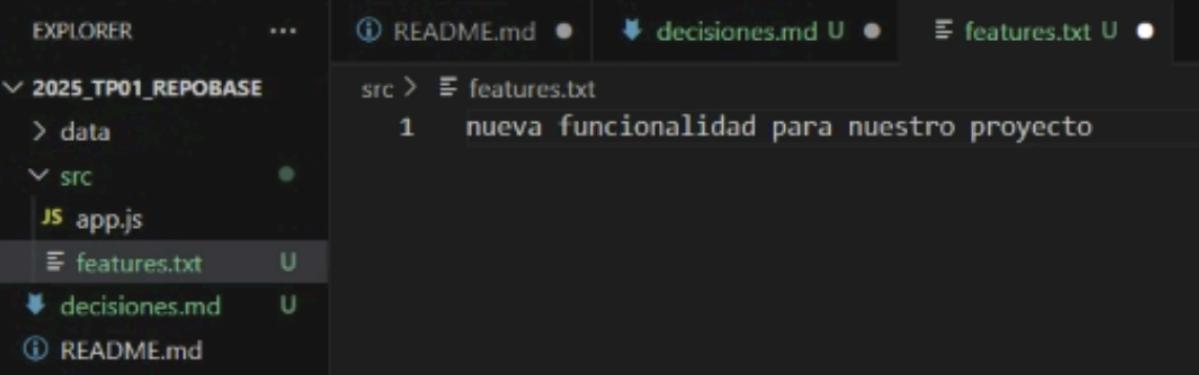
### ### 2. Desarrollar una funcionalidad

- Trabajá en una rama separada de `main`.

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch -c feature/nueva-funcionalidad
Switched to a new branch 'feature/nueva-funcionalidad'
```

- Hacé al menos \*\*2 commits atómicos\*\* con mensajes claros.

primer commit:

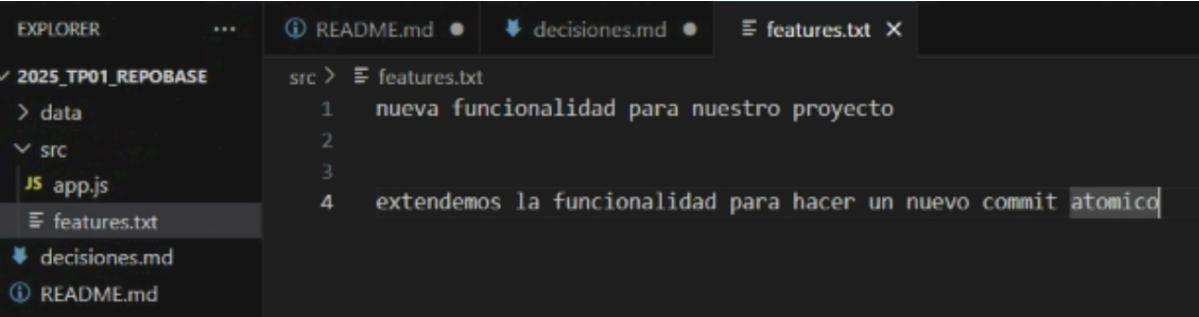


The screenshot shows the VS Code interface with the Explorer sidebar open. In the workspace, there is a folder named '2025\_TP01\_REPOBASE' containing 'data', 'src', and 'README.md'. Inside 'src', there is an 'app.js' file and a 'features.txt' file which is staged for commit (indicated by a 'U' icon). Another 'features.txt' file is shown in the main editor area with the following content:

```
src > features.txt
1 nueva funcionalidad para nuestro proyecto
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "agregamos el .md y el feature"
[feature/nueva-funcionalidad a7cdb70] agregamos el .md y el feature
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 decisiones.md
 create mode 100644 src/features.txt
```

segundo commit:



The screenshot shows the VS Code interface with the Explorer sidebar open. In the workspace, there is a folder named '2025\_TP01\_REPOBASE' containing 'data', 'src', and 'README.md'. Inside 'src', there is an 'app.js' file and a 'features.txt' file which is staged for commit (indicated by a 'U' icon). Another 'features.txt' file is shown in the main editor area with the following content:

```
src > features.txt
1 nueva funcionalidad para nuestro proyecto
2
3
4 extendemos la funcionalidad para hacer un nuevo commit atomico
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
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/features.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: src/features.txt
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "extendemos la funcionalidad en feature.txt"
[feature/nueva-funcionalidad 6992b6c] extendemos la funcionalidad en feature.txt
 1 file changed, 4 insertions(+)
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>
```

- Justifica la estrategia que usaste (¿por qué esa rama? ¿por qué esos commits?).

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add decisiones.md
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "docs: agregar justificación de estrategia de rama y commits atómicos"
[feature/nueva-funcionalidad 7327b64] docs: agregar justificación de estrategia de rama y commits atómicos
 1 file changed, 15 insertions(+)

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push
fatal: The current branch feature/nueva-funcionalidad has no upstream branch.
To push the current branch and set the remote as upstream, use

 git push --set-upstream origin feature/nueva-funcionalidad

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

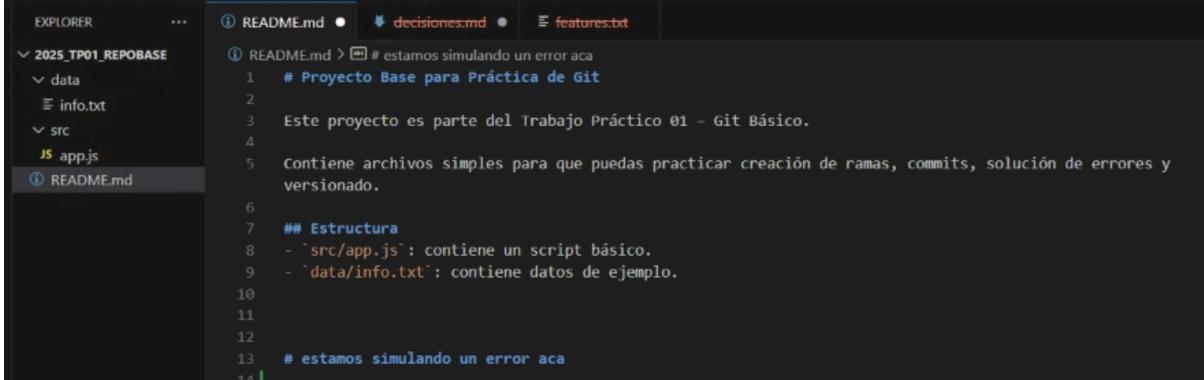
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push -u origin feature/nueva-funcionalidad
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.44 KiB | 737.00 KiB/s, done.
Total 11 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'feature/nueva-funcionalidad' on GitHub by visiting:
remote: https://github.com/ArnonNahmias/2025_TP01_RepoBase/pull/new/feature/nueva-funcionalidad
remote:
To https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
 * [new branch] feature/nueva-funcionalidad -> feature/nueva-funcionalidad
branch 'feature/nueva-funcionalidad' set up to track 'origin/feature/nueva-funcionalidad'.
```

Hicimos git push y nos dio un fatal error porque la rama en la que estábamos actualmente no tenía upstream, por lo que usamos la IA y nos sugirió el siguiente comando  
git push -u origin feature/nueva-funcionalidad

### ### 3. Corregir un error (simulado) y aplicar el fix

- Simulá un error en `main` y resolvelo en una rama `hotfix`.

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```



The screenshot shows a code editor interface with the following details:

- EXPLORER**: Shows a folder structure for "2025\_TP01\_REPOBASE" containing "data", "info.txt", "src", and "app.js". The "README.md" file is selected.
- README.md**: The content of the README file is displayed, including simulated errors at lines 13 and 14.
- Content of README.md**:

```
1 # Proyecto Base para Práctica de Git
2
3 Este proyecto es parte del Trabajo Práctico 01 - Git Básico.
4
5 Contiene archivos simples para que puedas practicar creación de ramas, commits, solución de errores y
6
7 ## Estructura
8 - `src/app.js`: contiene un script básico.
9 - `data/info.txt`: contiene datos de ejemplo.
10
11
12
13 # estamos simulando un error aca
14 |
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "metimos un error s
imulado en el readme para avanzar con la parte 3"
[main 1325aa0] metimos un error simulado en el readme para avanzar con la pa
rte 3
1 file changed, 4 insertions(+)
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 434 bytes | 434.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
 cf8df37..1325aa0 main -> main
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch -c hotfix/readme-bug
Switched to a new branch 'hotfix/readme-bug'
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch hotfix/readme-bug
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: README.md

no changes added to commit (use "git add" and/or "git commit -a")

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git aststus
git: 'aststus' is not a git command. See 'git --help'.

The most similar command is
 status

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch hotfix/readme-bug
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: README.md

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "hotfix: error en el readme solucionado"
[hotfix/readme-bug 6d03075] hotfix: error en el readme solucionado
 1 file changed, 4 insertions(+), 1 deletion(-)
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push
fatal: The current branch hotfix/readme-bug has no upstream branch.
To push the current branch and set the remote as upstream, use

 git push --set-upstream origin hotfix/readme-bug

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

Nuevamente el git push no nos funciono por el upstream por lo que nos imaginamos que era el mismo error que previamente solucionamos con la IA por lo que usamos el siguiente comando:

```
git push -u origin hotfix/readme-bug
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push -u origin hotfix/readme-bug
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 402 bytes | 402.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'hotfix/readme-bug' on GitHub by visiting
remote: https://github.com/ArnonNahmias/2025_TP01_RepoBase/pull/new/hotfix/readme-bug
remote:
To https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
 * [new branch] hotfix/readme-bug -> hotfix/readme-bug
branch 'hotfix/readme-bug' set up to track 'origin/hotfix/readme-bug'.
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>
```

- Aplicá el fix a `main` y también a tu rama de desarrollo.

Ahora lo que haremos es integrar el fix al main

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git merge --no-ff hotfix/readme-bug -m "Merge hotfix: corregir bug en README"
Merge made by the 'ort' strategy.
 README.md | 5 +----
 1 file changed, 4 insertions(+), 1 deletion(-)
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 235 bytes | 235.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ArnonNahmias/2025_TP01_RepoBase.git
 1325aa0..b39fb34 main -> main
```

Ahora lo que haremos es integrar el fix a la rama de desarrollo

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git log --oneline hotfix/readme-bug
6d03075 (origin/hotfix/readme-bug, hotfix/readme-bug) hotfix: error en el readme solucionado
1325aa0 metimos un error simulado en el readme para avanzar con la parte 3
cf8df37 Initial commit: Setup base repository structure
```

Utilizando la forma cherry-pick y haciendo uso de este comando obtenemos el hash del commit del hotfix que usamos para solucionar el error simulado en el readme. Para mediante cherry pick solamente traer este cambio a la rama de desarrollo

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch feature/nueva-funcionalidad
Switched to branch 'feature/nueva-funcionalidad'
Your branch is up to date with 'origin/feature/nueva-funcionalidad'.
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git log --oneline hotfix/readme-bug
6d03075 (origin/hotfix/readme-bug, hotfix/readme-bug) hotfix: error en el readme solucionado
1325aa0 metimos un error simulado en el readme para avanzar con la parte 3
cf8df37 Initial commit: Setup base repository structure
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git switch feature/nueva-funcionalidad
Switched to branch 'feature/nueva-funcionalidad'
Your branch is up to date with 'origin/feature/nueva-funcionalidad'.
```

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git cherry-pick 6d03075
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply 6d03075... hotfix: error en el readme solucionado
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
```

```
gitcajero.exe - Corriendo datos de ejemplo.
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<< HEAD (Current Change)
=====

Error solucionado

>>>>> 6d03075 (hotfix: error en el readme solucionado) (Incoming Change)

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
Your branch is up to date with 'origin/feature/nueva-funcionalidad'.

You are currently cherry-picking commit 6d03075.
 (fix conflicts and run "git cherry-pick --continue")
 (use "git cherry-pick --skip" to skip this patch)
 (use "git cherry-pick --abort" to cancel the cherry-pick operation)

Unmerged paths:
 (use "git add <file>..." to mark resolution)
 both modified: README.md

no changes added to commit (use "git add" and/or "git commit -a")

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
Your branch is up to date with 'origin/feature/nueva-funcionalidad'.

You are currently cherry-picking commit 6d03075.
 (all conflicts fixed: run "git cherry-pick --continue")
 (use "git cherry-pick --skip" to skip this patch)
 (use "git cherry-pick --abort" to cancel the cherry-pick operation)

Changes to be committed:
 modified: README.md
```

Oprimimos en donde dice accept both changes para que quede lo de rama hotfix junto con lo de la rama feature  
introducimos el comando  
git cherry-pick --continue  
aparece lo siguiente

Con el siguiente comando que nos sugirio la IA verificamos que todo se haya realizado correctamente:

```
git log --oneline --graph --decorate --all
```

- \*\*Elegí cómo lo integrás\*\* (`merge`, `cherry-pick`, etc.) y \*\*explicalo en `decisiones.md`\*\*.

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
Your branch is ahead of 'origin/feature/nueva-funcionalidad' by 1 commit.
 (use "git push" to publish your local commits)

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: decisiones.md

no changes added to commit (use "git add" and/or "git commit -a")

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git add .

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git status
On branch feature/nueva-funcionalidad
Your branch is ahead of 'origin/feature/nueva-funcionalidad' by 1 commit.
 (use "git push" to publish your local commits)

Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: decisiones.md

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git commit -m "agregamos la explicacion de la integracion al decisiones.md"
[feature/nueva-funcionalidad 33e56b3] agregamos la explicacion de la integracion al decisiones.md
 1 file changed, 8 insertions(+)

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.28 KiB | 1.28 MiB/s, done.
Total 6 (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/ArnonNahmias/2025_TP01_RepoBase.git
 7327b64..33e56b3 feature/nueva-funcionalidad -> feature/nueva-funcionalidad

D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>
```

#### ### 4. Hace un PR y aceptalo

```
D:\FACU 2025\ING SOFT 3\2025_TP01_RepoBase>git push -u origin feature/nueva-funcionalidad
branch 'feature/nueva-funcionalidad' set up to track 'origin/feature/nueva-funcionalidad'.
Everything up-to-date
```

The screenshot shows a GitHub pull request interface. At the top, there are navigation buttons for 'Filters', a search bar containing 'is:pr is:open', and other buttons for 'Labels' (9), 'Milestones' (0), and 'New pull request'. Below this is a large central area with a heading 'Welcome to pull requests!'. The text explains that pull requests help collaborate on code. A 'ProTip!' link at the bottom left indicates the page was updated on 2025-09-16.

ProTip! Updated in the last three days: [updated:>2025-09-16](#).

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base repository: Ingsoft3ucc/2025\_TP01\_RepoB... base: main head repository: ArnonNahmias/2025\_TP01\_Rep... compare: feature/nueva-funcionalidad

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

-o 5 commits 3 files changed 1 contributor

Commits on Sep 19, 2025

- agregamos el .md y el feature** ArnonNahmias committed 1 hour ago [a7cdb70](#)
- extendemos la funcionalidad en feature.txt** ArnonNahmias committed 54 minutes ago [6992b6c](#)
- docs: agregar justificación de estrategia de rama y commits atómicos** ArnonNahmias committed 48 minutes ago [7327b64](#)
- hotfix: error en el readme solucionado** ArnonNahmias committed 15 minutes ago [2966e80](#)
- agregamos la explicacion de la integracion al decisiones.md** ArnonNahmias committed 5 minutes ago [33e56b3](#)

Add a title [Help](#) [GitHub](#)

Feature/nueva funcionalidad --> Integrar nueva funcionalidad del tp1

Add a description

Write Preview

Este pull request agrega nueva funcionalidad desde la rama feature/nueva funcionalidad e incluye el archivo decisiones.md

Markdown is supported Paste, drop, or click to add files

Allow edits by maintainers [Create pull request](#)

## Feature/nueva funcionalidad --> Integrar nueva funcionalidad del tp1 #38

[Open](#) ArnonNahmias wants to merge 5 commits into [ingsoft3ucc:main](#) from [ArnonNahmias:feature/nueva-funcionalidad](#)

Conversation 0 Commits 5 Checks 0 Files changed 3 +34 -0

ArnonNahmias commented now  
Este pull request agrega nueva funcionalidad desde la rama feature/nueva funcionalidad e incluye el archivo decisiones.md

ArnonNahmias added 5 commits 1 hour ago

- agregamos el .md y el feature [a7cdb70](#)
- extendemos la funcionalidad en feature.txt [6992b6c](#)
- docs: agregar justificación de estrategia de rama y commits atómicos [7327b64](#)
- hotfix: error en el readme solucionado [2966e80](#)
- agregamos la explicacion de la integracion al decisiones.md [33e56b3](#)

Reviewers  
No reviews  
Still in progress? [Convert to draft](#)

Assignees  
No one assigned

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

The screenshot shows a GitHub pull request page for a pull request titled "Feature/nueva funcionalidad --> Integrar nueva funcionalidad del tp1 #38". The pull request has been merged by "ArnonNahmias" and has 5 commits. The commit history includes:

- agregamos el .md y el feature
- extendemos la funcionalidad en feature.txt
- docs: agregar justificación de estrategia de rama y commits atómicos
- hotfix: error en el readme solucionado
- agregamos la explicacion de la integracion al decisiones.md

Reviewers: No reviews. Still in progress? Convert to draft.

Assignees: No one assigned.

Labels: None yet.

Projects: None yet.

Milestone: No milestone.

Development: Successfully merging this pull request may close these issues. None yet.

Notifications: You're receiving notifications because you authored the thread. Unsubscribe.

### ### 5. Crear una versión etiquetada

- Marcá una versión estable con el tag `v1.0`.
- Explicá en `decisiones.md` qué convenciones usaste y por qué.

---

## ## 📄 Entregables

1. \*\*Repositorio en GitHub\*\* con todas las ramas, commits y el tag.
2. Archivo `decisiones.md` explicando:
  - Qué flujo de trabajo usaste y por qué.
  - Cómo integraste el fix.
  - Qué problemas tuviste y cómo los resolviste.
  - Cómo asegurarías calidad y trazabilidad en un equipo real.

---

## ## 🎤 Defensa Oral Obligatoria

Vas a tener que mostrar tu trabajo y responder preguntas como:

- ¿Qué hace `git rebase`? ¿Lo usaste o no? ¿Por qué?
- ¿Cómo revertís un commit ya push?
- Mostrame tu log y explicame qué hiciste en cada parte.

---

## ## ✅ Evaluación

| Criterio                                    | Peso |
|---------------------------------------------|------|
| Organización técnica del repositorio        | 30%  |
| Claridad y justificación en `decisiones.md` | 30%  |
| Defensa oral: comprensión y argumentación   | 40%  |

---

## ## ⚠️ Uso de IA

Podés usar IA (ChatGPT, Copilot), pero \*\*deberás declarar qué parte fue generada con IA\*\* y justificar cómo la verificaste.

Si no podés defenderlo, \*\*no se aprueba\*\*.

---

## ## 💥 Desafíosopcionales (para destacar)

- Mostrá un ejemplo de `git revert` sobre un commit innecesario.
- Resolvé un conflicto entre ramas.
- Usá `git stash` en una situación simulada y explicalo.