

ALUMNO: RODRIGO NICOLAS

PROGRAMACIÓN I

Trabajo Práctico N.º 2: Git & GitHub

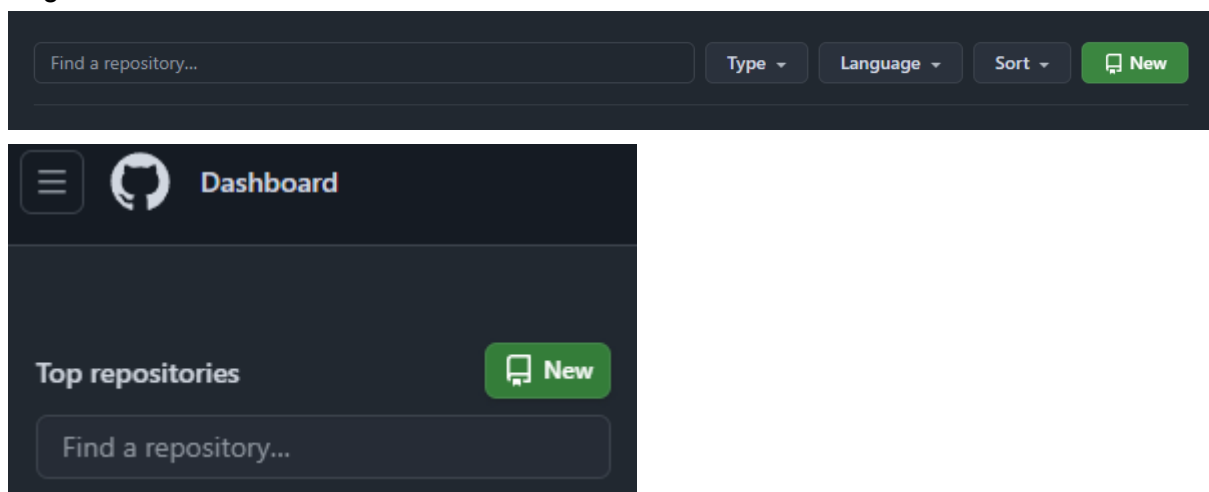
1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?

GitHub es una plataforma en la nube que permite almacenar, crear, modificar y compartir código. Utiliza el control de versiones Git creado por Linus Torvels.

- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub primero debemos crear una cuenta en la plataforma luego haciendo clic en New.



Podremos crear y darle nombre al nuevo repositorio.

- ¿Cómo crear una rama en Git?

Para crear una rama de git debemos insertar el comando:

```
$ git branch nuevaRama
```

Dentro del repo que estamos trabajando. Luego debemos hacer un checkout si queremos pasar a trabajar al a nueva rama.

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git debemos inserta el comando:

```
$ git checkout nuevaRama
```

ALUMNO: RODRIGO NICOLAS

Para saltar y crear una rama en nuevo pasa podemos hacerlo utilizando el comando:

```
$ git checkout -b nuevaRama
```

• **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git debemos seguir algunos pasos. Primero debemos estar en la rama a la que queremos fusionar los cambios generalmente main.

Luego ingresamos el siguiente comando:

```
$ git merge nuevaRama
```

• **¿Cómo crear un commit en Git?**

Para crear un commit en Git primero debemos añadir el archivo que hemos modificado con el comando:

```
$ git add nombreArchivo
```

```
$ git add . (agrega todos los archivos que han sido modificados)
```

Luego debemos ingresar el siguiente comando y detrás de -m y entre comillas el detalle del mensaje de los cambios que hemos realizados por convención los mensajes de los commits se realizan en el modo imperativo por ejemplos:

```
$ git commit - m "Realiza cambios a nombreArchivo"
```

• **¿Cómo enviar un commit a GitHub?**

Para enviar un commit a GitHub debemos ingresar el siguiente comando desde el repositorio clonado localmente luego de haber realizado los pasos anteriores de hacer git add y git commit:

```
$ git push origin nombre_de_la_rama
```

```
$ git push (si realizamos algunas configuraciones)
```

• **¿Qué es un repositorio remoto?**

Un repositorio remoto es una copia de un proyecto que se encuentra en un servidor remoto ya sea internet o una red local. Son importantes porque permite a varias personas colaborar al mismo tiempo en el mismo proyecto.

• **¿Cómo agregar un repositorio remoto a Git?**

Para agregar un repositorio remoto a Git debemos ir a la página del repositorio de GitHub crear un fork de ese repositorio y luego clonarlo localmente a nuestro sistema para poder usar ese proyecto o realizar cambios al mismo haciendo pull requests.

ALUMNO: RODRIGO NICOLAS

También podemos hacerlo de forma local mediante los siguientes comandos:

```
$ git remote add [nombre] [url]
$ git remote add [nombre] [url]
$ git remote add [nombre] [url]
```

- **¿Cómo empujar cambios a un repositorio remoto?**

Antes de empujar tus cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para evitar conflictos:

```
$ git pull origin nombre_de_la_rama
```

Empuja tus cambios al repositorio remoto:

```
$ git push origin nombre_de_la_rama
```

- **¿Cómo tirar de cambios de un repositorio remoto?**

Para tirar de cambios de un repositorio remoto debemos realizar el siguiente comando:

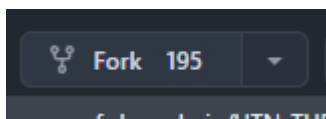
```
$ git pull origin main o (nombre_de_la_rama)
```

- **¿Qué es un fork de repositorio?**

Un fork es una copia de un repositorio a nuestra cuenta de GitHub que puede ser clonada de manera local si tenemos los accesos necesarios

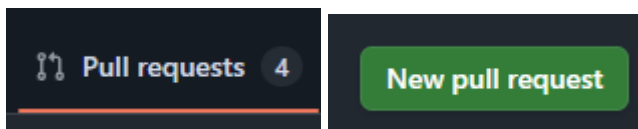
- **¿Cómo crear un fork de un repositorio?**

Para crear un fork de un repositorio debemos ir al repo que deseamos forkear y hacer click en el botón de fork:



- **¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**

Para hacer un pull request debemos ir a la solapa de pull request:



Agregar un mensaje global y luego generar el pull.

- **¿Cómo aceptar una solicitud de extracción?**

El autor del repositorio verá el pull request creado por el usuario y de considerarlo útil puede aceptarlo o no.

ALUMNO: RODRIGO NICOLAS

• **¿Qué es una etiqueta en Git?**

Es un sistema que permite a los controladores de versiones etiquetar puntos específicos importantes en el historial de versiones cómo por ejemplo v1.0.

• **¿Cómo crear una etiqueta en Git?**

Git usa dos tipos de etiquetado: ligero y anotado.

Etiquetas Anotadas:

Se crean con el comando:

```
$ git tag -a v1.4 -m "Mensaje de la etiqueta 1.4"
```

Permite ver información de la etiqueta.

```
$ git show v1.4
```

Etiquetas Ligeras:

Es un puntero a un commit específico

```
$ git tag
```

• **¿Cómo enviar una etiqueta a GitHub?**

Primero debemos crear una etiqueta en nuestro repositorio local con:

```
$ git tag o $ git tag -a -m
```

Una vez creada en nuestro repositorio local debemos empujarla al repositorio remoto en GitHub con:

```
$ git push origin v1.0
```

• **¿Qué es un historial de Git?**

Es un registro de los cambios realizados en un repositorio de código. Permite ver el log de los commits realizados por los programadores y contiene la siguiente información:

Identificador del commit

Autor

Fecha de realización

Mensaje enviado

• **¿Cómo ver el historial de Git?**

Para ver el historial de Git debemos estar en el repositorio que deseamos ver e ingresar el siguiente comando estando situados en la carpeta de nuestro proyecto:

```
$ git log
```

Esto nos permitirá ver los el listado de commits invertido es decir aparecen los últimos commits primero.

ALUMNO: RODRIGO NICOLAS

El comando `git log --oneline` es una forma compacta de ver el historial de commits en un repositorio Git. Muestra cada commit representado por una sola línea.

Si el proyecto ya tiene muchos commits se pueden buscar una cantidad determinada ingresando `git log -n` donde `n` es un número tanto así que queda:

```
$ git log -3 (muestra los últimos 3 commits)
```

Si queremos ver los cambios en el código también podemos hacerlo con `-p`. Da un output más extenso por lo que debemos navegar con los cursores y salir de la terminal con `CTRL + Z`.

• ¿Cómo buscar en el historial de Git?

Existen diferentes formas de ver el historial de commits.

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa `git log` con la opción `--grep`: `git log --grep="palabra clave"`

Para buscar commits que han modificado un archivo específico, usa `git log` seguido del nombre del archivo: `git log -- nombre_del_archivo`

Para buscar commits en un rango de fechas específico, usa las opciones `--since` y `--until`: `git log --since="2024-01-01" --until="2024-01-31"`

Para encontrar commits hechos por un autor específico, usa `--author`: `git log --author="Nombre del Autor"`

• ¿Cómo borrar el historial de Git?

Con el comando `$ git reset`

Con el comando `git reset <commit>` se actualiza hasta el último commit especificado
`git reset ->` Quita del stage todos los archivos y carpetas del proyecto.

`git reset nombreArchivo ->` Quita del stage el archivo indicado

`git reset nombreCarpeta/ ->` Quita del stage todos los archivos de esa carpeta.

`git reset nombreCarpeta/nombreArchivo ->` Quita ese archivo del stage (que a la vez está dentro de una carpeta).

`git reset nombreCarpeta/*.extensión ->` Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

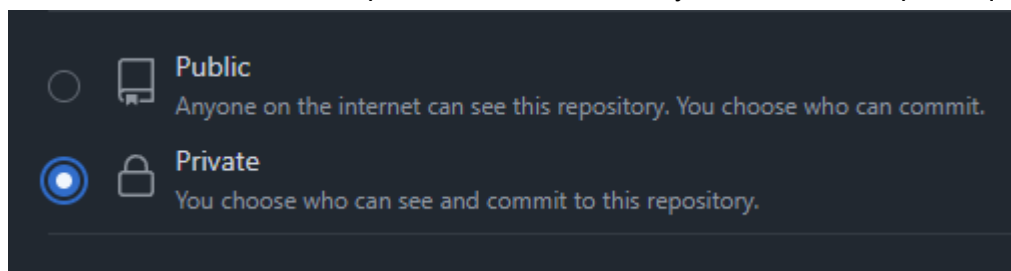
ALUMNO: RODRIGO NICOLAS

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado es un repositorio al que solo tienen acceso el creador y algunas personas con las autorizaciones aceptadas. A diferencia de los repositorios públicos donde cualquier persona puede hacer cambios y clonar el repositorio, solo las personas autorizadas pueden hacer esto en los repositorios privados. Estos repositorios son útiles cuando la información incluida en el código es sensible o proyecto no aun no esta terminado o no queremos que la información esté disponible públicamente.

• ¿Cómo crear un repositorio privado en GitHub?

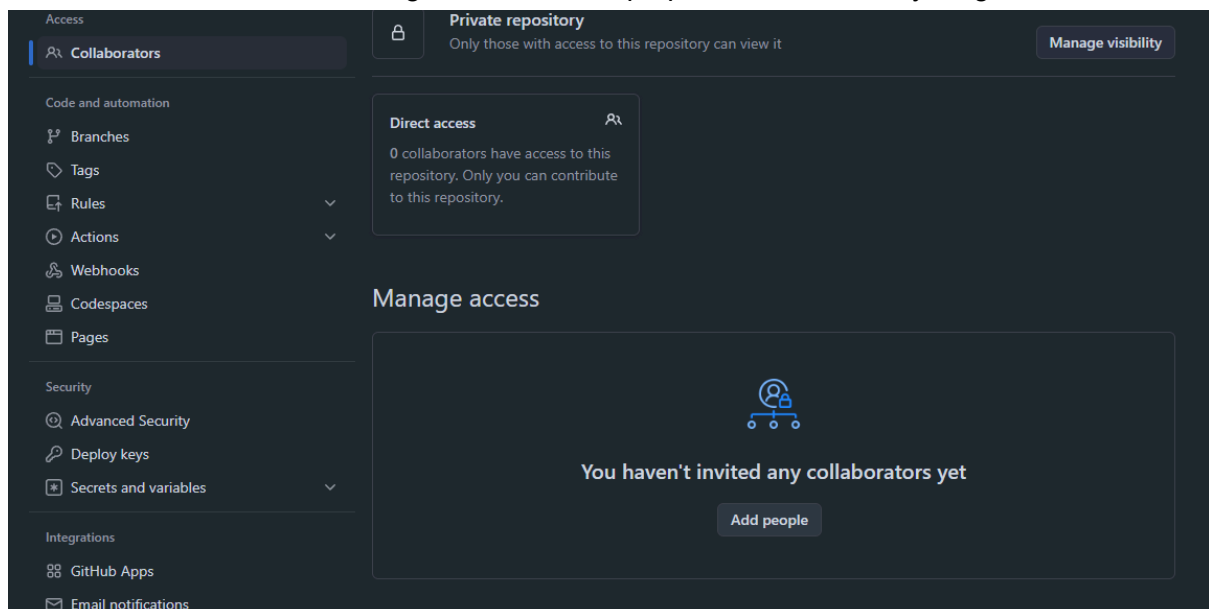
Debemos crear un nuevo repositorio desde GitHub y seleccionar la opción que dice “private”



Algunas de las limitaciones de los repositorios privados es que si no contamos con una cuenta premium no tendremos acceso a GitHub Pages.

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Debemos ir a la seccion “settings” de nuestro repo privado en GitHub y luego:



Hacer click en donde dice “Add people”. Esto nos permitirá añadir usuarios que puedan colaborar en nuestro proyecto con el nivel de acceso deseado.

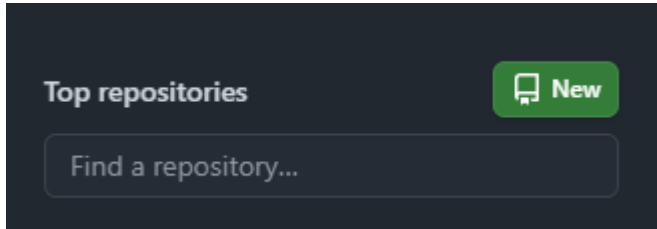
• ¿Qué es un repositorio público en GitHub?

Un repositorio público es un repositorio al que todos los usuarios tienen acceso.

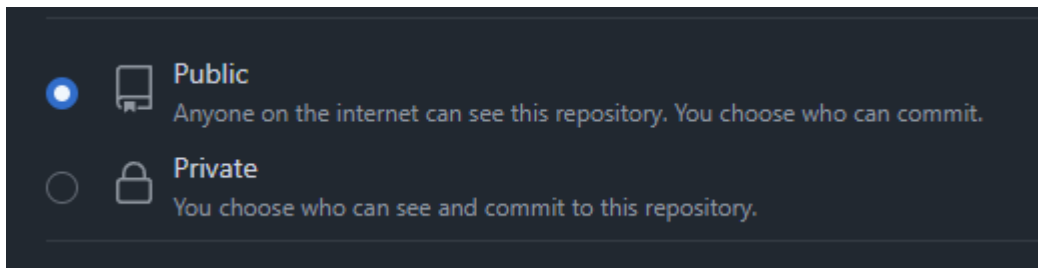
ALUMNO: RODRIGO NICOLAS

• ¿Cómo crear un repositorio público en GitHub?

Primero hacemos click en “New”



Luego seleccionamos la opción “Public”



• ¿Cómo compartir un repositorio público en GitHub?

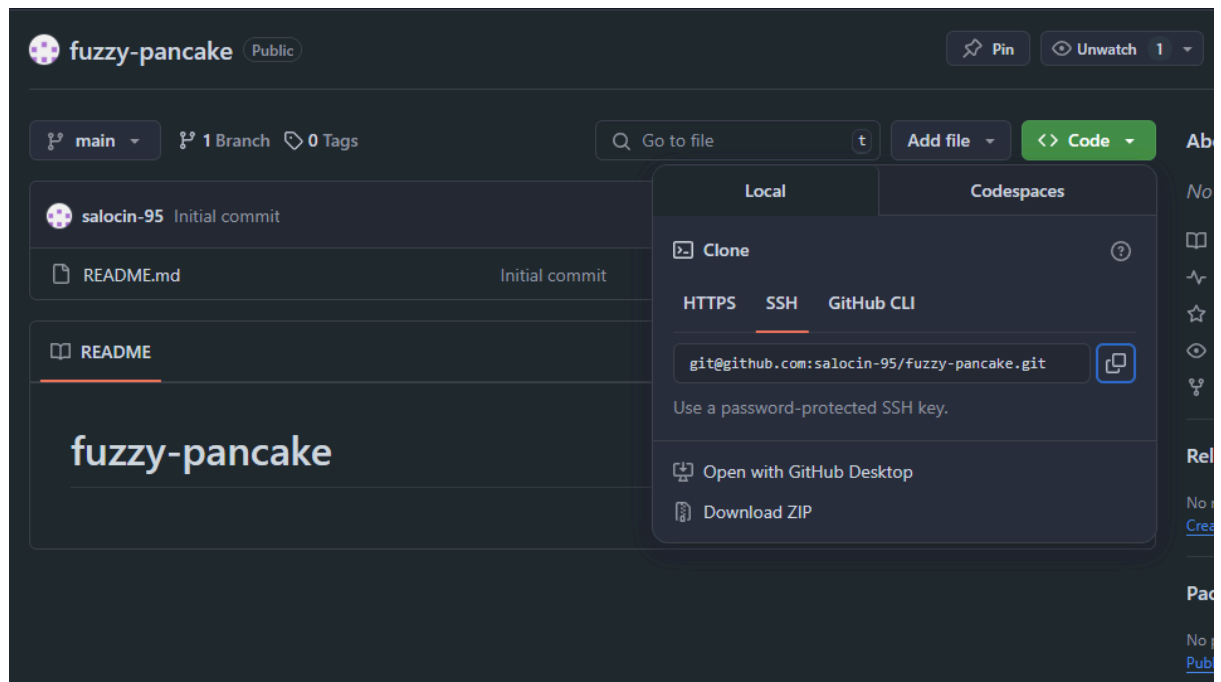
Podemos hacerlo compartiendo el link a repositorio por ejemplo

<https://github.com/usuario/repositorio>

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - o Dale un nombre al repositorio.
 - o Elije el repositorio sea público.
 - o Inicializa el repositorio con un archivo.

ALUMNO: RODRIGO NICOLAS



- Agregando un Archivo
o Crea un archivo simple, por ejemplo, "mi-archivo.txt".

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ echo "Ejercicio del trabajo practico" > mi-texto.txt

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (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)
      mi-texto.txt

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

- o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ git add .
```

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ git commit -m "Agrega mi-texto.txt"
[main 1e1f79c] Agrega mi-texto.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-texto.txt
```


ALUMNO: RODRIGO NICOLAS

o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ git push
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% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

• Creando Branchs

o Crear una Branch

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ git branch ejercicio/rama-nueva
```

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (main)
$ git checkout ejercicio/rama-nueva
Switched to branch 'ejercicio/rama-nueva'
```

o Realizar cambios

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (ejercicio/rama-nueva)
$ git add .

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (ejercicio/rama-nueva)
$ git status
On branch ejercicio/rama-nueva
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   mi-texto.txt
```


o agregar un archivo


```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (ejercicio/rama-nueva)
$ git commit -m "Agrega modificacion desde la rama nueva"
[ejercicio/rama-nueva b153ed5] Agrega modificacion desde la rama nueva
1 file changed, 1 insertion(+)
```

o Subir la Branch


ALUMNO: RODRIGO NICOLAS



```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/fuzzy-pancake (ejercicio/rama-nueva)
$ git push origin ejercicio/rama-nueva
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), 371 bytes | 371.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'ejercicio/rama-nueva' on GitHub by visiting:
remote:   https://github.com/salocin-95/fuzzy-pancake/pull/new/ejercicio/rama-nueva
remote:
To github.com:salocin-95/fuzzy-pancake.git
 * [new branch]      ejercicio/rama-nueva -> ejercicio/rama-nueva
```

 **fuzzy-pancake** Public Pin Unwatch 1

 **ejercicio/rama-nueva** had recent pushes 47 seconds ago Compare & pull request

main 2 Branches 0 Tags Go to file Add file Code

 **salocin-95** Agrega mi-texto.txt 1e1f79c · 7 minutes ago 2 Commits

 README.md	Initial commit	11 minutes ago
 mi-texto.txt	Agrega mi-texto.txt	7 minutes ago

README

fuzzy-pancake

ALUMNO: RODRIGO NICOLAS

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).

base: main ← compare: ejercicio/rama-nueva ✓ Able to merge. These branches can be automatically merged.

Add a title

Agrega modificacion desde la rama nueva

Add a description

Write Preview H B I

Add your description here...

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

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Use [Closing keywords](#) to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando: `git clone https://github.com/tuusuario/conflict-exercise.git`
- Entra en el directorio del repositorio: `cd conflict-exercise`

ALUMNO: RODRIGO NICOLAS

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo
● $ git clone git@github.com:salocin-95/psychic-system.git
Cloning into 'psychic-system'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch: git checkout -b feature-branch

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main)
● $ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (feature-branch)
○ $
```

• Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.

- Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in feature-branch"

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (feature-branch)
● $ git add README.md

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (feature-branch)
● $ git commit -m "Modifica una línea en README.md"
[feature-branch d389430] Modifica una línea en README.md
1 file changed, 1 insertion(+)
```

Paso 4: Volver a la rama principal y editar el mismo archivo:

- Cambia de vuelta a la rama principal (main): git checkout main

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (feature-branch)
● $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main)
○ $
```

• Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.

- Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in main branch"

ALUMNO: RODRIGO NICOLAS

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main)
• $ git commit -m "Modifica una línea en README.md en la main branch"
[main 9178412] Modifica una línea en README.md en la main branch
1 file changed, 1 insertion(+)
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main: git merge feature-branch
- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main)
• $ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main|MERGING)
• $
```

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:
<<<<<< HEAD Este es un cambio en la main branch. ===== Este es un cambio en la feature branch. >>>>>> feature-branch
- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge: git add README.md git commit -m "Resolved merge conflict"

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main|MERGING)
• $ git add README.md

Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main|MERGING)
• $ git commit -m "Resuelve el conflicto del merge"
[main ab495af] Resuelve el conflicto del merge
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub: git push origin main

ALUMNO: RODRIGO NICOLAS

```
Rodrigo@DESKTOP-6GL2J07 MINGW64 ~/Desktop/Facultad/codigo/psychic-system (main)
$ git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 769 bytes | 769.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To github.com:salocin-95/psychic-system.git
ec5f530..ab495af main -> main
```

- También sube la feature-branch si deseas: git push origin feature-branch

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.

The screenshot shows the GitHub interface for the repository 'psychic-system', which is public. At the top, there are buttons for 'Pin' and 'Unwatch'. Below this, the repository navigation bar shows 'main' as the selected branch, with '1 Branch' and '0 Tags'. A search bar 'Go to file' and buttons for 'Add file' and 'Code' are also present. The commit history section shows a single commit by 'salocin-95' titled 'Resuelve el conflicto del merge' with commit hash 'ab495af' and a timestamp of '1 minute ago'. Below the commit history, the 'README' file is selected, showing its content: 'psychic-system' followed by 'Ejercicio 3 del TP2 de Programacion I de TUP Este es un cambio en la main branch. Este es un cambio de la feature-branch'.