

Práctico 2: Git y GitHub

ALUMNA: DIANA FALLA

¿Qué es GitHub?

GitHub es una plataforma en la nube que permite a los desarrolladores crear, almacenar, compartir y colaborar en proyectos de código. Se basa en el sistema de control de versiones Git

• ¿Cómo crear un repositorio en GitHub?

Pasos para crear un repositorio en GitHub, puedes:

1. Ir a [GitHub](#) – Ingresar a tu cuenta
2. En la esquina superior derecha, seleccionar Nuevo repositorio
3. Escribir un nombre y una descripción
4. Elegir la visibilidad del repositorio (Pública o Privada)
5. Seleccionar Inicializar este repositorio con un archivo Léame
6. Hacer clic en Crear repositorio

• ¿Cómo crear una rama en Git?

Se puede hacer usando el comando `git branch` seguido del nombre de la nueva rama. Por ejemplo, para crear una rama llamada `primera_rama`, puedes usar `git branch primera_rama`

• ¿Cómo cambiar a una rama en Git?

Usar el comando `git checkout`. Por ejemplo, para cambiar a la rama `master` desde la rama `primera_rama`, puedes usar `git checkout master`

• ¿Cómo fusionar ramas en Git?

Usar el comando `git merge` seguido del nombre de la rama que quieres fusionar.

• ¿Cómo crear un commit en Git?

El comando `git commit` de Git crea una instantánea de los cambios que se han preparado en un repositorio. Es una de las funciones principales de Git.

Para hacer un commit:

1. Usar `git add` para seleccionar los cambios que se van a preparar (`git add .`)

2. Usar git commit para crear la instantánea de los cambios
3. Incluir un mensaje que describa los cambios (git commit es -m "mensaje")

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

Para subir cambios a GitHub, puedes usar los comandos git add, git commit, y git push

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es una copia de un proyecto que se encuentra alojada en un servidor remoto, en internet o en otra red. Se comparte entre varios miembros de un equipo

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

Para agregar un repositorio remoto nuevo, use el comando git remote add en el terminal, dentro del directorio donde está almacenado su repositorio. El comando git remote add toma dos argumentos: Un nombre remoto, por ejemplo, origin.

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

Para empujar cambios a un repositorio remoto con Git, puedes usar el comando git push

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

Para obtener los cambios de un repositorio remoto, puedes usar el comando git pull. Este comando descarga y fusiona los cambios del repositorio remoto en el repositorio local

- **¿Qué es un fork de repositorio?**

Un fork de repositorio es una copia exacta de un repositorio original, que se crea en una cuenta de GitHub o Bitbucket.

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

1. Buscar el repositorio
2. Hacer click en el botón "Fork" en la esquina superior derecha de la página del repositorio

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

Debemos dirigirnos a la solapa de Pull requests.

Le damos click en new pull request.

Nos aparece una ventana con el resumen, en donde podemos ver los cambios en comparación al repo original.

Siguiente paso darle click en Create pull request (colocamos algún mensaje global)

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

En sus pull requests del autor del repo podrá ver el mensaje que le enviamos, y si esta correcto realizar el cambio.

Si el usuario original determina que esta modificación es correcta y no genera problemas con la

rama master de su repositorio local remoto, puede hacer Merge pull request y de esta manera sumará a su repositorio los cambios.

- **¿Qué es un etiqueta en Git?**

En Git, una etiqueta es un nombre que se asigna a una confirmación específica en el historial de un proyecto.

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

Para crear una etiqueta en Git, puedes usar el comando `git tag`. Puedes crear etiquetas anotadas o ligeras.

Tipo de etiqueta	Características
------------------	-----------------

Anotada	Incluye metadatos como el nombre del etiquetador, la fecha y un mensaje
---------	---

Ligera	Más simple, solo contiene el nombre y un puntero a una confirmación
--------	---

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

Crear una etiqueta en tu repositorio local.

Puedes crear una etiqueta anotada o una etiqueta ligera (simplemente un puntero a un commit):

Ejemplo:

```
git tag v1.0
```

Podes verificar las etiquetas que has creado localmente con: `git tag`

- **¿Qué es un historial de Git?**

El historial de Git es un registro de todos los cambios realizados en un proyecto, almacenado como un gráfico de confirmaciones.

- **¿Cómo ver el historial de Git?**

Para ver el historial de Git, puedes usar el comando **git log**. Este comando muestra el historial de confirmaciones de un repositorio en orden cronológico inverso.

- **¿Cómo buscar en el historial de Git?**

Se puede filtrar y localizar commits específicos, usando diversos comandos:

1. La opción `--grep`: `git log --grep="palabra clave"`
2. `git log -- nombre_del_archivo`
3. `git log --since="2024-01-01" --until="2024-01-31"`
4. `git log --author="Nombre del Autor"`

- **¿Cómo borrar el historial de Git?**

El comando **git reset**

Se puede usar de diferentes formas, para quitar el STAGE de:

- `git reset --` (todos los archivos y carpetas del proyecto.)
- `git reset nombreArchivo` – (Archivo indicado)
- `git reset nombreCarpeta/nombreArchivo`
- `git reset nombreCarpeta/*.extensión` – Si cumple con la condición indicada
- `git reset nombreCarpeta/ -` (Todos los archivos de esa carpeta)

- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado en GitHub es un espacio de almacenamiento de código, archivos y revisiones que solo está disponible para el dueño y los colaboradores que se le autoricen.

- **¿Cómo crear un repositorio privado en GitHub?**

Para crear un repositorio privado en GitHub:

1. Ir a la esquina superior derecha de cualquier página de GitHub
2. Seleccionar Nuevo repositorio
3. Escribir un nombre para el repositorio
4. Añadir una descripción (opcional)
5. Elegir la visibilidad del repositorio
6. Seleccionar Privado
7. Hacer clic en Crear repositorio

- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Primero acceder al repositorio

- Haz clic en "Settings" del repositorio.
- Selecciona "Collaborators" que se encuentra en el menú de la izquierda.
- En la sección "Collaborators", haz clic en el botón "Add people" e ingresa el nombre de usuario de GitHub de la persona que deseas invitar.
- Selecciona el nivel de acceso que deseas otorgar: Read, Triage, Write, Maintain, o Admin.
- Luego hacer click en el botón "Add" para enviar la invitación.

- **¿Qué es un repositorio público en GitHub?**

Un repositorio público en GitHub es un espacio de almacenamiento que se puede acceder desde cualquier parte del mundo. En él se pueden guardar archivos, código y el historial de cambios de cada archivo.

- **¿Cómo crear un repositorio público en GitHub?**

En la esquina superior derecha, seleccionar Nuevo repositorio

Escribir un nombre para el repositorio

Añadir una descripción opcional

Elegir la visibilidad del repositorio como "público"

Seleccionar Inicializar este repositorio con un archivo Léame

Hacer clic en Crear repositorio

- **¿Cómo compartir un repositorio público en GitHub?**

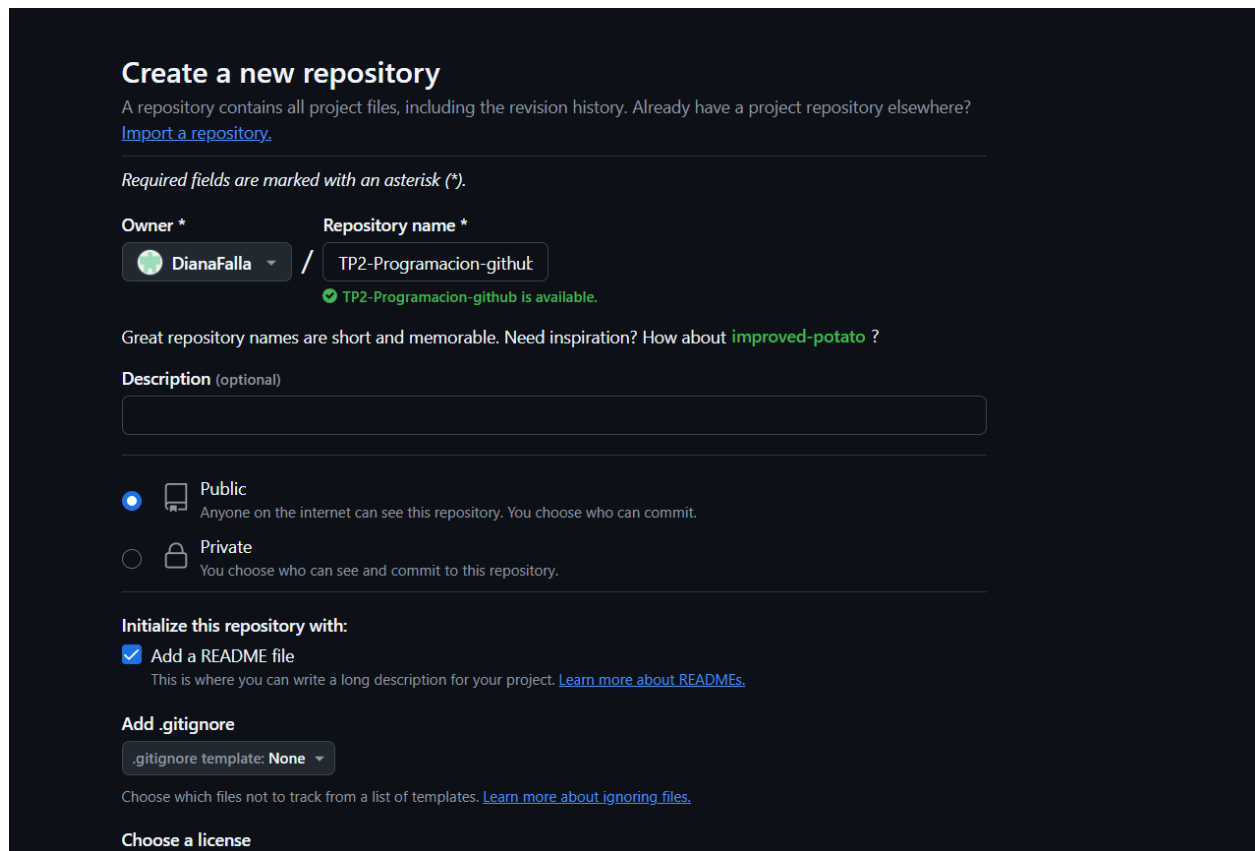
Para compartir un repositorio en GitHub, puedes invitar a personas o equipos a colaborar en él. También puedes hacer público tu repositorio.

Paso	Instrucción
1	Ve a la página principal del repositorio
2	Haz clic en Configuración
3	En la sección Acceso, haz clic en Colaboradores y equipos
4	Haz clic en Agregar personas o Agregar equipos
5	Selecciona el rol de repositorio que quieres conceder
6	Haz clic en Agregar NOMBRE al REPOSITORIO

2) Realizar la siguiente actividad:

- **Crear un repositorio.**
 - o Dale un nombre al repositorio.
 - o Elige el repositorio sea público.

o Inicializa el repositorio con un archivo.




The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there's a note that required fields are marked with an asterisk. The 'Owner' field is set to 'DianaFalla' and the 'Repository name' field is 'TP2-Programacion-github'. A green checkmark indicates that the repository name is available. There's a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about **improved-potato** ?'. The 'Description' field is optional and currently empty. Under the 'Visibility' section, 'Public' is selected, with a note that anyone on the internet can see the repository. 'Private' is also an option. The 'Initialize this repository with:' section has 'Add a README file' checked, with a link to learn more about READMEs. The 'Add .gitignore' section shows a dropdown menu set to 'None', with a link to learn more about ignoring files. At the bottom, there's a 'Choose a license' option.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * **Repository name ***


 DianaFalla / TP2-Programacion-github

✔ TP2-Programacion-github is available.

Great repository names are short and memorable. Need inspiration? How about **improved-potato** ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

• Agregando un Archivo

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

o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"`

en la línea de comandos.

```

MINGW64:/c/Users/Usuario/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION (master)
$ git clone https://github.com/DianaFalla/TP2-Programacion-github.git
Cloning into 'TP2-Programacion-github'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION (master)
$ cd TP2-Programacion-github

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (main)
$ echo "Este es mi primer archivo" > primer-archivo.txt

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (main)
$ git add .
warning: in the working copy of 'primer-archivo.txt', LF will be replaced by CRLF the next time Git touches it

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (main)
$ git commit -m "Agregando mi archivo"
[main 40ba9bc] Agregando mi archivo
1 file changed, 1 insertion(+)
create mode 100644 primer-archivo.txt

```

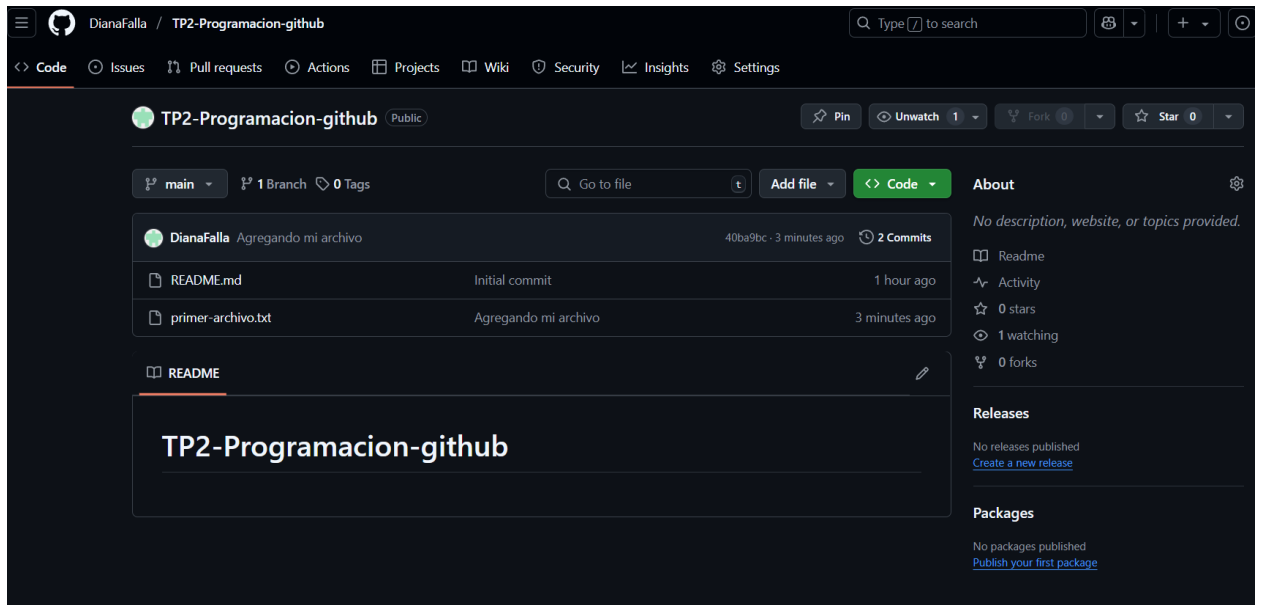
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

```

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (main)
$ git branch
* main

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (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% (2/2), done.
Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/DianaFalla/TP2-Programacion-github.git
24bdfce..40ba9bc main -> main

```

- Creando Branchs

- o Crear una Branch
- o Realizar cambios o agregar un archivo
- o Subir la Branch

MINGW64:/c/Users/Usuario/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git branch

main

* nuevaRama

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ echo "Modificacion realizada desde nuevaRama" > primer-archivo.txt

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git add .

warning: in the working copy of 'primer-archivo.txt', LF will be replaced by CRLF the next time Git touches it

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git status

On branch nuevaRama

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: primer-archivo.txt

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git commit -m "Agregando modificacion a nuevaRama"

[nuevaRama 01b4033] Agregando modificacion a nuevaRama

1 file changed, 1 insertion(+), 1 deletion(-)

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git status

On branch nuevaRama

nothing to commit, working tree clean

Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/TP2-Programacion-github (nuevaRama)

\$ git push origin nuevaRama

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), 342 bytes | 342.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

remote:

remote: Create a pull request for 'nuevaRama' on GitHub by visiting:

remote: <https://github.com/DianaFalla/TP2-Programacion-github/pull/new/nuevaRama>

remote:

To <https://github.com/DianaFalla/TP2-Programacion-github.git>

* [new branch] nuevaRama -> nuevaRama

DianaFalla / TP2-Programacion-github

Q Type 7 to search

+

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

TP2-Programacion-github

Public

Pin

Unwatch 1

Fork 0

Star 0

nuevaRama had recent pushes 4 seconds ago

Compare & pull request

main 1 Branch 0 Tags

Go to file

Add file

<> Code

DianaFalla Agregando mi archivo 40ba9bc · 14 minutes ago 2 Commits

README.md Initial commit 1 hour ago

primer-archivo.txt Agregando mi archivo 14 minutes ago

README

TP2-Programacion-github

About

No description, website, or topics provided

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

<> Code

Issues

Pull requests 1

Actions

Projects

Wiki

Security

Insights

Settings

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: main

compare: nuevaRama

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

Agregando modificacion a nuevaRama #1

No description available

View pull request

1 commit

1 file changed

1 contributor

Commits on Apr 16, 2025

Agregando modificacion a nuevaRama

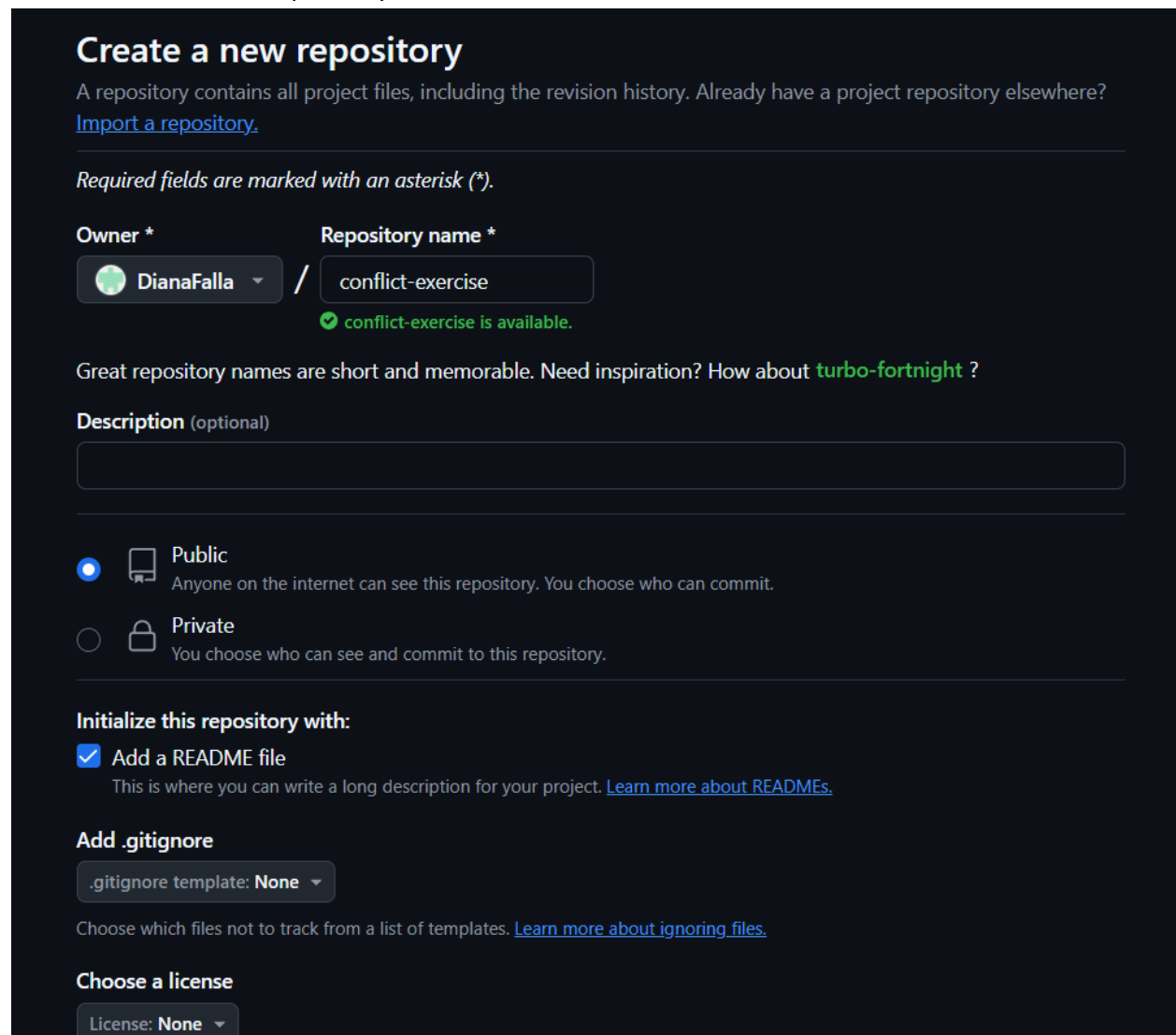
DianaFalla committed 5 minutes ago

01b4033

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".




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * **Repository name ***


 DianaFalla / conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **turbo-fortnight** ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

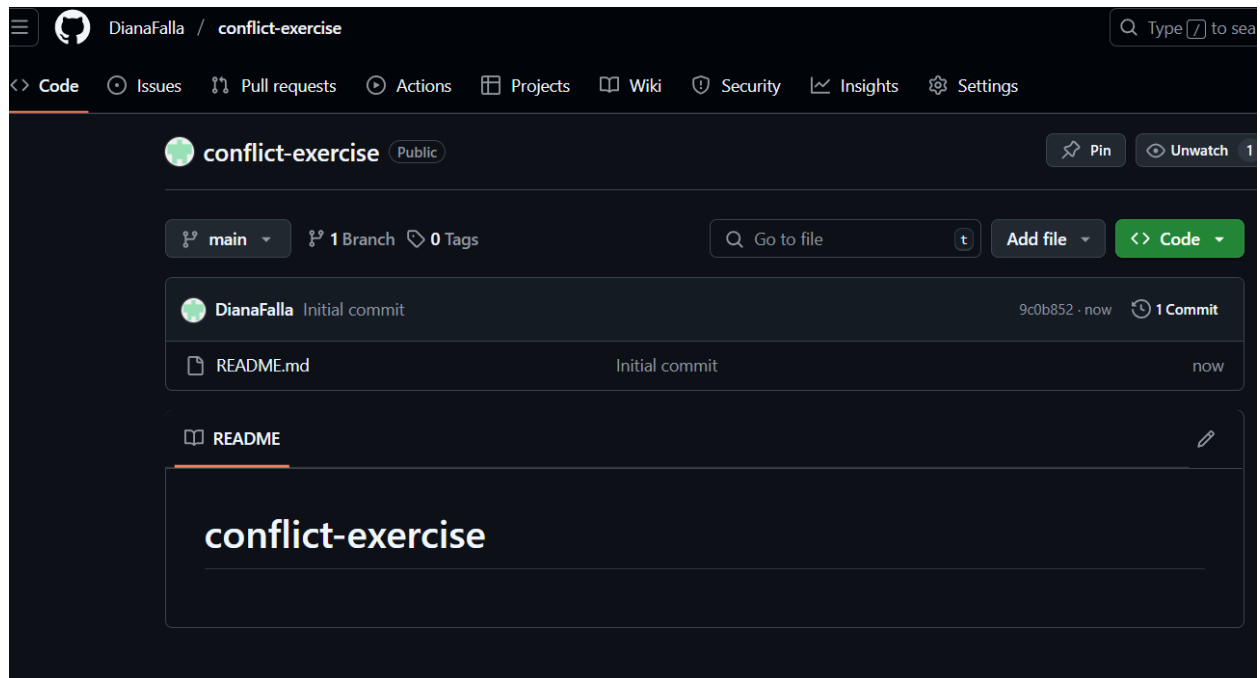
Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**



Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

```
MINGW64:/c/Users/Usuario/Documents/DIANA/UTN/PROGRAMACION
Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION (master)
$ git clone https://github.com/DianaFalla/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

Entra en el directorio del repositorio:

Paso 3: Crear una nueva rama y editar un archivo

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

Paso 5: Hacer un merge y generar un conflicto

Paso 6: Resolver el conflicto

Paso 7: Subir los cambios a GitHub

Paso 8: Verificar en GitHub

```
Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION (master)
$ cd conflict-exercise

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (feature-branch)
$ git add README.md

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch be72d5c] Added a line in feature-branch
1 file changed, 1 insertion(+), 1 deletion(-)

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main)
$ git add README.md

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main d124a2e] Added a line in main branch
1 file changed, 3 insertions(+), 1 deletion(-)

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (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.

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main|MERGING)
$ git add README.md

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main|MERGING)
$ git commit -m "Resolved main conflict"
[main 46cc10a] Resolved main conflict

Usuario@LAPTOP-SIGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
```

```
Usuario@LAPTOP-5IGVATP3 MINGW64 ~/Documents/DIANA/UTN/PROGRAMACION/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/DianaFalla/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/DianaFalla/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
```

The screenshot shows the GitHub repository page for 'conflict-exercise' by user 'DianaFalla'. The repository is public and has 2 branches and 0 tags. The main branch is selected. The repository has 4 commits, with the latest commit 'Resolved main conflict' by DianaFalla 5 minutes ago. The README file is also shown, with the latest commit 'Resolved main conflict' 5 minutes ago. The README content is: 'ESTE ES UN CAMBIO EN LA MAIN BRANCH' and 'Este es un cambio en la feature branch.' The right sidebar shows the 'About' section with no description, website, or topics provided. It also shows the 'Releases' section with no releases published and a link to 'Create a new release'. The 'Packages' section shows no packages published and a link to 'Publish your first package'.

conflict-exercise Public

main 2 Branches 0 Tags

Go to file Add file Code

DianaFalla Resolved main conflict 46cc10a · 5 minutes ago 4 Commits

README.md Resolved main conflict 5 minutes ago

README

ESTE ES UN CAMBIO EN LA MAIN BRANCH

Este es un cambio en la feature branch.

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package