**Mejores prácticas para Branches**

**Git Bash y Github desktop**

**Git Bash:**

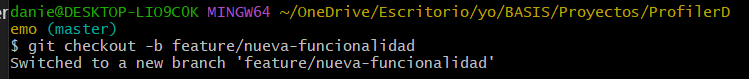
* **Trabajo en ramas separadas**

¿Por qué es importante?

* Evita modificaciones directas a la rama principal (main/master) la cual suele mantener el código estable y listo para producción.
* Permite un mejor ambiente de desarrollos, pruebas y depuración para nuevas funcionalidades sin interrumpir el flujo de trabajo de otros desarrolladores.

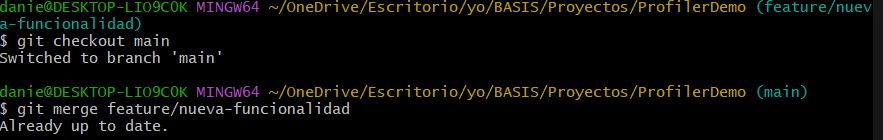
Implementación:

1. Previo al inicio de una nueva tarea, es necesario crear una rama específica para esa funcionalidad:



*Comando: git checkout -b feature/nueva-funcionalidad*

1. Realizar los cambios necesarios en la rama indicada, realizando commits y probando el código.
2. Al finalizar la funcionalidad, hacer merge con la rama principal:



*Comando 1: git checkout main*

*Comando 2: git merge feature/nueva-funcionalidad*

Ejemplo:

* Se tiene una aplicación y se necesita añadir una nueva funcionalidad para mostrar la estadística de los usuarios, se deberá crear la nueva rama para dicha funcionalidad:

*git checkout -b feature/mostrar-estadisticas*

* Se deben realizar los cambios en el código y confirmarlos:

*git add estadisticas.cs*

*git commit -m "Añadida funcionalidad de estadísticas"*

* Hacer merge en los cambios:

*git checkout main*

*git merge feature/mostrar-estadisticas*

* **Nombrar las ramas de manera descriptiva y coherente**

¿Por qué es importante?

* Ayuda con la fácil identificación del propósito de cada rama.
* Facilita el trabajo en equipo cuando se estandarizan los nombres.

Buenas prácticas para nombres de ramas:

1. Uso de prefijos claros:

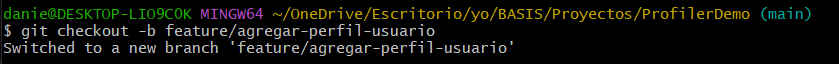
* feature/ → funcionalidades nuevas
* bugfix/ → corrección de errores
* hotfix/ → correcciones urgentes
* release/ → preparación de versiones

1. Uso de guiones para la separación de palabras:

* bugfix/vista-principal

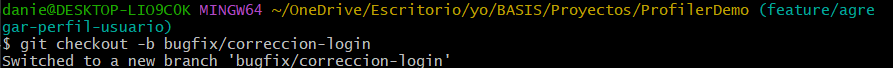
Ejemplo:

* Creación de una nueva rama para una nueva funcionalidad:



*Comando: git checkout -b feature/agregar-perfil-usuario*

* Creación de una rama para corregir un error:



*Comando: git checkout -b bugfix/correccion-login*

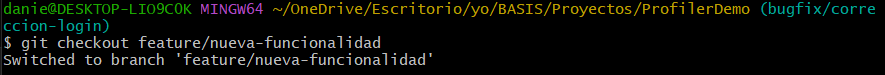
* **Mantener la rama sincronizada con main:**

¿Por qué es importante?

* Reduce conflictos al combinar cambios.
* Garantiza que el trabajo esté en la versión más actualizada del código principal.

Implementación:

1. Cambiar a la rama de trabajo:



*Comando: git checkout feature/nueva-funcionalidad*

1. Traer los últimos cambios de main:



*Comandos:*

*git fetch origin*

*git merge main*

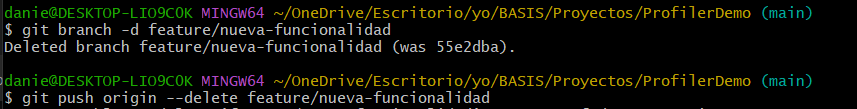
* **Eliminar ramas que ya no son necesarias:**

¿Por qué es importante?

* Mantenimiento de un repositorio limpio y organizado.
* Reducción de riesgos en el caso de trabajar en ramas obsoletas.

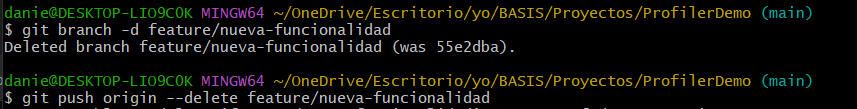
Implementación:

1. Después de haber realizado el merge se debe eliminar la rama de manera local:



*Comando: git branch -d feature/nueva-funcionalidad*

1. Eliminar la rama en el repositorio remoto:



*comando: git push origin --delete feature/nueva-funcionalidad*

* **Eliminar ramas que ya no son necesarias:**

¿Por qué es importante?

* Mantiene un historial lineal y con lectura fácil.
* Evita bifurcaciones innecesarias del historial.

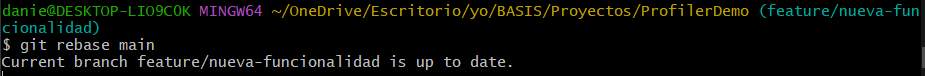
Implementación:

1. Cambiar la rama de trabajo:



*Comando: git checkout feature/nueva-funcionalidad*

1. Rebase con la rama principal:



*Comando: git rebase main*

**Página web:**

* **Protección de la rama principal**

¿Por qué es importante?

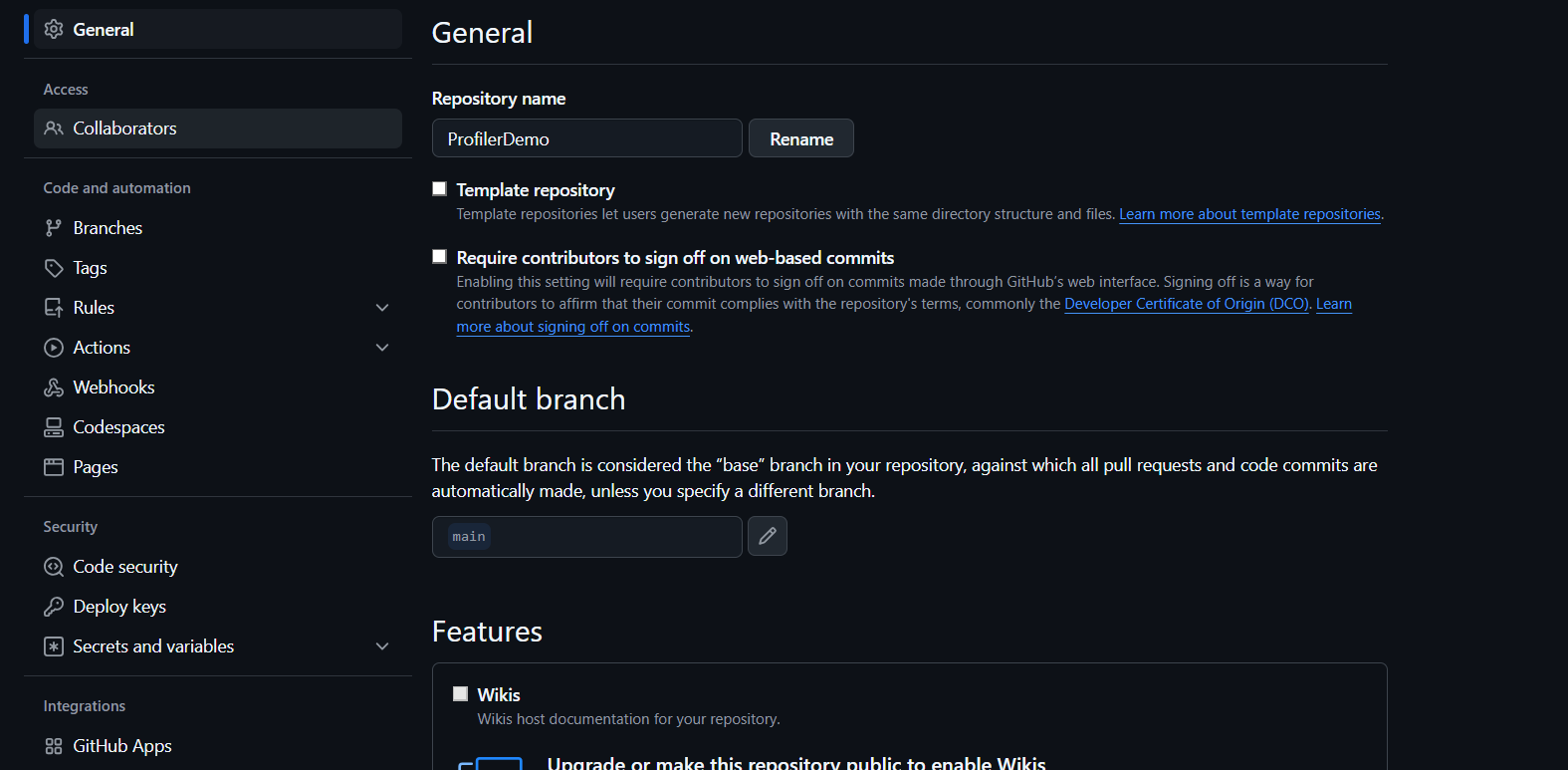
* Evita que se suban cambios sin revisión.
* Mantiene la estabilidad del código base.

Implementación:

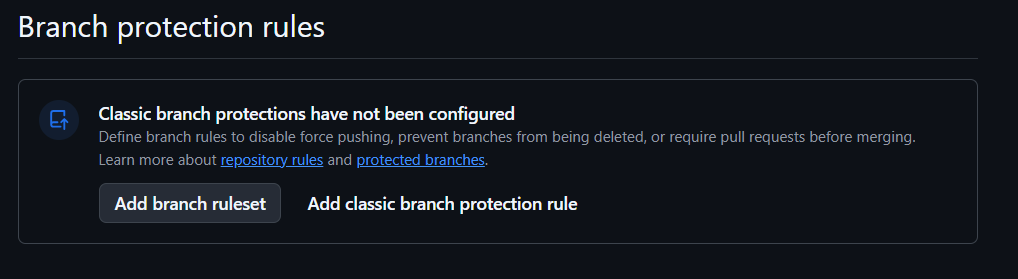
1. Configurar la rama principal como protegida en github.

¿Cómo?

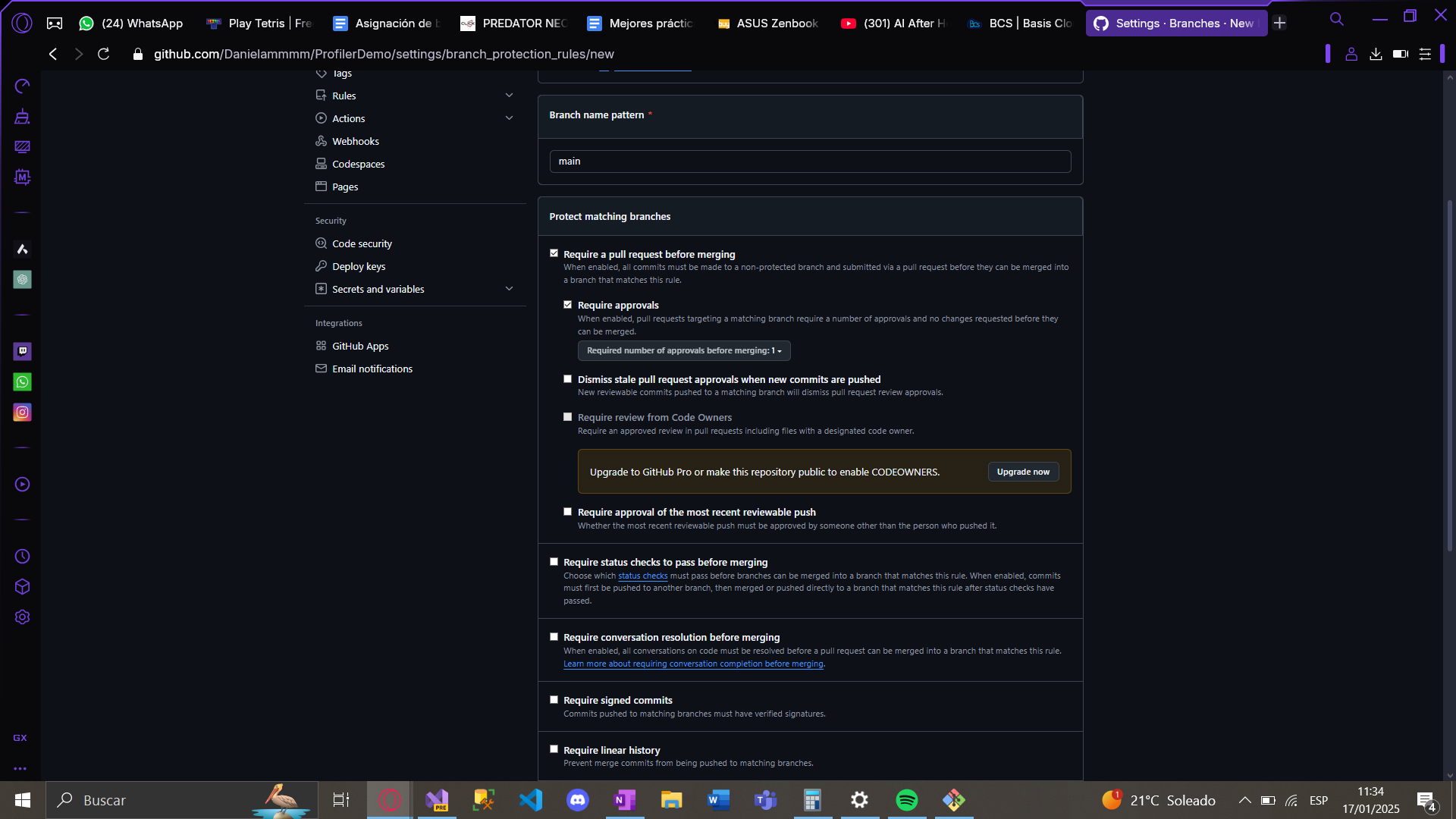
1. Ir a settings en el repositorio y luego branches:



1. Ir a Add classic branch protection rule



1. Agregar las reglas necesarias y el nombre de la rama:



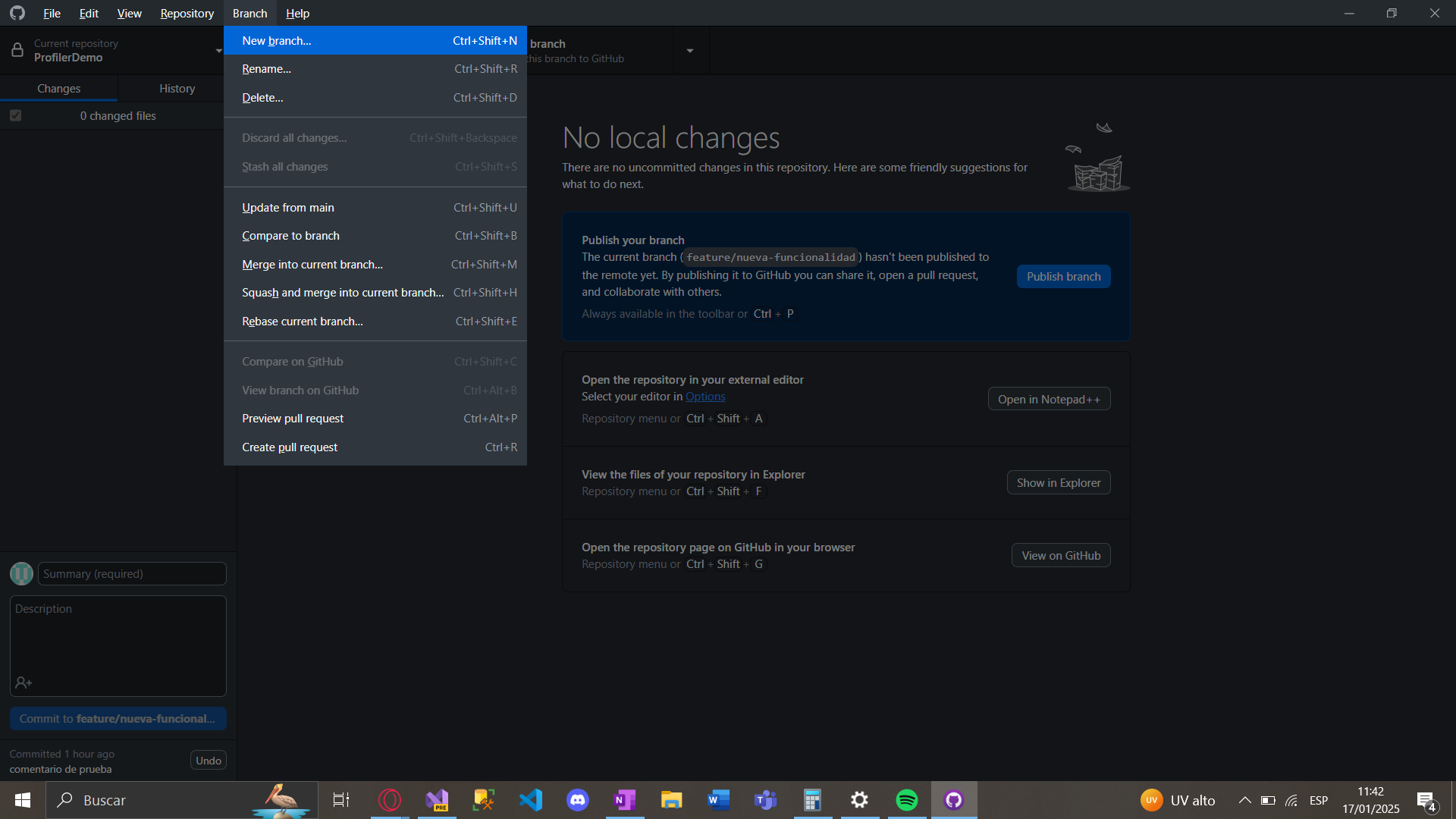
1. Crear la regla.

**Implementación en Github Desktop:**

* **Trabajo en Ramas Separadas:**

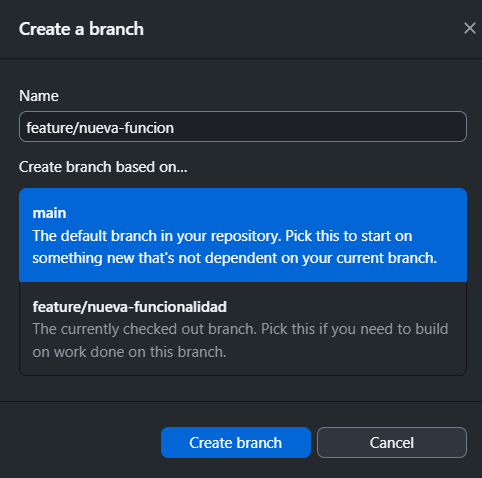
Implementación:

1. Ir al menú branch → New Branch:



*Shortcut: Ctrl+Shift+N*

1. Crear la nueva rama y asociarla con el punto de partida (generalmente main):



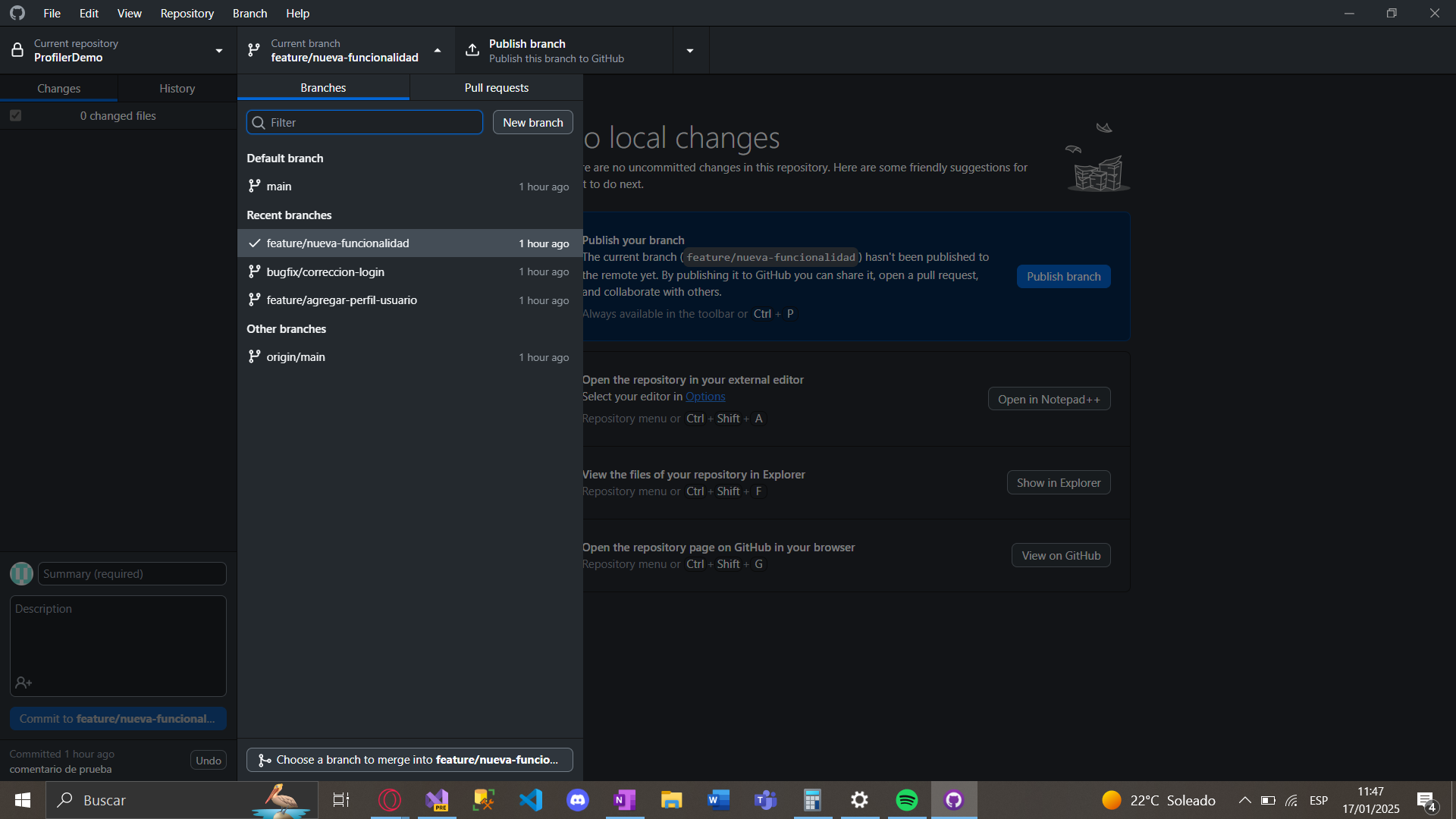
Nota: los prefijos y buenas prácticas para nombrar ramas se aplican aquí también.

1. Crear la rama.

* **Mantener la rama sincronizada con main:**

Implementación:

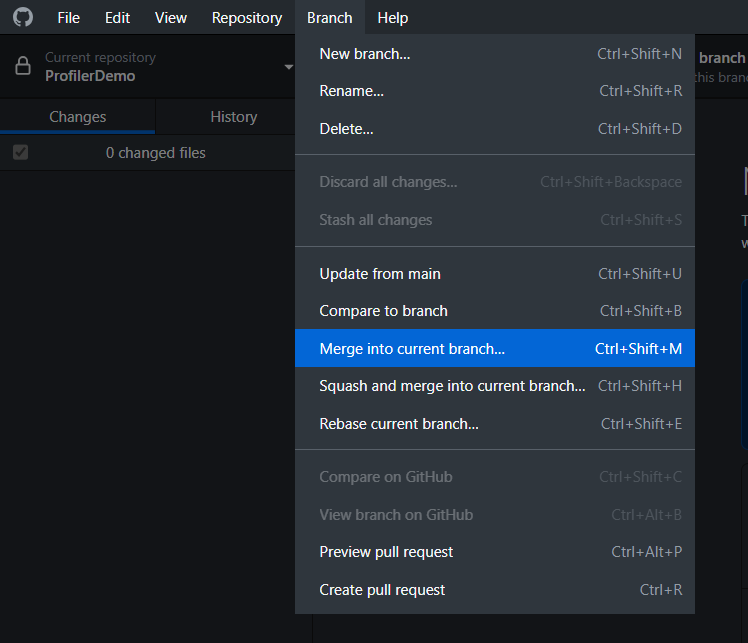
1. Cambiar a la rama en la que se está trabajando desde el menú superior izquierdo.



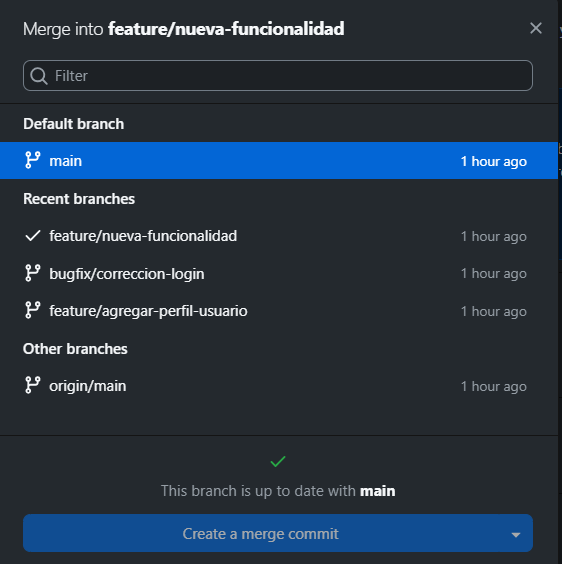
Nota: Asegurarse que la rama main esté actualizada:

Si no esta actualizada, cambiar a la rama main y hacer clic en Fetch origin, luego en Pull origin para traer los últimos cambios:

1. Hacer click en Branch > Merge into current branch:



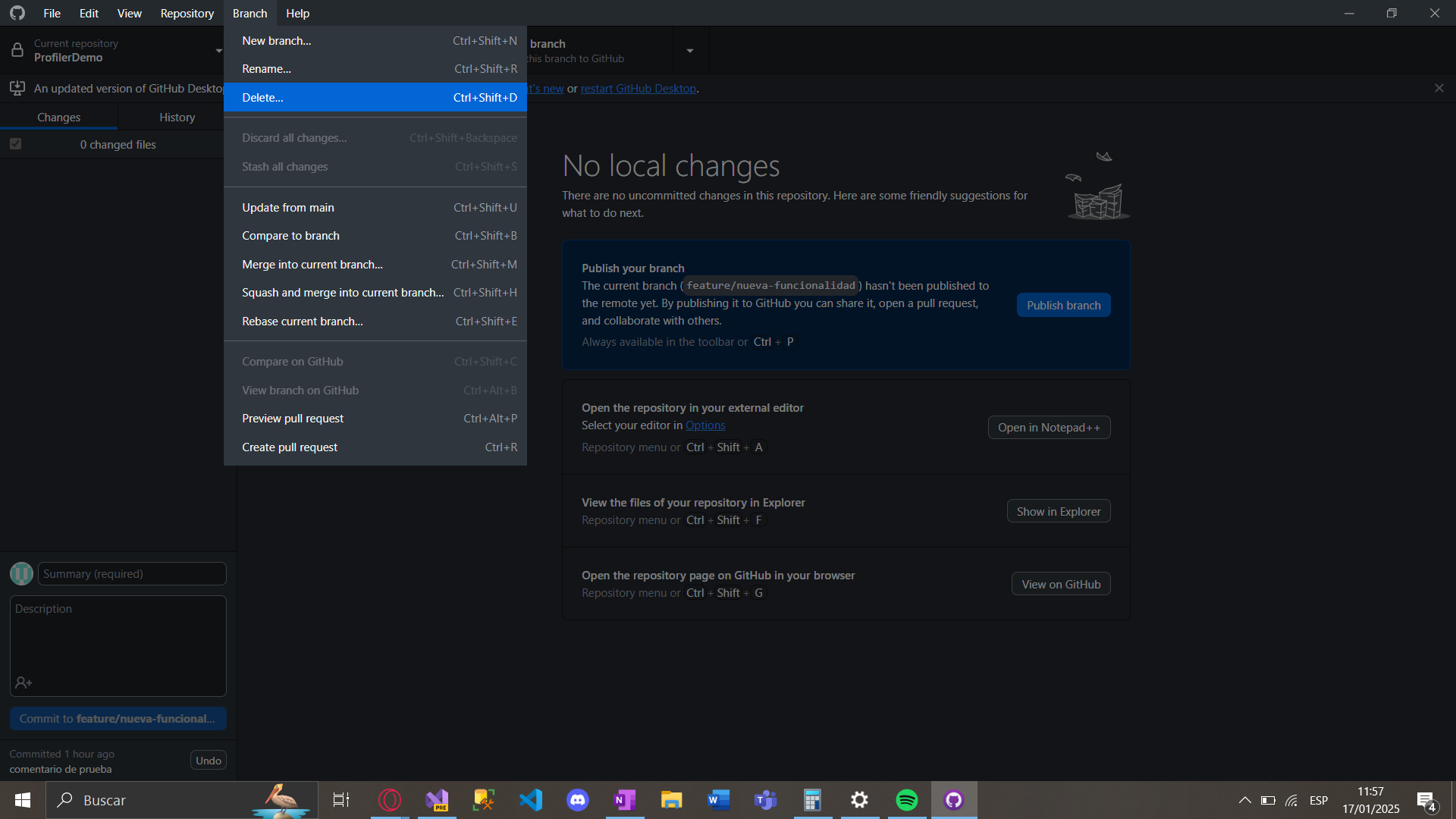
1. Seleccionar main como la rama a fusionar en la rama de trabajo:



### **Eliminar ramas que ya no son necesarias**

Implementación:

1. Cambiar a la rama que se desea eliminar
2. Ir al menú Branch > Delete:



1. Confirmar la eliminación..

* **Sincronizar Ramas Remotas**

Implementación:

1. Crear una rama nueva y hacer push al remoto:

* Cambiar a la nueva rama desde GitHub Desktop.
* Hacer clic en Push origin en la parte superior derecha para subir la rama al repositorio remoto.

1. Cuando esté listo para combinar, ir a GitHub (web) y crear un pull request.