

			
---	---	--	---

Nombre del alumno-a:  
**Aarón Pérez Ramírez**

**NOTA: Antes de nada, cumplimentar el nombre completo del alumno -a**

## OBJETIVOS

---

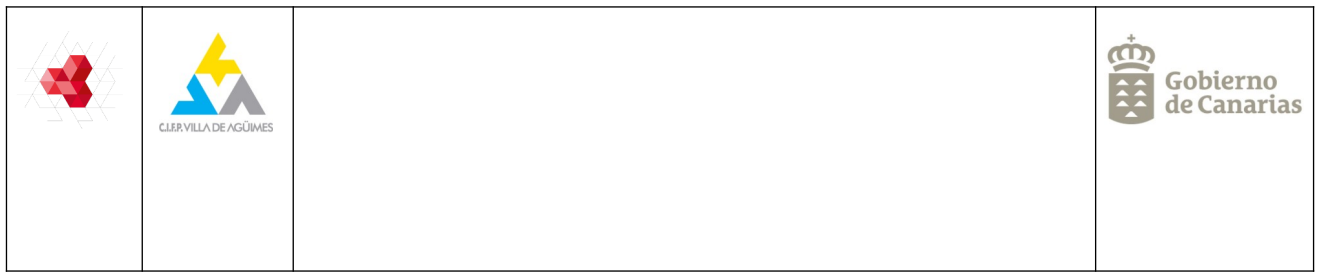
- Conocer y habituarse a la metodología de trabajo, así como la plataforma CAMPUS DE LAS ENSEÑANZAS PROFESIONALES
- Escribir sentencias simples, aplicando la sintaxis del lenguaje y verificando su ejecución sobre navegadores Web.

## RECURSOS

---

- El equipo de aula asignado al propio alumno.
- Carpeta compartida DAW en el equipo del profesor y accesible por red local.
- Software de ofimática Libre Office, (ya preinstalado en el equipo por defecto, no es necesario instalar nada)
- Git y tortoise.
- Software de desarrollo.
- Software para comprimir-descomprimir archivos (nuestro caso, Winrar o 7-ZIP o similar, ya preinstalado en el equipo por defecto)

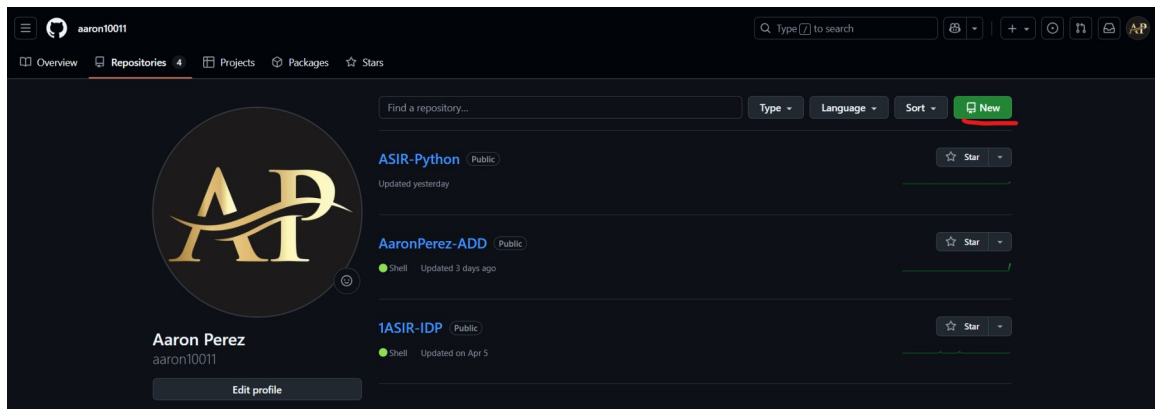




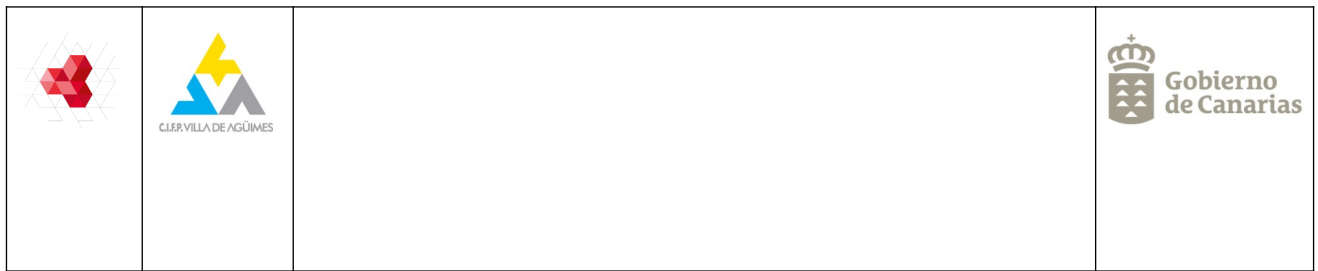
## ENUNCIADO

### 1 Inicialización del Repositorio

- Crea una carpeta llamada `proyecto-final-2025`.
- Inicializa un repositorio Git dentro de esta carpeta.
- Crea un repositorio remoto en GitHub llamado:  
`Repo_AlumnoXXX_20092025` (donde `xxx` son tus iniciales).



Dentro de GitHub nos vamos a nuestro perfil y creamos un nuevo repositorio.



### Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).  
Required fields are marked with an asterisk (\*).

**1 General**

**Owner \*** AP aaron10011 / **Repository name \***

✓ Repo\_AlumnoAPR\_20092025 is available.

Great repository names are short and memorable. How about **studious-octo-engine**?

**Description**

0 / 350 characters

**2 Configuration**

**Choose visibility \*** Public

Choose who can see and commit to this repository

**Add README** Off

READMEs can be used as longer descriptions. [About READMEs](#)

**Add .gitignore** No .gitignore

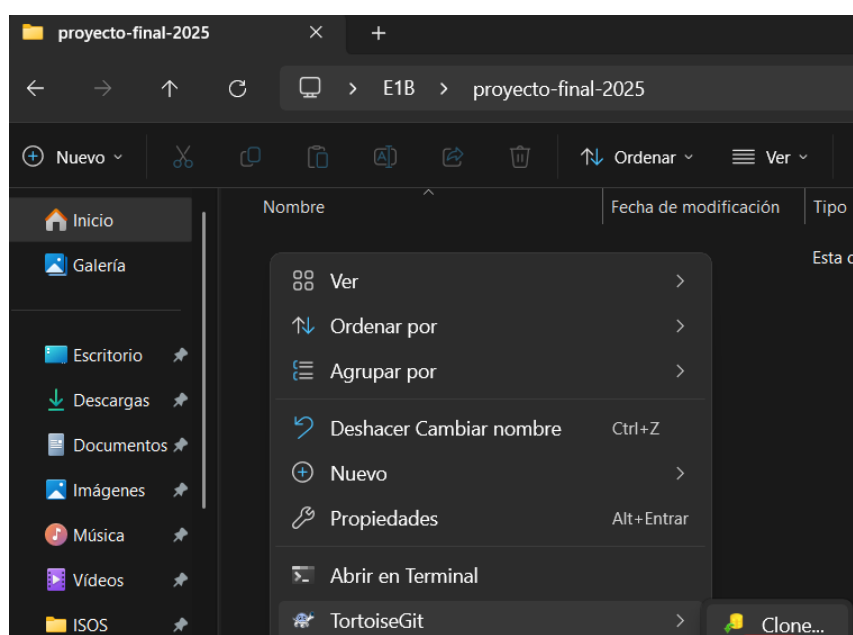
.gitignore tells git which files not to track. [About ignoring files](#)

**Add license** No license

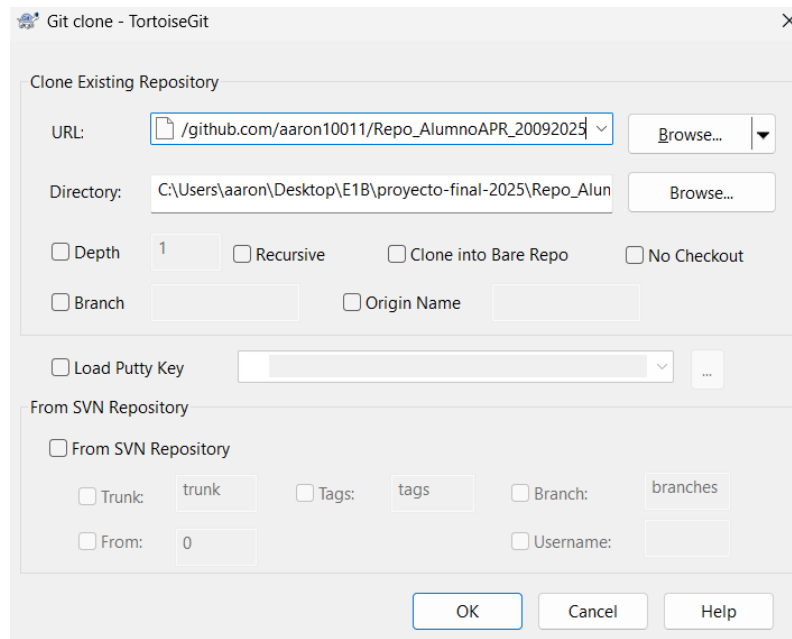
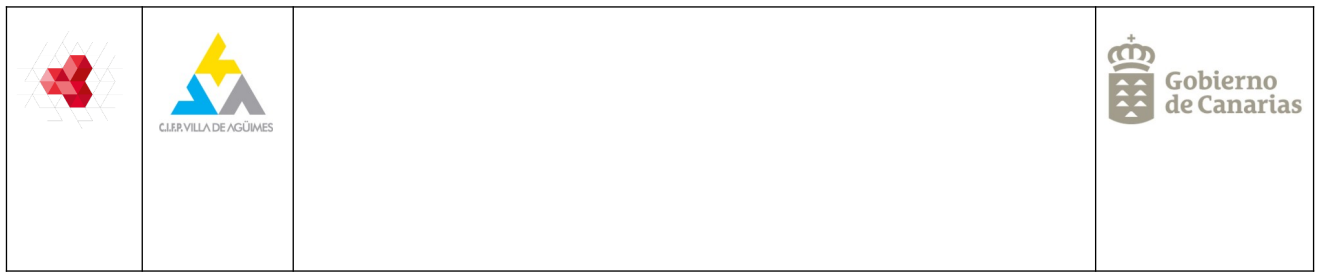
Licenses explain how others can use your code. [About licenses](#)

**Create repository**

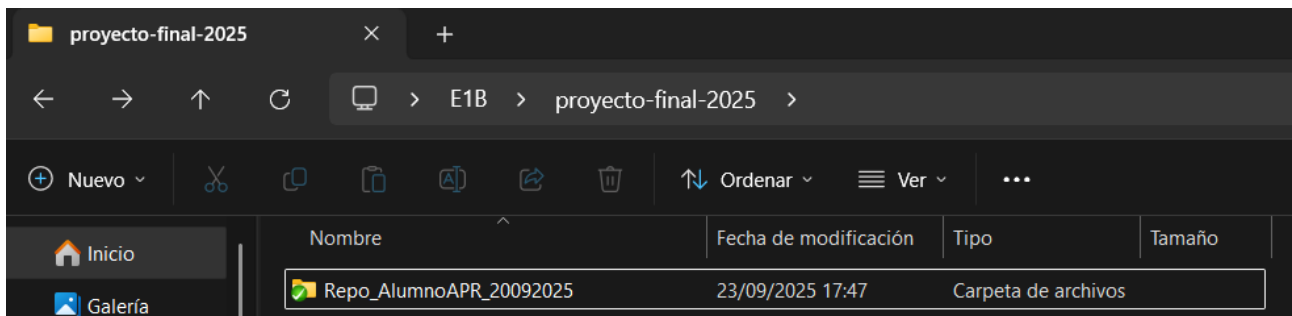
Creamos el repositorio de GitHub, no añadimos el README ya que lo haremos despues.



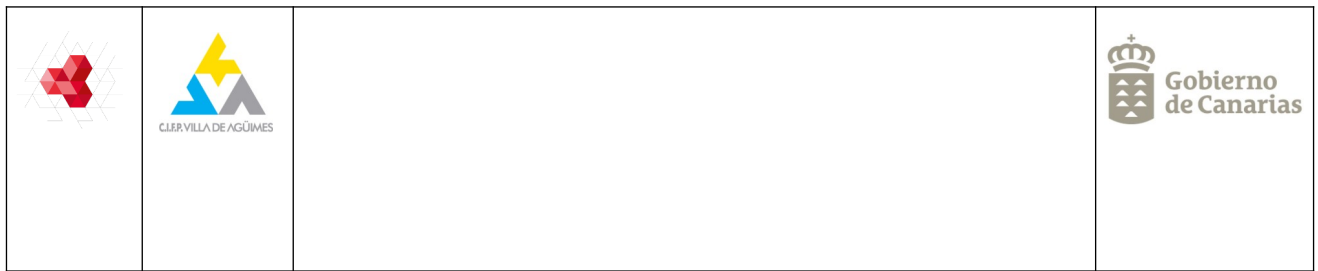
Creamos la carpeta y clonamos nuestro repositorio dentro de la carpeta



Pegamos la URL de nuestro repositorio y le damos a Ok



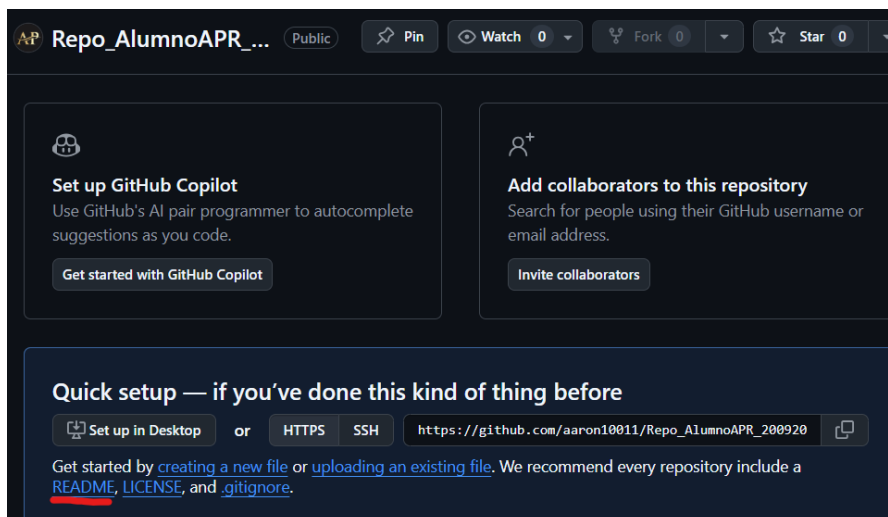
Repositorio clonado



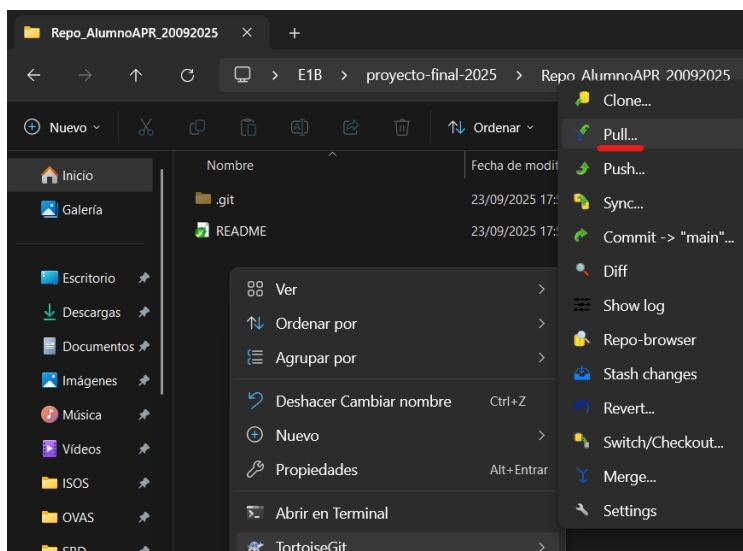
## 2 Creación de la Estructura del Proyecto

- Crea un archivo `README.md` con una descripción breve del proyecto.
- Crea una carpeta `frontend` para el código de la interfaz.
- Dentro de `frontend`, crea un archivo de ejemplo:




`AlumnoXXX_20092025_UI.html`.

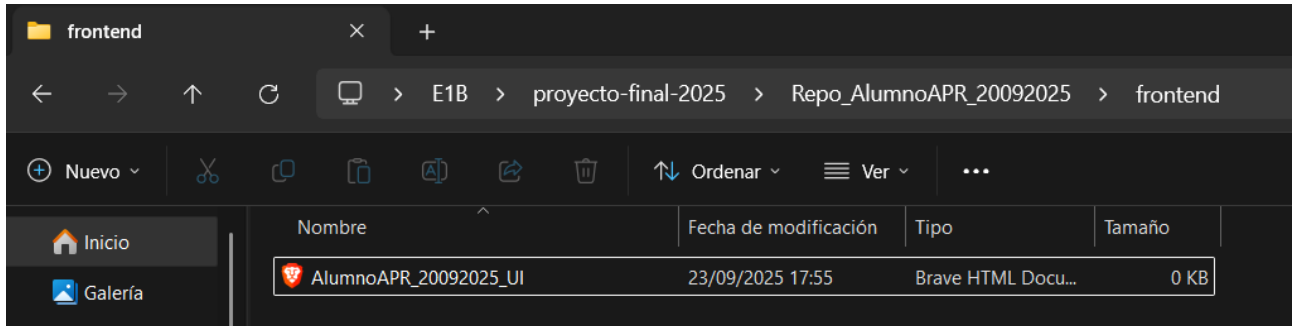


Dentro de nuestro repositorio de GH la damos a añadir un README

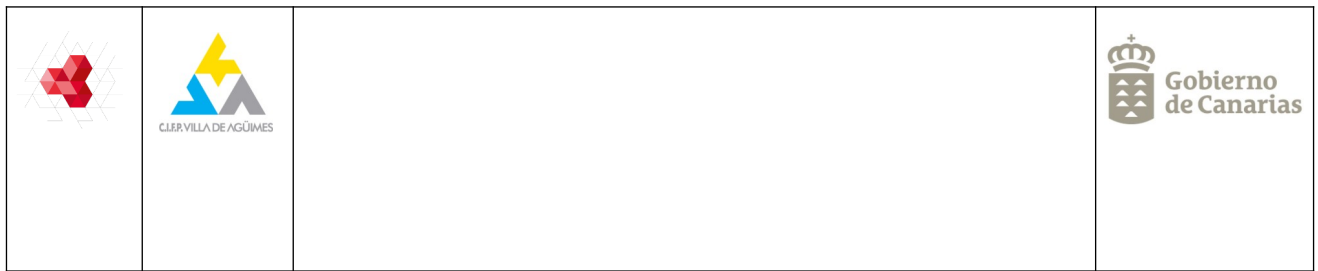


Una vez creado le damos a Pull dentro de donde lo clonamos para que nos aparezca el README

	 CIUDAD DE AGÜIMES		 Gobierno de Canarias
---	--	--	--



Dentro creamos la carpeta frontend y creamos el archivo html.



### 3 Crear Ramas

Desde la rama principal `main` :

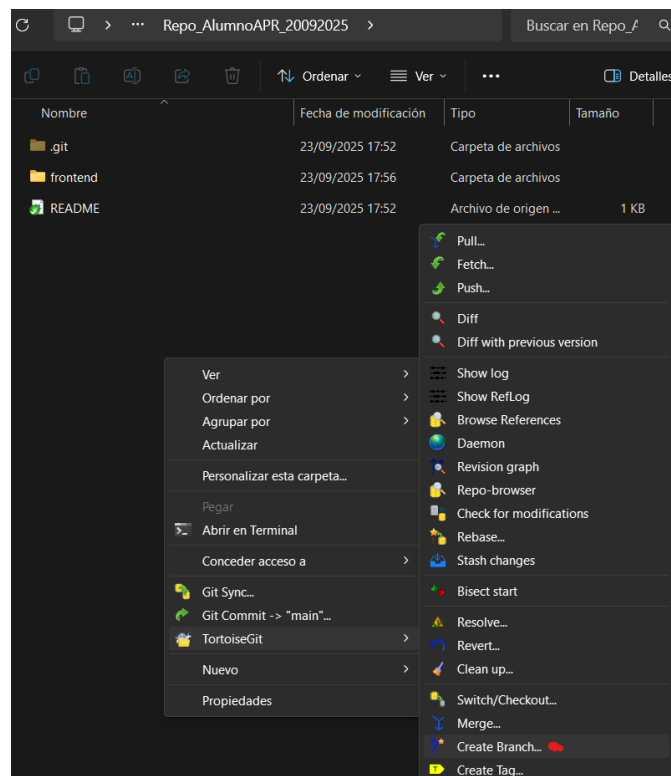
- `develop_AlumnoXXX_20092025`
- `fixes_AlumnoXXX_20092025`

Desde la rama `develop_AlumnoXXX_20092025` :

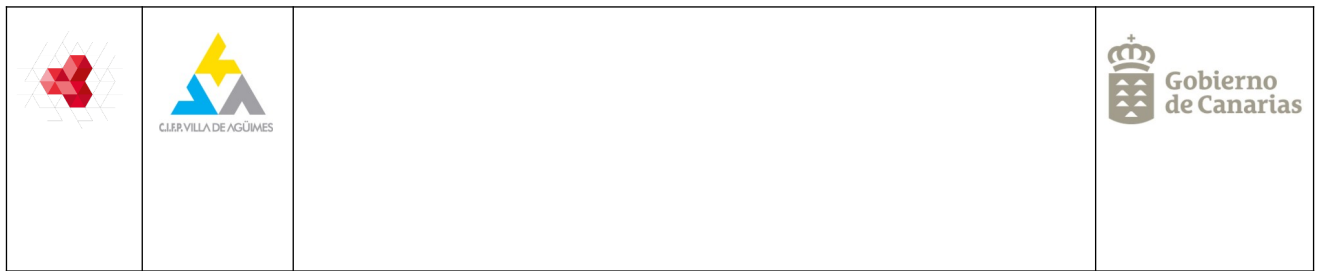
- `FeatureLogin_AlumnoXXX_20092025`
- `FeatureDashboard_AlumnoXXX_20092025`

Verificación:

- Confirma la creación de cada rama usando **Revision Graph** o equivalente.



Le damos a crear nueva rama.



C:\Users\aaaron\Desktop\E1B\proyecto-fin...\Repo\_AlumnoAPR\_20092025 -...

Name

Branch

Base On

☐ HEAD (main)

☒ Branch  ...

☐ Tag

☐ Commit  ...

Options

☒ Track ☐ Force ☐ Switch to new branch

Description

OK Cancel Help

Creamos la rama develop desde main.

C:\Users\aaaron\Desktop\E1B\proyecto-fin...\Repo\_AlumnoAPR\_20092025 -...

Name

Branch

Base On

☐ HEAD (main)

☒ Branch  ...

☐ Tag

☐ Commit  ...

Options

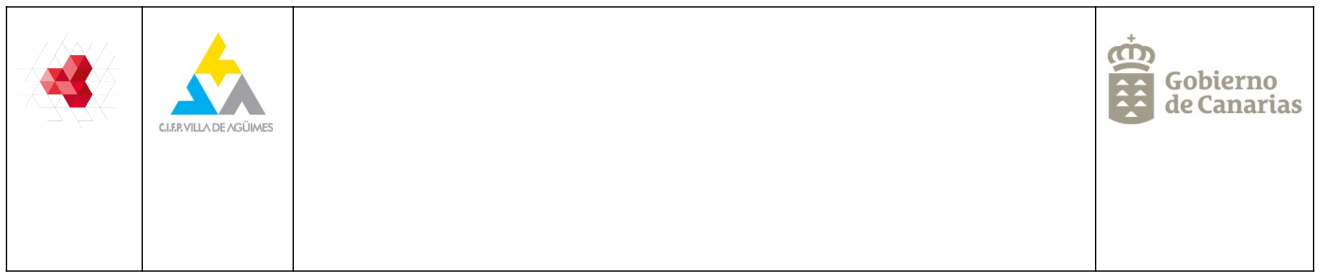
☒ Track ☐ Force ☐ Switch to new branch

Description

OK Cancel Help

Creamos la rama fixes desde main.






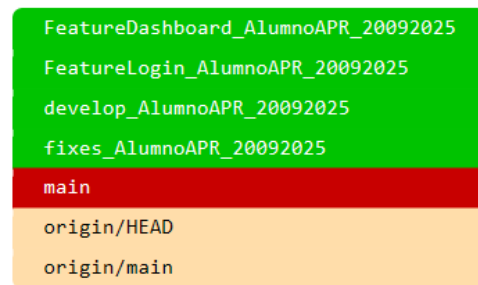
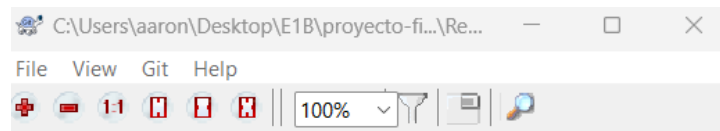
A screenshot of a 'Create Branch' dialog box from a Git client. The window title is 'C:\Users\aaaron\Desktop\E1B\proyecto-fin...\Repo\_AlumnoAPR\_20092025 -...'. The 'Name' section has a 'Branch' field containing 'FeatureLogin\_AlumnoAPR\_20092025'. The 'Base On' section has three radio buttons: 'HEAD (main)', 'Branch' (which is selected), and 'Tag'. The 'Branch' radio button is selected, and its dropdown menu shows 'develop\_AlumnoAPR\_20092025'. There are also 'Tag' and 'Commit' options with empty dropdowns. The 'Options' section has three checkboxes: 'Track' (checked), 'Force' (unchecked), and 'Switch to new branch' (unchecked). There is a 'Description' text area at the bottom. At the very bottom are 'OK', 'Cancel', and 'Help' buttons.

Creamos la rama FeatureLogin desde develop

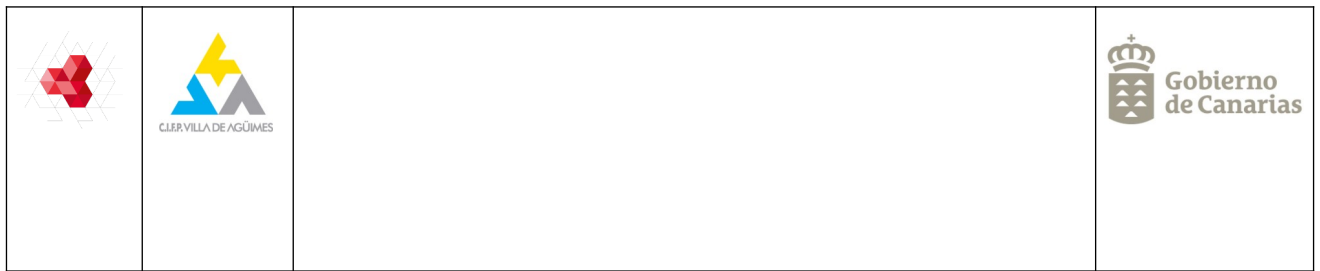
A screenshot of a 'Create Branch' dialog box from a Git client. The window title is 'C:\Users\aaaron\Desktop\E1B\proyecto-fin...\Repo\_AlumnoAPR\_20092025 -...'. The 'Name' section has a 'Branch' field containing 'FeatureDashboard\_AlumnoAPR\_20092025'. The 'Base On' section has three radio buttons: 'HEAD (main)', 'Branch' (which is selected), and 'Tag'. The 'Branch' radio button is selected, and its dropdown menu shows 'develop\_AlumnoAPR\_20092025'. There are also 'Tag' and 'Commit' options with empty dropdowns. The 'Options' section has three checkboxes: 'Track' (checked), 'Force' (unchecked), and 'Switch to new branch' (unchecked). There is a 'Description' text area at the bottom. At the very bottom are 'OK', 'Cancel', and 'Help' buttons.

Creamos la rama FeatureDashboard desde develop

			
---	---	--	---



Confirmación de la creacion de cada rama usando RevisionGraph.

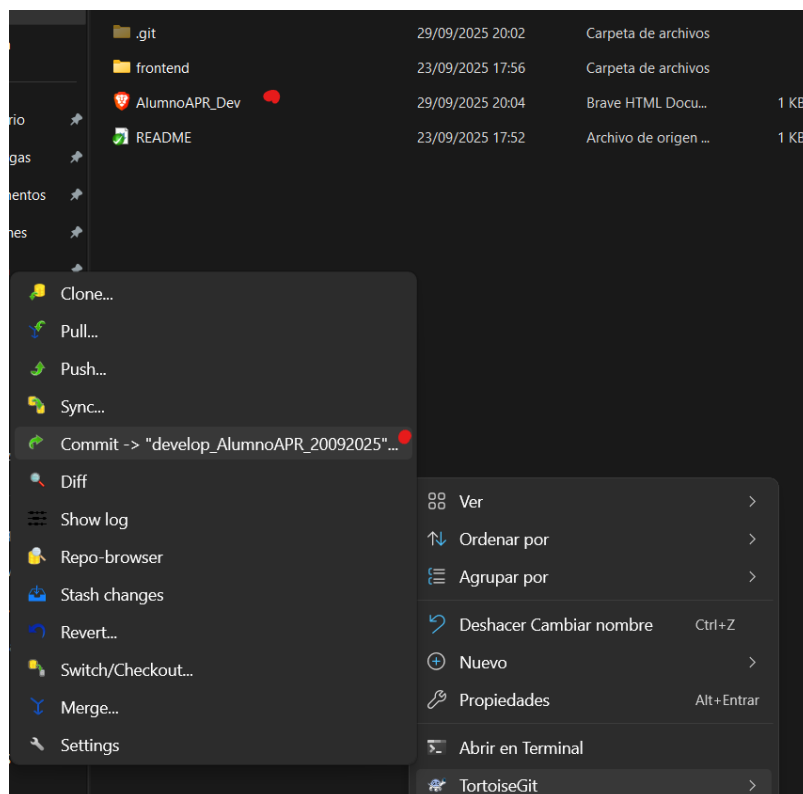


#### 4 Cambios en cada rama

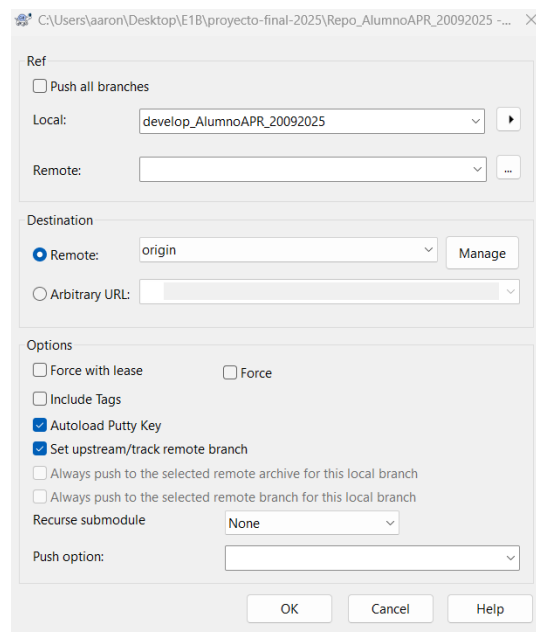
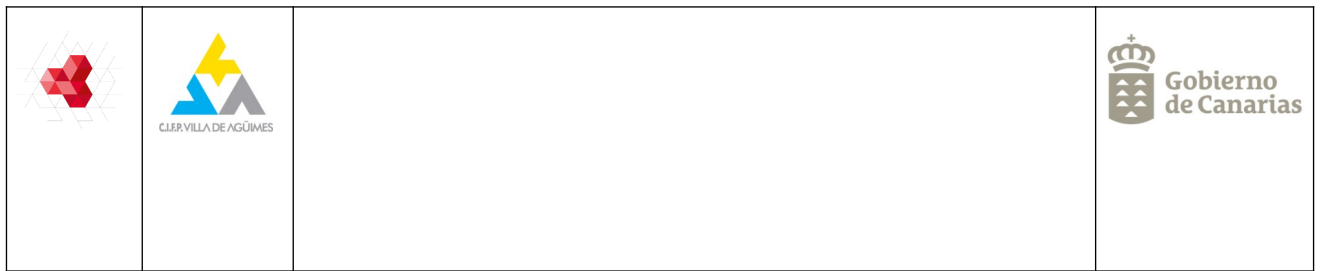
Realiza los cambios indicados en la rama local correspondiente y súbelos al repositorio remoto:

Rama	Archivo	Contenido
develop_AlumnoXXX_20092025	AlumnoXXX_Dev.html	Párrafo: "Development environment"
fixes_AlumnoXXX_20092025	AlumnoXXX_Fix.html	Párrafo: "Bug fixes applied"
FeatureLogin_AlumnoXXX_20092025	AlumnoXXX_Login.html	Párrafo: "Login feature implemented"
FeatureDashboard_AlumnoXXX_20092025	AlumnoXXX_Dashboard.html	Párrafo: "Dashboard feature implemented"

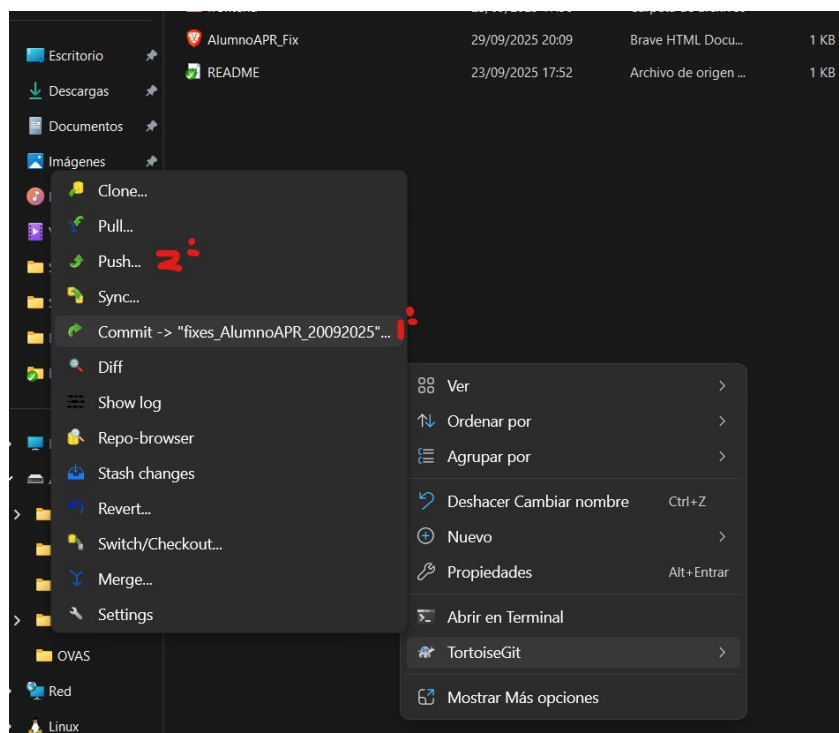
25



Creamos el archivo y hacemos commit desde la rama de develop.

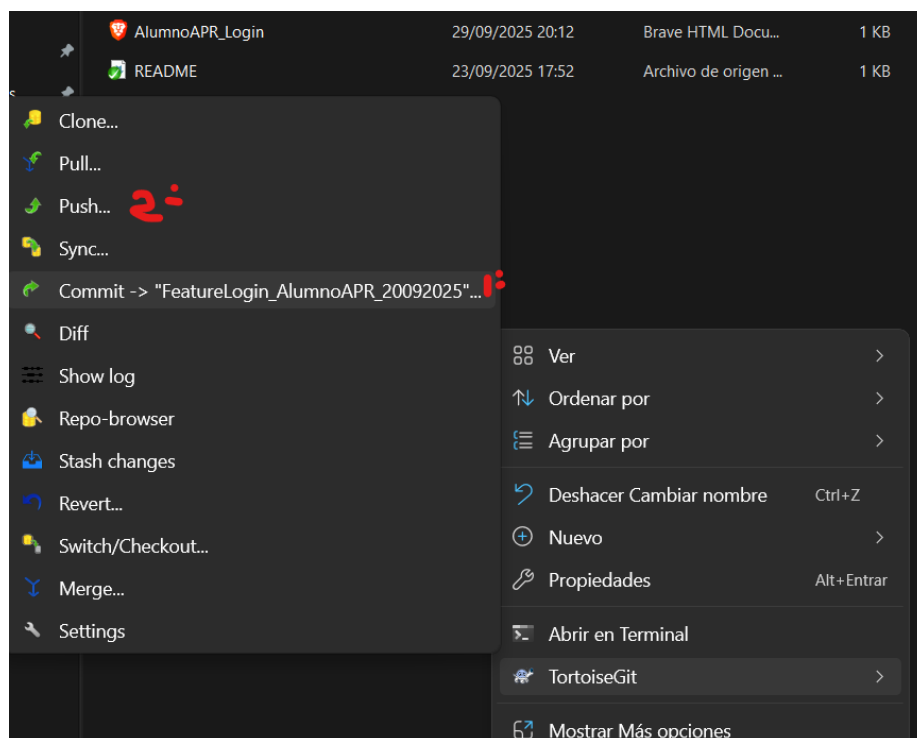


Luego hacemos un push hacia el repositorio remoto



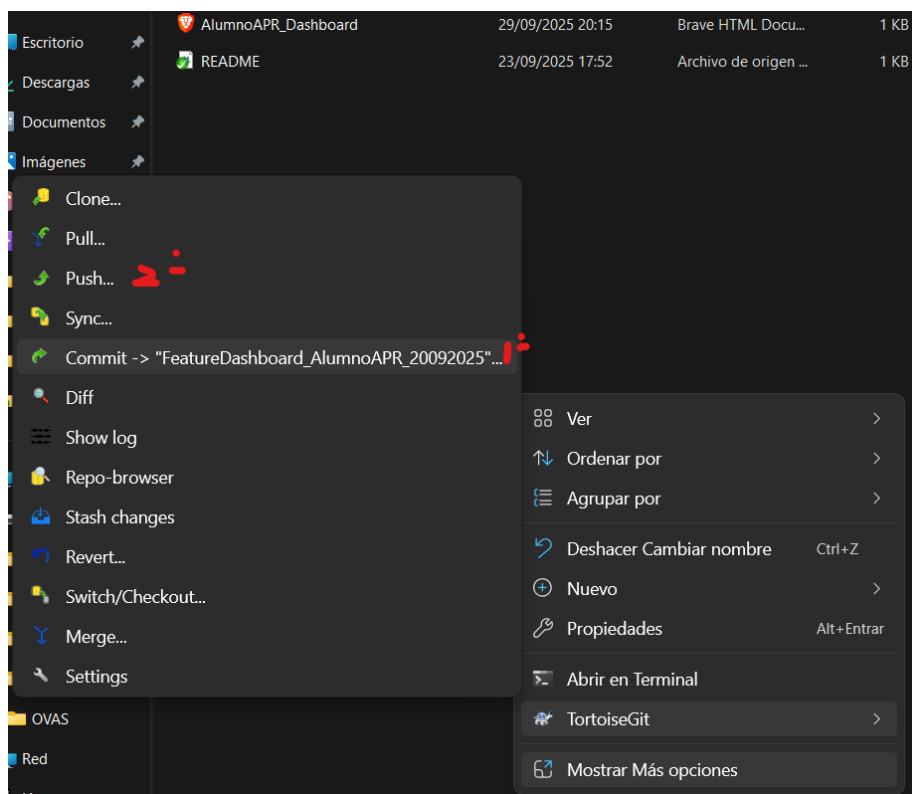
Hacemos lo mismo en la rama Fixes pero creando el archivo AlumnoAPR\_Fix

	 CIUDAD DE AGÜIMES		 Gobierno de Canarias
---	--	--	--






Hacemos lo mismo en la rama FeatureLogin con el archivo AlumnoAPR\_Login

			
---	---	--	---

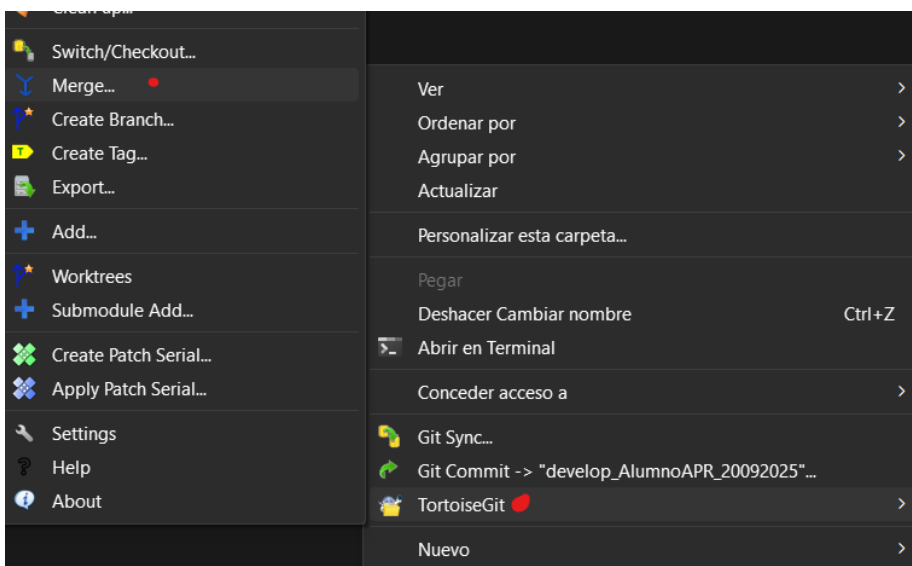


Hacemos lo mismo en la rama Dashboard con el archivo AlumnoAPR\_Dashboard

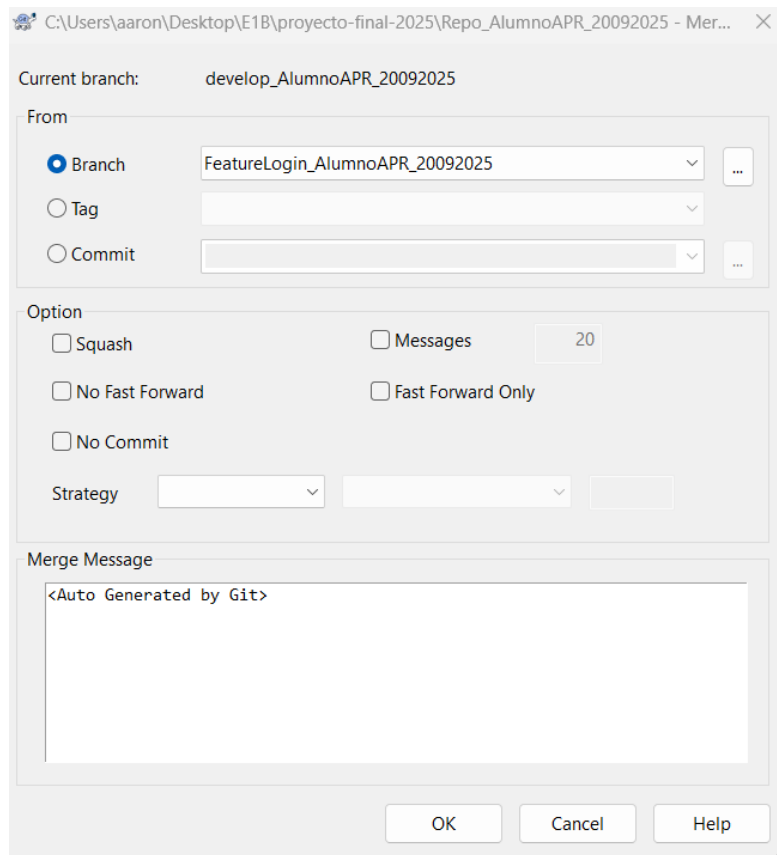
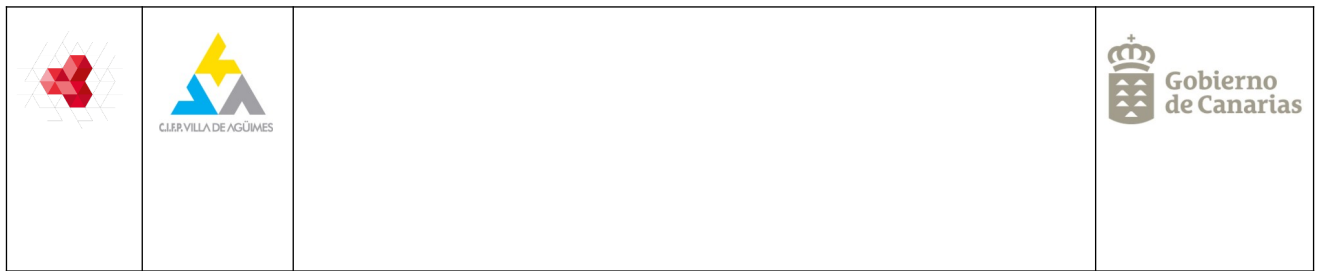
			
---	---	--	---

## 5 Integración de ramas

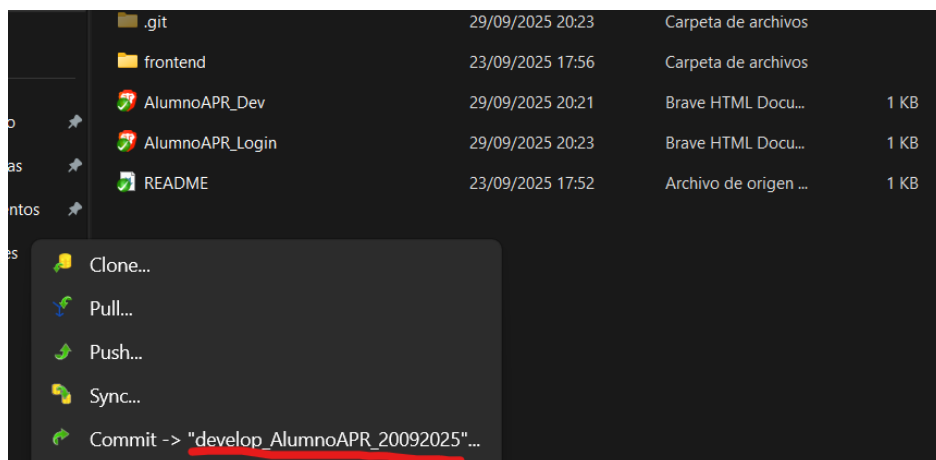
- Haz un **merge** de `FeatureLogin_AlumnoXXX_20092025` en `develop_AlumnoXXX_20092025`.
- Realiza un **rebase** de `FeatureDashboard_AlumnoXXX_20092025` sobre `develop_AlumnoXXX_20092025`.
- Finalmente, fusiona `develop_AlumnoXXX_20092025` con `main`.



Hacemos un switch a la rama develop y le damos a realizar un merge

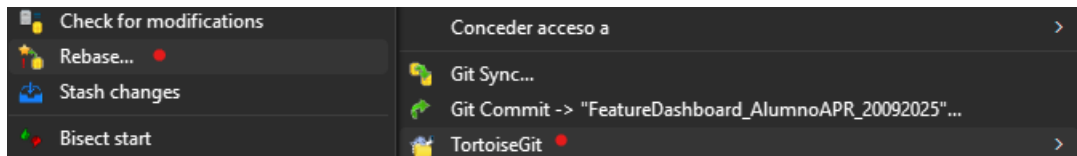
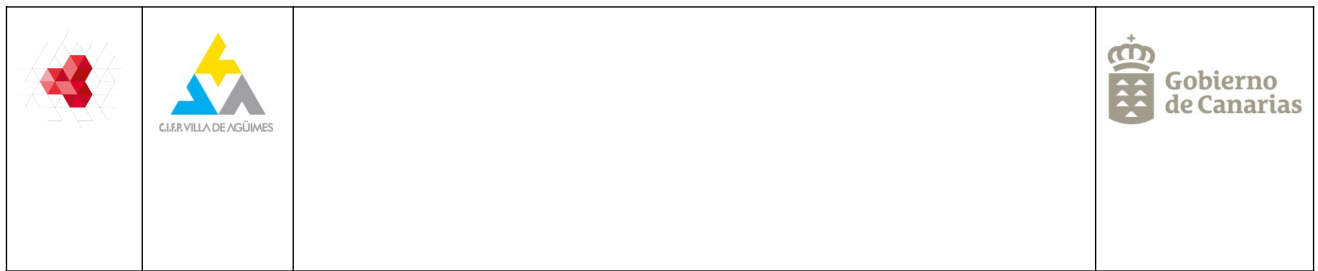


En “Branch” seleccionamos la rama de FeatureLogin y le damos a OK

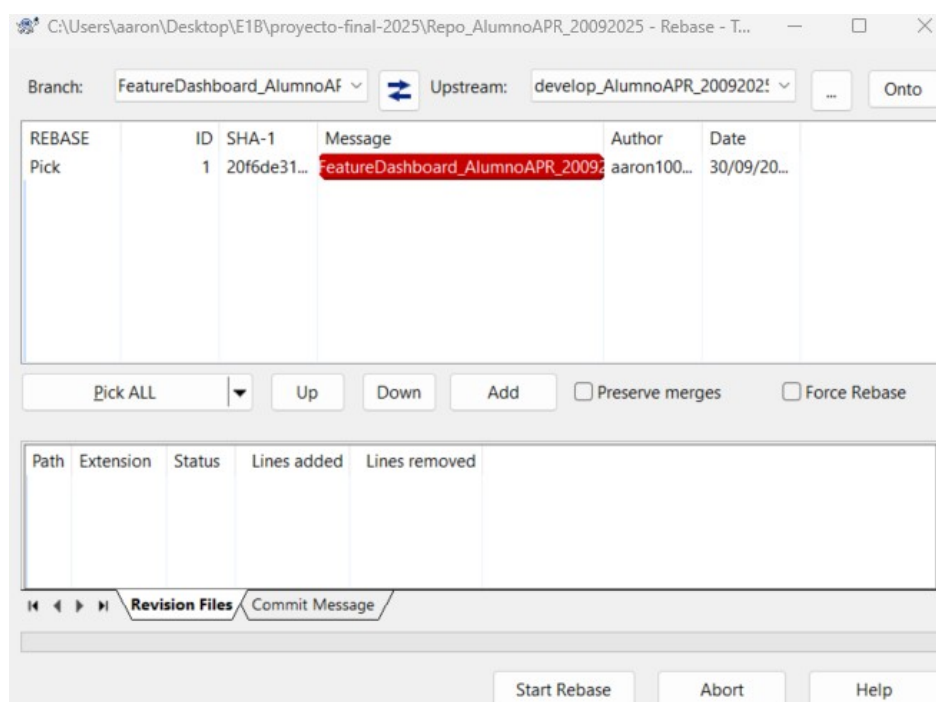


Comprobamos que se ha realizado correctamente

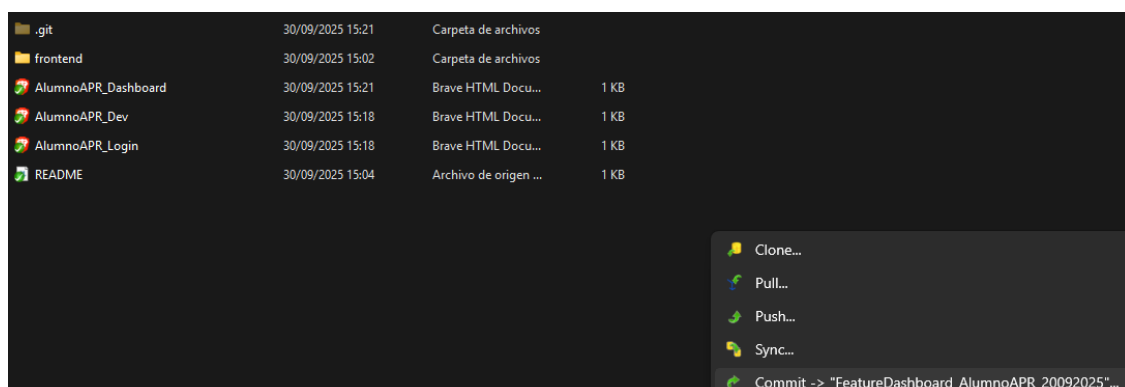




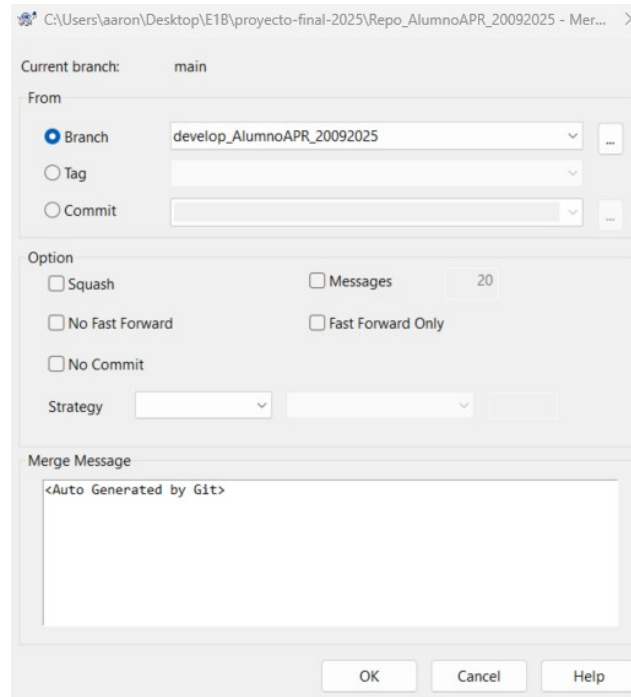
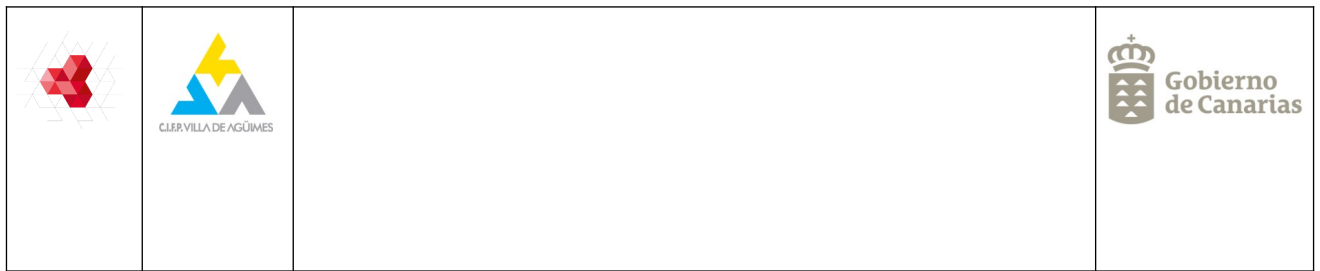
Desde la rama FeatureDashboard seleccionamos la opcion de hacer un rebase



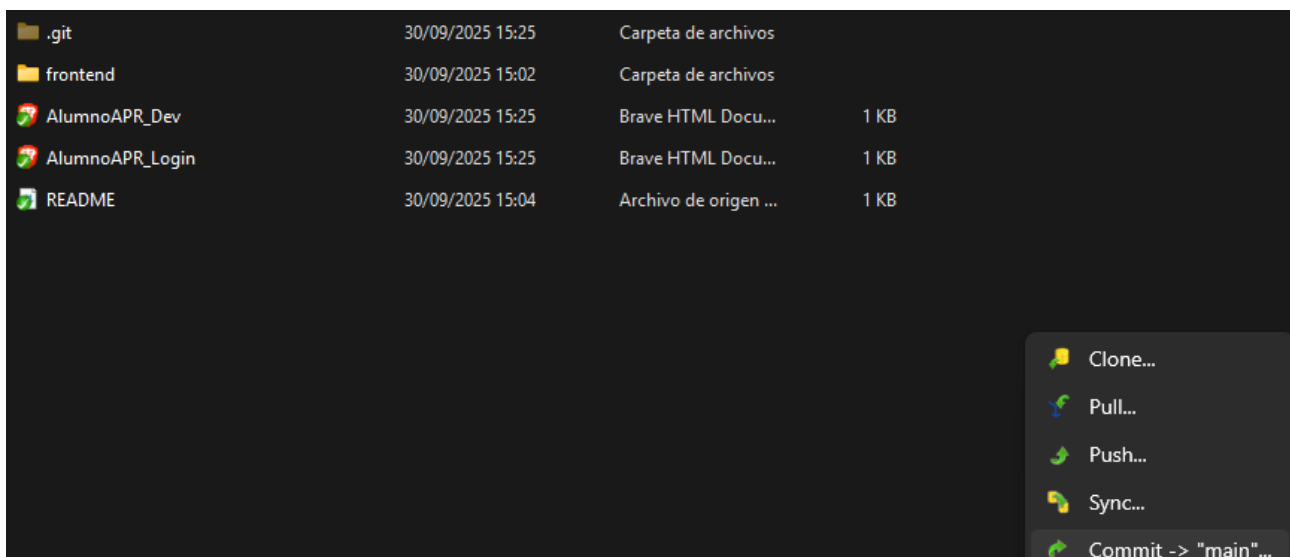
En Branch seleccionamos la rama de FeatureDashboard y en Upstream seleccionamos la de develop y le damos a Start Rebase.



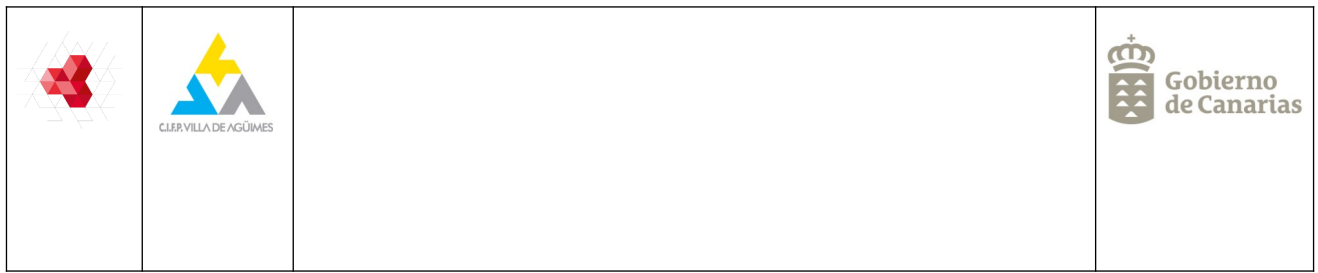
Vemos que el rebase se ha completado de forma correcta en FeatureDashboard.



Por último hacemos un merge de develop en main para fusionarlos.



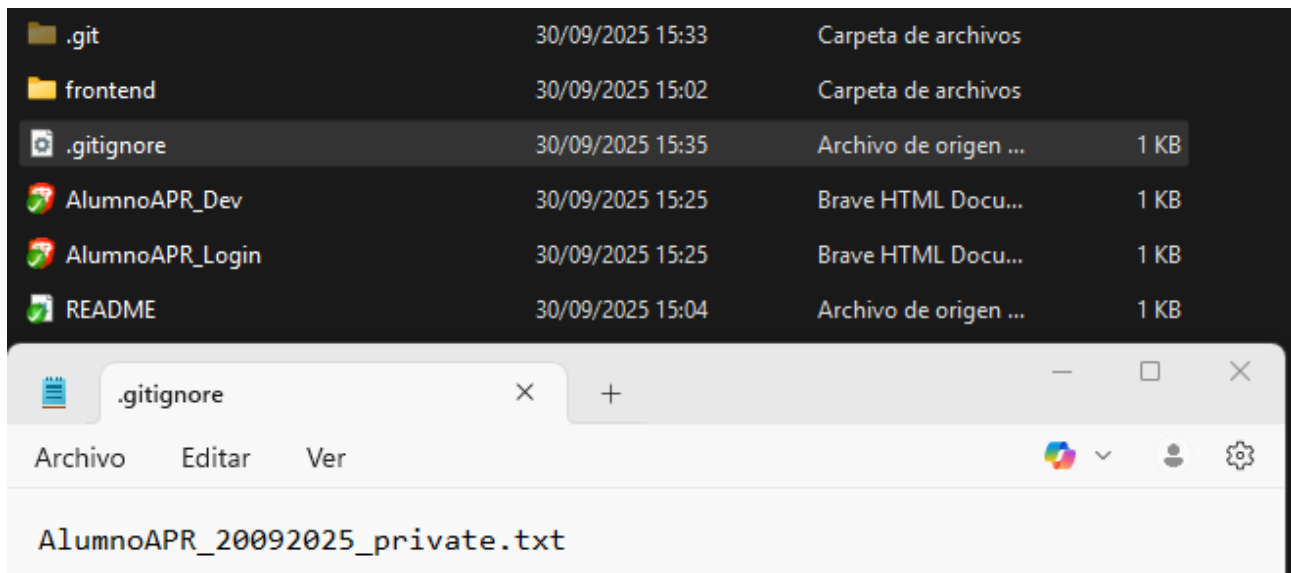
Vemos que se han fusionado correctamente.






## 6 Ignorar fichero

- Crea un archivo `.gitignore` y agrega:

```
AlumnoXXX_20092025_private.txt
```



Creamos el archivo `.gitignore` agregando `AlumnoAPR_20092025_private.txt`

			
---	---	--	---

## 7 Presentar Informe

- Documenta paso a paso el proceso, incluyendo todos los comandos Git utilizados.
- Explica cómo resolviste conflictos si surgieron.
- Describe la estructura final del repositorio y cómo se integran los cambios en `main`.

La estructura final del repositorio termina siendo con 4 ramas, dos desde main y dos desde la develop.

La rama main se queda con los archivos de `.gitignore`, `AlumnoAPR_Dev`, `AlumnoAPR_Login` y la carpeta frontend con el archivo `AlumnoAPR_20092025_UI`.

La rama develop se queda con los mismos archivo que main, pero sin este último que creamos (`.gitignore`).

La rama FeatureLogin se queda con la carpeta frontend con el archivo `AlumnoAPR_20092025_UI` y el archivo `AlumnoAPR_Login`.

La rama FeatureDashboard se queda con la carpeta frontend con el archivo `AlumnoAPR_20092025_UI`, el archivo `AlumnoAPR_Dashboard`, `AlumnoAPR_Dev` y `AlumnoAPR_Login`.

Por último la rama Fixes se queda con la carpeta frontend con el archivo `AlumnoAPR_20092025_UI` y el archivo `AlumnoAPR_Fix`.