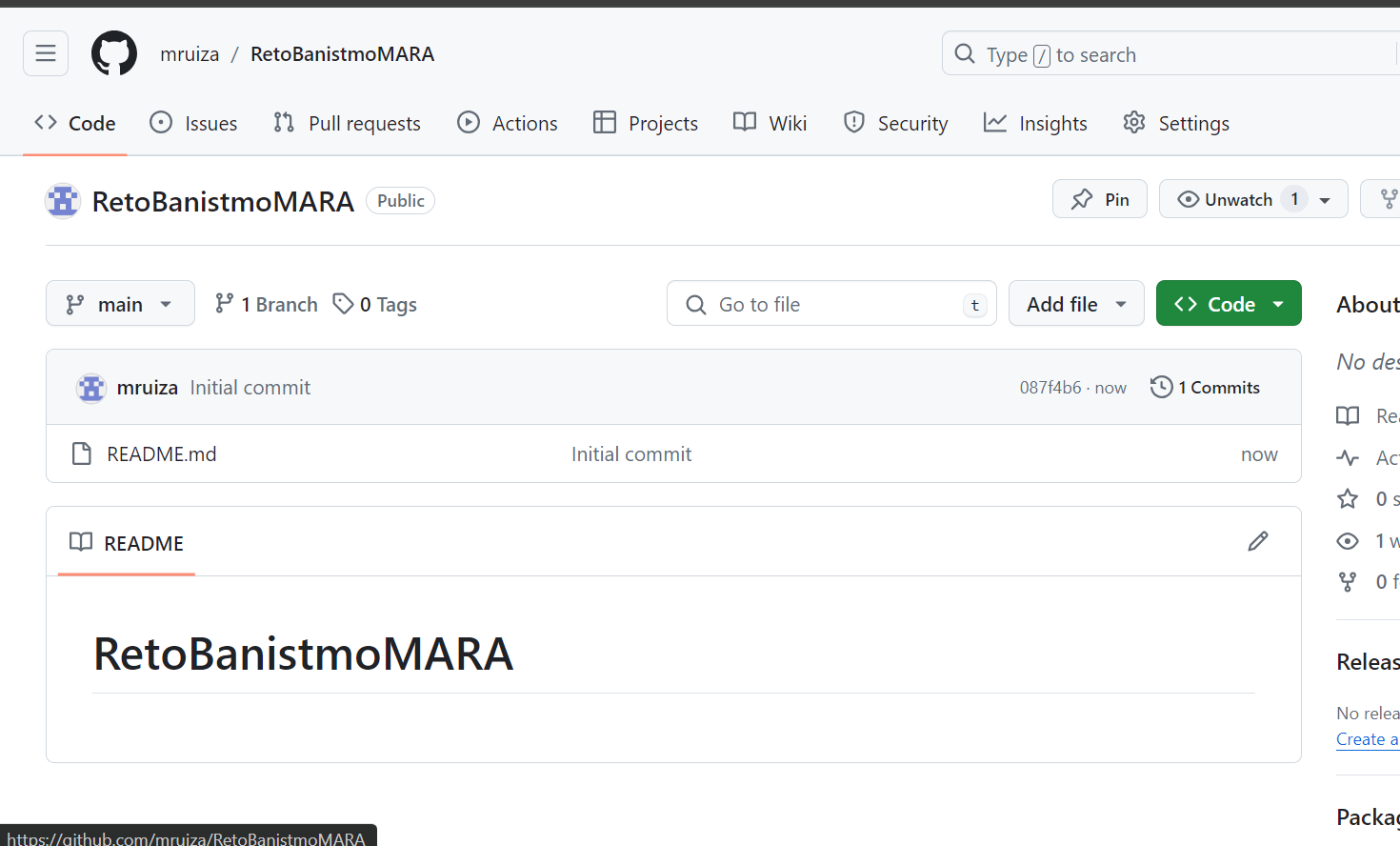
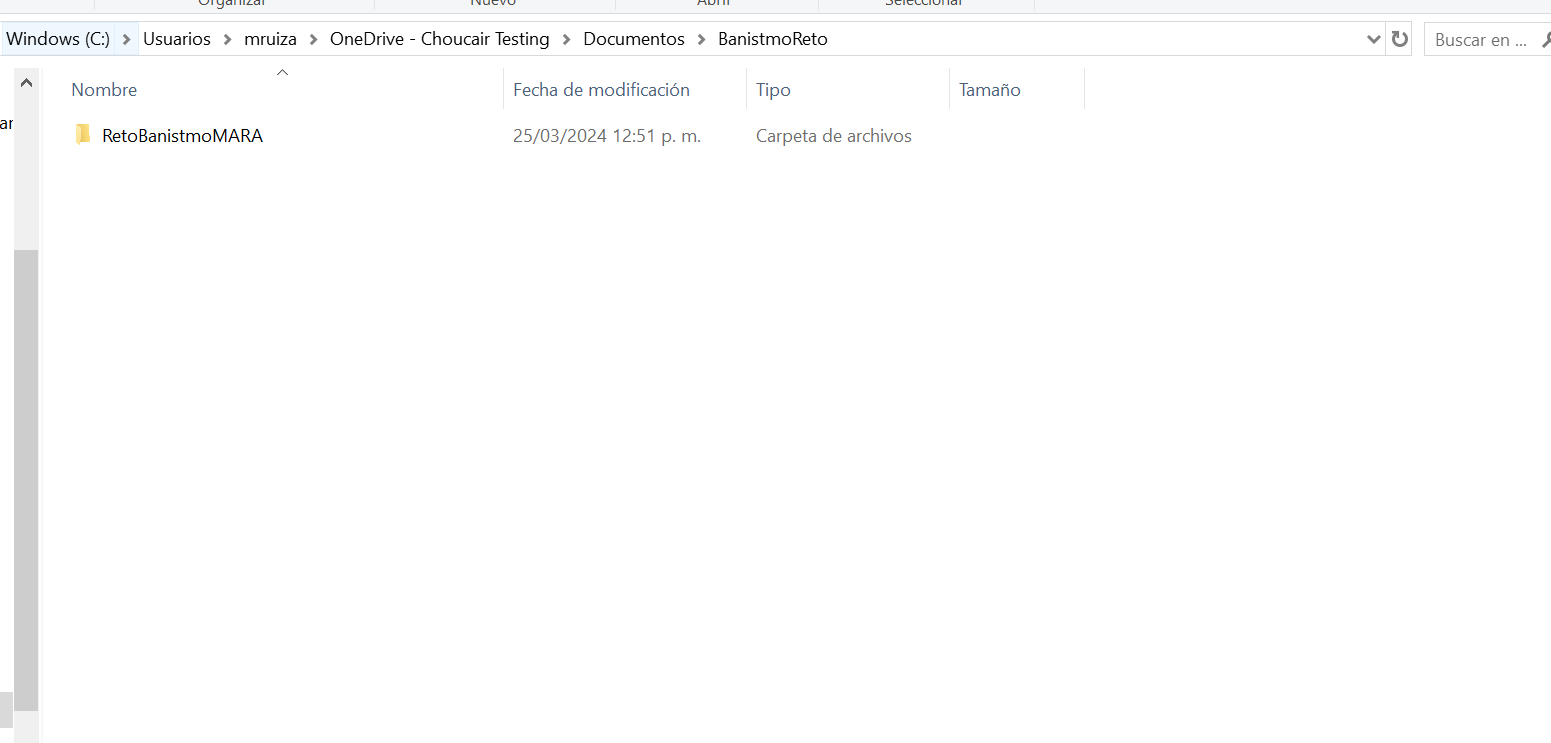
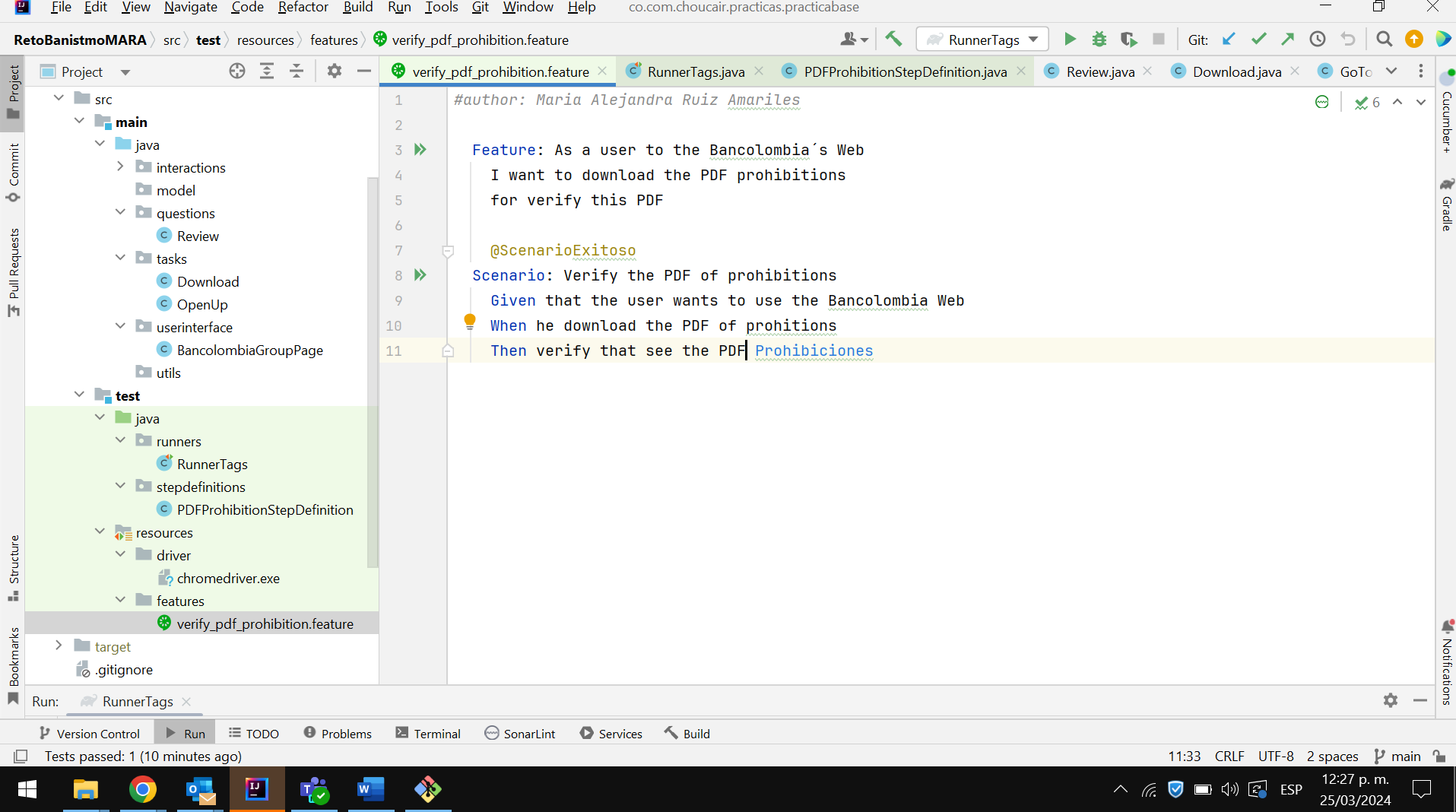
Reto Banistmo Maria Alejandra Ruiz Amariles

URL del repositorio: <https://github.com/mruiza2023/RetoBanistmoMARA.git>

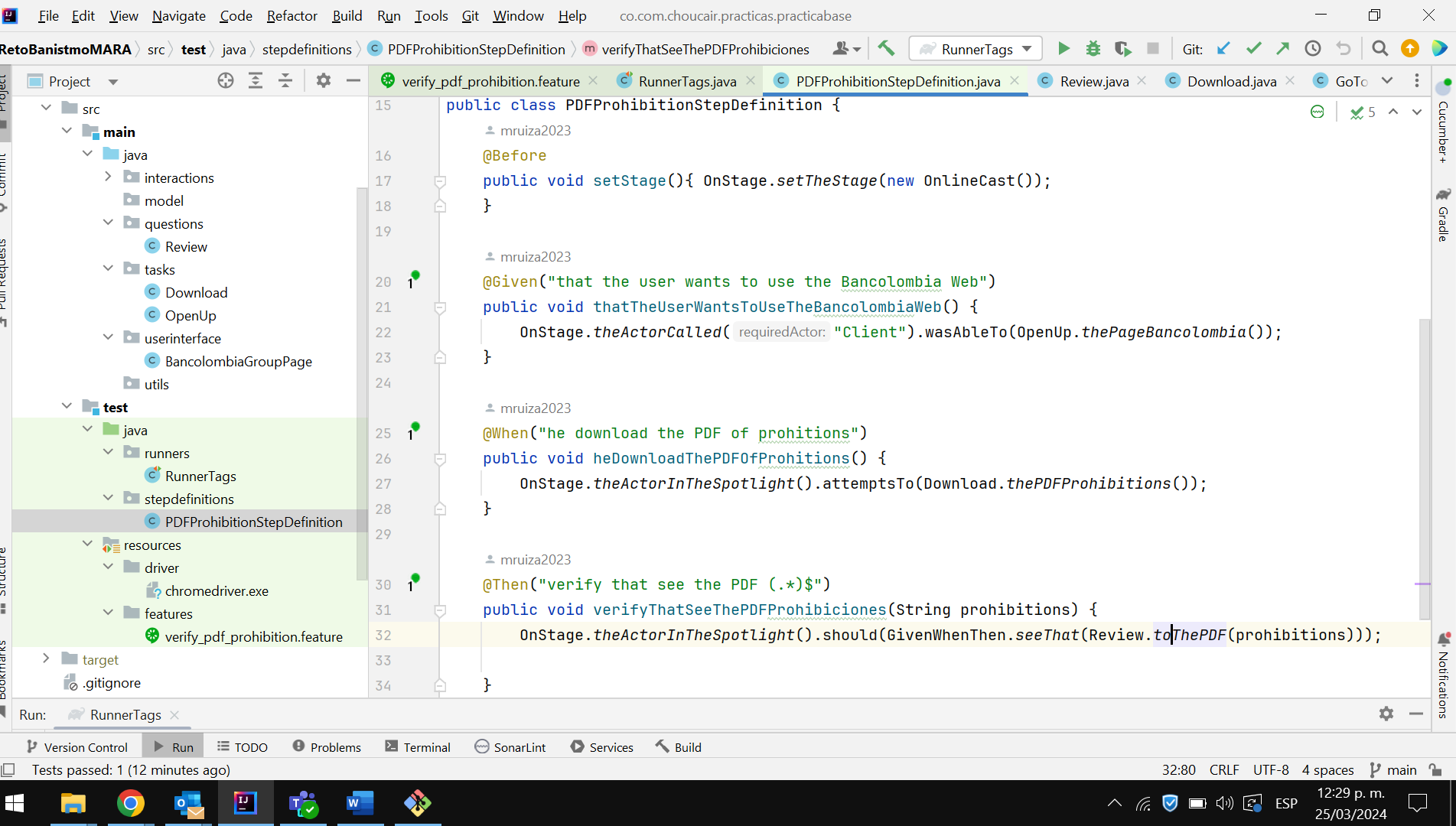
1. Inicié por crear un repositorio remoto en GIThub, en donde se subirán los cambios que se realicen al proyecto.
2. Clone el repositorio creado en mi equipo local con el comando **git clone “url del repositorio”** en este caso <https://github.com/mruiza/RetoBanistmoMARA.git>



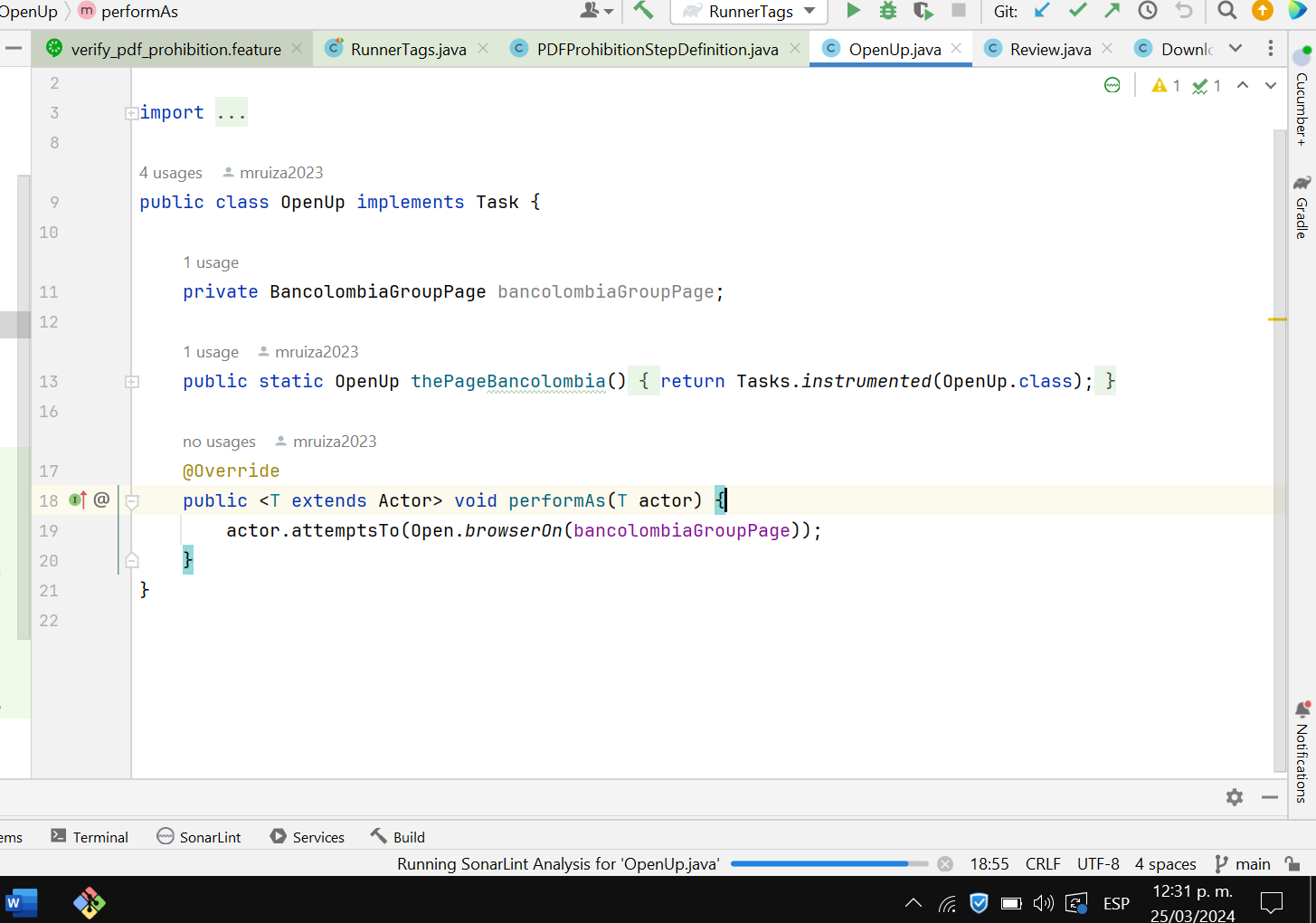
1. Utilizando Cucumber y Gherkin, creé mi feature “verify\_pdf\_prohibitions.feature” con un escenario exitoso, la cual se encuentra en la carpeta de Features, se define el Given, When, Then , teniendo en cuenta el flujo que se muestra en el pdf del reto.



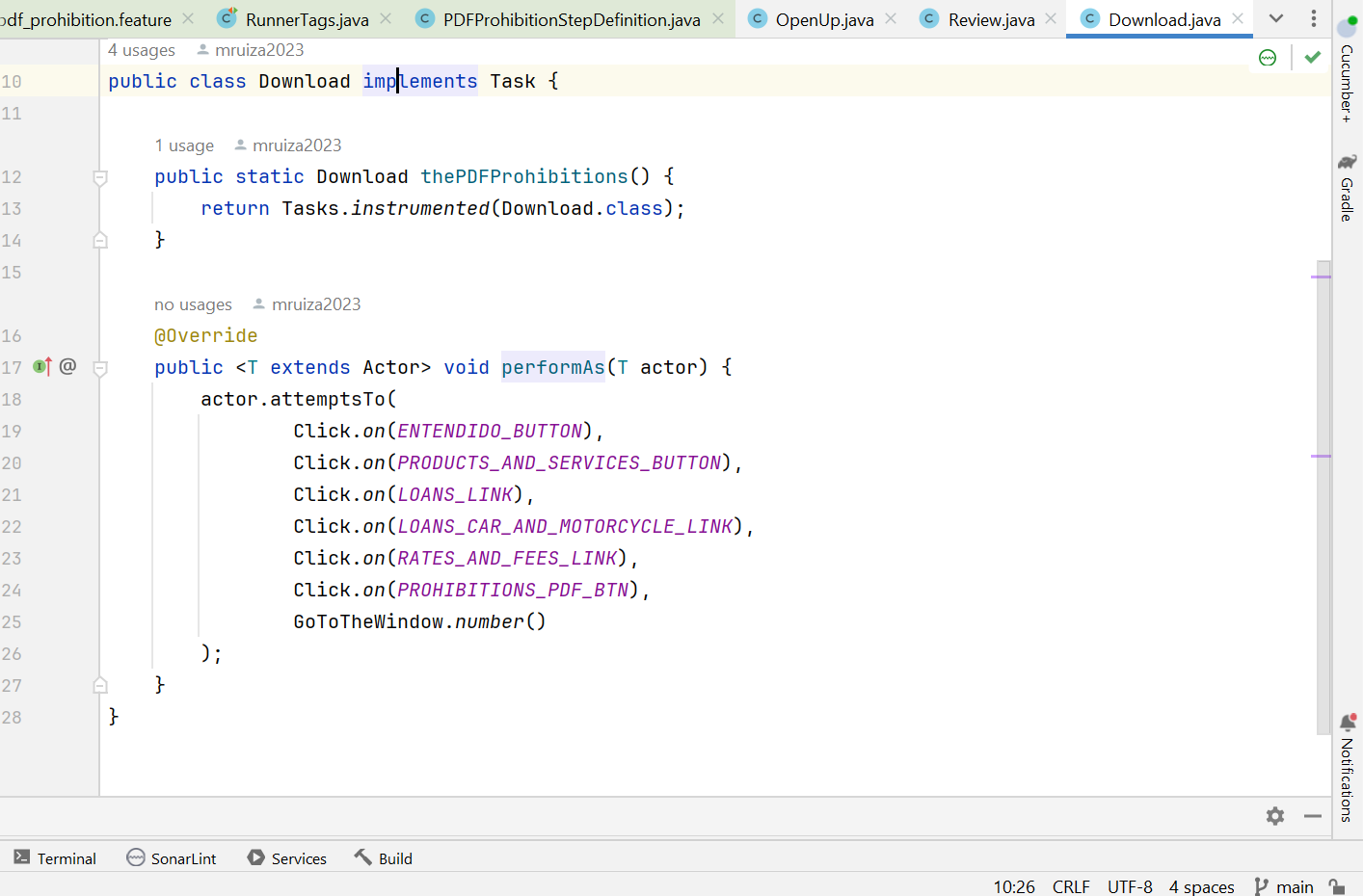
1. Se crea la clase StepDefinition, donde se realiza la programación de los métodos según la feature, se configura también el ambiente mediante la sentencia OnStage.



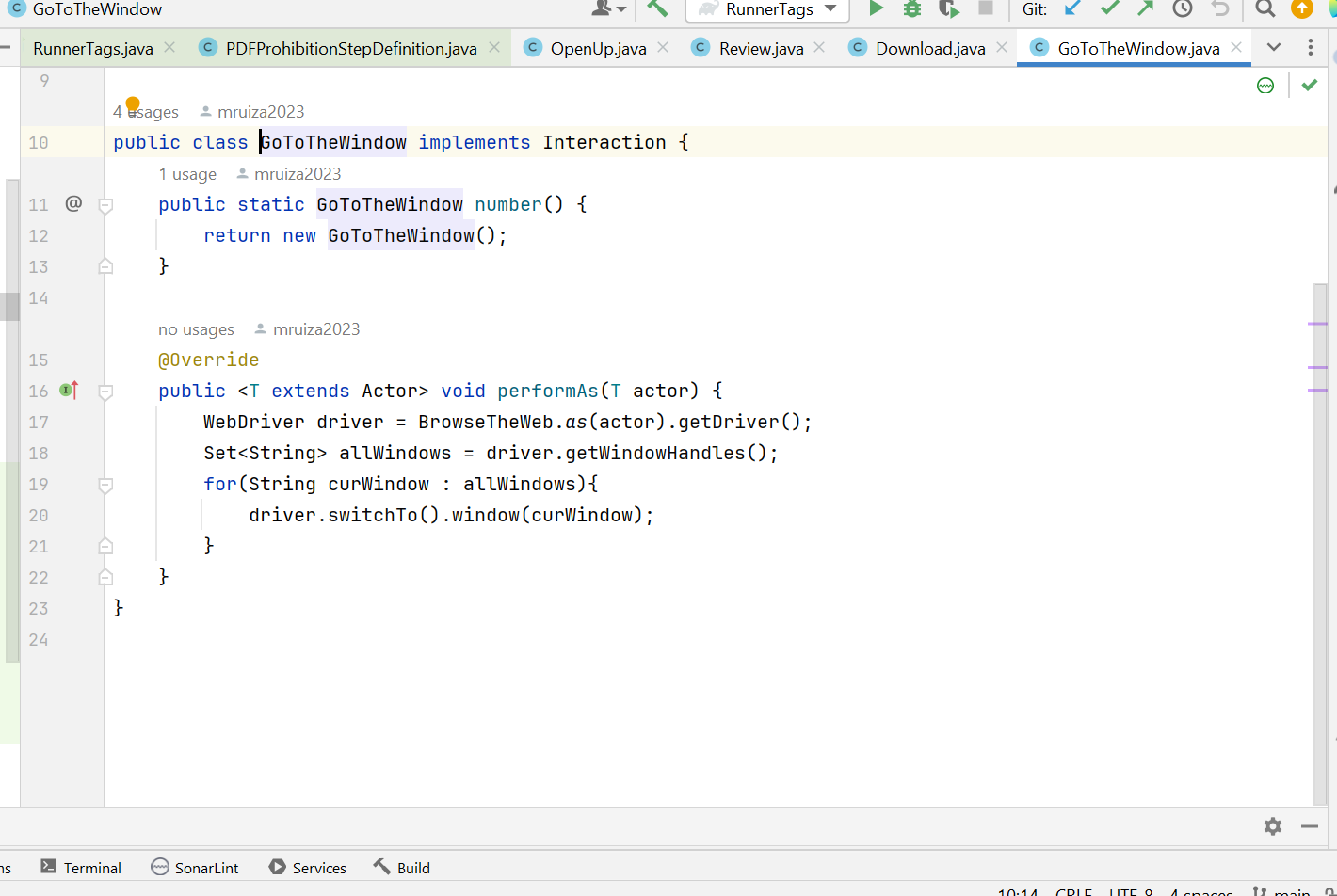
1. Para el método del Given, se crea una clase tipo Task (Porque es un conjunto de acciones que hará el actor) que me permitirá abrir el navegador y la página web de Banistmo.



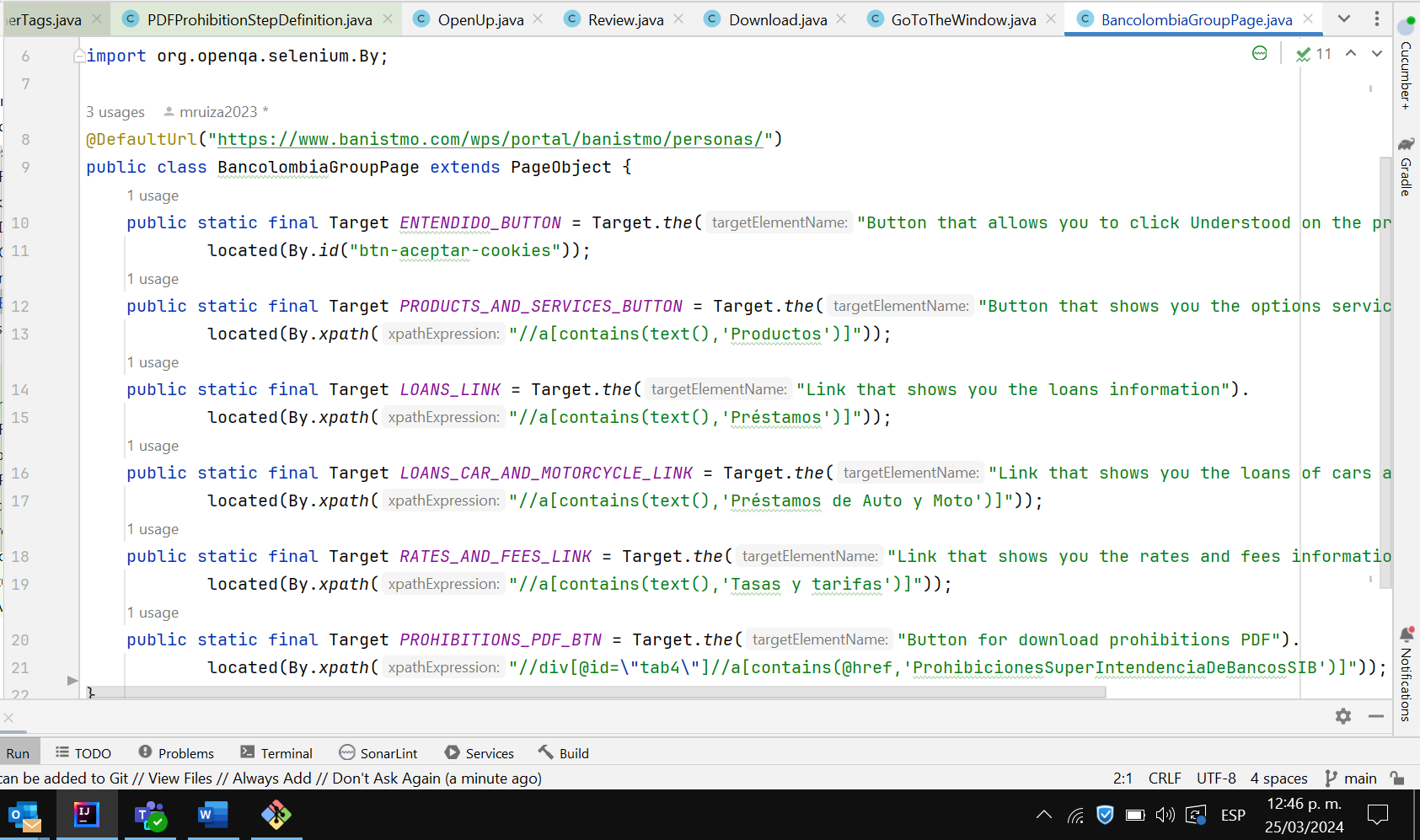
1. Para el método When se crea una clase tipo Task para crear los pasos que requiero para descargar el pdf.



Dentro de ella, se hace el llamado también de la interacción GoToTheWindow que me permite obtener la cantidad de pestañas abiertas y cambiarme a otra pestaña mediante el método switchTo. (Tipo Interacción porque es una acción que hará el actor y que puede ser reutilizada en otras clases).

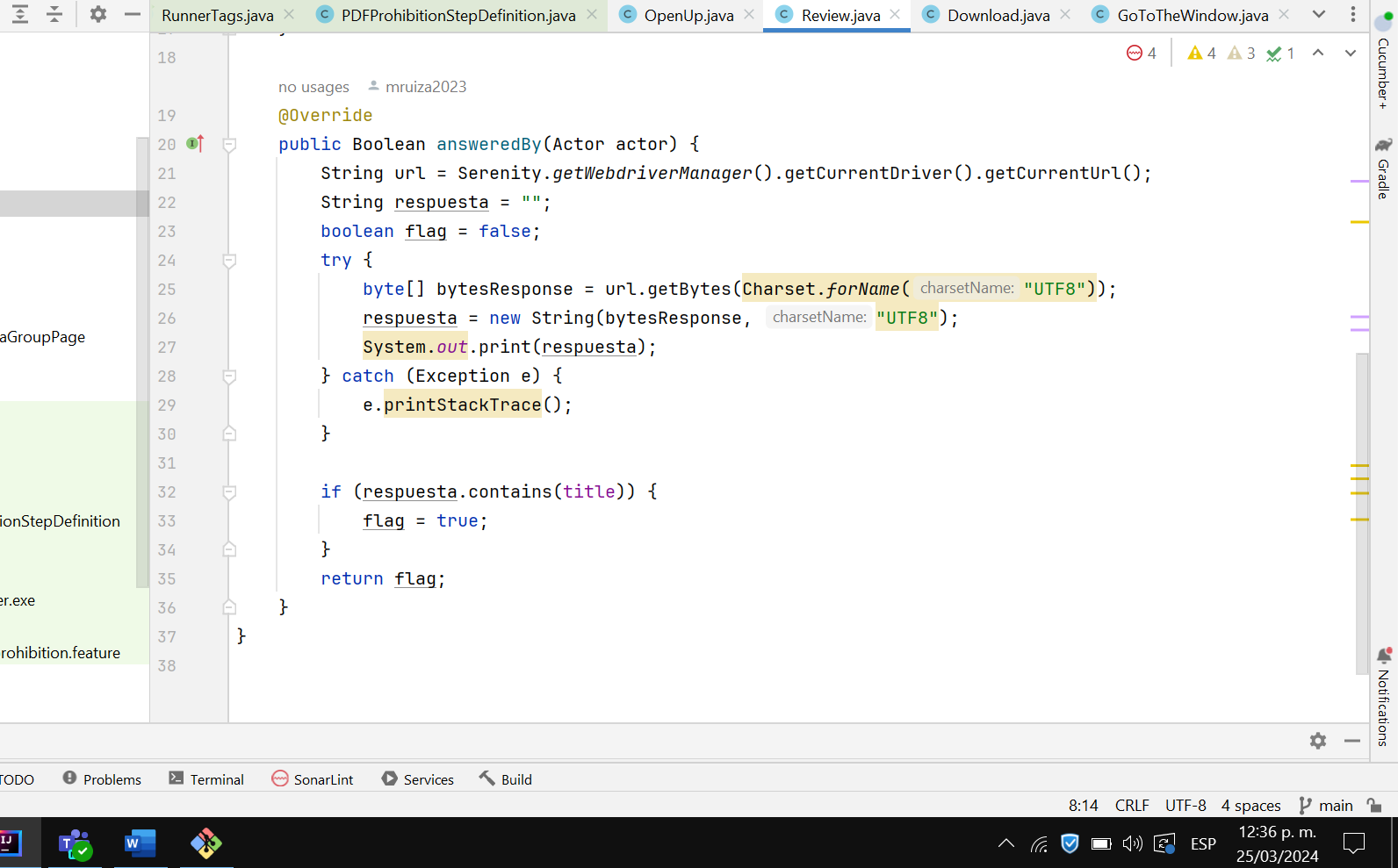


1. Los objetos utilizados en las clases anteriores se definieron en la clase tipo PageObject BancolombiaGroupPage.



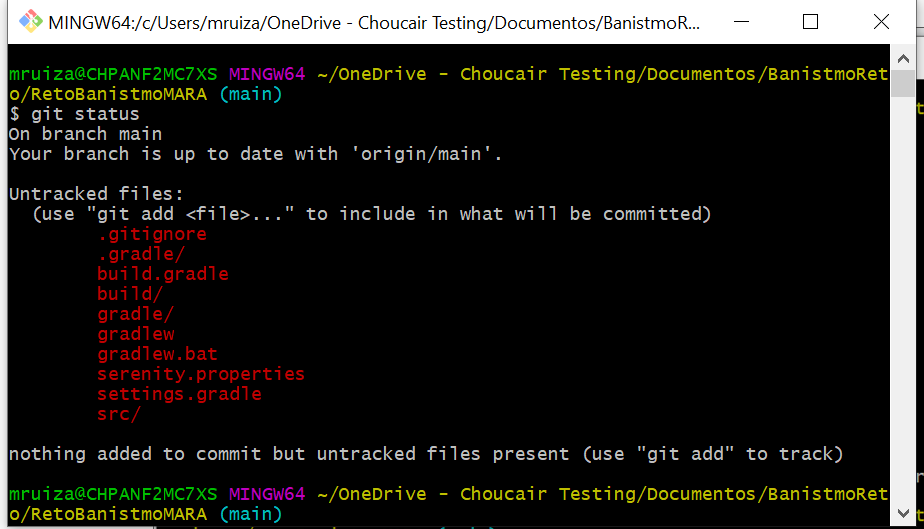
1. Para el ultimo paso “Then” se verifica que el PDF sea el correcto mediante la clase tipo Question. La verificación se realiza obteniendo la url y verificando que en ella se muestre el titulo del pdf solicitado.

Luego, mediante el if comparamos que lo que tenemos en la url sea igual al nombre enviado desde la feature.

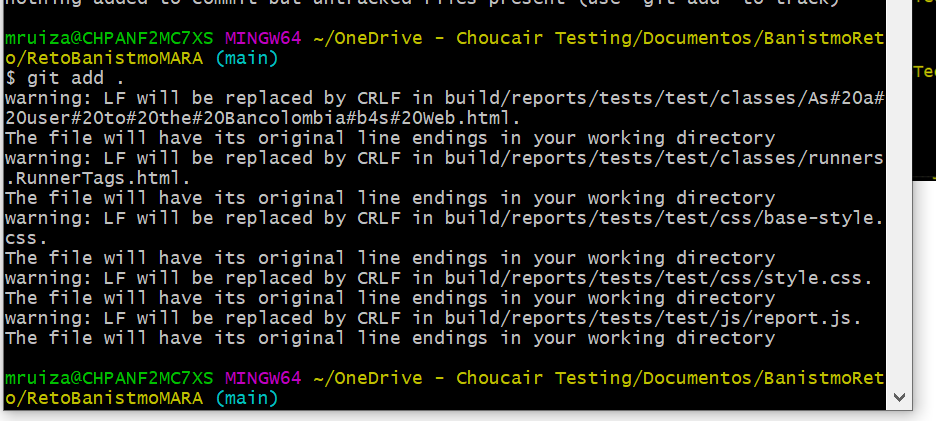


**Comandos GIT utilizados**

1. **Git clone:** se utilizó para clonar mi repositorio remoto, y trabajar desde el repositorio local.
2. **Git status:** Para verificar los archivos en los cuales se realizaron modificaciones.



1. **Git add .::** Para agregar los cambios realizados al stagging área de git.

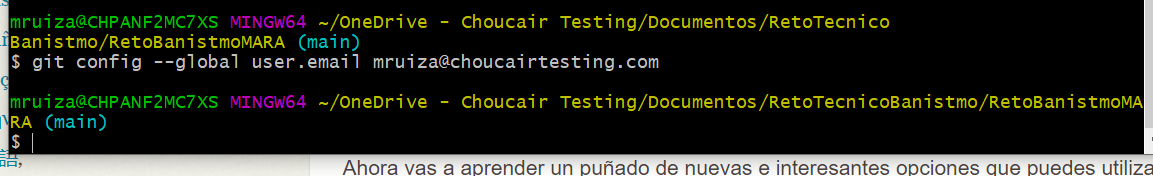


1. **Git commit -m “”:** Lo utilicé para confirmar los cambios que se realizaron en el stagging área

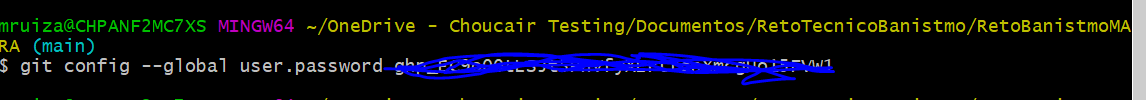
Texto

Descripción generada automáticamente

1. **Git config –global user.email:** Utilizado para configurar en el proyecto el email con las cuales accederé al repositorio remoto



1. **Git config –global user.password** Utilizado para configurar en el proyecto el password con las cuales accederé al repositorio remoto.



1. **Git push:** Para subir los cambios realizados al repositorio remoto

