**Requerimiento 1**

Se pide implementar el código de la actividad 1 mediante java y eclipse. Además, hay que tener en cuenta que hay que cada integrante del equipo tendrá que trabajar con GIT y todo el equipo tendrá un repositorio GIT común (Github).

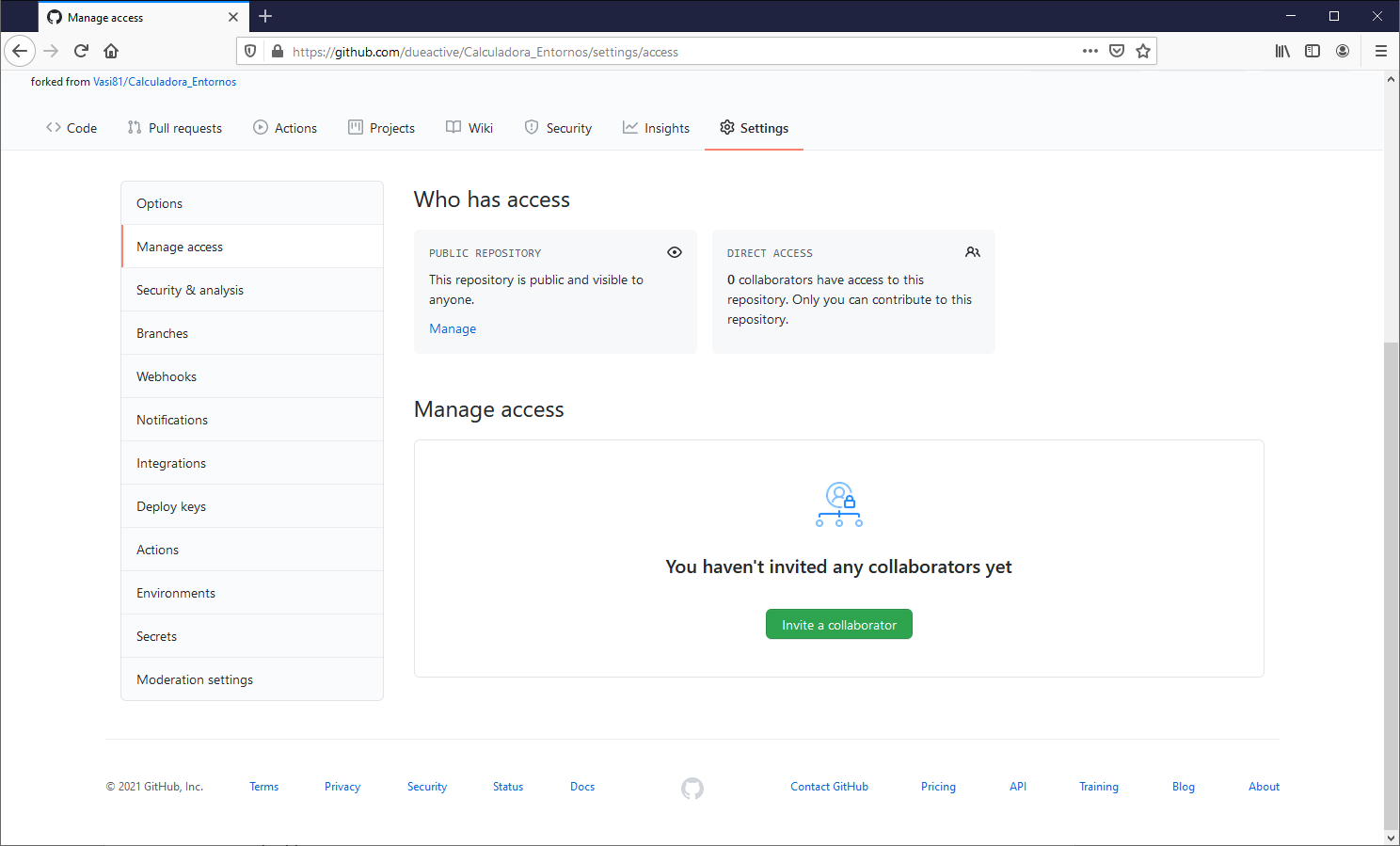
Cada integrante del equipo tendrá la labor de implementar una clase completa haciendo los commits necesarios a su repositorio local y sincronizarse con el resto del equipo a través del repositorio remoto.

\*\*Antes de nada, unificamos el nombre del paquete para nuestras clases.

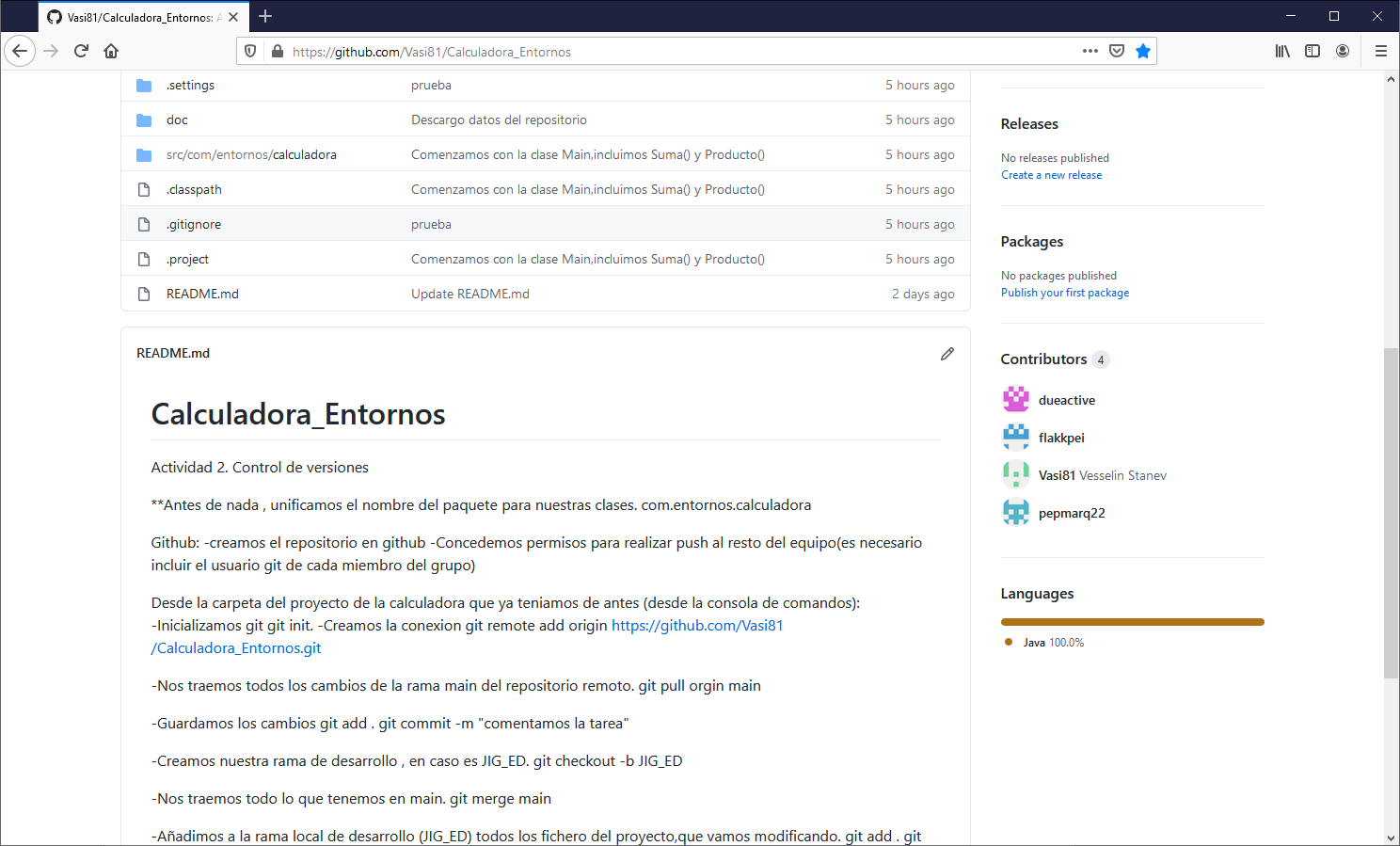
**com.entornos.calculadora**

-creamos el repositorio en Github (<https://github.com/Vasi81/Calculadora_Entornos.git>).

-Asignamos los integrantes al proyecto para que tengan permisos sobre ese repositorio, de lo contrario no podrán realizar acciones como PUSH.



Integrantes:



\*\*Importante, para evitar futuros problemas con el fichero. gitignore lo definimos de la siguiente manera:

/bin/

\*.project

\*.classpath

/.settings/

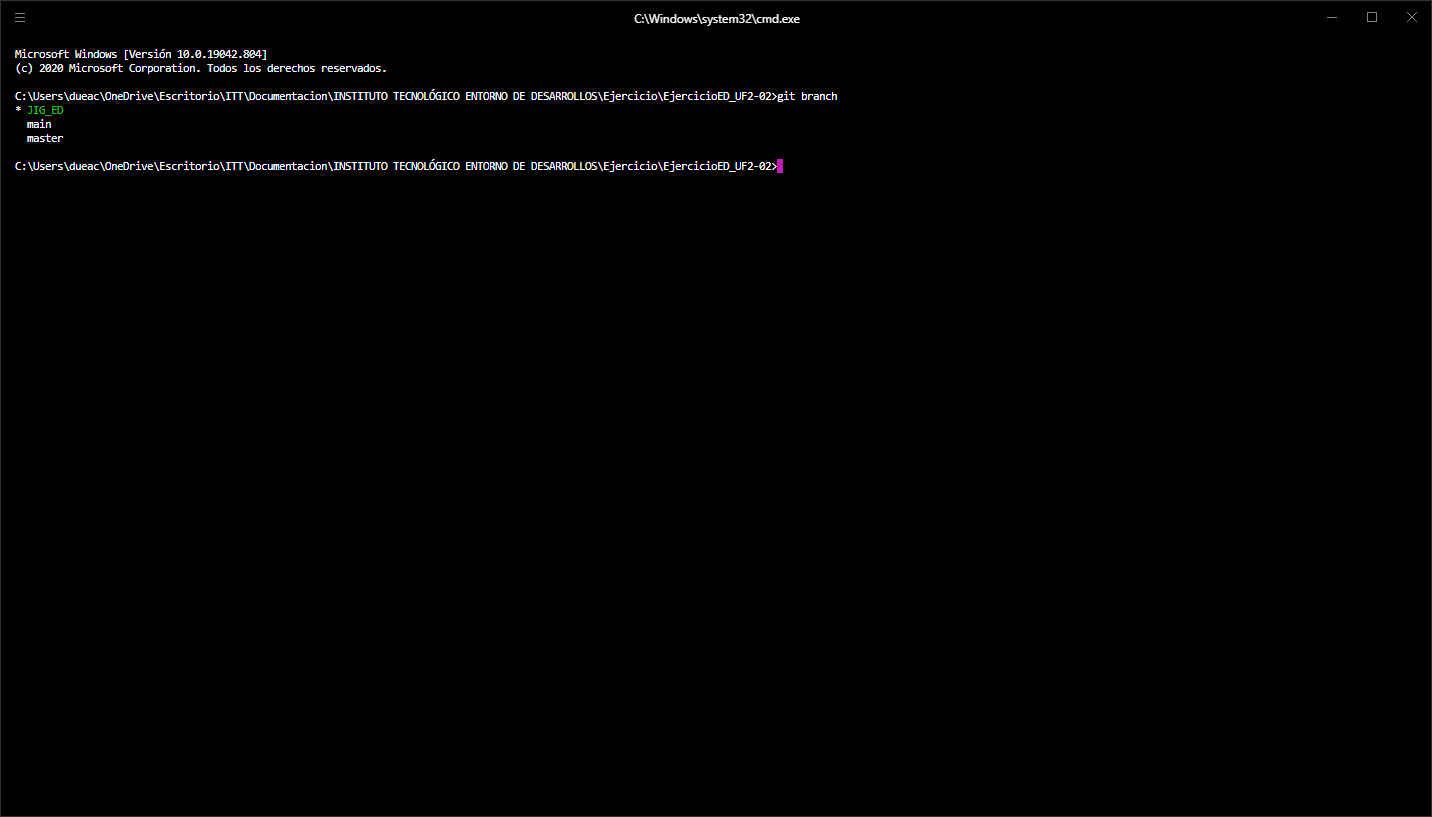
De esta manera solo subiremos al repositorio los ficheros .java y la carpeta doc , cuando generemos la nueva documentación en JavaDoc y no tendremos problemas con el resto de las configuraciones del proyecto de cada miembro del equipo.

Una vez que tenemos el repositorio remoto, comenzamos a nivel local cada integrante del equipo.

Desde la carpeta que contiene el proyecto de la Calculadora de cada miembro, desde la consola de comando escribimos:

**git init** (Inicializa el proyecto con git).

A continuación, verificamos las ramas activas que tenemos, **git branch**.



Como en el repositorio remoto tenemos la rama main (master), en caso de no tener una rama main/master la creamos en nuestro repositorio local :

 git checkout -b main (Como rama master).

git checkout -b Nombre Rama para desarrollo (Como rama de desarrollo, en esta rama realizaré todos los cambios en el desarrollo).

Después podemos conectarnos al repositorio remoto que usaremos los integrantes del equipo para ir subiendo los cambios (pull) o descargando los cambios realizados por otrp compañero (push) , Creamos la conexión:

**git remote add origin** [**https://github.com/Vasi81/Calculadora\_Entornos.git**](https://github.com/Vasi81/Calculadora_Entornos.git)

**git checkout -b JIG\_ED** (creamos una rama para desarrollo).

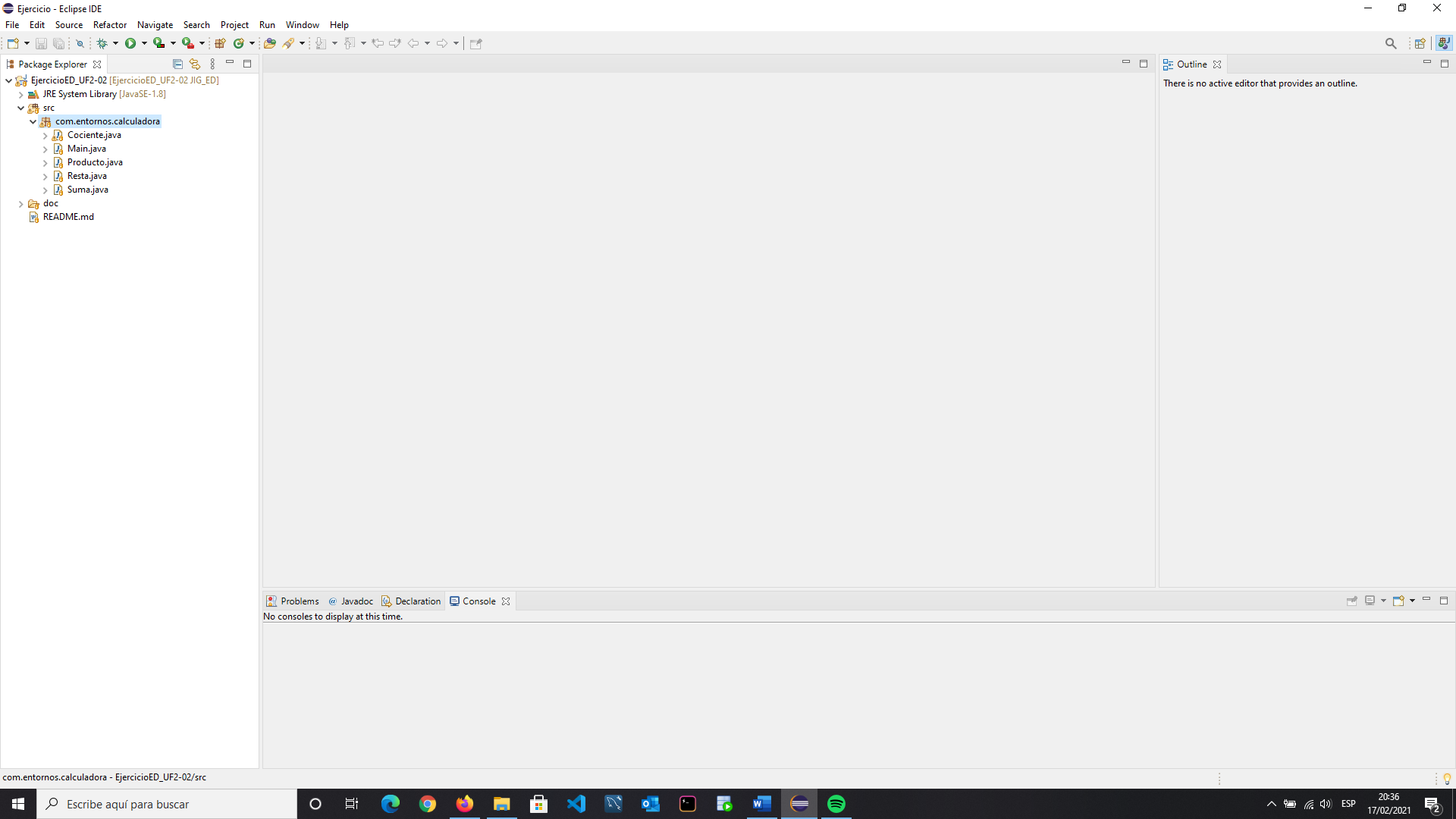
**git pull orgin main** : Descargamos el repositorio remoto en la rama de desarrollo para verificar si tenemos algún cambio por descargar.

A continuación, escribo el primer commit para comenzar con la implementación del código.

**git add .**

**git commit -m “incluimos los cambios que se vayan aplicando”.**

Desde Eclipse comenzamos con los cambios o la implementación del código y si nos fijamos en Eclipse nos indicará la rama en la que estamos trabajando.



**Requerimiento 2**

Se pide que cada integrante trabaje con una rama propia, pero cada vez que haga una subida al repositorio compartido fusione su rama con la rama máster para poder ir viendo los cambios del resto del equipo.

Comenzamos con los cambios que vamos realizando y según los realizamos los podemos ir incluyendo en git o podemos aglutinar varios cambios en un solo commit.

**git add .** (Para que aplique a todos los ficheros afectados o podemos indicar el fichero que queremos incluir).

**git commit -m “Comentario sobre el cambio”**

Esto lo tendríamos que realizar por cada cambio o conjunto de cambios que vayamos realizando.

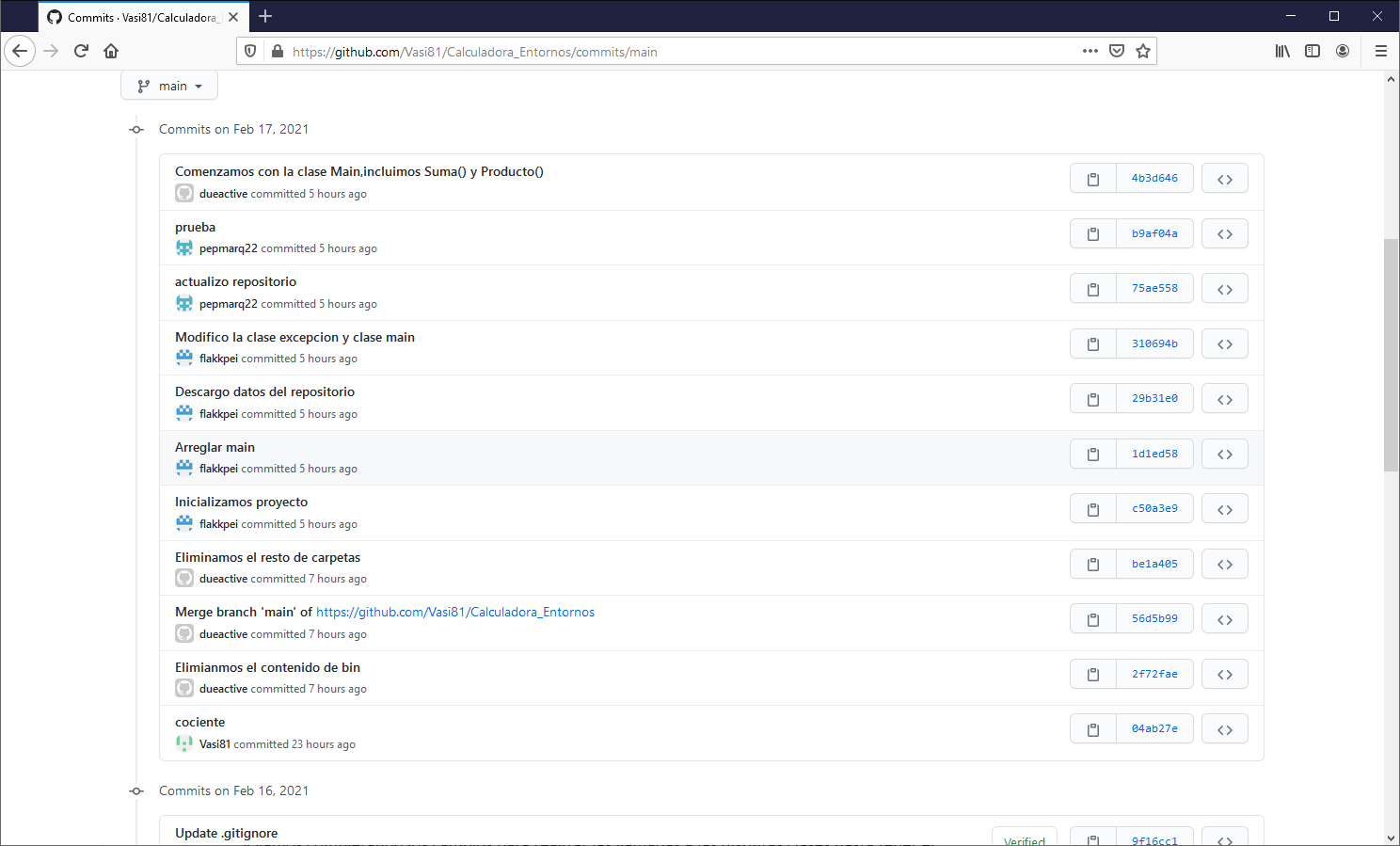
Como en la actividad anterior no quedamos satisfechos por como realizamos la documentación de JavaDoc , hemos repasado todo lo referente a la documentación con JavaDoc, por lo tanto esto también se refleja en los commits.

Una vez tengamos todos los cambios con sus respectivos commits , nos cambiamos de rama , a la rama main de nuestro repositorio local.

Desde esta rama realizamos git merge Rama de desarrollo, para fusionar lo que tenemos en la rama de desarrollo con la rama main.

Una vez que ambas ramas tienen el mismo contenido, podemos subir dichos cambios al repositorio remoto.

**git push origin main (Subimos los cambios al repositorio remoto).**



**Requerimiento 3**

Se pide una clase main que permita probar las clases de la calculadora. Para ello cada integrante tendrá que modificar el código que llama a los métodos de su clase en la clase main, es decir, todos los integrantes del equipo tendrán que trabajar sobre la misma clase main. Con que cada integrante pruebe uno de los métodos que ha realizado, es más que suficiente.

Nótese, que esta parte es probable que, de problemas de sincronización, por lo que se espera que los integrantes del equipo hablen y decidan como afrontar el problema.

Para este caso nos organizamos para subir una primera versión de la clase main y a continuación, cada uno de los miembros , de manera ordenada , nos descargamos dicha clase y vamos completando los cambios para realizar las llamadas a las distintas clases hasta tener el main con todas las clases implementadas.

De esta manera tendremos la clase main para todas las clases.

Con la documentación operamos de la misma forma, una vez tengamos todas las clases en el repositorio podemos generar la docuemntacionde todas las clases.

Una vez que tengamos la documentación (doc) subimos los cambios:

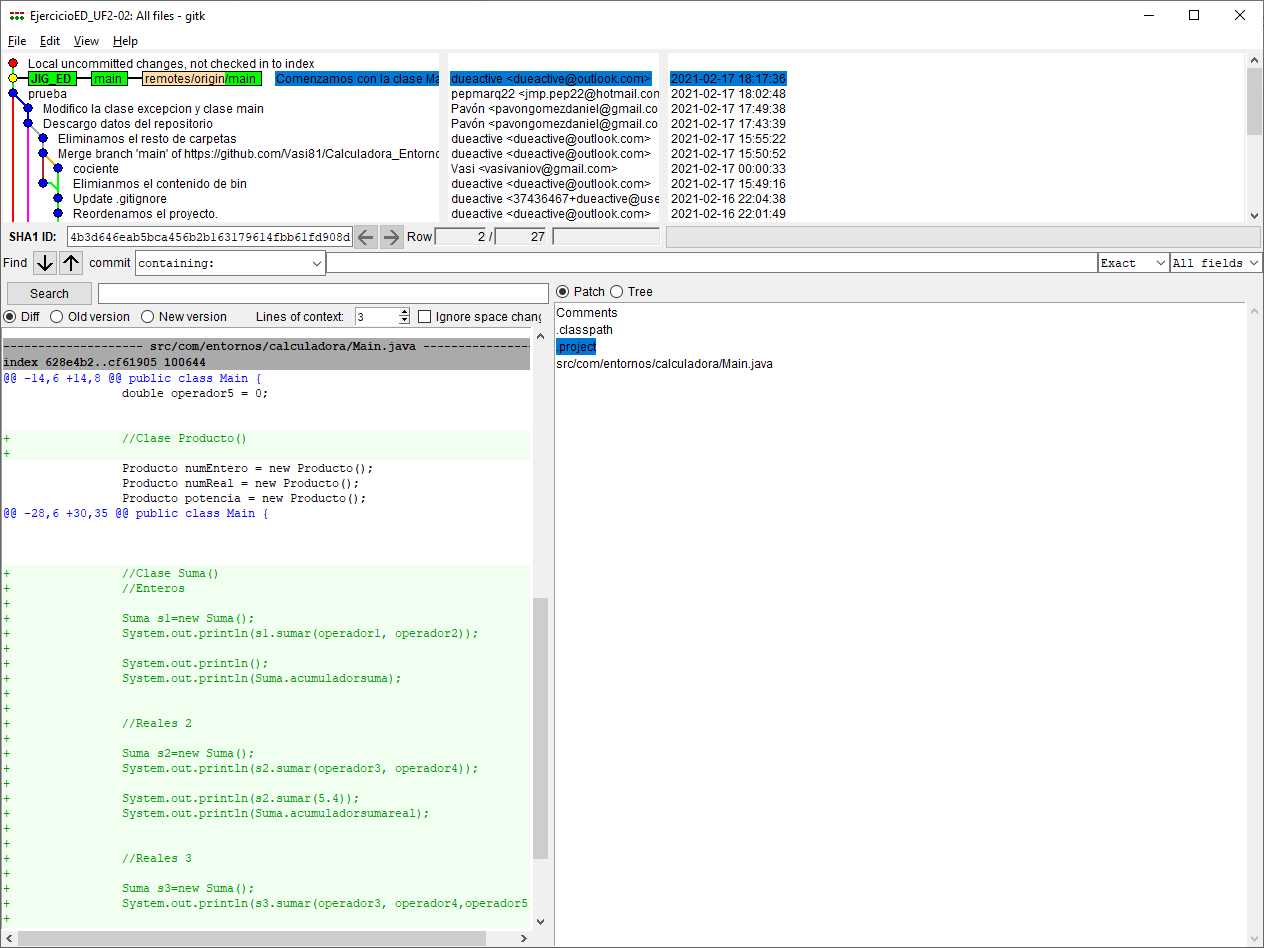
git add .

git commit -m “Generamos la documentación JavaDoc”.

git checkout main

git merge JIG\_ED

git push origin main.



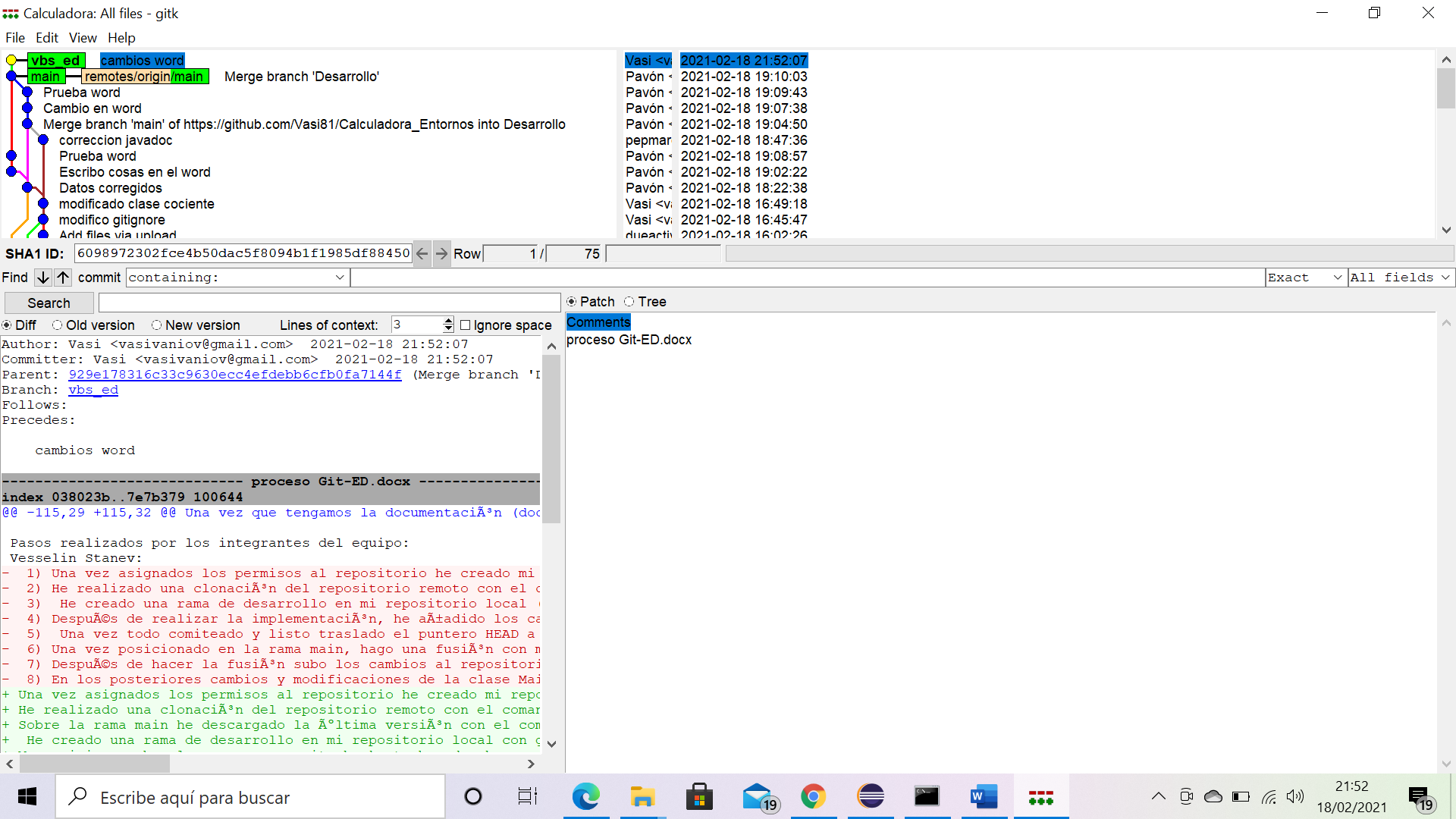
Gitk

Pasos realizados por los integrantes del equipo:

Vesselin Stanev:

1. Una vez asignados los permisos al repositorio he creado mi repositorio local con **git init** y la conexión con el repositorio remoto **git remote add** <https://github.com/Vasi81/Calculadora_Entornos>)
2. He realizado una clonación del repositorio remoto con el comando **git clone** <https://github.com/Vasi81/Calculadora_Entornos>)
3. Sobre la rama main he descargado la última versión con el comando **git pull origin main**
4. He creado una rama de desarrollo en mi repositorio local con **git checkout -b vbs\_ed** sobre la que voy a trabajar para implementar la clase Cociente.
5. Me posiciono sobre la nueva rama: **git checkout vbs\_ed** y hago una fusión con la rama main para traerme la última versión que he descargado **git merge main**
6. Después de realizar la implementación sobre la rama vbs\_ed, he añadido los cambios al stage **git add .** y he realizado los commits oportunos **git commit -m”……”**
7. Una vez todo comiteado y listo traslado el puntero HEAD a mi rama principal: **git checkout main.**
8. Una vez posicionado en la rama main, hago una fusión con mi rama de desarrollo: **git merge vbs\_ed**
9. Después de hacer la fusión subo los cambios al repositorio remoto: **git push origin main**
10. En los posteriores cambios y modificaciones de la clase Main, he seguido prácticamente los mismos pasos.

gitk



- Daniel Pavón Gómez

1. Lo primero que he hecho ha sido crear un proyecto vacío donde voy a alojar el workbench de git.
2. Después he hecho un **“git init”** para iniciar git en esa carpeta.
3. Después he creado una conexión con el repositorio de github creado por un compañero. “**Git remote add origin** <https://github.com/Vasi81/Calculadora_Entornos>”
4. Una vez conectado al repositorio de github lo que voy a hacer va a ser crear la rama main. **“Git checkout -b main”**.
5. Desde esta rama voy a bajarme todos los datos del repositorio con **“Git pull origin main”**
6. Ahora me va a dar un error debido a que el proyecto creado ya tiene las clases .classpath y .project y dará conflicto. Para poder bajarlo bien lo que haré sera un **“Git add .”** y un **“Git commit -m “Quitamos datos clonados”**.
7. Ahora vuelvo a hacer el “Git pull origin main” pero ahora me deberá hacer un merge por lo que lo haremos de la siguiente forma. **“git pull origin master –allow-unrelated-histories”**
8. Ya tenemos el repositorio bajado en nuestra rama main, ahora crearemos una rama nueva donde desarrollaremos nuestro proyecto. **“Git checkout -b Desarrollo”**
9. Ahora tengo que trasladar el repositorio que está en la rama main a nuestra rama para eso haremos un merge hacia aquí. **“Git merge main”**.
10. Ahora que ya lo tengo voy a programar mi parte de la calculadora.
11. Una vez ya la tengo programada voy a hacer un **“Git add .”** Para pasar todos los datos al stage y he realizado los commits que me han hecho falta. **“Git commit -m “comentario”.**
12. Ahora que tengo todo listo me voy a la rama main **“Git checkout main”** y voy a pasar a este todos los datos nuevos que he cambiado con **“Git merge Desarrollo”**
13. Una vez que ya tengo todo en la rama main solo tengo que hacer un **“Git push origin main”** y ya estaría todo subido al repositorio.