



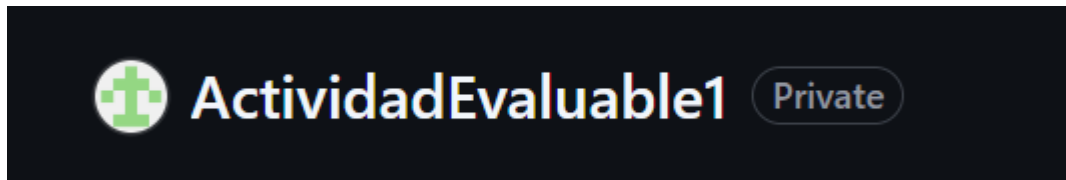
Trabajo Git en conjunto

Uso de metodologías SCRUM y Jira

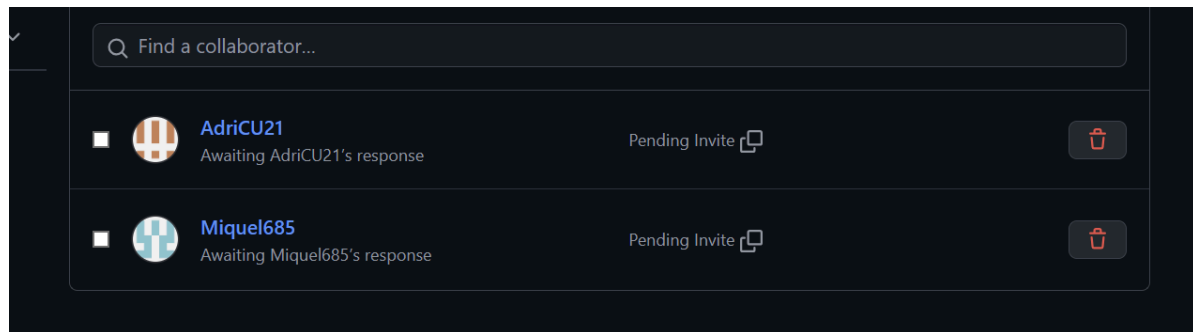


Creación y vinculación de Github.

Paso 1: Creación del repositorio Github.



Paso 2: Invitamos a los colaboradores para poder tener acceso al repositorio.

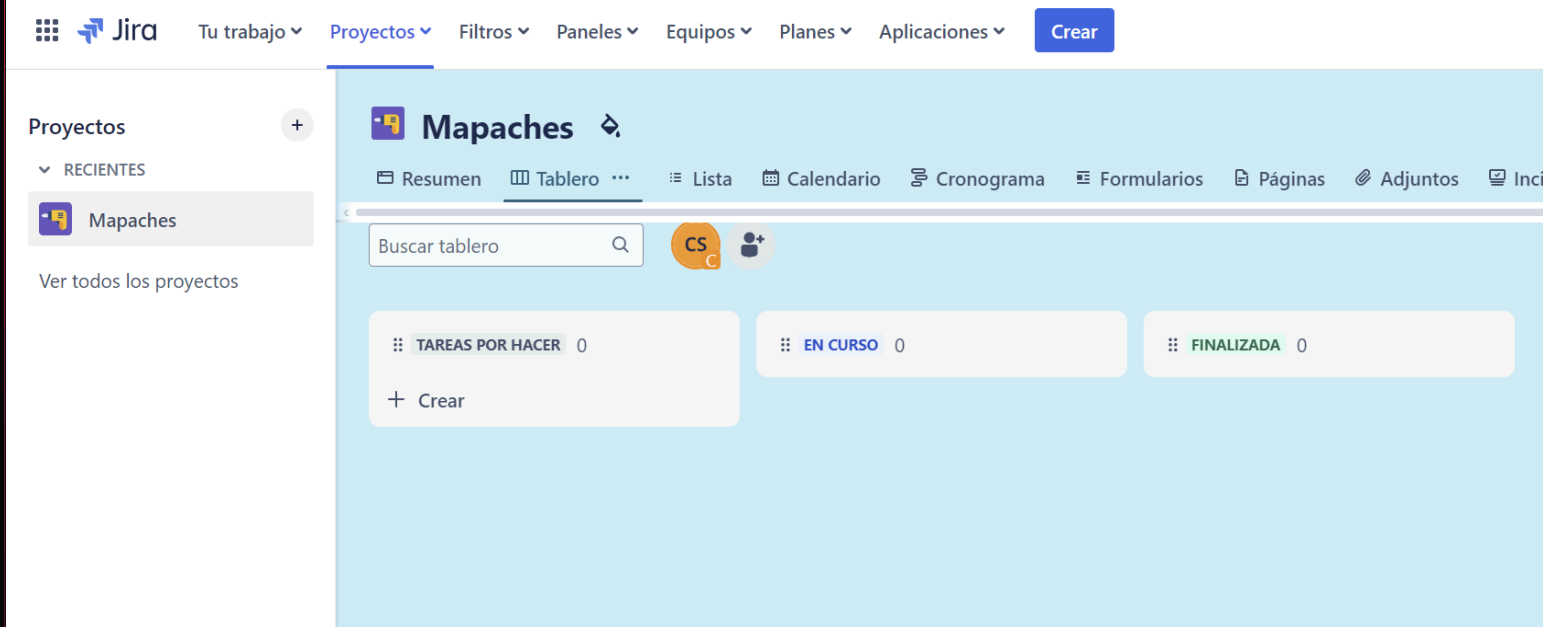


Jira, herramienta de trabajo.

La herramienta digital de organización que usaremos será Jira, puesto que es un producto de Trello y algunos miembros ya tienen experiencia con este tipo de herramientas.



Jira nos proporciona una interfaz gráfica del lugar de trabajo, donde aquí nos asignaremos las tareas pendientes, y tendremos una cronología y un resumen del trabajo realizado por parte de nuestro equipo.



Resumen de estado

Tendrás que crear algunos elementos para tu proyecto a fin de obtener un resumen del estado del progreso del equipo. [Crear un elemento](#)



Tareas por hacer	0
En curso	0
Finalizada	0
Total	0

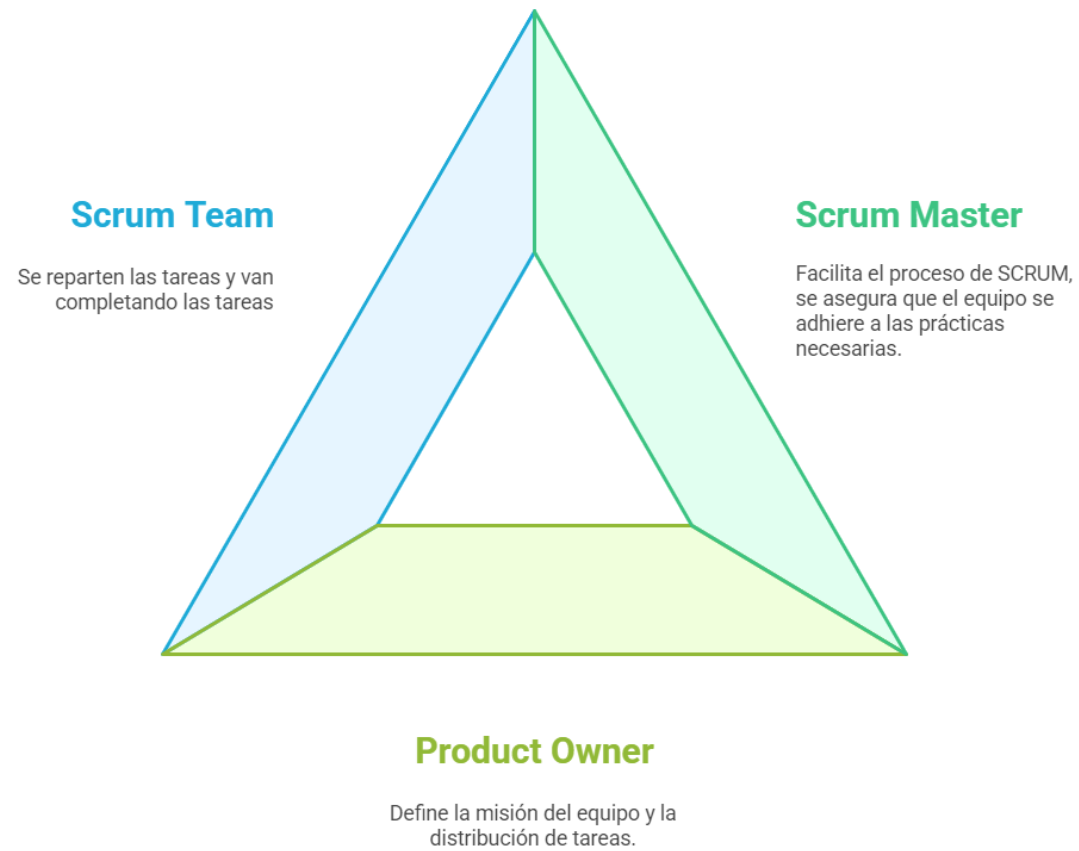
Selección de roles - Scrum

Adrián: Scrum Team

Miquel: Product Owner, (Scrum Team)

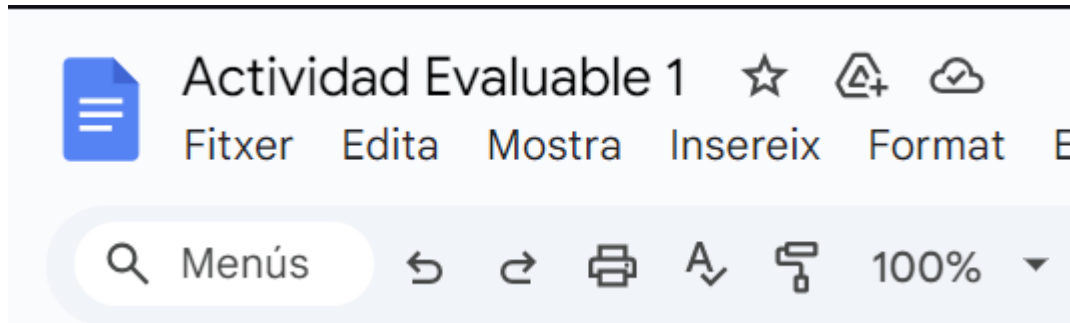
Carlos: Scrum Master, (Scrum Team)

ROLES DE SCRUM



Creación de este propio documento.

Haremos un seguimiento del trabajo y de las partes implicadas mediante un docs, el cuál vamos a usar como tutorial para que así pueda verse todos los procesos y pasos a seguir.



Creación de carpeta local.

Crearemos una carpeta cada uno para poder almacenar el código en local para cada miembro del equipo.

Cada uno de los integrantes va a tener su carpeta en la cuál trabajará y es aquí donde se harán todas las modificaciones antes de subirlas a github.

Organización de Tareas y su tiempo.

Así debe verse el menú.

```
INVENTARI CIFP PERE DE SON GALL
1. Afegir producte
2. Eliminar producte per nom
3. Eliminar producte per posició
4. Substituir producte
5. Modificar producte
6. Mostrar productes
7. Mostrar resum de l'inventari
8. Sortir
Insereix la teva opció: |
```

Necesitamos tener:

- clase Main
 - Debe contener todas las llamadas a métodos de diferentes clases
 - No puede haber repetición de códigos
- clase Añadir producto
- clase Eliminar producto por nombre

- clase Eliminar producto por posición
- clase Sustituir producto
- clase Modificar producto
- clase Mostrar resumen del inventario
→ Debe tener un desplegable visual
- clase Salir
- Refactorización de código

Organización del trabajo

Semana 1

La primera semana de trabajo arrancamos con el primer Sprint, este no va a ser un Sprint convencional, ya que primero hay que agarrar el ritmo y entrenar al equipo para que se acostumbren a los sprints.

Este Sprint va a consistir en 4 días de meetings diarios en los cuáles vamos a establecer los siguientes puntos vitales:

- Objetivos del día
- Duración de los objetivos (estimación del trabajo)
- Desglosar el trabajo (Partes de código)

Semana 2

Reunión Pommodoro 1

Primera reunión para dejar claro los objetivos y empezar a trabajar en el proyecto.
Nos hemos reunido a las 15:15 y ha durado hasta las 15:40.

Nos hemos puesto de acuerdo en que lo primero que vamos a hacer es crear un Main, este albergará la llamada a todos los métodos que haremos más adelante.

Esta es la opción más eficiente, ya que de este modo vamos a reducir dependencias y todo va a ser más atómico. Nos ahorraremos los problemas de merge y push.

Reunión Pommodoro 2

Sesión de trabajo conjunta 1.

Empezamos a las 18:40 y acabamos a las 19:30.

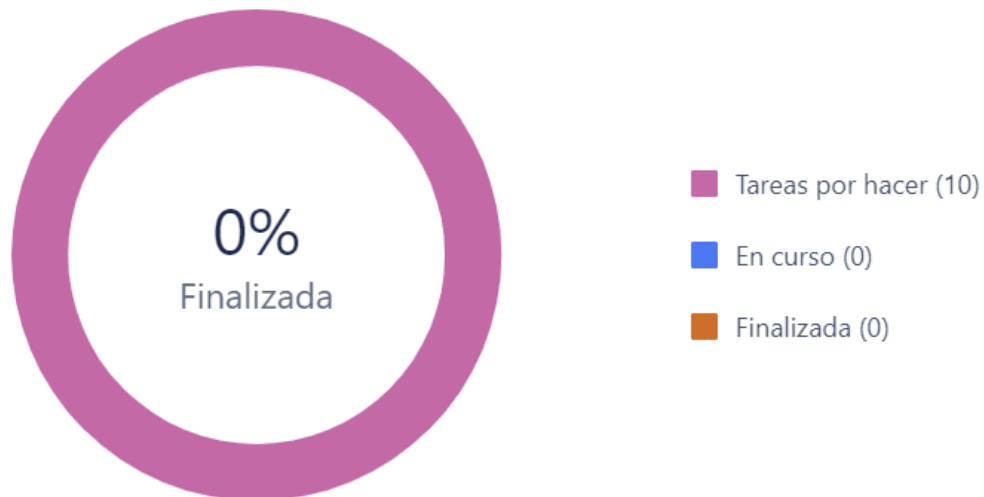
En esta sesión nos hemos reunido para organizar todas las tareas y dividir las en partes más pequeñas. Estas tareas las ponemos en la app Jira y así tenemos un dashboard visual de nuestras tareas.

<input type="checkbox"/>	Tipo	# Clave	≡ Resumen	➔ Estado	≡ Categoría	@ Persona asignada	📅 Fecha de ve...	⬆ Priori	+
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-1	Main (llamadas a métodos)	TAREAS POR HAC...		AU ADRIÁN CAMACHO ...		=	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-4	clase Añadir producto	TAREAS POR HAC...		CS CARLOS ALEXANDR...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-5	clase Eliminar prod por nombre	TAREAS POR HAC...		MA MIQUEL ROSSELLÓ ...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-6	clase Eliminar prod por posición	TAREAS POR HAC...		AU ADRIÁN CAMACHO ...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-7	clase Sustituir producto	TAREAS POR HAC...		CS CARLOS ALEXANDR...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-8	clase Modificar producto	TAREAS POR HAC...		MA MIQUEL ROSSELLÓ ...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-9	clase Mostrar resumen del inventario	TAREAS POR HAC...		AU ADRIÁN CAMACHO ...		^	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-10	clase Salir	TAREAS POR HAC...		AU ADRIÁN CAMACHO ...		≡	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-11	Refactorización de clases 1-4	TAREAS POR HAC...		CS CARLOS ALEXANDR...		∨	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MAP-12	Refactorización de clases 5-8	TAREAS POR HAC...		MA MIQUEL ROSSELLÓ ...		∨	

Nos hemos puesto de acuerdo en que hay un total de 10 tareas totales, contando subtareas de algunas tareas más grandes. Estas tareas las hemos asignado de forma equitativa entre los 3 miembros del grupo. El SCRUM Master (Carlos), se ha encargado de asignar la prioridad de las tareas, el objetivo de esto es enfocarnos en lo importante priorizando las tareas más valiosas para este sprint.

Resumen de estado

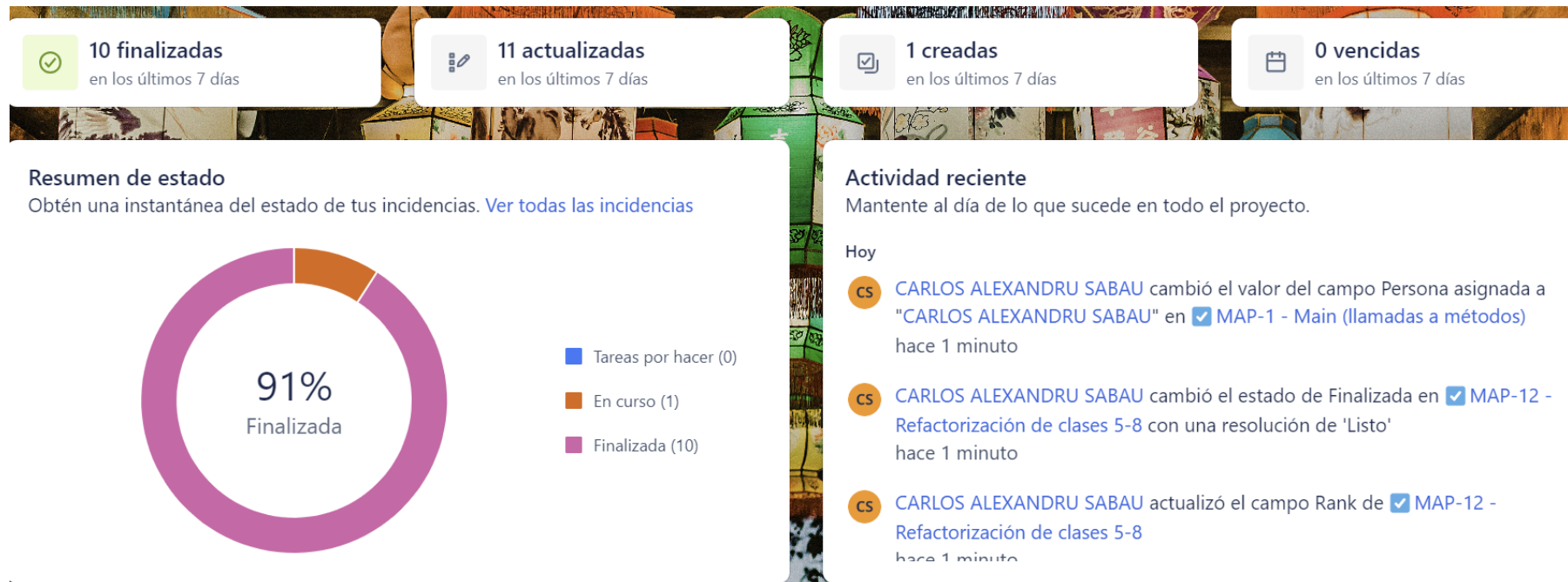
Obtén una instantánea del estado de tus incidencias. [Ver todas las incidencias](#)



Cada semana haremos un update de este panel para ver el progreso del equipo y monitorizar la eficiencia.

ULTIMO SPRINT

Para este último sprint, nos hemos organizado y hemos hechos varios meetings, de los cuales discutimos y resolvimos problemas cruciales para el desarrollo del programa.



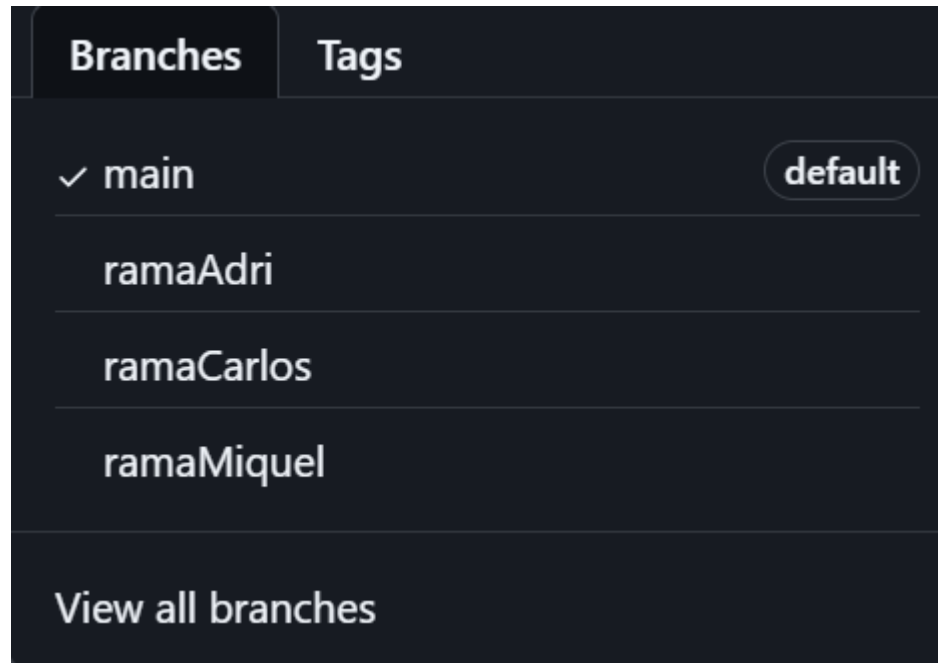
Decidimos por organizarnos e ir implementando función por función en el main, cada uno implementando su función creada en el main y subiendo los cambios de la siguiente forma:

```
int opcion = menu.menuInventari();  
switch (opcion) {  
    case 1 -> {  
        inventario = anadirProducto.anadirProductoInventario(inventario, valor());  
    }  
    case 2 -> {  
        inventario = eliminarProdNombre.eliminarProdNombreInventario(inventario, valor());  
    }  
    case 3 -> {  
        inventario = eliminarProdPosicion.eliminarPosicion(inventario, valorNumero());  
    }  
}
```

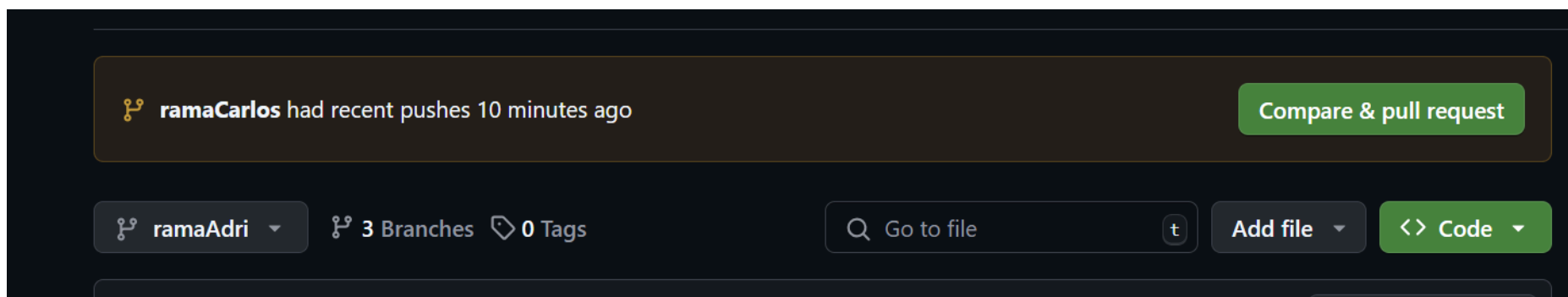
Todos los miembros hemos ido implementando en el switch que tenemos en el main las funciones.

Avances de programación y git

Primero todos nos creamos nuestras ramas para poder empezar el trabajo individual.



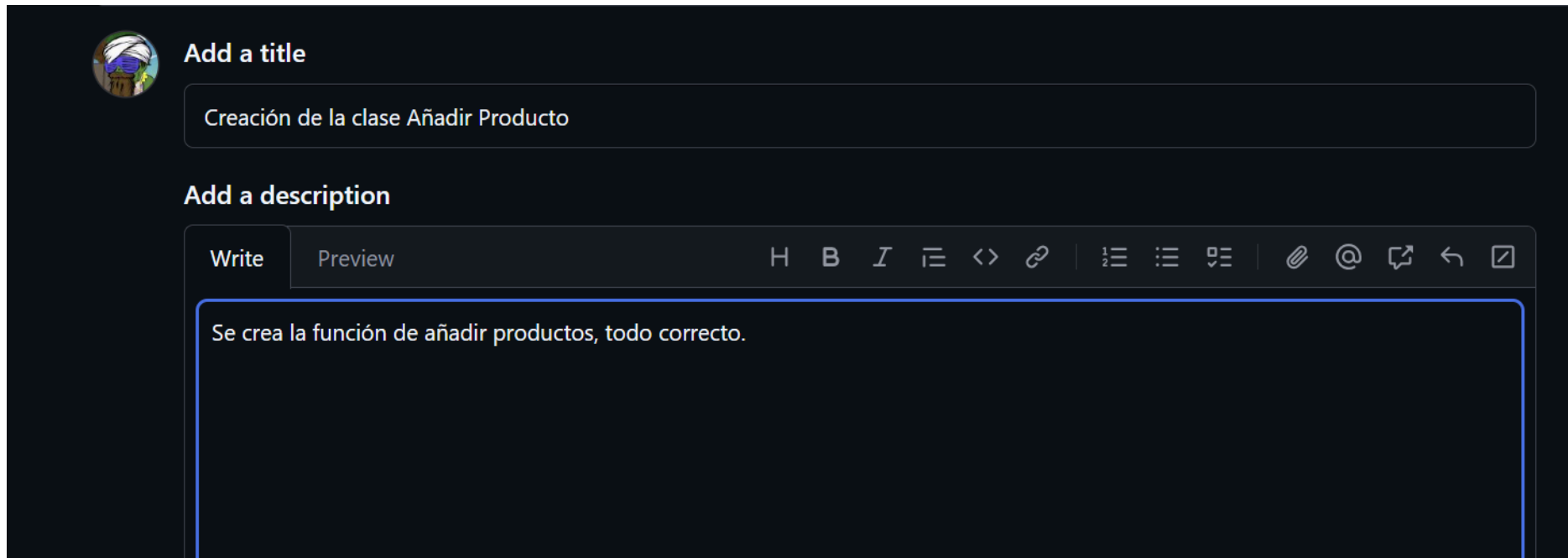
A la hora de subir algún cambio o push, los pasos a seguir son los siguientes:
Se revisa y compara el pull request



Luego vemos el código añadido, lo revisamos y si todo está bien y tiene buena lógica y buena praxis, pasa a la siguiente fase.



Añadimos una descripción breve de lo que se ha hecho, en este caso se crea la nueva funcionalidad de Añadir Producto.













The screenshot shows a GitHub commit interface. At the top left is a circular profile picture of a person with a white turban and a blue and green patterned shirt. To the right of the profile picture is the text "Add a title". Below this is a text input field containing the text "Creación de la clase Añadir Producto". Below the title field is the text "Add a description". Below this is a text input field with a rich text editor toolbar. The toolbar includes buttons for "Write" and "Preview", and various formatting options: H (heading), B (bold), I (italic), a list icon, a code icon (<>), a link icon, a link icon with a chain, a list icon, a list icon, a list icon, a link icon, an @ symbol, a share icon, a back arrow, and a close icon. The text input field contains the text "Se crea la función de añadir productos, todo correcto."

Add a title

Creación de la clase Añadir Producto

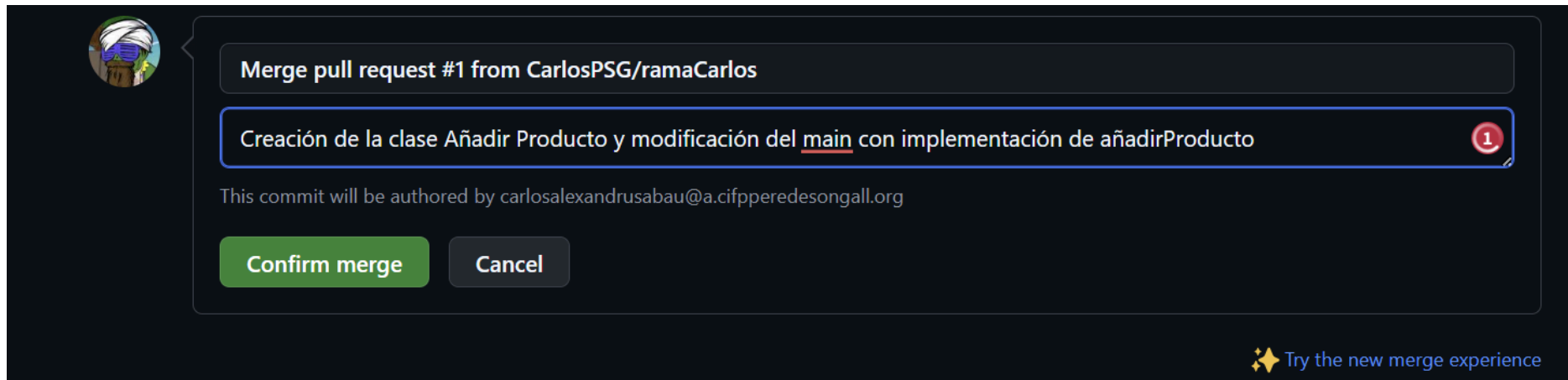
Add a description

Write Preview

H B I  <>       @   

Se crea la función de añadir productos, todo correcto.

Y finalmente acabaremos confirmando el merge con el main, para que aparezca en el main como un cambio definitivo.



Este es otro ejemplo, en donde creamos otra funcionalidad.

```
25 src/sustituirProducto.java
...  ...  @@ -0,0 +1,25 @@
1 + import java.util.Objects;
2 +
3 + public class sustituirProducto {
4 +     public static void main(String[] args) {
5 +
6 +     }
7 +
8 +     public static String[] reemplazarProducto (String[] inventario, String opcion){
9 +         System.out.println("Sustitución de producto: ");
10 +         String res = Main.valor();
11 +
12 +         int posicion = 0;
13 +
14 +         for (int i = 0; i < inventario.length; i++) {
15 +             if(Objects.equals(inventario[i], res)) {
16 +                 posicion = i;
17 +
18 +                 inventario[posicion] = res;
19 +
20 +                 break;
21 +             }
22 +         }
23 +         return inventario;
24 +     }
```

Comandos Github

Carlos ha creado el repositorio git y nosotros lo hemos clonado.

```
miguel@DESKTOP-PMVFP4U MINGW64 ~/Desktop/carlitosxpro
$ git clone https://github.com/CarlosPSG/ActividadEvaluable1.git
Cloning into 'ActividadEvaluable1'...
warning: You appear to have cloned an empty repository.

miguel@DESKTOP-PMVFP4U MINGW64 ~/Desktop/carlitosxpro
$ cd ActividadEvaluable1/
```

```
Usuario@PC-1122110 MINGW64 ~/Desktop/ActividadEvaluable1
$ git clone https://github.com/CarlosPSG/ActividadEvaluable1.git
Cloning into 'ActividadEvaluable1'...
warning: You appear to have cloned an empty repository.

Usuario@PC-1122110 MINGW64 ~/Desktop/ActividadEvaluable1
$ cd ActividadEvaluable1/

Usuario@PC-1122110 MINGW64 ~/Desktop/ActividadEvaluable1/ActividadEvaluable1 (main)
$ |
```

Leyenda de comandos y ayudas.

Si queremos clonar un repositorio → git clone linkDelRepo

Si por ejemplo queremos fusionar el main con nuestra rama para tener todo copiado, lo que tendremos que hacer es movernos a nuestra rama y de allí hacer un merge.

Ejemplo:

git checkout ramaCarlos

git merge main

De esta forma habremos copiado toda la carpeta del main en nuestra rama y podremos seguir trabajando de forma independiente

Como recuperar información perdida.

Resulta que yo sin saber eliminé un .iml super importante para que funcionara el proyecto.

El caso, acabé mergeando los cambios al main y por lo tanto nada funcionaba.

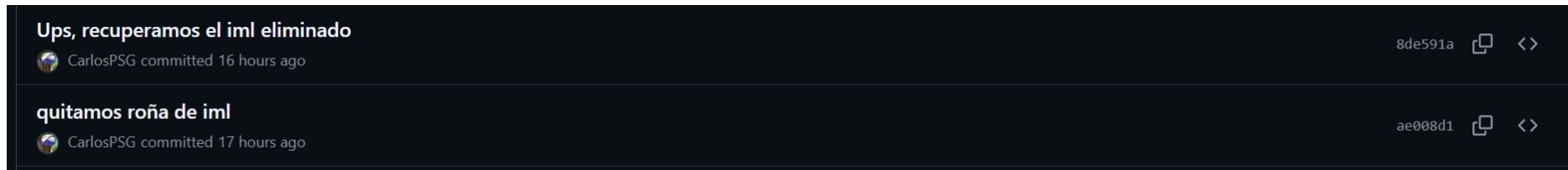
Por suerte, para esto sirven los merge, así que para volver atrás un commit en el tiempo tenemos que conseguir el log de operaciones.

```
litter@ALIEN MINGW64 ~/proyectoCarlos/ActividadEvaluable1 (main)
$ git reflog
ae008d1 (HEAD -> main, origin/ramaCarlos, origin/main, origin/HEAD, ramaCarlos) HEAD@{0}
: merge origin/ramaCarlos: Fast-forward
4176697 HEAD@{1}: checkout: moving from ramaCarlos to main
ae008d1 (HEAD -> main, origin/ramaCarlos, origin/main, origin/HEAD, ramaCarlos) HEAD@{2}
: commit: quitamos roña de iml
d6f9d1a HEAD@{3}: commit: cambios main y de funciones
b89e2d0 HEAD@{4}: commit: funcion numeros
ea9dc05 HEAD@{5}: merge origin/main: Merge made by the 'ort' strategy.
bfa43ec HEAD@{6}: checkout: moving from main to ramaCarlos
4176697 HEAD@{7}: checkout: moving from ramaCarlos to main
bfa43ec HEAD@{8}: reset: moving to HEAD
bfa43ec HEAD@{9}: checkout: moving from main to ramaCarlos
4176697 HEAD@{10}: checkout: moving from ramaCarlos to main
bfa43ec HEAD@{11}: checkout: moving from main to ramaCarlos
```

Como podemos ver, yo hice un commit de como quitaba la “roña” del iml, así que sabemos el ID que necesitamos recuperar es ese que está marcado en rojo.

```
litter@ALIEN MINGW64 ~/proyectoCarlos/ActividadEvaluable1 (main)
$ git checkout 4176697 -- src/ActividadEvaluable1.iml
```

Hacemos un checkout del ID y de la ruta y ya está, habremos vuelto atrás en el tiempo.



Reunión final, introspección.

Esta reunión se ha llevado a cabo en el ambiente de trabajo y de forma presencial.

Ha tenido una duración de aproximadamente 50 minutos, empezando a las 18:10.

El motivo de la reunión ha sido una conversación entre todos los miembros del equipo para dar voz a nuestras quejas respecto a la metodología del trabajo o a las metas del equipo.

Todos los miembros hemos llegado a la conclusión de que el trabajo se podría haber agilizado y se podrían haber ahorrado gran parte de la cantidad de errores mediante una mejor comunicación entre el PO, Scrum Master y la parte del equipo de desarrollo. No obstante, también estamos de acuerdo en que los errores nos han ayudado a mejorar significativamente la calidad del trabajo realizado y nos ha aportado más conocimiento a todos los miembros del grupo.