PRÁCTICA GENERAL

AMPLIACIÓN DE INGENIERÍA DEL SOFTWARE

GRADO EN INGENIERÍA INFORMÁTICA

GRUPO - M

Jesús Garcerán Sáez

Sergio Martín Vaquero

Daniel Aguado Gala

Alberto Hernández Jiménez

Alberto San José Bastante

Universidad Rey Juan Carlos

ETSII Escuela Técnica Superior de Ingeniería Informática

ÍNDICE

1. INTRODUCCIÓN

Esta práctica se basa en la realización de una calculadora científica guiándonos por la planificación de proyectos dada en clase. Consideramos la calculadora un proyecto puesto que es un conjunto de etapas, actividades y tareas para alcanzar un objetivo, que implica un trabajo no inmediato en un plazo relativamente largo.

Lo primero que hicimos en este proyecto fue la planificación de la práctica. Definimos el tipo de equipo y los requisitos. Tras esto, comenzamos a realizar los diagramas de casos de uso y HTA y definimos las tareas a realizar. Realizamos una planificación de tareas con su correspondiente asignación y completamos el diagrama PERT.

Tras hacer el diagrama PERT, comenzamos la realización de la implementación. A la vez, íbamos actualizando el calendario cada día que trabajábamos en el proyecto con el fin de ver si cumplíamos la planificación. La implementación la hicimos usando GitHub como herramientas de control de versiones.

Por último, al terminar la implementación, definimos los casos de prueba basándonos en la técnica de caja negra de "Partición de clases de equivalencia". Realizamos los casos de prueba y comprobamos los resultados obtenidos con los esperados. Tras esto, completamos la documentación e hicimos la presentación para clase.

TIPO DE EQUIPO

Para la realización de esta práctica hemos contado con un tipo de equipo de estilo **Descentralizado Democrático (DD).** Este organigrama de equipo tiene las siguientes características:

- No tiene líder de grupo permanente, en función de la tarea manda uno u otro.
- Todas las decisiones sobre la práctica se toman en grupo por consenso.
- La comunicación en el grupo se hace de manera horizontal, y las opiniones e ideas de todos los integrantes del grupo tienen el mismo valor.

Con este tipo de quipo hemos conseguido tener una moral más alta en el grupo y una mayor satisfacción. La comunicación ha sido constante.

Por tanto, el paradigma de organización ha sido un **paradigma abierto**, con las siguientes características:

- Mucha comunicación en el grupo.
- Se cuenta con el grupo para la toma de cualquier decisión.

Hemos intentado funcionar con un **Desarrollo Ágil**, puesto que somos un equipo pequeño. De esta manera, la práctica ha ido avanzando poco a poco sin tener problemas de testing o de ofrecer al cliente un programa que no se ajusta a lo pedido.

Los equipos ágiles se caracterizan por:

- Una gran autonomía al a hora de trabajar por parte de todos los integrantes.
- Mínima planificación de las tareas, vamos avanzando poco a poco.
- Realización de breves reuniones diarias en equipo para coordinar el trabajo diario.
- Se considera clave la auto-organización continua y la colaboración entre todos los integrantes del grupo.
- Obtenemos nuestro enfoque según los requisitos del producto y los estándares.

Junto con el desarrollo ágil realizamos "reuniones diarias" para ver cómo íbamos avanzando en el proyecto, basándonos en la guía SCRUM.

2. MANTENIMIENTO DEL SOFTWARE

La parte inicial de la práctica. En este apartado, buscamos obtener la información necesaria para poder comenzar a realizar el proyecto. Por tanto, realizamos una planificación del proyecto, identificando las partes más importantes para comenzar con buen pie:

- Requisitos. Son los servicios que tendrá que proporcionar el sistema y de sus restricciones operativas. Reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema.
- Planificación de tareas. Dividimos el proyecto en tareas, asignando a cada participante del grupo una serie de estas. Además, realizamos una planificación de cuánto tiempo le dedicaremos a la tarea con el fin de poder entregar el proyecto a tiempo. Por último, comparamos la planificación realizada al principio con el calendario real del proyecto, para ver las diferencias y errores que hemos tenido en cuanto a planificación.

a) Requisitos

El concepto de requisito es bastante ambiguo. Se puede definir como "lo que debe hacer el sistema", aunque depende mucho de lo que entendemos el equipo de desarrollo. Dentro de los requisitos definimos dos tipos de requisitos, los funcionales y no funcionales.

i) Requisitos funcionales y no funcionales

REQUISITOS FUNCIONALES

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.

Mediante consenso, hemos definido los siguientes requisitos funcionales, ya que son los que entendimos a partir de la información dada por el cliente.

ID	Requisito	Tipo
RF01	Paréntesis (jerarquía de operaciones)	Jerarquía
RF02	Impedir la utilización de dos signos seguidos (formulas erróneas)	Jerarquía
RF03	Tener la funcionalidad completa por teclado (por ejemplo, para realizar el inverso pulsar “i”).	General
RF04	Permitir realizar operaciones con números decimales.	General
RF05	Poder borrar un carácter o una operación	General
RF06	Poder realizar operaciones científicas (inverso, raíz cuadrada, etc.)	General
RF07	Introducir números por teclado	General
RF08	Realizar operaciones básicas de calculadora (suma, resta, multiplicación y división).	Jerarquía

REQUISITOS NO FUNCIONALES

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad.

Al igual que los requisitos funcionales, los hemos definid o de manera grupal a partir de la información precisada por el cliente.

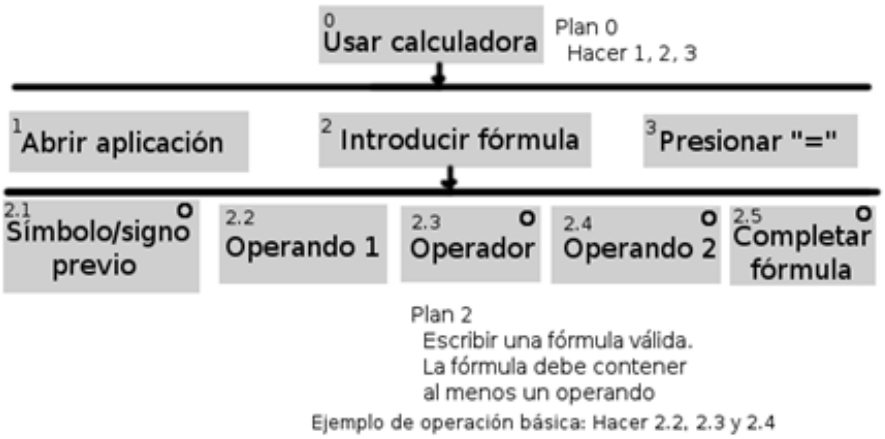
ID	Requisito	Tipo
RNF01	El lenguaje de programación será Java	General
RNF02	Colocación habitual de botones numéricos en la calculadora	Interfaz
RNF03	Añadir interfaz con apariencia de calculadora	Interfaz
RNF04	Debe ser una aplicación robusta y tratar errores	General
RNF05	Visualizar la operación que estas realizando en la parte superior de la pantalla de la calculadora.	Interfaz
RNF06	Poder redimensionar la calculadora con un diseño “responsive”	General
RNF07	Ofrecer botón de ayuda al usuario	Interfaz

ii) **Diagrama HTA**

Una vez enumerados los requisitos candidatos, necesitamos, para entenderlos mejor, comprender el Contexto del Sistema y capturar los Requisitos Funcionales. Esto es algo que se suele hacer al comenzar un programa, y sirve para clarificar las ideas y compartirlas con el cliente.

- **Comprender el Contexto del Sistema (Diagrama HTA)**

Para comprender el contexto del sistema (el entorno en el que se enmarcará el sistema), hemos realizado un diagrama HTA.

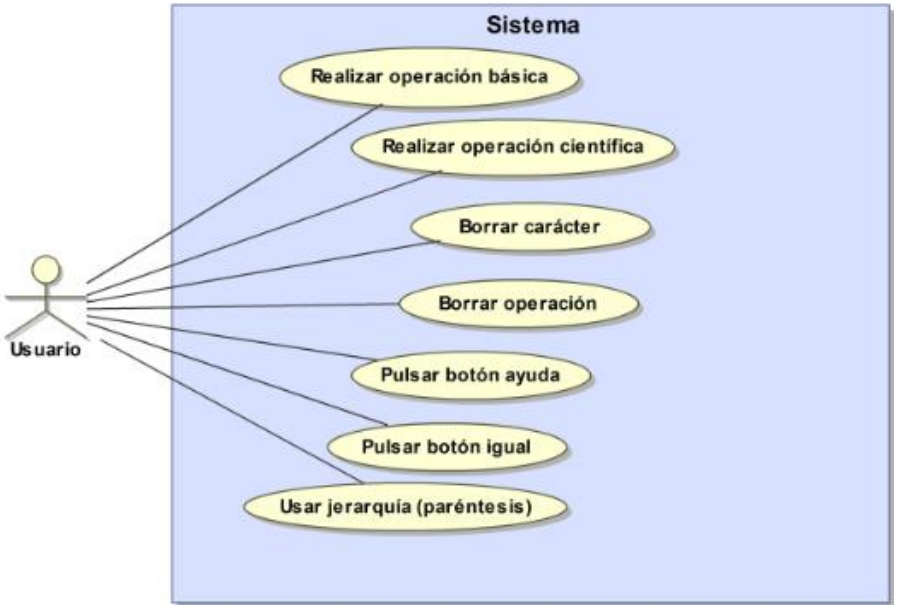


iii) Diagrama de Casos de Uso

- **Capturar los requisitos funcionales (Diagrama de casos de uso)**

El diagrama de casos de uso permite a los desarrolladores de software y a los clientes llegar a un acuerdo sobre los requisitos. Cada caso de uso representa la forma en que los actores usan el sistema.

El diagrama realizado ha sido el siguiente:



Cada caso de uso ha sido descrito de la siguiente manera:

Caso de uso: “Realizar operación básica”
Actor: Usuario
Descripción: El usuario pulsará un número (con una serie de pulsaciones de botón indeterminado de pulsaciones de botón) y tras esto seleccionará la operación deseada a realizar (suma, resta, multiplicación, división, etc.). Por último, pulsará otra serie de botones para obtener un número, o, por el contrario, empezaremos a usar jerarquía (con paréntesis).

Caso de uso: “Realizar operación científica”
Actor: Usuario
Descripción: El usuario pulsará un número (con una serie de pulsaciones de botón indeterminado de pulsaciones de botón) y tras esto seleccionará la operación deseada a realizar (raíz cuadrada, inverso, doble, etc.). La operación se hará directamente y el sistema mostrará el resultado de la operación.

Caso de uso: “Pulsar botón de ayuda”
Actor: Usuario
Descripción: El usuario pulsará el botón ayuda y se mostrará en pantalla una nueva pestaña en la que se le explicará al usuario como usar la calculadora, las operaciones que puedes realizar con ella y todo tipo de información relevante sobre el sistema.

Caso de uso: “Borrar carácter”
Actor: Usuario
Descripción: El usuario pulsará el botón “CE” y el sistema borrará el último carácter que ha escrito el usuario. Se puede pulsar el botón tantas veces como sea necesario siempre que existan caracteres.

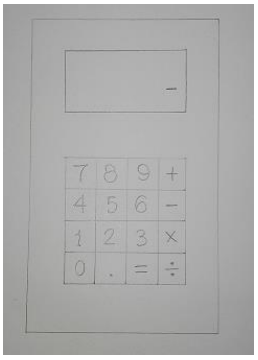
Caso de uso: “Borrar operación”
Actor: Usuario
Descripción: El usuario pulsará el botón “C” y el sistema borrará la operación realizada hasta el momento, es decir, reseteará la operación realizada hasta el momento para empezar de cero. El sistema borrará toda la información almacenada hasta el momento y permitirá al usuario escribir otra operación.

Caso de uso: "Pulsar botón igual"
Actor: Usuario
Descripción: El usuario pulsará el botón igual y el sistema realizará la operación o la suma de operaciones con su correspondiente jerarquía. El sistema, una vez realizada la operación, mostrará el resultado en pantalla.

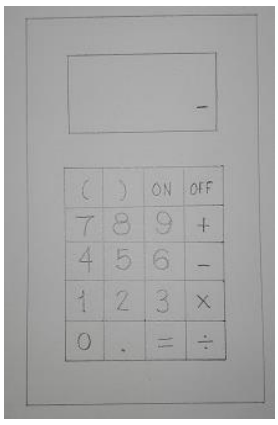
Caso de uso: "Usar jerarquía (paréntesis)"
Actor: Usuario
Descripción: El usuario pulsará el paréntesis izquierdo, y tras esto realizaremos una operación (ya sea compleja o básica) en la que podremos escribir otro paréntesis en su interior. Una vez escrita la operación, pulsaremos el botón paréntesis derecho para obtener la jerarquía y poder obtener el resultado.

iv) Prototipos

Primer prototipo. El primer prototipo es una base de la calculadora que deseamos realizar, el resultado obtenido es el de una calculadora simple sin nada destacable, con el aspecto habitual de una calculadora



Segundo prototipo. Tras la fijarnos en los requisitos, lo primero que definimos fue que sería necesario tener un botón de encendido y apagado. Además, puesto que la jerarquía era algo que deberíamos tener muy presente a la hora de realizar la calculadora, añadimos ambos apartados. El resultado fue el siguiente:



Tercer y último prototipo. Nos fijamos en todos los requisitos que habíamos cerrado. Vimos necesario el ampliar las funciones, y por tanto, guiándonos por la apariencia de una calculadora científica típica, realizamos el prototipo final. Vimos necesario un botón de ayuda para explicar el funcionamiento de la calculadora, que puede que no sea necesario, pero toda ayuda es bien recibida. El resultado final fue el siguiente:

b) Planificación de tareas

El objetivo del proyecto es el de realizar una calculadora científica, cuya funcionalidad principal sea la de la jerarquía de operaciones. Debe ser una calculadora para ordenador, por tanto, debe tener la funcionalidad básica mediante botones, aunque también podría usarse el teclado. Obviamente, la calculadora debe realizar todas las operaciones básicas que realiza cualquier calculadora, y permitir trabajar con todo tipo de números.

i. Identificar tareas

Lo primero que realizamos es la identificación de tareas a realizar por el equipo. Este es uno de los primeros trabajos que se hizo para la correcta planificación del proyecto. Las tareas a realizar se definieron de manera conjunta por todos los integrantes del grupo y con un nivel de detalle medio. Las tareas que definimos fueron las siguientes:

1. Definir los requisitos de usuario.
2. Definir tipo de equipo.
3. Definir las tareas a realizar.
4. Organizar las tareas entre los integrantes del equipo.
5. Hacer el diagrama de casos de uso.
6. Hacer el diagrama HTA.
7. Hacer el prototipo en papel de la interfaz gráfica.
8. Hacer el prototipo “detallado”.
9. Hacer el diagrama PERT.
10. Planificación de la realización de pruebas.
11. Realizar la interfaz de la calculadora.

12. Implementar la funcionalidad de los botones numéricos (cuando pulsas que salga en pantalla).
13. Implementar la realización de operaciones básicas (suma, resta, multiplicación y división).
14. Implementar el botón de igual para la realización de la operación.
15. Implementar “no permitir más de un carácter de operación a la vez”, permitir solo realizar una operación.
16. Implementar el borrado de un solo carácter y de una operación.
17. Implementar la realización de las operaciones científicas (inverso, doble, raíz cuadrada).
18. Implementar la funcionalidad de botones numéricos por teclado.
19. Implementar la funcionalidad avanzada por teclado.
20. Mostrar en la parte superior de la pantalla la operación que estamos realizando.
21. Crear la pestaña ayuda para el usuario, informando de los “atajos de teclado” y de cómo funciona la calculadora.
22. Implementar la realización de priorización de operaciones mediante paréntesis.
23. Realizar la documentación del proyecto.
24. Casos de prueba y resultados obtenidos.

ii. Asignar tareas y estimar

Una vez definimos las tareas, realizamos una asignación de tareas a los distintos integrantes del equipo. Esta asignación se realizó, al igual que la definición de tareas, de manera conjunta y cada persona eligió la parte que iba a realizar, de manera que nadie asigno a otra persona una tarea.

Esta asignación se hizo siguiendo la filosofía de equipo de **Descentralizado Democrático**, como hemos comentado anteriormente.

Una vez asignadas las tareas, pasamos a la estimación de cuánto tiempo pensábamos que teníamos que dedicarle a cada tarea de las definidas. De tal forma, se realizó una planificación temporal. Definimos la siguiente tabla:

Tarea	Duración (en días)
A Definir los requisitos de usuario	2
B Definir tipo de equipo	1
C Definir las tareas a realizar	1
D Organizar las tareas entre los integrantes del equipo	1
E Hacer el diagrama PERT	2
F Hacer el diagrama de casos de uso	1
G Hacer el diagrama HTA	1
H Hacer el prototipo en papel de la interfaz gráfica	1
I Hacer el prototipo "detallado"	2
J Planificación de la realización de pruebas	2
K Realizar la interfaz de la calculadora	3
L Implementar la funcionalidad de los botones numéricos (cuando pulsas que salga en pantalla)	2
M Implementar la realización de operaciones básicas	2
N Implementar el botón de igual para la realización de la operación	2
O Implementar "no permitir más de un carácter de operación a la vez	3
P Implementar el borrado de un solo carácter y de una operación	2
Q Implementar la realización de las operaciones científicas	3
R Implementar la funcionalidad de botones numéricos por teclado	2
S Implementar la funcionalidad avanzada por teclado	3
T Mostrar en la parte superior de la pantalla la operación que estamos realizando	1
U Crear la pestaña ayuda para el usuario	1
V Implementar la realización de priorización de operaciones mediante paréntesis	3
W Casos de prueba y resultados obtenidos	3
X Realizar la documentación del proyecto	2

Esta estimación se hizo en base a nuestro conocimiento y experiencia como desarrolladores. Además, se tuvo en cuenta el tiempo que se le iba a dedicar (aproximadamente) al trabajo, ya que, al no poder dedicarle totalmente nuestro tiempo diario a este proyecto, le íbamos a dedicar intervalos de tiempo variables.

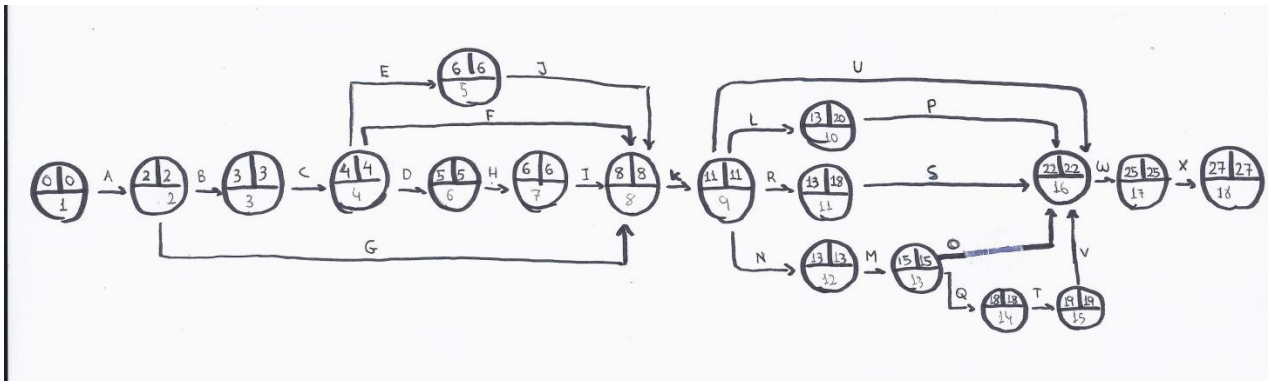
iii. Definir diagrama PERT

Son diagramas basados en redes de precedencia. Es una representación gráfica del proyecto que relaciona las actividades, y permite visualizar las que son críticas. El diagrama PERT definido, según las tareas y nuestra estimación temporal, fue el siguiente:

DIAGRAMA PERT

Actividades	Precedentes	Duración	TEi	TLi	TEj	TLj	HT	HL	HI	RRHH
A	-	2	0	0	2	2	0	0	0	Todos
B	A	1	2	2	3	3	0	0	0	Todos
C	B	1	3	3	4	4	0	0	0	Jesús
D	C	1	4	4	5	5	0	0	0	Todos
E	C	2	4	4	6	6	0	0	0	Sergio, Jesús
F	C	1	4	4	8	8	3	3	3	Jesús
G	A	1	2	2	8	8	5	5	5	Alberto dlP
H	D	1	5	5	6	6	0	0	0	Alberto
I	H	2	6	6	8	8	0	0	0	Alberto
J	E	2	6	6	8	8	0	0	0	Alberto
K	J,I,F,G	3	8	8	11	11	0	0	0	Sergio
L	K	2	11	11	13	19	6	0	0	Alberto
M	N	2	13	13	15	15	0	0	0	Alberto
N	K	2	11	11	13	13	0	0	0	Alberto
O	M	3	15	15	22	22	4	4	4	Alberto
P	L	2	13	20	22	22	7	7	0	Alberto
Q	M	3	15	15	18	18	0	0	0	Alberto
R	K	2	11	11	13	18	5	0	0	Alberto
S	R	3	13	18	22	22	6	6	0	Alberto
T	Q	1	17	17	18	18	0	0	0	Alberto
U	K	1	11	11	22	22	10	10	10	Alberto
V	T	3	19	19	22	22	0	0	0	Alberto dlP
W	U,P,S,O,V	3	22	22	25	25	0	0	0	Jesús
X	W	2	25	25	27	27	0	0	0	Jesús

El diagrama PERT se quedó de la siguiente manera:



Las conclusiones obtenidas fueron que las tareas L y R eran las únicas que tenían un poco de margen. El resto eran tareas críticas que se deberían realizar según el tiempo estimado y no deberíamos sobrepasarlo.

iv. Calendario real

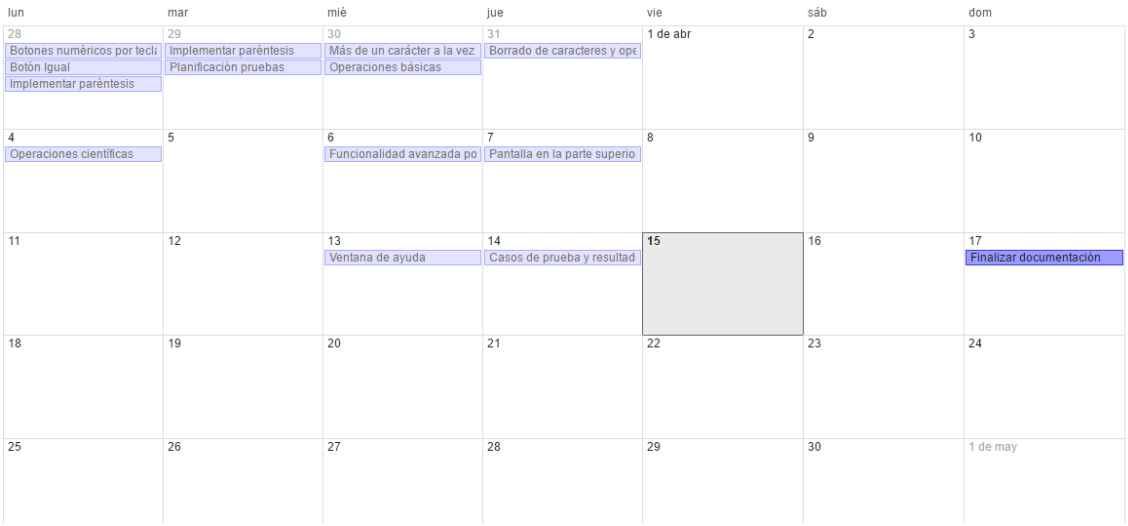
Una vez realizada la planificación comenzamos a realizar el proyecto, en cuanto a desarrollo se refiere. Cada día que se trabajaba en el proyecto era apuntado en un calendario con el fin de ver si se cumplía la estimación realizada.

CALENDARIO

Marzo

lun	mar	mié	jue	vie	sáb	dom
29	1 de mar	2 Definición de requisitos	3 Definición de requisitos Tareas a realizar	4	5	6
7	8	9 Organizar tareas	10 Diagrama PERT	11 Diagrama PERT	12	13 Diagrama C. de uso Diagrama HTA Prototipo
14	15	16 Prototipo	17 Prototipo detallado	18 Interfaz de usuario	19 Botones numéricos Interfaz de usuario	20 Botones numéricos por tecla Interfaz de usuario
21	22	23	24	25	26	27
28 Botones numéricos por tecla Botón Igual Implementar paréntesis	29 Implementar paréntesis Planificación pruebas	30 Más de un carácter a la vez Operaciones básicas	31 Borrado de caracteres y ope	1 de abr	2	3

Abril



Los problemas o inconvenientes surgidos fueron los siguientes:

En un primer momento teníamos pensado intentar hacer cada día una tarea como mínimo, teniendo en cuenta su duración, dado que si teníamos una tarea de dos días de duración, debíamos hacerla en días consecutivos. Pero esta idea no pudimos llevarla a cabo debido a que no teníamos prácticas de otras asignaturas o no todos teníamos el mismo horario por lo que no podíamos atender las dudas que nos surgían al momento.

En cuanto a las duraciones que asignamos a cada tarea las hemos cumplido considerablemente bien. Aunque tuvimos algún retraso a la hora de realizar la tarea de “Realizar la interfaz de la calculadora”, pero no nos supuso ningún problema para la realización de las demás.

A su vez, cuando se acercaban las vacaciones de Semana Santa y dado que llevábamos bastante adelantada la práctica decidimos descansar esa semana, puesto que la mayoría de los componentes del grupo íbamos a estar fuera en esas fechas y sin poder realizar la práctica en buenas condiciones.

Teníamos planeado acabar como muy tarde toda la documentación el día 17 de abril (como se ve en el calendario), uno anterior a la fecha límite de entrega, para poder comprobar que estaba todo correcto sin ninguna prisa y por si había que realizar modificaciones y lo hemos conseguido, sin agobiarnos mucho.

3. GESTIÓN DE CONFIGURACIÓN

Le gestión de configuración es el conjunto de actividades de seguimiento desde el inicio hasta el final del proyecto software.

a) Herramienta de control de versiones

La herramienta que hemos usado para el control de versiones ha sido GitHub. GitHub es una plataforma de desarrollo colaborativo de software, que sirve para alojar proyectos, utilizando el sistema de control de versiones Git.

Está herramienta nos proporciona una serie de herramientas bastante útiles para el trabajo en equipo. Destacamos entre ellas:

- Una wiki para el mantenimiento de las distintas versiones.
- Un sistema de seguimiento de problemas para detallas problemas o sugerencias con respecto al proyecto entre el equipo.
- Una herramienta de revisión de código que permite añadir anotaciones o debatir en cualquier punto de un fichero.
- Un visor de ramas donde se pueden comparar los progresos realizados.

Estrategia de versionado
Esta herramienta sigue una estrategia de versionado **Copiar-Modificar-Mezclar**. En esta estrategia, en el caso de un conflicto por que un usuario quiere subir un archivo desincronizado debe sincronizarlo para poder subirlo. De esta forma se solucionan los problemas de versiones entre distintos usuarios.

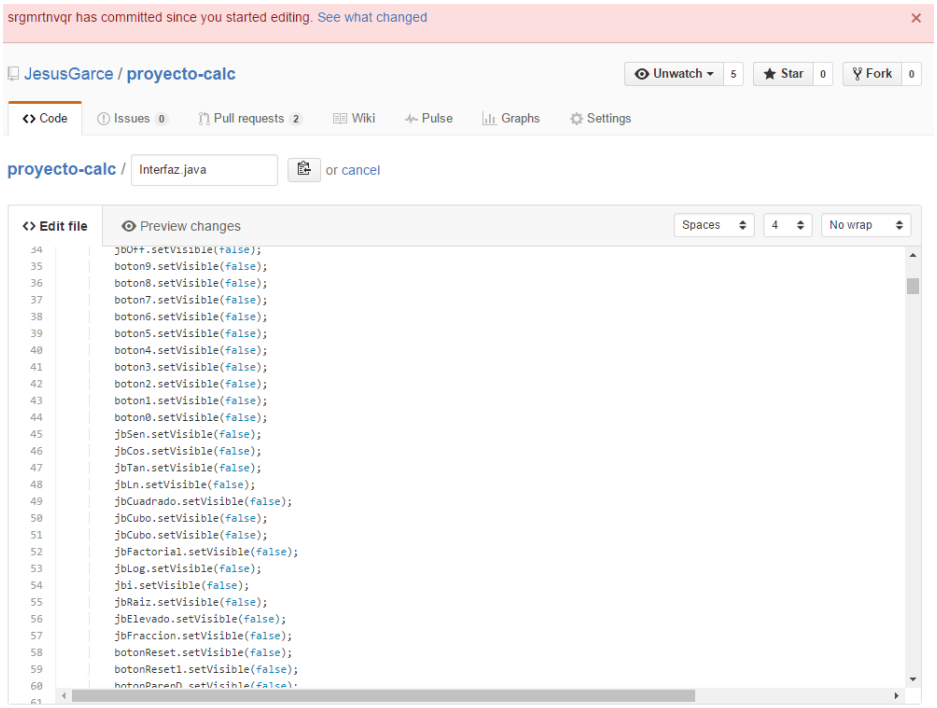
Tipo de repositorio
Es un sistema de control de versiones **distribuido**. Esto es, el repositorio está distribuido por todos los participantes, y estos tienen en local toda la información que necesitan. Sin embargo, no ocupa mucho espacio, ya que tendremos únicamente las partes que nos interesa (la rama de la que hemos partido).
Este tipo de repositorio provoca que ramas diverjan. Es un problema que se soluciona haciendo “merge”, uniendo los cambios sobre una línea principal.

b) Conflictos de versiones

Al usar varios usuarios un mismo proyecto y modificarlo a la vez es posible que surjan conflictos en las distintas versiones de un archivo. En la realización de esta práctica nos hemos encontrado con dos conflictos de versiones a los que aquí hacemos referencia. Ambos conflictos se han resuelto finalmente editando manualmente las líneas conflictivas.

Los conflictos surgidos han sido los siguientes:

Conflicto al editar un archivo .java a la vez en el editor de GitHub



Este conflicto se dio cuando dos usuarios intentaron modificar un archivo .java en el editor de GitHub.

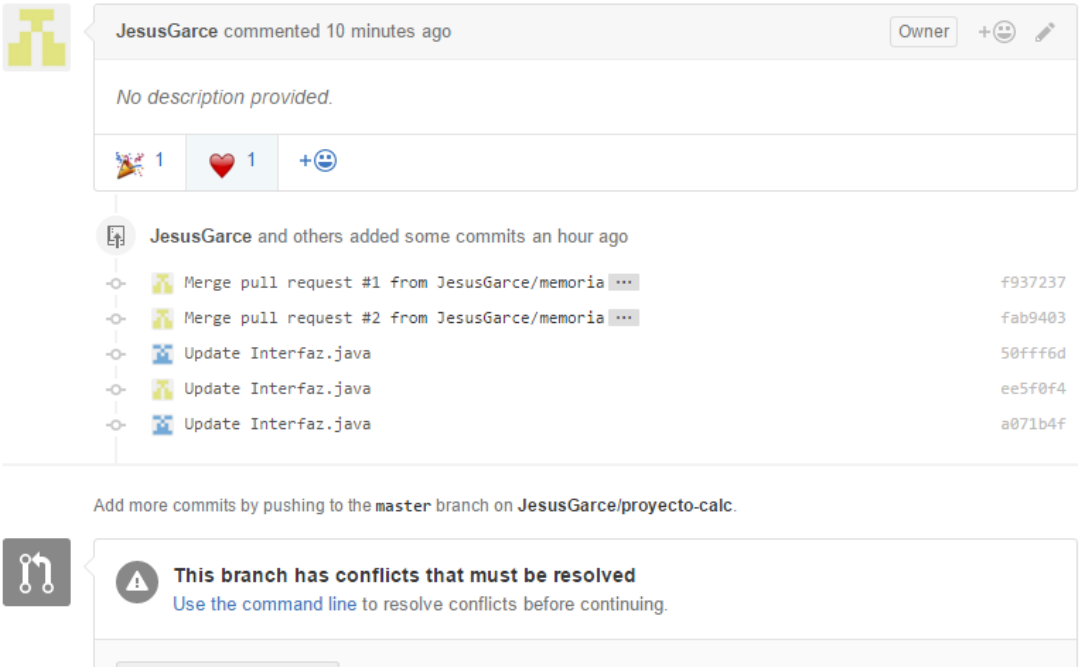
¿Dónde y cómo ocurrió?
Se ha producido en la parte de interfaz, en el código de la calculadora. *Srgmrtnzvqr* intentó realizar un pequeño cambio en la interfaz mientras *JesusGarce* estaba realizando cambios de nombres de variables con el fin de hacer el texto más legible. Cuando *Srgmrtnzvqr* termino de realizar el pequeño cambio guardo (realizo un "commit") y cerró el editor. Al terminar de cambiar los nombres de las variables y guardar el resultado surgió el error mostrado en pantalla.

¿Cómo se resolvió?
El usuario realizó una comprobación de los cambios que había realizado su compañero y se mezcló el archivo antes de guardar. Básicamente, siguiendo la filosofía de Copiar-Modificar-Mezclar. Tras esto, realizo el "commit" sin ningún error.

¿Qué tipo de control de cambios habría que realizar?
Se debería hacer un control de cambios de Gestión (u organizado). Es un error que ha ocurrido durante el proceso de desarrollo pero una vez hemos definido finalmente la interfaz. Por tanto, se debería aprobar la nueva versión de interfaz modificada con el fin de mejorar.

¿Es elemento de la configuración o línea base?
Puesto que se trata de un archivo dentro del código fuente, se trata de un elemento de configuración. Los elementos de configuración se tratan como una única entidad en el proceso de gestión de configuración. Además, era durante el proceso de desarrollo, no marcaba una fase final de ningún ciclo de vida.

Conflicto al actualizar el documento de la práctica



Este conflicto se dio cuando dos usuarios modificaron el documento de la práctica a la vez para realizar distintos cambios.

¿Dónde y cómo ocurrió?
Se ha producido en la parte de documentación de la práctica. El usuario *srgmrtnvqr* estaba realizando las capturas de pantalla de los casos de prueba, y el usuario *JesusGarce* escribía los errores surgidos en la planificación. Ambos habían creado una rama para modificar el documento. Cuando el usuario *srgmrtnvqr* termino de modificar el documento, realizo un "pull request" y cerró el proyecto. *JesusGarce* termino una hora después y al intentar hacer el "pull request" e intentar mezclar, le surgió el siguiente conflicto.

¿Cómo se resolvió?
JesusGarce resolvió los conflictos con la línea de comandos de GitHub, tal y como recomienda una vez surgido el error. Se consiguió mezclar y subir el documento actualizado con ambos cambios.

¿Qué tipo de control de cambios habría que realizar?
Puesto que es un error que surgió antes de aprobarse la documentación, se podría considerar necesario un tipo de control informal, hablando propiamente de la documentación. Eran cambios que se realizaban con el fin de completar la documentación, antes de que está sea aprobada y definida.

¿Es elemento de la configuración o línea base?
La documentación es un elemento de configuración del software (documentación), y el error se dio durante el proceso de "explotación". Sucedió mientras se realizaba la documentación por tanto, como no marca el final de ninguna fase, si no que era durante el proceso, no lo consideramos línea base. Así pues, es un elemento de configuración.

4. PRUEBAS DEL SOFTWARE

Las pruebas del software son importantes con el fin de solucionar fallos. Un fallo tiene muchísimos costes en software.

Un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados, y tiene una serie de características:

- Es un proceso de ejecución de un programa con el fin de descubrir un error.
- Un buen caso de prueba tiene una alta probabilidad de encontrar un error
- Una prueba tiene éxito si descubre un error no detectado con anterioridad.

Las pruebas del software tienen una serie de principios:

a) A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos
b) Deberán planificarse muchos antes de que empiecen.
c) El principio de Pareto es aplicable (80% resultado, 20% esfuerzo).
d) Se deben incluir tanto entradas correctas como incorrectas.
e) Las pruebas deberían empezar por lo pequeño y progresar a lo grande.
f) Imposibilidad de hacer pruebas exhaustivas.
g) Realización de pruebas por equipos independientes.

a) Técnica de caja negra

El enfoque de caja negra tiene una serie de principios:

- Se centra en los requisitos funcionales del software.
- Permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.
- No es una alternativa a las técnicas de prueba de caja blanca.
- Se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores.
- Tiende a aplicarse durante fases posteriores a la prueba.

Detecta errores en las siguientes categorías: funciones incorrectas, errores de interfaz, errores en estructuras de datos o accesos a bases de datos, errores de rendimiento y errores de inicialización.

La técnica que vamos a usar es la de **Partición en clases de equivalencia**. Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Una **clase de equivalencia** representa un conjunto de estados válidos o no válidos para condiciones de entrada.
Como, en nuestro caso, una condición de entrada requiere un valor específico, definimos una clase de equivalencia válida y una no válida, y en nuestra clase de equivalencia válida solo tendremos un resultado.

Hemos elegido esta técnica puesto que, al ser una calculadora lo que vamos a tratar, es más difícil descubrir cuáles son los “límites” de las clases, ya que en una calculadora un resultado debe ser el correcto, y cualquier otro resultado es erróneo. Por tanto, vemos más adecuado usar clases de equivalencia.

b) Identificación de casos de prueba

Para identificar los casos de prueba vamos a identificar cuáles son las clases que tenemos, con el fin de tratar todos los casos que podemos identificar en una calculadora científica. Separamos en diversos bloques:

- **Operaciones básicas.** Una calculadora, al escribir nuestra operación básica (suma, resta, multiplicación, división) debe darnos el resultado correcto.
- **Jerarquía de operaciones.**
    - Al escribir distintas operaciones en una misma línea se debe aplicar la jerarquía de operaciones
    - Al escribir paréntesis para marcar que operación se debe hacer antes y después debe tener un resultado correcto.
- **Operaciones científicas.**
    - Al realizar operaciones científicas con dos operandos debe dar un resultado correcto.
    - Al realizar operaciones científicas con un operando el resultado debe ser el adecuado.
- **Operaciones científicas + Jerarquía de operaciones.**
    - Al escribir operaciones científicas junto con operaciones básicas debe dar el resultado, tratando ambas operaciones.
    - Al escribir diversas operaciones científicas deben ser tratadas adecuadamente para dar el resultado correcto.
    - Al escribir operaciones científicas junto con operaciones básicas y paréntesis el resultado debe ser el adecuado.
    - Al escribir operaciones científicas junto con paréntesis el resultado debe ser correcto.
- **Funcionalidad extra**. Se debe comprobar que la funcionalidad de la calculadora, aparte de la ejecución de operaciones, actúa de manera correcta (hacer borrado de un carácter, de una expresión, permitir poner puntos, etc.)
- **Errores.** Se deben tratar los distintos errores que pueda tener el usuario al escribir la expresión a realizar por la calculadora.
- **Funcionalidad teclado.** Se debe comprobar que funcionan todas las teclas de la calculadora pulsando con el teclado.
- **Interfaz.** La interfaz debe tener una funcionalidad correcta sin errores visibles por el usuario. Todos los botones deben funcionar de manera adecuada.

c) Casos de prueba identificados

Basado en la identificación anterior, hemos definido, con el fin de tratar todas las opciones definidas en todos los bloques, los siguientes casos de prueba. Estos casos de prueba, al igual que en el apartado anterior, los hemos definido en bloques:

Operaciones básicas

1. Caso de prueba: **53 + 25**. Resultado esperado: **78**
2. Caso de prueba: **12.52 * 10.24.** Resultado esperado: **128.2048**
3. Caso de prueba: **15 / 0.58**. Resultado esperado: **25.86**
4. Caso de prueba: **-12 * 3**. Resultado esperado: **-36**
5. Caso de prueba: **9.87 * (-0.53)**. Resultado esperado. -**5.2311**

Jerarquía de operaciones

6. Caso de prueba: **(53 – 2.5) * 1.26**. Resultado esperado: **63.63**
7. Caso de prueba: **53 – 2.5 * 1.26**. Resultado esperado: **49.85**
8. Caso de prueba: **5 / 2.03 + 14 * 9**. Resultado esperado: **128.4630542**
9. Caso de prueba: **5 / (2.03 + 14) * 9**. Resultado esperado: **2.807236432**
10. Caso de prueba: (**5 / 2.03) + 14 * 9**. Resultado esperado: **128.4630542**
11. Caso de prueba: **8.32 / 2 – 1.2 * 2.31 * 0.59**. Resultado esperado: **2.52452**
12. Caso de prueba: **8.32 / (2 – 1.2 * (2.31 * 0.59))**. Resultado esperado: **22.82453638**
13. Caso de prueba: **8.32 * (1.29 / (12 * 3.4 – (9.12 / 8) * 4))** Rtdo. esperado: **0.29615894**
14. Caso de prueba: **8.04 * (6.32 / (0.23 * (4.01 / 6 * (2.59) + 3)))** Rtdo. esperado: **46.6975**

Operaciones científicas

15. Caso de prueba: **9.32^2**. Resultado esperado: **86.8624**
16. Caso de prueba: **√2.85**. Resultado esperado: **1.688194302**
17. Caso de prueba: **52!.** Resultado esperado: **8.06582x10^67**
18. Caso de prueba: **6.32^{-1}**. Resultado esperado: **0.1582278481**
19. Caso de prueba: **sin (9.6)**. Resultado esperado: **0.1502255891**
20. Caso de prueba: **cos (-56)**. Resultado esperado. **0.6374239897**
21. Caso de prueba: **ln (3.2)**. Resultado esperado. **1.16315081**
22. Caso de prueba: **log (56)**. Resultado esperado. **1.748188**
23. Caso de prueba: **$(-6.04)^5$.**Resultado esperado. **-7841.036**

Operaciones científicas + Jerarquía de operaciones

24. Caso de prueba: **5.61^2 / 1.08**. Resultado esperado: **29.14083333**
25. Caso de prueba: **√64 – 3.5 / 2**. Resultado esperado: **6.25**
26. Caso de prueba: **(√64 – 3.5) / 2**. Resultado esperado: **2.25**
27. Caso de prueba: **-2.35^{-1} + 3.25 – 3.21^2**. Resultado esperado: **-6.628568085**
28. Caso de prueba: **032^{-1} * 5.21 /√98 + 6.13 * 3.2 / 24.2**. Rdo. esperado: **2.455233124**
29. Caso de prueba: **032^{-1} * (5.21 /√98 + 6.13 * 3.2 / 24.2)**. Rdo. esperado: **4.177712463**
30. Caso de prueba: **032^{-1} * (5.21 /(√98 + 6.13) * 3.2) / 24.2**. Rdo. esperado: **0.1343081969**
31. Caso de prueba: **sin (25) / (2.41 – 0.98) * 3.2**. Resultado esperado: **0.856354534**

Funcionalidad extra

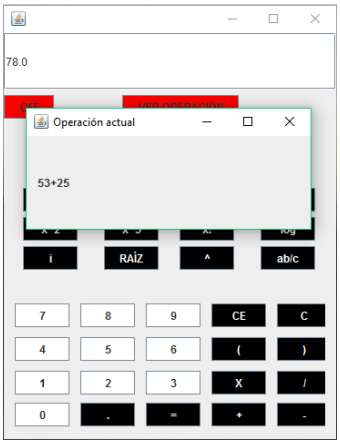
32. Caso de prueba: **Pulsar "AYUDA"**. Resultado esperado: **Se abre pestaña "AYUDA"**
33. Caso de prueba: **Escribir "5 * 3" y pulsar "CE".** Rdo. esperado: **Se borra la expresión**
34. Caso de prueba: **Escribir "5 * 3" y pulsar "C".** Rdo. esperado: **Queda "5 *".**

Errores

35. Caso de prueba: *** 8.41**. Resultado esperado: **8.41 (No deja escribir ("*").**
36. Caso de prueba: **5 * + 3**. Resultado esperado: **5 * 3 (No deja escribir "+").**
37. Caso de prueba: **2 +**. Resultado esperado: **Error.**
38. Caso de prueba: **2.34 * (5 + 2.12**. Resultado esperado: **Error.**
39. Caso de prueba: **-2.41 – 0.124) / 0.62**. Resultado esperado: **Error.**
40. Caso de prueba: **9*12.01 + ((1.23+0.59))**. Resultado esperado: **109.91**
41. Caso de prueba: **5.12 * 0..12**. Resultado esperado: **5.12 * 0.12 (No deja escribir ".").**
42. Caso de prueba: **√-64.** Resultado esperado: **Error.**
43. Caso de prueba: **. * 32.** Resultado esperado: **0**.

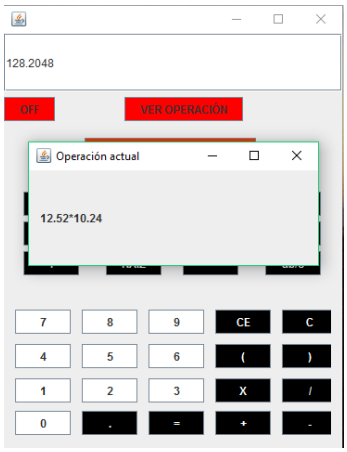
d)Capturas de pantalla de los casos de prueba

1. Caso de prueba: **53 + 25**



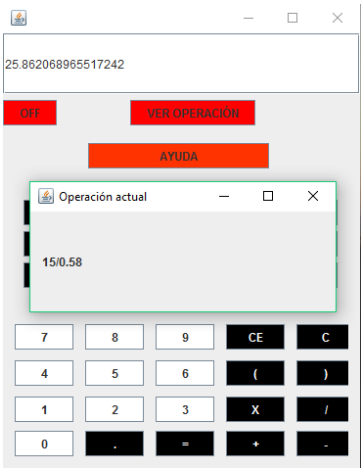
Resultado esperado: **78 |** Resultado obtenido: **78 |** Prueba: **Correcta**

2. Caso de prueba: **12.52 * 10.24.**



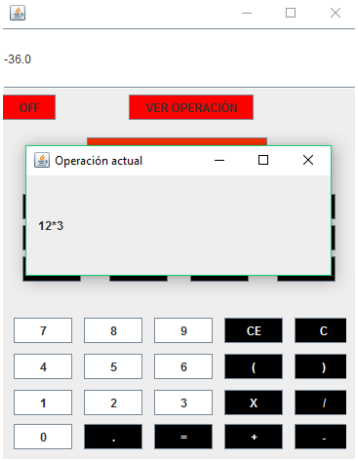
Resultado esperado: **128.2048 |** Resultado obtenido: **128.2048 |** Prueba: **Correcta**

3. Caso de prueba: **15 / 0.58**.



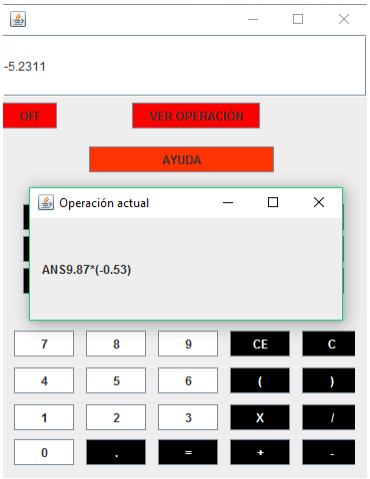
Rtdo. esperado: **25. 8620689551724 |** Rtdo obtenido: **25.8620689551724 |** Prueba: **Correcta**

4. Caso de prueba: **-12 * 3**.



Resultado esperado: **-36 |** Resultado obtenido: **-36 |** Prueba: **Correcta**

5. Caso de prueba: **9.87 * (-0.53)**.

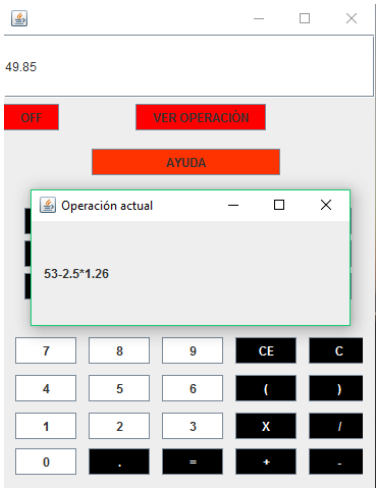


Resultado esperado: **-5.2311 |** Resultado obtenido: **-5.2311 |** Prueba: **Correcta**

6. Caso de prueba: **(53 – 2.5) * 1.26**.

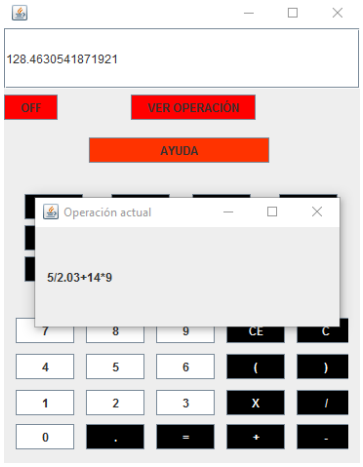
Resultado esperado: **63.63|** Resultado obtenido: **63.63 |** Prueba: **Correcta**

7. Caso de prueba: **53 – 2.5 * 1.26**.



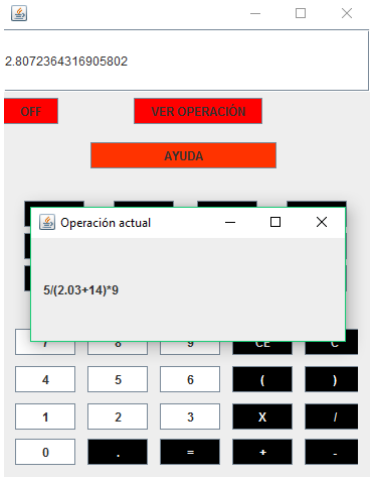
Resultado esperado: **49.85|** Resultado obtenido: **49.85 |** Prueba: **Correcta**

8. Caso de prueba: **5 / 2.03 + 14 * 9**.



Resultado esperado: **128.46305|** Resultado obtenido: **128.46305|** Prueba: **Correcta**

9. Caso de prueba: **5 / (2.03 + 14) * 9**.

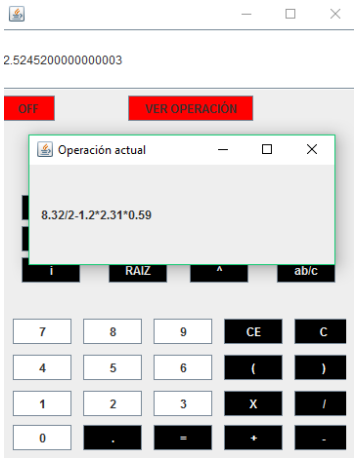


Resultado esperado: **2.80723643|** Resultado obtenido: **2.80723643|** Prueba: **Correcta**

10. Caso de prueba: **(5 / 2.03) + 14 * 9**.

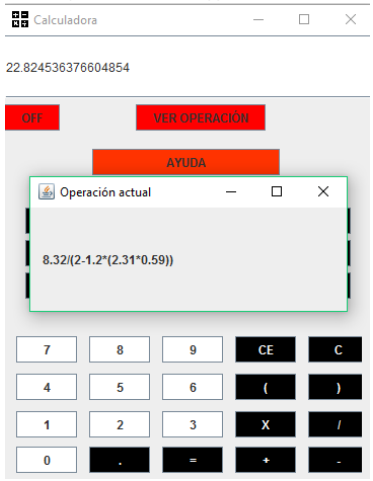
Resultado esperado: **128.463054|** Resultado obtenido: **128.463054|** Prueba: **Correcta**

11. Caso de prueba: **8.32 / 2 – 1.2 * 2.31 * 0.59**.



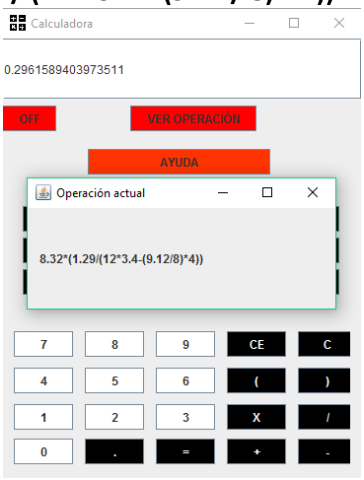
Resultado esperado: **2.52452|** Resultado obtenido: **2.52452|** Prueba: **Correcta**

12. Caso de prueba: **8.32 / (2 – 1.2 * (2.31 * 0.59))**.



Resultado esperado: **22.8245364|** Resultado obtenido: **22.8245364|** Prueba: **Correcta**

13. Caso de prueba: **8.32 * (1.29 / (12 * 3.4 – (9.12 / 8) * 4))**



Resultado esperado: **0.29615894|** Resultado obtenido: **0.29615894|** Prueba: **Correcta**

14. Caso de prueba: **8.04 * (6.32 / (0.23 * (4.01 / 6 * (2.59) + 3)))** Rtdo. esperado: **46.6975**

Resultado esperado: **46.6975|** Resultado obtenido: **46.6975|** Prueba: **Correcta**

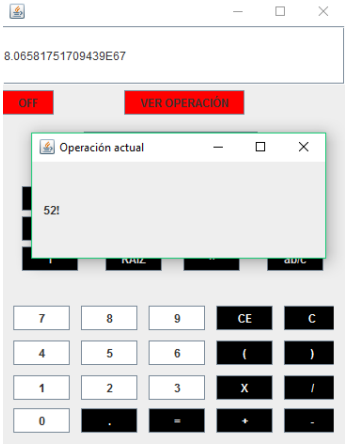
15. Caso de prueba: **9.32^2**.

Resultado esperado: **86.8624|** Resultado obtenido: **86.8624|** Prueba: **Correcta**

16. Caso de prueba: **√2.85**.

Resultado esperado: **1.6881943|** Resultado obtenido: **1.6881943|** Prueba: **Correcta**

17. Caso de prueba: **52!.**



Resultado esperado: **8.06582E67|** Resultado obtenido: **8.06582E67|** Prueba: **Correcta**

18. Caso de prueba: **6.32^{-1}**.

Resultado esperado: **0.15822785|** Resultado obtenido: **0.15822785|** Prueba: **Correcta**

19. Caso de prueba: **sin (9.6)**.

Resultado esperado: **0.1502256|** Resultado obtenido: **0.16676875|** Prueba: **Incorrecta**

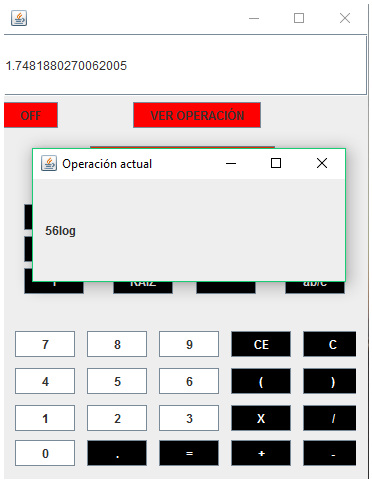
20. Caso de prueba: **cos (-56)**.

Resultado esperado: **0.63742399|** Resultado obtenido: **0.5591929|** Prueba: **Incorrecta**

21. Caso de prueba: **ln (3.2)**.

Resultado esperado: **1.16315081|** Resultado obtenido: **1.16315081|** Prueba: **Correcta**

22. Caso de prueba: **log (56)**.

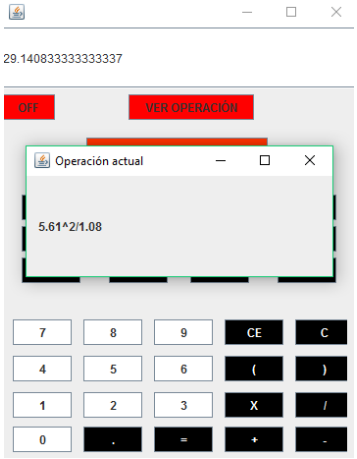


Resultado esperado: **1.748188|** Resultado obtenido: **1.748188|** Prueba: **Correcta**

23. Caso de prueba: **$(-6.04)^5$.**

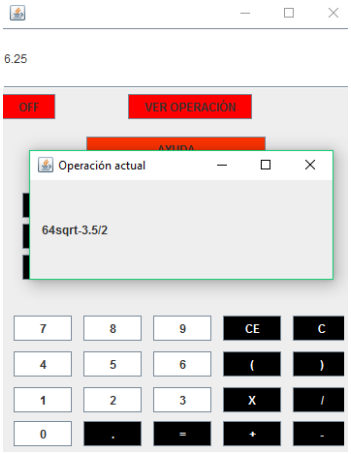
Resultado esperado: **-7841.036|** Resultado obtenido: **-8038.67912|** Prueba: **Incorrecta**

24. Caso de prueba: **5.61^2 / 1.08**.



Resultado esperado: **29.1408333|** Resultado obtenido: **29.1408333|** Prueba: **Correcta**

25. Caso de prueba: **√64 – 3.5 / 2**.

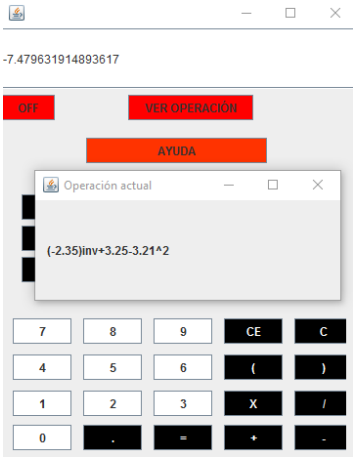


Resultado esperado: **6.25|** Resultado obtenido: **6.25|** Prueba: **Correcta**

26. Caso de prueba: **(√64 – 3.5) / 2**.

Resultado esperado: **2.25|** Resultado obtenido: **2.25|** Prueba: **Correcta**

27. Caso de prueba: **$-2.35^{-1} + 3.25 - 3.21^{2}$**.

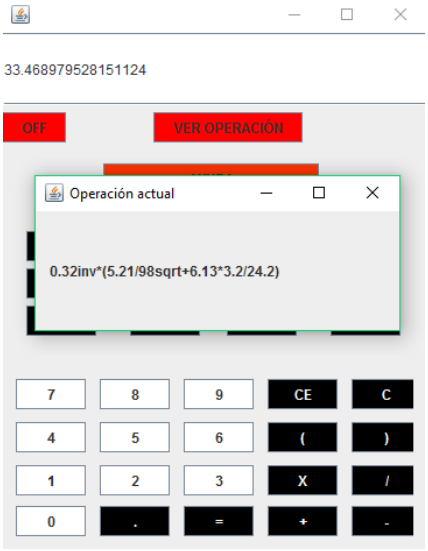


Resultado esperado: **-7.479632 |** Resultado obtenido: **-7.479632 |** Prueba: **Correcta**

28. Caso de prueba: **0.32^{-1} * 5.21 /√98 + 6.13 * 3.2 / 24.2**.

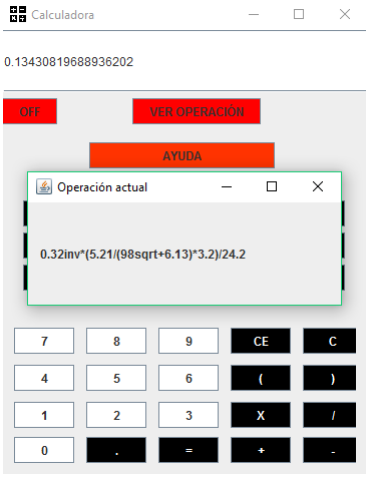
Resultado esperado: **2.4552331 |** Resultado obtenido: **2.4552331|** Prueba: **Correcta**

29. Caso de prueba: **0.32^{-1} * (5.21 /√98 + 6.13 * 3.2 / 24.2)**. Rdo. esperado: **4.177712463**



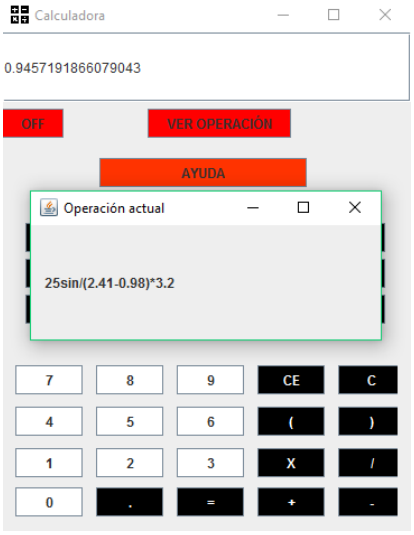
Resultado esperado: **4.17771246|** Resultado obtenido: **33.4689796|** Prueba: **Incorrecta**

30. Caso de prueba: **0.32^{-1} * (5.21 /(√98 + 6.13) * 3.2) / 24.2**



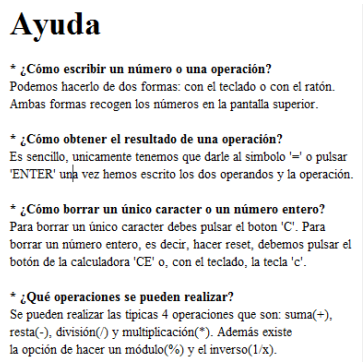
Resultado esperado: **0.1343082|** Resultado obtenido: **0.1343082|** Prueba: **Correcta**

31. Caso de prueba: **sin (25) / (2.41 – 0.98) * 3.2**.



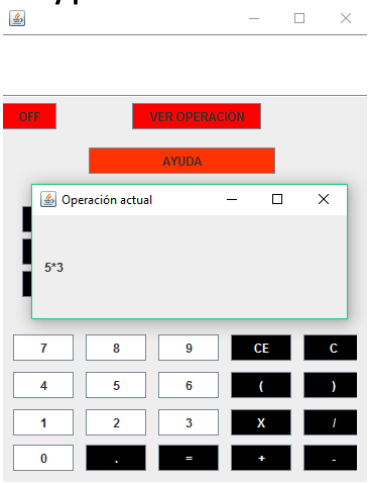
Resultado esperado: **0.856354534|** Resultado obtenido: **0.9457192|** Prueba: **Incorrecta**

32. Caso de prueba: **Pulsar "AYUDA"**.



Resultado esperado: **Se abre pestaña "AYUDA"|**
Resultado obtenido: **Se abre pestaña "AYUDA"|** Prueba: **Correcta**

33. Caso de prueba: **Escribir "5 * 3" y pulsar "CE".**

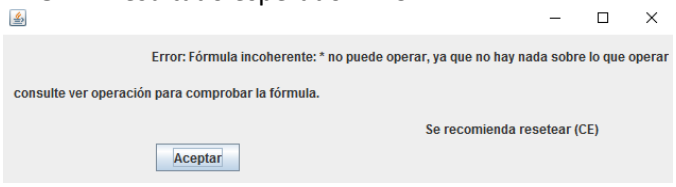


Resultado esperado: **Se borra la expresión |**
Resultado obtenido: **Se borra la expresión |** Prueba: **Correcta**

34. Caso de prueba: **Escribir “5 * 3” y pulsar “C”.**

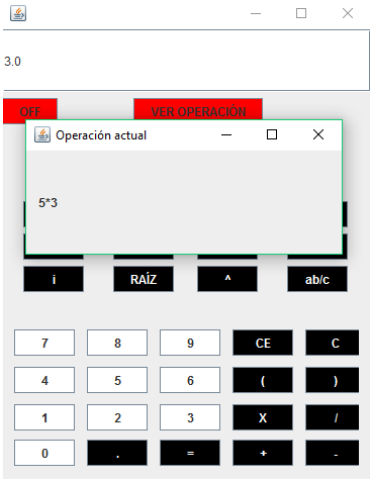
Resultado esperado: **Queda “5 *”.|** Resultado obtenido: **Queda “5 *”. |** Prueba: **Correcta**

35. Caso de prueba: *** 8.41**. Resultado esperado: **Error**



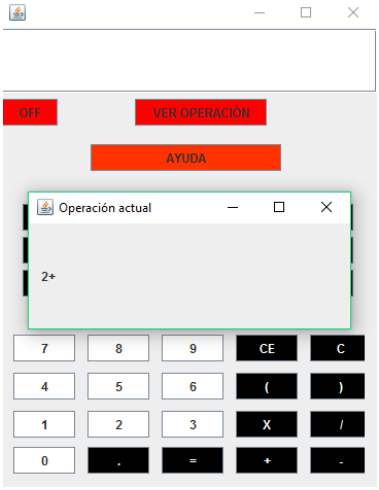
Resultado esperado: **Error.|** Resultado obtenido: **Error. |** Prueba: **Correcta**

36. Caso de prueba: **5 * + 3**.



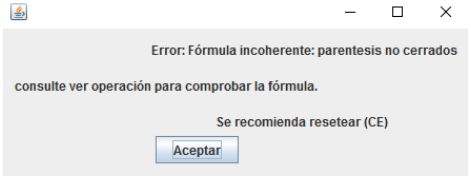
Resultado esperado: **5*3 (No deja escribir “+”).|** Resultado obtenido: **5*3. |** Prueba: **Correcta**

37. Caso de prueba: **2 +**.



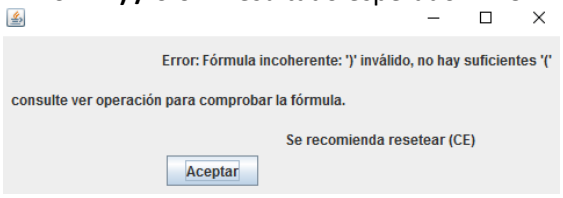
Resultado esperado: **Error.|** Resultado obtenido: **No aparece resultado. |** Prueba: **Incorrecta**

38. Caso de prueba: **2.34 * (5 + 2.12**. Resultado esperado: **Error.**



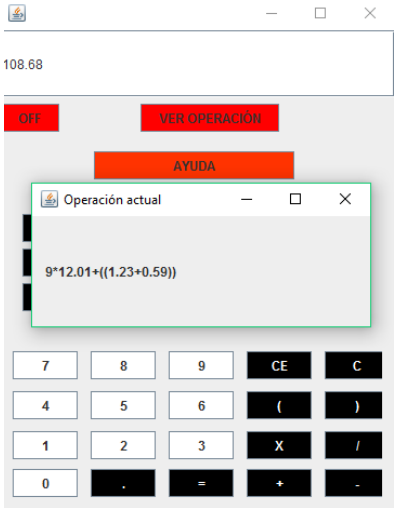
Resultado esperado: **Error.|** Resultado obtenido: **Error. |** Prueba: **Correcta**

39. Caso de prueba: **-2.41 – 0.124) / 0.62**. Resultado esperado: **Error.**



Resultado esperado: **Error.|** Resultado obtenido: **Error. |** Prueba: **Correcta**

40. Caso de prueba: **9*12.01 + ((1.23+0.59))**. Resultado esperado: **109.91**

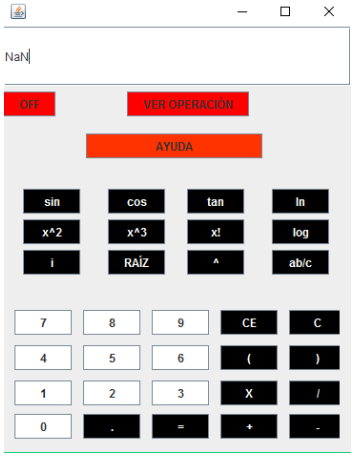


Resultado esperado: **109.91.|** Resultado obtenido: **108.68. |** Prueba: **Incorrecta**

41. Caso de prueba: **5.12 * 0..12**. Resultado esperado: **0.6144 (No deja escribir “.”).**

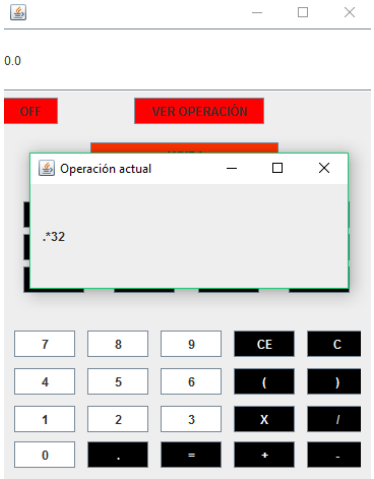
Resultado esperado: **0.6144.|** Resultado obtenido: **0.6144. |** Prueba: **Correcta**

42. Caso de prueba: **√-64.** Resultado esperado: **Error.**



Resultado esperado: **Error.|** Resultado obtenido: **Error. |** Prueba: **Correcta**

43. Caso de prueba: **. * 32.** Resultado esperado: **0**.



Resultado esperado: **0.|** Resultado obtenido: **0. |** Prueba: **Correcta**

RESULTADOS FINALES

Bloque	Correctas	Incorrectas
Básicas	5	0
Jerarquía	9	0
Científicas	6	3
Jerarquía + Científicas	6	2
Funcionalidad extra	3	0
Errores	7	2
TOTAL	**36**	**7**

Tras hacer un estudio en profundidad de la implementación hemos encontrado 7 errores sobre 43 pruebas. Es decir, aproximadamente el 19% de las pruebas realizadas han resultado erróneas. Algunos de estos errores tienen una fácil resolución. Los errores ocurridos han sido, en general, causa de mal redondeo en las operaciones científicas. Así pues, ha sido una prueba de errores satisfactoria puesto que hemos encontrado diversos errores que hay que solucionar.

Así pues, consideramos que la prueba de errores ha sido bien realizada. Tras esta prueba, el equipo debería ponerse a trabajar duramente para solucionarlos. Sin embargo, puesto que la fecha de entrega es próxima, no aseguramos solucionar todos los errores. Además, no tenemos conocimiento matemático suficiente como para precisar más las operaciones científicas. Otros errores, surgidos de no realizar bien el tratamiento de los errores en calculadora, son errores no muy importantes.

De manera general, podemos decir que los errores son menores, no son errores importantes que afecten en exceso a la funcionalidad de la calculadora. Por ello, nos sentimos satisfechos tanto con la funcionalidad de nuestro proyecto como por las pruebas que hemos realizado con el fin de comprobar esta funcionalidad.

5. CONCLUSIÓN

Esta práctica grupal tenía como objetivo el hecho de aprender a realizar de forma correcta todo el ciclo de vida de software. Así pues, en un grupo de 5 personas debíamos realizar una calculadora con jerarquía de operaciones, encargándonos de toda la gestión del proyecto.

Lo primero en realizar fue la planificación de está. Realizamos los diagramas pertinentes y definimos los requisitos que hemos considerado para una calculadora. De estos, algunos los hemos considerado imprescindibles y otros secundarios. Así pues, por ejemplo, la implementación de funciones sobre el teclado no se ha realizado, puesto que la hemos considerado secundaria y le hemos dado preferencia a otros requisitos. De esta manera, la calculadora recibe números y operaciones básicas mediante el teclado, pero no operaciones científicas, como logaritmos. Otro requisito que no hemos podido cumplir al final ha sido el diseño "responsive" de la calculadora, ya que esta no puede agrandarse. Una vez definidos los requisitos, comenzamos a identificar las tareas que debíamos realizar, y a asignarlas a cada uno de los integrantes del grupo.

Tras la planificación hemos comenzado a realizar el diseño de la calculadora mediante una serie de prototipos. Cuando todos los del grupo aceptamos los prototipos, comenzamos a realizar la implementación usando GitHub como repositorio. Nos surgieron una serie de errores, y al principio nos costó bastante entenderla. Al final, a base de pelearnos con la herramienta, conseguimos poder trabajar con ella de forma satisfactoria.

Al mismo tiempo, mientras se hacia la implementación, se iba rellenarlo el documento con toda la información correspondiente a la práctica. Este documento se puede considerar la parte global y más importante de la práctica, puesto que es donde tenemos almacenada toda la información correspondiente a la práctica y planificación.

Nos ha sido una práctica entretenida, y que nos ha servido para aprender a gestionar un proyecto software en grupo. Al principio nos fue un poco difícil organizarnos, puesto que no nos conocíamos y no habíamos trabajado nunca juntos. Tras el desarrollo de la práctica nos fuimos entendiendo mejor y conociéndonos, lo cual permitió llevar a cabo la práctica de forma correcta.

Hemos trabajado todos los integrantes del grupo, en distinta medida. De manera general, Sergio se ha encargado de la interfaz, parte de la documentación y estimación de tareas. Alberto de la implementación y diagrama HTA. Alberto San José del prototipo. Jesús de la mayor parte de la documentación, diagramas PERT y casos de uso. Y, por último Daniel, encargado de la presentación, la cual, por equivocación, no hemos considerado como tarea.

En general, no tenemos ningún comentario negativo acerca de cómo ha sido llevada la práctica por parte del profesorado, y como se nos ha sido explicada. Por otro lado, la consideramos una práctica bastante adecuada para la asignatura, puesto que tratamos una gran parte de puntos importantes de ella y nos sirve para entender conceptos teóricos, y, además, ponerlos en práctica.

En conclusión, nos ha sido una práctica que pensamos que nos puede servir en el futuro, que nos ha dado experiencia y nos ha ayudado a entender conceptos. Estamos de acuerdo en que está práctica se realice en futuros cursos en esta asignatura.