



Práctica

Ampliación de ingeniería del software

Grado en Ingeniería Informática (2015/2016)

Cristian Camarzana Romero

Andrea Delgado Pablo

Sergio Huertas Moratalla

Vanesa Maseli Martín

Beatriz Reyes Sánchez

# INDICE

1. Planificación
   1. Análisis de requisitos
   2. Previsiones y objetivos
   3. Análisis de tareas
2. Organización
   1. Tipo de equipo
   2. Tabla de tareas
   3. Diagrama de Pert
   4. Calendario
   5. Distribución de las tareas
3. Realización
   1. Diseño
      1. Diagrama de casos de uso
      2. Diagrama de clases
      3. Prototipo de baja fidelidad
      4. Prototipo de alta fidelidad
   2. Implementación
   3. Pruebas
4. Conclusiones obtenidas

# Planificación

## Análisis de requisitos

En este momento de la vida del software es necesario corregir y mejorar algún aspecto del producto. El cliente ha solicitado la modificación completa de la interfaz de usuario y la agregación de nuevas funcionalidades para una mejor experiencia de uso. A continuación analizaremos los distintos requisitos agrupados en requisitos funcionales y no funcionales que son necesarios cumplir:

* Requisitos funcionales.
  + **RF01**. Se debe poder seguir realizando las operaciones que se incluían en el código proporcionado al principio.
  + **RF02**. Añadir la funcionalidad de hacer operaciones con paréntesis.
  + **RF03**. Cambiar el signo de un número.
  + **RF04**. Poder eliminar una sola cifra de un número.
  + **RF05**. Limpiar la pantalla, eliminado todos los números y signos de operación que aparecen.
  + **RF06**. Permitir borrar un número.
  + **RF07**. Realizar una interfaz usable y atractiva para el usuario, basándose en la calculadora que incluye Windows.
* Requisitos no funcionales.
  + **RNF01**. El lenguaje de programación usado para desarrollar el software debe ser java.
  + **RNF02**. Se debe crear un ejecutable que pueda ser utilizado en cualquier sistema operativo.

## Previsión y objetivos

En esta práctica, nos han proporcionado un código Java cuyo funcionamiento consiste en realizar operaciones básicas con una calculadora. Sin embargo el código se ha quedado obsoleto para las nuevas funcionalidades que se necesitan.

En primer lugar nos hemos encontrado una interfaz gráfica que no tiene el aspecto de una calculadora por lo que hemos decidido eliminarla totalmente y crear una nueva.

En segundo lugar, nuestro propósito es añadir una nueva funcionalidad de manera que la calculadora tenga en cuenta las operaciones con paréntesis respectando la jerarquía.

Creemos que podremos realizar estos cambios con éxito en el tiempo que nos exige el cliente de manera que funcione correctamente y tenga la apariencia adecuada.

## Análisis de tareas

Una vez analizados los requisitos que se nos piden añadir o modificar toca identificar las tareas y analizarlas para más tarde poder asignárselas a los diferentes miembros del equipo.

Las tareas, donde cada letra entre paréntesis que le acompaña será su identificador en el posterior diagrama, son las siguientes:

* **Análisis de requisitos (A).** Como ya habíamos tratado con anterioridad lo primero que se debe hacer es analizar los requisitos que se nos piden para poder saber que debemos hacer.
* **Realizar la interfaz (B).** Unos de los requisitos que se nos pedían era cambiar la interfaz de la aplicación para que tuviera una apariencia que se acercara más a una calculadora normal. Por ello, esta es una de las tareas importantes del proyecto.
* **Realizar pruebas sobre la interfaz y corregir fallos (C).** Una vez realizada la interfaz de usuario es necesario comprobar que funciona correctamente y que no existen errores, por lo que se deben hacer una serie de pruebas que certifiquen eso. En caso de encontrar algún error deberá ser corregido.
* **Añadir funcionalidad (D).** El otro requisito que debíamos cumplir era añadir funcionalidad a la aplicación. Para ello, debemos modificar el código para permitir que esta recoja la opción de hacer una cuenta en la que se incluyan paréntesis.
* **Realizar pruebas sobre la funcionalidad y corregir fallos (E).** Al igual que ocurría con la interfaz, en la funcionalidad también es necesario comprobar que no existen errores y que, por lo tanto, funciona como debería. Si durante las pruebas detectamos un error deberá ser corregido antes de continuar con el proyecto.
* **Unificar funcionalidad e interfaz (F).** Entre las tareas anteriores se encontraban la de modificar la interfaz y la de modificar el código, ambas cosas por separado, por lo que una tarea necesaria es esta, que nos permitirá fusionar ambas tareas para que la aplicación tome la forma deseada por el usuario.
* **Realizar pruebas y modificar posibles errores (G).** Una vez llegados a esta tarea, la aplicación tendrá ya la apariencia real que nos pidió el usuario, por lo que es necesario realizar ciertas pruebas para detectar distintos errores que pueda haber. En caso de encontrar algún error deberíamos corregirlos.
* **Realización de la documentación (H).** Esta tarea es fundamental en el desarrollo del proyecto, ya que encarga de la explicación de todos los aspectos incluidos en él. Por ello, debe realizarse durante todo el transcurso de este, ya que se deben añadir información de todas las tareas que se realizan.
* **Preparación de la exposición (I).** Una vez realizada toda la documentación necesaria, se debe realizar la presentación basada en la anterior que será mostrada al cliente para explicar todos los pasos que se han seguido.

# Organización

## Tipo de equipo

Este caso utilizaremos un organigrama de equipo denominado descentralizado democrático, en el que no existe un líder claro y las decisiones se toman por consenso entre todos los miembros del grupo.

Esto es así debido a que todos tenemos más o menos los mismos conocimientos tanto informáticos como de liderazgo por lo que no podríamos nombrar a uno o más líderes que dirigieran el proyecto.

En cuanto los paradigmas de organización hemos elegido el denominado como paradigma abierto, en el que existe comunicación y colaboración entre todos los integrantes del equipo y las decisiones son consensuadas por todos ellos, al igual que hemos mencionado en el organigrama elegido.

Las razones por las que hemos decido este paradigma y no otro son más o menos que las anteriores, ya que no nombraremos a un líder por lo que se deberá hablar con todo el grupo las decisiones individuales que se quieran tomar.

## Tabla de tareas

En esta ocasión nos toca asignar el tiempo que se necesitará para realizar cada una de las tareas que hemos identificado anteriormente y su orden dentro del tiempo total del proyecto.

Para ello, en la siguiente tabla mostraremos la precedencia entre las tareas, el tiempo que consume cada tarea y los recursos humanos asociados a cada una de ellas.

|  |  |  |  |
| --- | --- | --- | --- |
| Tarea | Precedencia | Tiempo | Recursos humanos |
| A | - | 1 |  |
| B | A | 2 |  |
| C | B | 1 |  |
| D | A | 6 |  |
| E | D | 2 |  |
| F | C, E | 1 |  |
| G | F | 3 |  |
| H | - | 7 |  |
| I | G, H | 3 |  |

## Diagrama de Pert

En el apartado anterior hemos especificado las tareas que teníamos, con sus precedencias y su tiempo asociado. Por ello, lo siguiente que se debe hacer es dibujar el diagrama de Pert, ya que se ve más gráficamente todo el contenido. Dicho diagrama se muestra a continuación.

2

E

1

C

0

02

1

7

H

A

B

2

D

6

F

1

G

3

IG

3

13

13

16

16

1

11

3

8

7

7

9

9

9

9

10

10

También es necesario realizar la tabla de tiempo y holguras, la cual se muestra más abajo, asociada al diagrama anterior para tener todos los datos recogidos.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TEi | TLi | TEj | TLj | tij | HT | HL | HI |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 1 | 1 | 1 |  |  |  |
| B | 1 | 1 | 3 | 8 | 2 |  |  |  |
| C | 2 | 8 | 9 | 9 | 1 |  |  |  |
| D | 1 | 1 | 7 | 7 | 6 |  |  |  |
| E | 7 | 7 | 9 | 9 | 2 |  |  |  |
| F | 9 | 9 | 10 | 10 | 1 |  |  |  |
| G | 10 | 10 | 13 | 13 | 3 |  |  |  |
| H | 0 | 0 | 13 | 13 | 7 |  |  |  |
| I | 13 | 13 | 16 | 16 | 3 |  |  |  |

## Calendario

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Marzo | | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  | Análisis requisitos |  |  |  |  |
|  |  |  |  |  |  |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Documentación | | | |  |  |  |
| AF. |  |  |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  | Realizar interfaz | |  |  |  |  |
| Añadir funcionalidad | | | | |  |  |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 28 | 29 | 30 | 31 |  |  |  |
|  | Doc. | Pruebas interfaz |  |  |  |  |
|  | Pruebas funcionalidad | |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Abril | | | | | | |
|  |  |  |  | 1 | 2 | 3 |
|  |  |  |  | Doc. |  |  |
|  |  |  |  |  |  |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Unión  I-F | Pruebas unión interfaz-funcionalidad | | | Doc. |  |  |
|  |  |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | Preparación de la exposición | | |  |  |  |
|  |  |  |  |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Entrega |  |  | Exposición |  |  |  |
|  |  |  |  |  |
| 25 | 26 | 27 | 28 | 29 | 30 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## Distribución de las tareas

En los apartados anteriores identificamos las tareas y analizamos su duración, por lo que ahora es necesario asignar cada tarea a una persona que se encargará de realizarla. En la tabla siguiente se recogen estos datos.

|  |  |
| --- | --- |
| Tarea | Persona encargada |
| Análisis de requisitos |  |
| Realizar la interfaz |  |
| Añadir funcionalidad |  |
| Unificar funcionalidad e interfaz |  |
| Realizar pruebas y modificar posibles errores |  |
| Realización de la documentación |  |
| Realización de la presentación |  |

# Realización

## Diseño

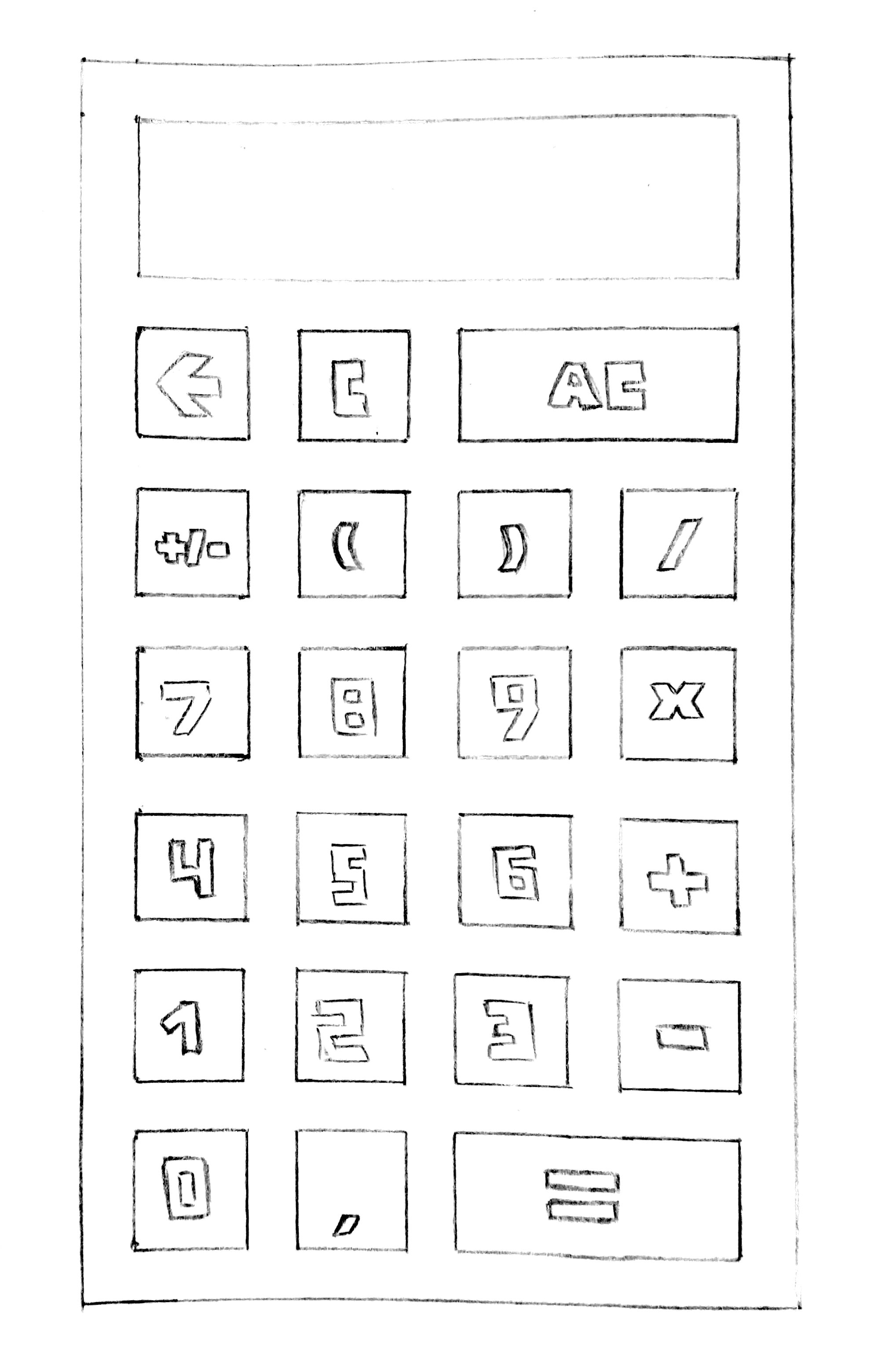
### Diagrama de casos de uso

### Diagrama de clases

|  |
| --- |
| Calculadora AIS |
| * E :Stack * E1: Stack * P: Stack * P1: Stack * igual\_reset: Boolean * decimal: Boolean * s: String * numero\_parents: Integer |
| * evaluar(String, String, String):Double * comprobar(String): Boolean * operar() * (métodos asociados a los botones) |

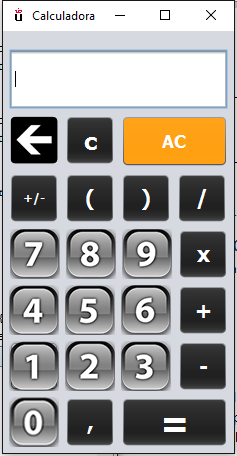
### Prototipo de baja fidelidad

Aquí mostraremos el primer diseño de nuestra interfaz gráfica que pensamos en realizar.



### Prototipo de alta fidelidad

En este apartado mostramos el diseño final de la interfaz gráfica de nuestra aplicación, dónde tendremos al igual que en el caso anterior una única pantalla.



## Implementación

Una vez terminada la fase de diseño, nos toca implementar el código de la aplicación. En este caso, hay dos tareas diferenciadas en la parte de implementación que son el diseño de la interfaz y la corrección de la funcionalidad de la calculadora.

En primer lugar, la interfaz de usuario está compuesta por 22 botones con diferentes funcionalidades cada uno y por una pantalla que muestra los botones que han sido pulsados. A continuación desglosaremos los botones por grupos.

* Botones “0”-“9”. Principales protagonistas, utilizados para poder introducir los números por pantalla. Además al pinchar en un botón este se añade en una pila de números que luego será necesaria para poder resolver las operaciones. Estos tienen un color distinto para que sean fácilmente reconocibles por el usuario.
* Botones “/”, “x”, “+” y “-“. Junto con los anteriores son vitales para que esta aplicación pueda tener el nombre de calculadora, ya que su funcionalidad es mostrar las operaciones por pantalla. También, como en el caso anterior, cada operador se introduce en una pila de operadores.
* Botón “,”. Este botón es necesario para poder introducir números con decimales y por lo tanto poderlos incluir en nuestras operaciones.
* Botón “=”. En la forma en la que está implementada esta aplicación, este botón cobra gran importancia, ya que se encarga de llamar a la función encargada de resolver toda la operación que se va añadiendo en la pantalla.
* Botón “+/-“. Dicho botón se ha añadido a la calculadora para poder realizar operaciones con números positivos o negativos indistintamente.
* Botones “(” y “)”. Estos botones permiten las operaciones con paréntesis, requisito fundamental que se nos pedía al comienzo. Al igual que ocurría con los símbolos de las operaciones, estos son introducidos en la pila de operadores.
* Botón “<-“. Este botón junto a los dos siguientes son utilizados para borrar algún símbolo de la pantalla. En este caso, al apretar este botón únicamente eliminaremos el último componente de un número.
* Botón “C”. Como hemos explicado anteriormente, este botón también se utiliza para borrar, pero aquí borraríamos todo el número entero.
* Botón “AC”. Su funcionalidad es borrar toda la pantalla, para poder comenzar una operación distinta a la anterior.

En segundo lugar nos queda explicar la parte que relacionada con el código usado para aumentar la funcionalidad de la calculadora que se nos daba como partida. En este caso, nos ha sido suficiente con implementar 3 métodos, los cuales se explican más a delante.

* Evaluar. Este método ha sido implementado para poder realizar las operaciones que se nos muestran por pantalla.

Al estar guardados todos los símbolos como cadenas de caracteres, este método recibe tres variables de tipo string cada una, dos de ellas sacadas de la pila de números, por lo que representaran números y otro sacado de la pila de operadores por lo que será un operador.

Este método es muy sencillo, convierte los dos números en double y dependiendo del operador que haya recibido realiza una operación u otra.

El resultado de la operación es un double por lo que el método devolverá este tipo de variable.

* Comprobar. Este método es utilizado para detectar los errores en cuanto a la entrada y además mostrar un mensaje por pantalla dando información sobre el mismo.

Para saber si existe o no, necesita recibir como parámetro la cadena de caracteres que en ese momento se está mostrando por pantalla.

Como ya hemos dicho, sirve para detectar errores por lo que comprueba todos los posibles errores que creemos que se podrían dar y muestra el mensaje correspondiente a ese error para que el usuario pueda modificarlo.

Para que el método operar pueda o no parar la ejecución o no en caso de existir algún error, este método devuelve un tipo boolean, que será true en caso de no encontrar un error y false en caso contrario.

* Operar. Es el método principal de la aplicación, ya que es el encargado de llamar a las demás funciones cuando son necesarias. Este es el método que se llama cuando se presiona el botón “=”.

En este caso, el método no recibe nada, ya que todo lo que necesita son las pilas de números y operadores, las cuales son variables globales.

La función de este método es la de comprobar que hay en ambas pilas e ir extrayendo los números necesarios para pasárselos al método evaluar y obtener resultados. Por lo tanto, es el responsable del control del orden de las operaciones.

Cuando acaba con todas las operaciones, el número resultante lo deja en la pila de números por si se quiere seguir haciendo operaciones con ese resultado. Por este motivo, este método no devuelve ningún valor.

## Pruebas

# Control de versiones

## 4.1. Estrategia utilizada

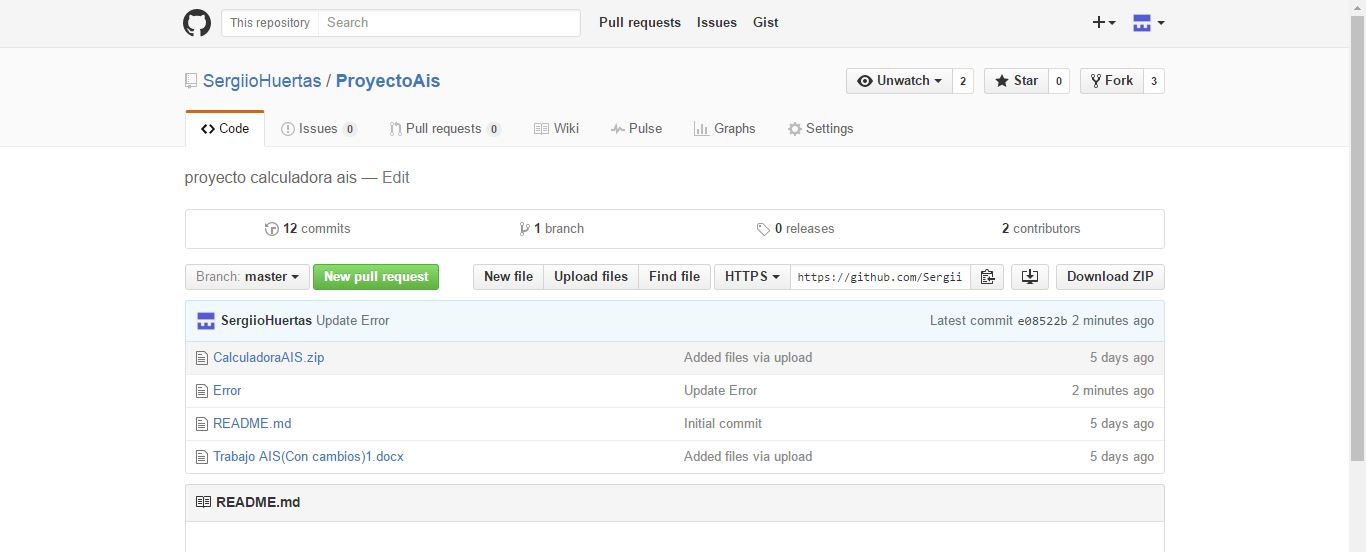
Para la realización del control de versiones vamos a utilizar GitHub, que utiliza una estrategia de versionado de desarrollo paralelo. Consiste en que si en un proyecto trabajan varios equipos crearemos una rama para cada uno de ellos, de esta forma tendremos el código estable e integrado de las tareas de desarrollo.

Se trata de un repositorio distribuido dónde los clientes aparte de descargarse la última versión de los archivos también replican completamente el repositorio. Por lo tanto si hubiese algún fallo en el servidor el proyecto podría restaurarse a partir de un repositorio de cualquier cliente ya que todos los nodos que manejan información pueden actuar como cliente o servidor.

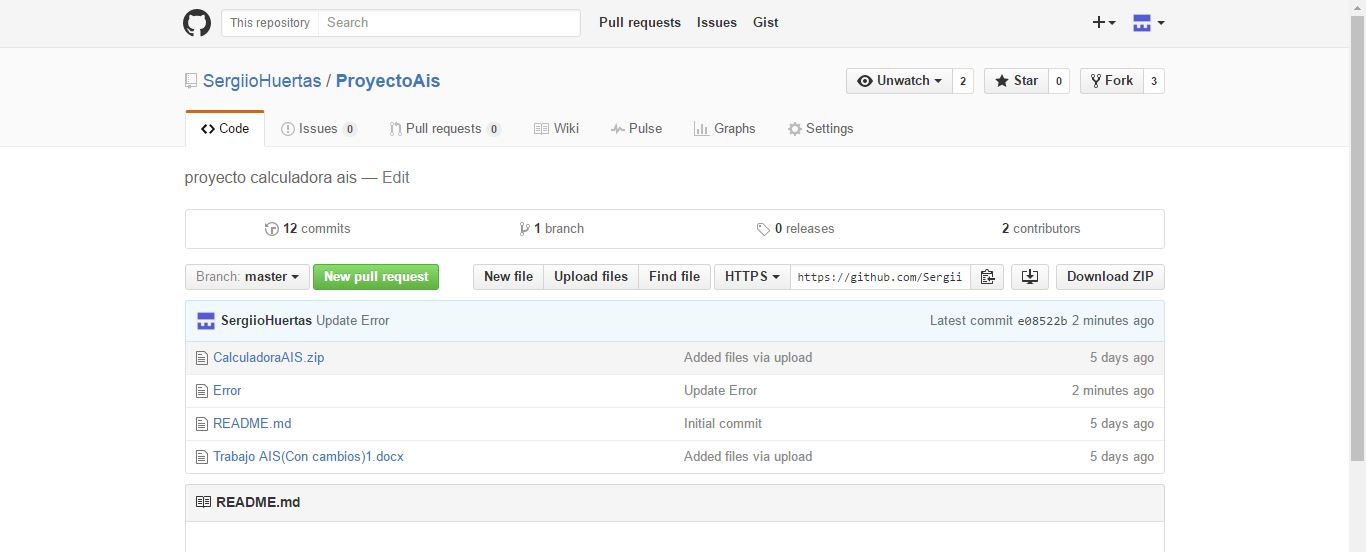
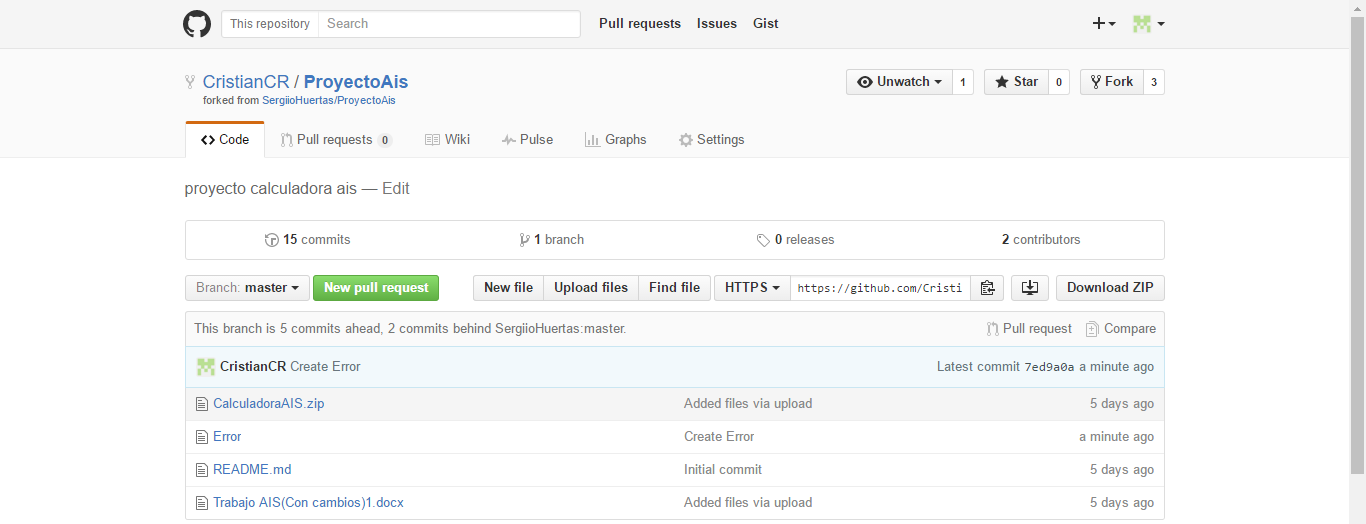
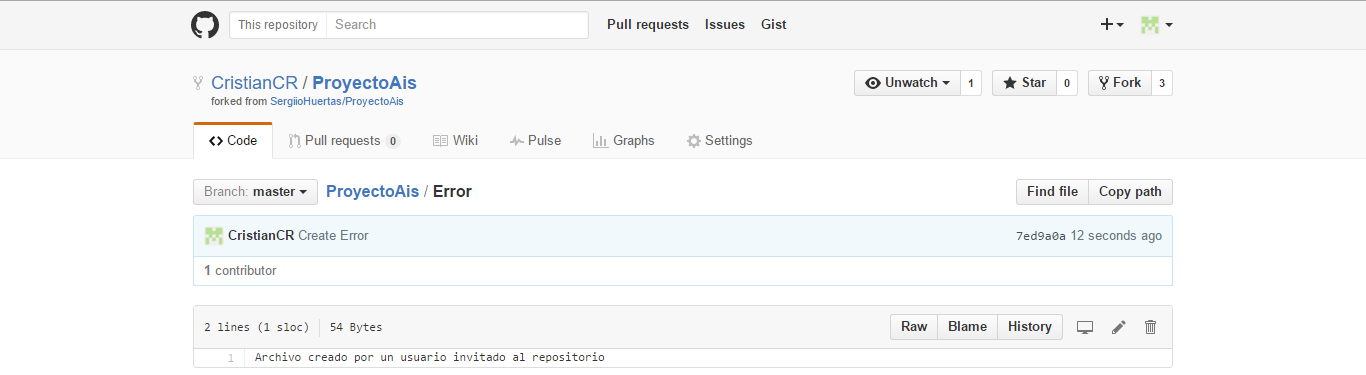
## 4.2. Errores producidos

El primer error que nos hemos encontrado ha sido en el momento en el que dos personas del proyecto han cambiado un proyecto que tenía el mismo nombre pero distinto contenido.

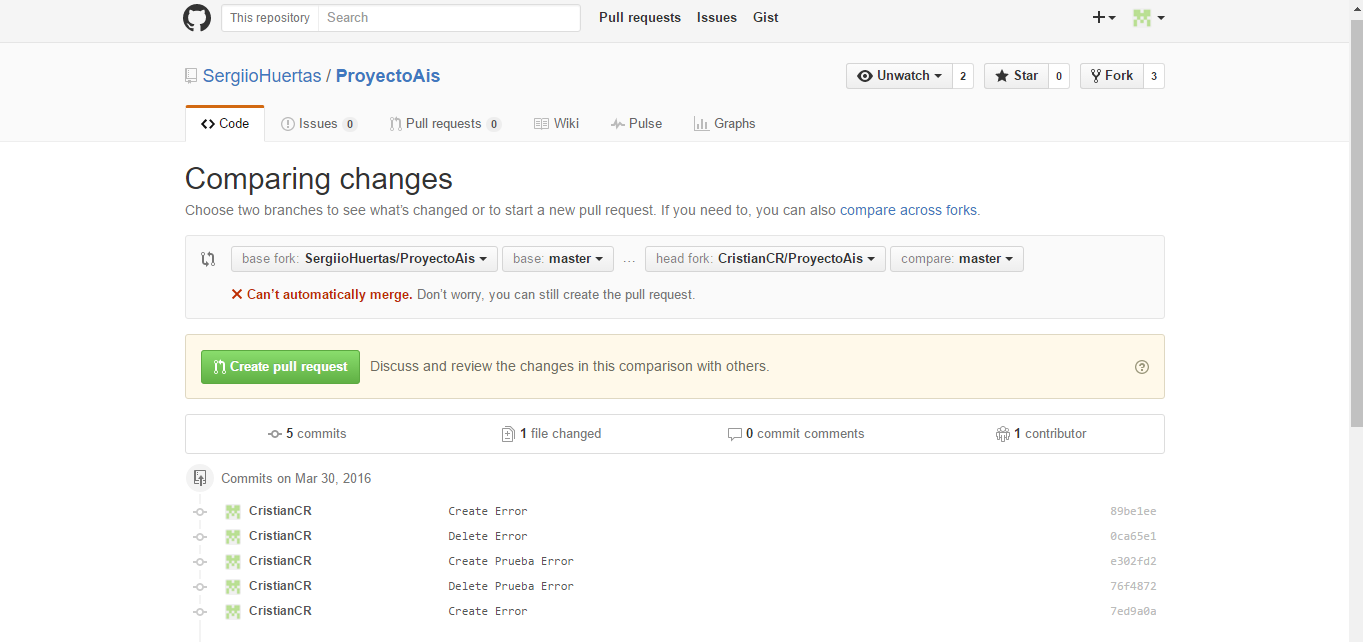
En primer lugar el administrador y el cliente crean cada uno un documento distintos pero con el mismo nombre en sus repositorios.



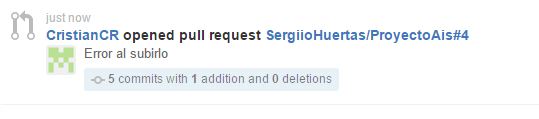
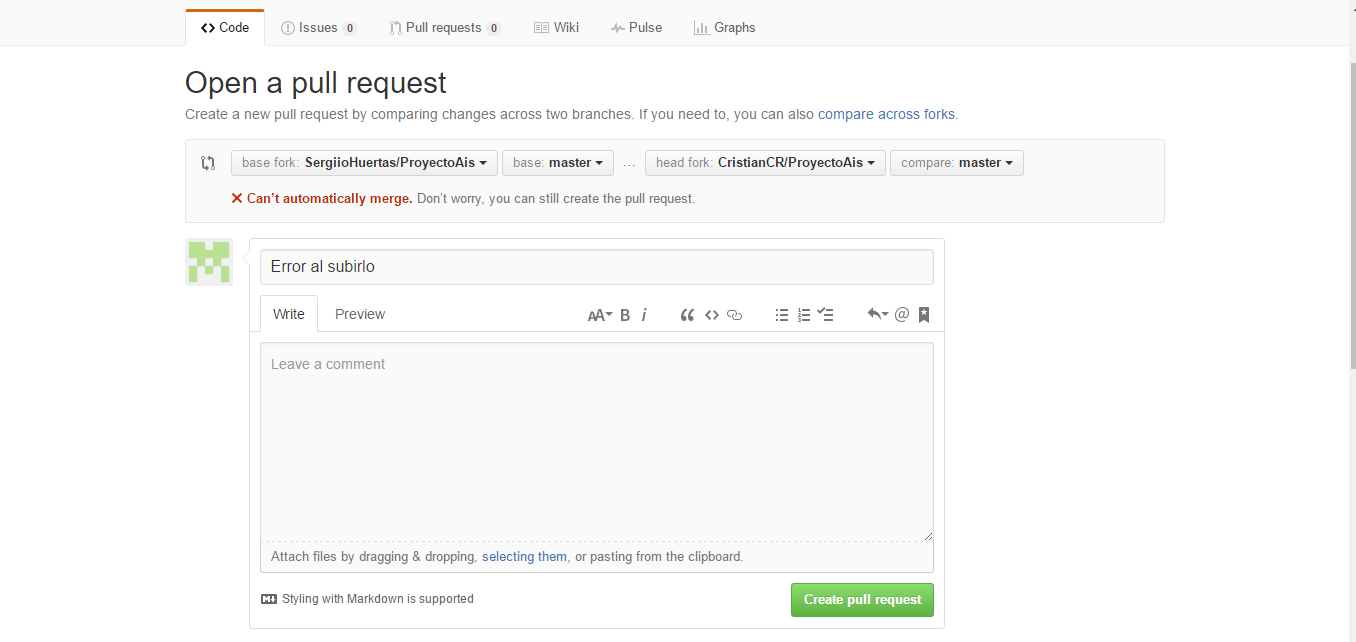
En el repositorio de cada uno aparecerá el documento que han añadido en su repositorio, pero al intentar el usuario subirlo al repositorio principal este tendrá un conflicto con el que ha creado el administrador en su repositorio.



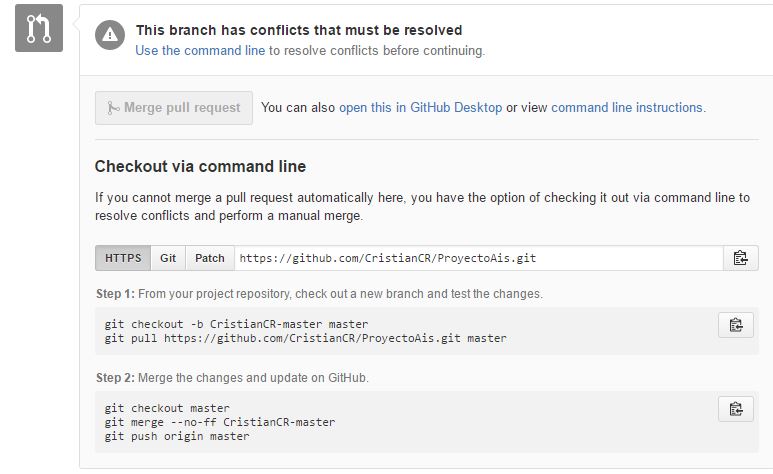
A continuación el cliente subirá el archivo al repositorio principal y esto provocará que se produzca el error notificando al administrador del repositorio.



El cliente podrá abrir una disputa con el administrador para poder buscar una solución al error. Al administrador le aparecerá una notificación de la disputa.



Por último el administrador podrá solucionar el problema del conflicto mediante la línea de comandos.



Planificación. Desviaciones-por que

Control de versiones. Conflictos-como solucionados