INFORME DE PROYECTO DE GRADO

# RESOLUCIÓN DE ECUACIONES POR MÉTODOS ARITMÉTICOS EN ENSEÑANZA MEDIA


Martín Poli - Camila Rojí

**Tutores**

Sylvia da Rosa - Federico Gómez

**Clientes**

Nora Ravaioli - Teresa Pérez



UdelaR-InCo Montevideo, Uruguay

Agosto 2017

# Resumen

En este proyecto se desarrolló una aplicación cuyo objetivo es servir de apoyo a profesores de matemática y estudiantes de la enseñanza media, en la temática relacionada con ecuaciones y su resolución. Se diseñó y elaboró de acuerdo con especificaciones provistas por las profesoras de Matemática en Educación Secundaria y de Didáctica de la Matemática en formación docente, Nora Ravaioli y Teresa Pérez.

Las secciones de este informe se organizan como sigue: en la sección 1, se presenta una Introducción con los antecedentes y la motivación del software solicitado, y, en la siguiente sección, se presentan los requisitos de la aplicación y la sintaxis de las ecuaciones que el software debe permitir crear y resolver.

Luego, en la tercer sección, se comentan los principales problemas encontrados en el desarrollo de la aplicación. Estos se describen divididos por problemas de presentación (Front-end) y problemas de la lógica de resolución de ecuaciones (Back-end), a su vez, en esta sección se explican los distintos algoritmos realizados para resolver estos problemas.

En la cuarta sección, se describe el diseño realizado para desarrollar la aplicación, donde se presenta la descomposición en sub-sistemas, el diagrama de distribución, los modelos de Entidad-Relación para las distintas bases de datos y el diseño realizado para la interfaz de usuario.

En el siguiente capítulo, se describen las tecnologías elegidas, los motores de base de datos y el lenguaje utilizado para desarrollar la aplicación. En esta sección también se indica dónde se encuentran disponibles los instaladores para descargar la aplicación y el manual de usuario que explica sus funcionalidades.

En la siguiente sección, se presentan las distintas etapas de testing realizadas para asegurar el cumplimiento de requisitos especificados por el cliente y la conformidad del mismo. En el último capítulo, se presentan las conclusiones del proyecto, así como recomendaciones de futuras ampliaciones que se pueden realizar.

# Índice

# 1. Introducción

En la actualidad la presentación formal de la ecuación se realiza en segundo año de educación media básica, donde es común iniciar el estudio con los mecanismos de resolución de ecuaciones de primer grado y progresivamente ir aumentando el grado de dificultad, basándose en los distintos tipos de ecuaciones (lineal, cuadrática, cúbica, racional, irracional, etc). Las profesoras de Matemática en Educación Secundaria y de Didáctica de la Matemática en formación docente, Nora Ravaioli y Teresa Pérez, en adelante "el cliente", creen que esa introducción prematura de técnicas de resolución en ecuaciones de primer grado, con el objetivo de automatizar las "reglas de pasaje", genera pasividad cognitiva, es decir, el estudiante al ver una ecuación, comienza a aplicar mecánicamente reglas para despejar la variable, sin tener claro los criterios de prioridad y sin comprender que el objetivo es hallar el valor de la incógnita que verifica la ecuación. Se plantearon entonces investigar si un primer acercamiento a la resolución de las ecuaciones utilizando estrategias aritméticas, favorece al desarrollo de los procedimientos de resolución a partir de la comprensión conceptual. Para ello realizaron un estudio con estudiantes de primero y de segundo del ciclo básico de enseñanza media, en el cual, las profesoras introducen al grupo de estudiantes de primero en cuestiones relativas a contenidos matemáticos y aspectos operativos, para luego guiarlos en la resolución de ecuaciones por métodos artiméticos utilizando distintas aplicaciones. Al mismo tiempo, realizan una prueba de control en el grupo de segundo año. Al año siguiente, esa misma prueba es propuesta a los estudiantes del grupo de primero del año anterior, lo que permite a las profesoras obtener datos del desempeño de estudiantes que trabajaron utilizando los métodos aritméticos y de estudiantes que han sido introducidos al tema ecuaciones de forma tradicional.

El análisis de los resultados de este estudio, les permitió concluir que la utilización de métodos aritméticos de resolución de ecuaciones en el inicio del trabajo algebraico, parecería favorecer la transición del pensamiento aritmético al algebraico, y la consolidación de la noción de ecuación. Por otro lado, se pudo constatar que la utilización de las aplicaciones permitió a los estudiantes trabajar a su ritmo, en forma autónoma, sin depender exclusivamente del docente para verificar el acierto o el error de sus procedimientos, y también que lograban resolver una mayor cantidad de ecuaciones.

Al mismo tiempo, las profesoras pudieron hacer una valoración de las distintas aplicaciones utilizadas, detectando fortalezas y debilidades y generando la idea de construir un software específico que sintetizara las primeras y superara las segundas. A continuación se describen algunas de las aplicaciones examinadas.

## 1.1. Aplicaciones examinadas

Las aplicaciones analizadas por las profesoras fueron desarrolladas por el Instituto Freudenhtal de la Universidad de Utrecht (Holanda) y otras por la universidad de Utah (Estados Unidos). En ellas las soluciones de las ecuaciones siempre son propuestas por los estudiantes, mientras que las aplicaciones cumplen el rol de verificar los resultados propuestos, utilizando estrategias

algebraicas para la resolución de ecuaciones.

Las aplicaciones analizadas fueron las siguientes:

### 1.1.1. True makers

En esta aplicación se lista un conjunto de ecuaciones con una sola aparición de la variable $x$ en la que se tiene que ingresar el valor de $x$ que verifica la ecuación.

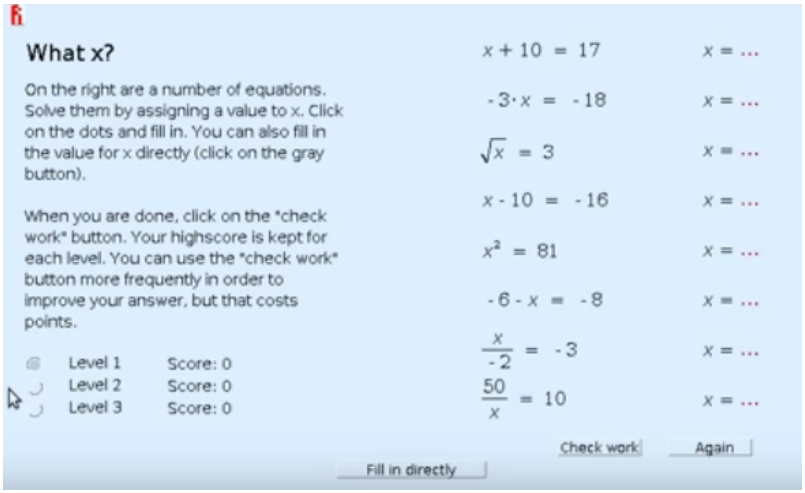



Figura 1: Aplicación True Makers

Esta aplicación tiene la ventaja de ser simple de usar, donde las ecuaciones se resuelven exclusivamente utilizando estrategias aritméticas. La limitación que las profesoras encontraron es que acepta expresiones de la forma $-5^2 = 25$ como correctas sin necesidad del uso de paréntesis, y además en las ecuaciones del tipo $x^2 = 25$ acepta la solución 5 o la -5, pero no permite ver el conjunto solución.

### 1.1.2. Cover up o recubrimiento

Esta aplicación introduce un método de resolución de ecuaciones que permite registrar los pasos del razonamiento al resolver las ecuaciones. Las profesoras denotaron esta característica como muy ventajosa debido a que, en general, los estudiantes tienen dificultades para explicitar su razonamiento, especialmente cuando los procedimientos son intuitivos como lo son los aritméticos.

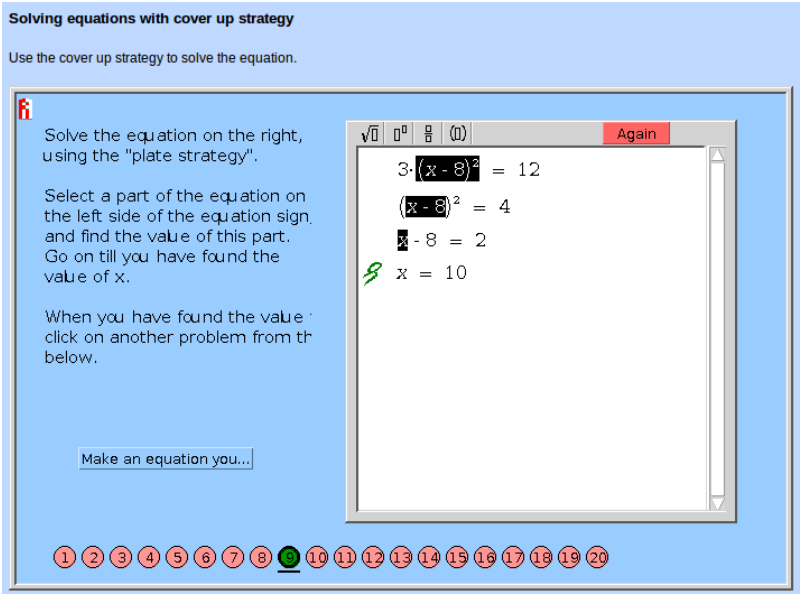



Figura 2: Aplicación Cover up

Otra ventaja que ellas consideraron importante es que esta aplicación permite el trabajo con ecuaciones polinómicas de cualquier grado así como también con ecuaciones no polinómicas, con la única condición de que la variable aparezca una sola vez. Una desventaja que encontraron, es que solamente permite resolver ecuaciones que contengan la incógnita en el lado izquierdo de la igualdad. Otra desventaja, es que al igual que en la aplicación *True Makers*, en ecuaciones del tipo $x^2 = 25$ se acepta el resultado 5 o -5 sin permitir visualizar el conjunto solución. Como se puede ver en la figura, la aplicación aceptó la solución 10, sin considerar que el 6 también es solución a la ecuación.

### 1.1.3. Balanza Algebraica

Esta aplicación permite la resolución de ecuaciones lineales haciendo uso de una balanza en la cual, se representa la ecuación colocando una determinada cantidad de $x$ y $1s$ en dos platillos, de forma de obtener la misma igualdad que la planteada. Para resolver la ecuación se debe sumar, restar, multiplicar o dividir de ambos lados hasta tener en un lado de la balanza solo la $x$ y en el otro lado una determinada cantidad de $1s$. Al mismo tiempo, cada vez que se realiza una operación en ambos miembros de la ecuación, la aplicación escribe el resultado obtenido.

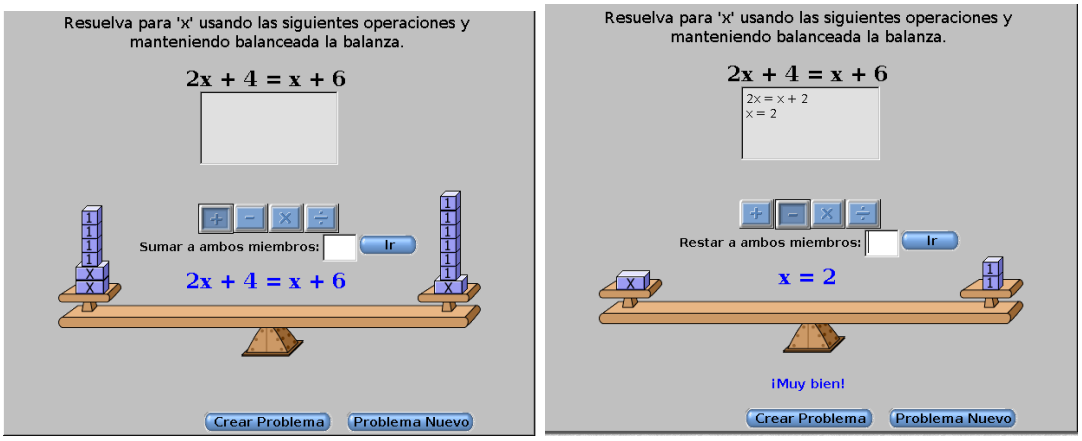



Figura 3: Balanza algebraica

Según las profesoras, esta aplicación tiene la ventaja de que el concepto de equivalencia permanece presente en los distintos pasos de la resolución, además de dar un ejemplo de registro para cuando se resuelven ecuaciones sin el apoyo de la aplicación. La mayor limitación de esta aplicación, es que sólo tiene sentido en el dominio de los naturales.

### 1.1.4. Balanza Algebraica-Negativos

Esta aplicación, permite resolver ecuaciones lineales con números enteros. En esta versión de la balanza, para representar los pesos negativos se utilizan globos.

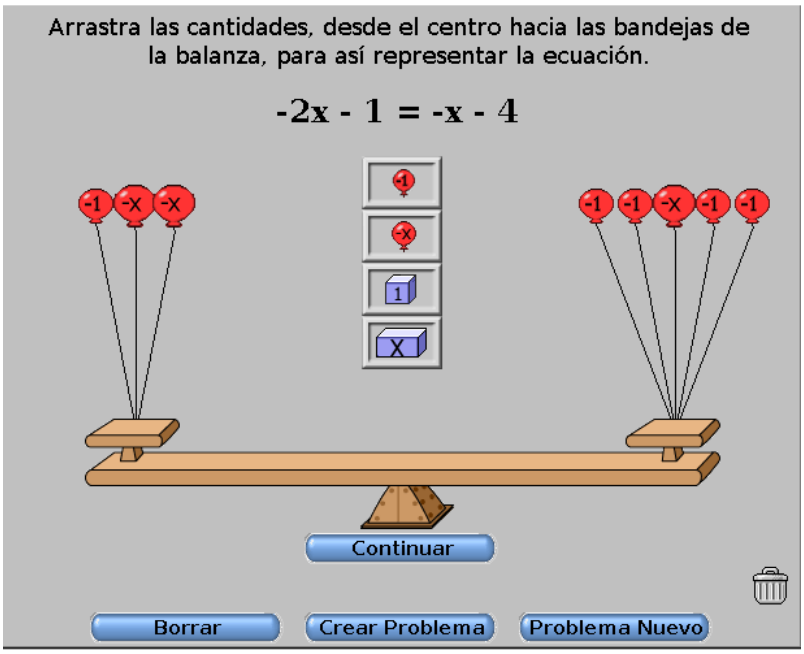



Figura 4: Balanza algebraica negativos

Con respecto a esta aplicación, las profesoras comentaron que al utilizar los globos como pesos negativos puede generar alguna dificultad en la comprensión del modelo. El dominio son los números enteros.

## 1.2. Comentarios y Problema Planteado

Las profesoras valoraron que la aplicación *Cover up* es la aplicación más adecuada para introducir ecuaciones, a pesar de las dificultades encontradas, que se resumen a continuación:

- Los textos se encontraban en Inglés o en Holandés.
- En la ecuación, la incógnita sólo puede aparecer en el lado izquierdo de la igualdad.
- En el caso de las ecuaciones cuadráticas, acepta como solución cualquiera de las raíces, pero no exige explicitar las dos cuando existen.

Además de las dificultades encontradas, las profesoras elaboraron la siguiente lista de especificaciones técnicas deseables en una aplicación de este tipo:

- Tener una base de datos de ecuaciones accesibles.
- Que se puedan agregar/borrar ecuaciones a/de la base de datos.

- Que la aplicación se pueda usar con o sin conexión a internet.
- Que guarde el trabajo realizado por el alumno, de modo que el docente pueda acceder a ello.
- Que retorne algún tipo de feedback al alumno.
- Que ofrezca una interfaz atractiva.

A partir de su análisis plantearon el problema de la construcción de un software que permita la resolución de ecuaciones utilizando el método cover up, que se describe en la siguiente sección, y con las especificaciones técnicas que se resumen arriba.

En este informe se describe la solución al problema desarrollado en el proyecto y la utilización del resultado por estudiantes y profesores de la enseñanza media.

## 1.3. Método Cover up

Como nuestro producto a desarrollar debe resolver las ecuaciones utilizando el método cover up, en esta sección explicamos en qué consiste el método para obtener una mejor comprensión de la aplicación. Este método también conocido como "de la tapadita", permite encontrar las soluciones de la ecuación sólo realizando operaciones aritméticas. A continuación vamos a realizar un ejemplo utilizando una ecuación que a simple vista parece compleja, pero que utilizando esta estrategia resulta simple de resolver.

Si miramos la ecuación $13 - \dfrac{56}{21 + \dfrac{70}{3+x}} = 12$, despejar la $x$ por el método tradicional no resulta sencillo. El método cover up consiste en ocultar las partes de la ecuación que parecen complejas en un principio para facilitar su resolución, en esta caso, como la ecuación a simple vista es difícil de resolver, podemos ocultar la parte de la ecuación correspondiente a $\dfrac{56}{21 + \dfrac{70}{3+x}}$, obteniendo para resolver $13 - \ldots = 12$, por lo que se puede decir que $\dfrac{56}{21 + \dfrac{70}{3+x}}$ debe valer 1, por lo tanto si ahora se oculta $21 + \dfrac{70}{3+x}$, el problema planteado a resolver es $\dfrac{56}{\ldots} = 1$, resultando que $21 + \dfrac{70}{3+x}$ debe ser igual a 56, continuando con el proceso ocultamos $\dfrac{70}{3+x}$, por lo tanto, quedaría planteado $21 + \ldots = 56$, y por ende, se puede decir que $\dfrac{70}{3+x}$ es igual a 35 resultando que $3 + x$ debe valer 2, y por lo tanto, se obtiene que el valor de $x$ es -1.

Por lo tanto, el método consiste en seleccionar una parte de la ecuación y hallar su valor, lo que lleva a plantear una sub ecuación. En ésta, se repite el proceso hasta que la subecuación es de la forma x = valor.

## 1.4. Otros trabajos relacionados

Para conocer cuáles son las aplicaciones que se enfocan a resolver ecuaciones y las funcionalidades que proveen, realizamos una investigación en internet analizando las aplicaciones encontradas.

En la actualidad hay una gran cantidad de aplicaciones que resuelven problemas matemáticos de distintos tipos. En general, estas aplicaciones resuelven ecuaciones de mayor grado en comparación de las ecuaciones a las que estamos enfocados y permiten utilizar más de una variable, pero la mayoría de ellas se limitan a retornar el resultado del problema planteado.

La idea principal es que nuestro software pueda ser utilizado como introducción a la resolución de las ecuaciones en los primeros años de enseñanza media, es decir, no es necesario abarcar ecuaciones de grado mayor a tercer grado, ni de más de una variable. Sí es fundamental que sea el estudiante el que propone la solución y no que sea la aplicación la que retorna el resultado. Principalmente, nuestra aplicación tiene que validar los resultados propuestos por los estudiantes en ecuaciones que se puedan resolver utilizando estrategias aritméticas.

Algunas de las aplicaciones que encontramos fueron:

### 1.4.1. WolframAlpha

Esta aplicación permite plantear ecuaciones de distintos grados, además de permitir utilizar varias variables. Al ingresar la ecuación esta aplicación retorna la solución de la ecuación además de graficar la misma. En su versión “Pro”, también permite ver la solución paso a paso de la ecuación.

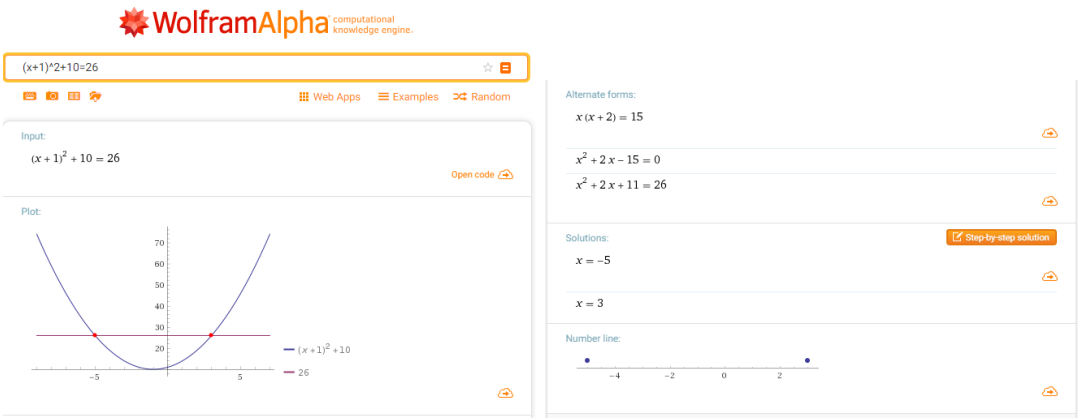



Figura 5: WolframAlpha - Resolución de una ecuación

### 1.4.2. MathWay

Esta aplicación, a diferencia de *WalframAlpha* permite escribir la ecuación utilizando una botonera similar al de una calculadora, ofreciendo la misma funcionalidad de retornar el conjunto solución de la ecuación planteada, sin embargo esta aplicación no grafica la ecuación, como sí lo hace la anterior.


Figura 6: Mathway - Resolución de una ecuación

### 1.4.3. Ecuation Solving

Esta aplicación es bastante similar a las ecuaciones anteriores. También utiliza un tablero para escribir las ecuaciones, permitiendo ingresar ecuaciones de distinto grado y una vez que se cliquea en el botón “Solve” se retornan los resultados de la misma.

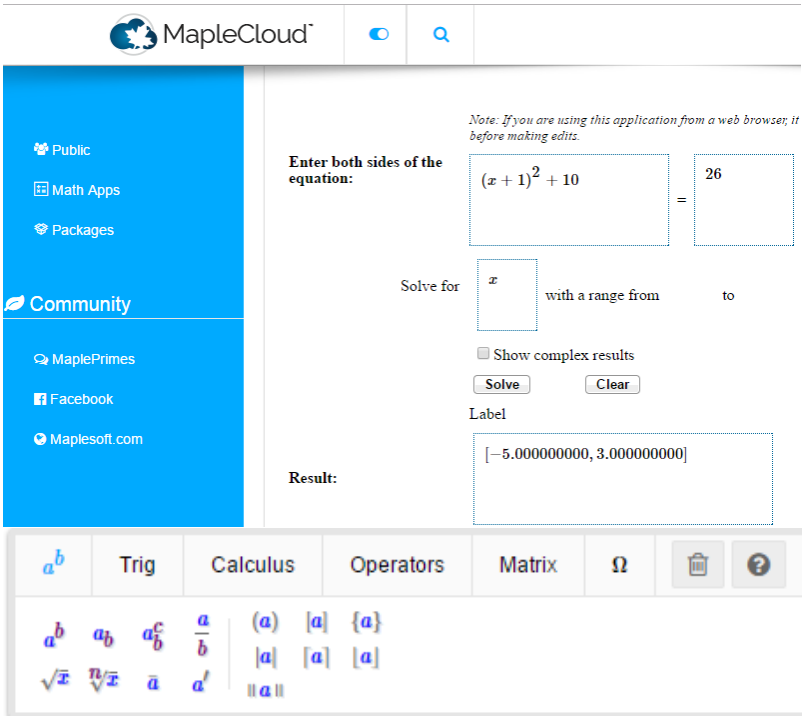



Figura 7: Ecuation Solving - Resolución de una ecuación

### 1.4.4. PAT2Math

Esta aplicación permite a los estudiantes ir resolviendo por partes la ecuación hasta encontrar el resultado de la variable. En esta aplicación, las ecuaciones están separadas por distintos niveles con la intención de introducir el método tradicional de despeje. Para esto al iniciar un nivel, la aplicación muestra los despejes necesarios para resolver la ecuación, y luego presenta otras 5 ecuaciones para que se resuelvan de la misma forma. Una vez que se resuelven todas las ecuaciones planteadas se desbloquea el siguiente nivel de ecuaciones.

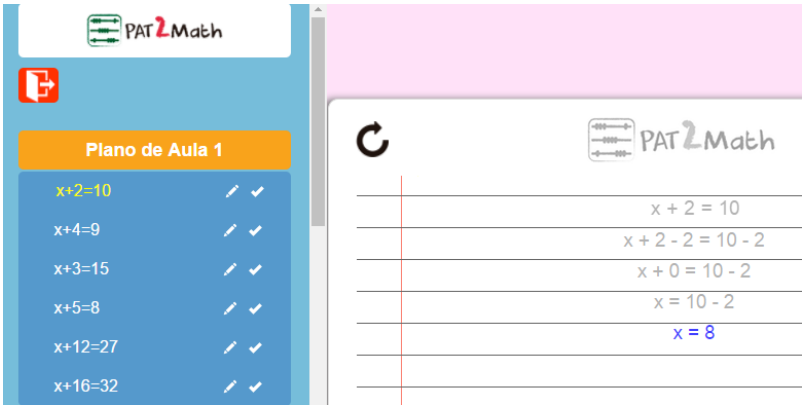



Figura 8: PAT2Math - Despeje realizado por la aplicación como introducción al nivel de ecuaciones.

Una desventaja que notamos de esta aplicación es, que en la resolución de las ecuaciones es necesario plantear con el teclado todo el paso, por ejemplo, cuando se quiere ingresar la solución de la ecuación igual a 3, se debe ingresar con el teclado la expresión $x = 3$, y en el caso en que se deseen despejar ecuaciones más complejas, encontramos esta forma de ingresar los pasos un poco incómoda.

Además de estos tipos de aplicaciones encontradas, hay otra variedad de aplicaciones entre las cuales están las desarrolladas para celulares que permiten tomar una fotografía al problema planteado, que la aplicación reconoce y devuelve la solución.

Como pudimos ver, hay una gran diversidad de aplicaciones que resuelven problemas matemáticos, desde calculadoras hasta aplicaciones que resuelven cálculos complejos con varias variables de distintos grados, otros que grafican funciones y otros con fines didácticos como *PAT2Math*.

La diferencia principal que tienen las primeras tres aplicaciones con la desarrollada en este proyecto, es que en ninguna de estas es el estudiante el que propone la solución, sino que se limitan a retornar el resultado del problema. La aplicación *PAT2Math* permite ingresar los resultados de las ecuaciones, pero la opinión del cliente sobre introducir prematuramente las "reglas de pasaje" es que genera pasividad cognitiva, comentado en la sección 1. La característica principal de nuestro software, determinada por las necesidades de las profesoras, es que se pueda utilizar como herramienta didáctica para la introducción a la resolución de ecuaciones, de forma que los estudiantes puedan comprender qué es una igualdad y utilizar sus conocimientos previos para encontrar el número que verifica dicha igualdad.

# 2. Requisitos de la aplicación

## 2.1. Descripción general

De acuerdo al problema planteado se desarrolló un software para apoyar el desarrollo de la exploración de resolución de ecuaciones en etapas iniciales, basada en las competencias aritméticas de los alumnos y orientado a la consolidación de la noción de ecuación. Siguiendo las especificaciones planteadas por el cliente, la aplicación cuenta con una base de datos con ecuaciones disponibles permitiendo resolverlas utilizando el método de cover up en forma interactiva, a través de una interfaz atractiva diseñada para los equipos de enseñanza media brindados por el Plan Ceibal. La aplicación funciona asimismo para otros equipos similares que cuenten con Linux Ubuntu como sistema operativo y la versión correspondiente de Java (ver 2.3.6).

Dada la inquietud del cliente por la calidad de conexión a Internet en algunas zonas del país, se desarrolló una aplicación de escritorio que pueda ser utilizada sin conexión a Internet. Las funcionalidades esenciales se definieron como: creación de ecuaciones, resolución de las mismas y la posibilidad de consultar el historial de resolución de las ecuaciones ya trabajadas. Además, se cuenta con un repositorio central que contiene ecuaciones compartidas, que podrá ser utilizado cuando se disponga de conexión a Internet. Las ecuaciones creadas localmente en el equipo del usuario, se pueden compartir con el resto de los estudiantes y docentes de ciclo básico, ya que la aplicación permite subir y descargar ecuaciones a través del repositorio central, siempre y cuando se cuente con conexión a Internet.

La aplicación cuenta principalmente con las siguientes funcionalidades:

- Permite crear ecuaciones mediante una botonera que provee los operadores raíz, potencia, cociente y paréntesis.
- Cuenta con un conjunto de ecuaciones cargadas por defecto junto con la instalación.
- Permite resolver las ecuaciones guardadas localmente donde las soluciones de la ecuación son provistas por el usuario en cada paso.
- Guarda el trabajo realizado por el estudiante paso a paso mientras realiza la resolución.
- Lista las ecuaciones guardadas localmente ordenadas por sus estados que pueden ser *sintrabajar*, *trabajadas*, *resueltas* o *eliminadas*.
- Permite subir las ecuaciones ingresadas por el usuario que no fueron subidas previamente, al repositorio central.
- Permite al usuario descargar ecuaciones del repositorio central indicando la cantidad deseada o un intervalo de tiempo.

En las siguientes secciones se desarrollan los requisitos funcionales y no funcionales definidos dentro del alcance del proyecto.

## 2.2. Requerimientos funcionales

### 2.2.1. Crear ecuación

A diferencia de la aplicación de cover up (ver 1.1.2), utilizada actualmente, que sólo permite definir ecuaciones de la forma $f(x) = k$, la aplicación desarrollada en este proyecto, permite crear ecuaciones de la forma $f(x) = k$ o $k = f(x)$[1], donde $k$ es una expresión formada por constantes y operadores aritméticos y $f$ es una expresión algebraica de hasta tercer grado (lineal, cuadrática, cúbica), racional, con radicales o combinaciones de estas, con una única variable y una única ocurrencia.

Se valida que la sintaxis de la ecuación sea correcta (ver 2.4), alertando al usuario con mensajes que lo ayuden a solucionar eventuales errores. Las ecuaciones creadas correctamente se mantienen guardadas localmente para su posterior resolución y pueden ser compartidas en el repositorio central.

### 2.2.2. Resolver ecuación

La resolución de las ecuaciones se hace en etapas mediante métodos aritméticos y culmina cuando el estudiante encuentra todas las soluciones posibles. Si la ecuación cuenta con más de una solución posible, cuando el usuario encuentra una solución, la aplicación le indica mediante un mensaje que existen más soluciones y le da la posibilidad de continuar la resolución desde la etapa en que se detectó otro posible resultado. Para los casos en que la ecuación no tiene solución, luego de una determinada cantidad de intentos se muestra un mensaje indicando que la ecuación no tiene solución. Tanto el número de intentos como los mensajes utilizados fueron definidos en conjunto con el cliente para que la devolución al alumno sea lo más adecuado para su aprendizaje. Los mensajes detallados se encuentran en los anexos. Se destaca que la aplicación es una herramienta que ayuda al desarrollo de la resolución, pero es siempre el usuario quien propone la solución de la misma en cada paso.

### 2.2.3. Logueo local de la resolución

La aplicación guarda todos los pasos realizados por el estudiante al resolver la ecuación, para permitir visualizar las resoluciones y en el caso de que no se haya finalizado continuar con la resolución. También se puede volver a resolver una ecuación ya resuelta, o reiniciar una resolución que no ha sido terminada.

[1] Es decir, donde la incógnita puede aparecer a la derecha o a la izquierda de la igualdad.

### 2.2.4. Listado de ecuaciones locales

Al iniciar la aplicación se muestran las ecuaciones cargadas por defecto y se le brinda la posibilidad al usuario de poder listar las ecuaciones creadas, descargadas, eliminadas o mostrar todas las almacenadas localmente. El listado de ecuaciones está ordenado según sus estados que pueden ser *sintrabajar*, *trabajadas*, *resueltas* o *eliminadas*, los cuales se representan con los colores azul, naranja, verde o gris respectivamente (ver 4.5). Desde el listado el usuario puede acceder a la resolución de una ecuación, ver los historiales de resolución o eliminar las ecuaciones que fueron creadas o descargadas por él.

### 2.2.5. Ver resolución de la ecuación

Como se mencionó en los puntos anteriores es posible visualizar el historial de resolución de una ecuación que fue trabajada anteriormente por el usuario. La aplicación muestra todos los pasos realizados para resolver la ecuación ordenados en forma cronológica y permite guardar una captura del historial que posteriormente podrá ser enviada al docente.

### 2.2.6. Eliminar ecuación local

Se permite eliminar una ecuación que fue creada o descargada previamente, no así las ecuaciones iniciales (ver 2.3.2). La eliminación de la ecuación será solo a nivel de la base local del usuario, el repositorio central no se ve afectado por esta acción. Algo importante a resaltar es que cuando se desea eliminar una ecuación que ya fue trabajada, ésta solo se da de baja permitiendo consultar su historial de resolución desde el listado de ecuaciones eliminadas. Esto fue solicitado por el cliente de forma de no perder el trabajo realizado por el estudiante.

### 2.2.7. Subir ecuaciones al repositorio central

Siempre que el usuario lo solicite y se cuente con conexión a Internet, podrá compartir ecuaciones en el repositorio central. El usuario debe seleccionar de las ecuaciones creadas localmente las que desea subir y la aplicación verifica que se cuente con conexión a Internet. Si no se logra establecer una conexión con el repositorio central se muestra un mensaje indicando que lo intente más tarde. Es importante resaltar que en el repositorio central solo se guardan las ecuaciones sin los logueos de resolución que puedan existir localmente en la bases de los usuarios.

### 2.2.8. Descargar ecuaciones

La aplicación también permite al usuario descargar ecuaciones del repositorio central cuando cuente con conexión a Internet, indicando la cantidad deseada o un intervalo de tiempo. El repositorio central selecciona del conjunto de ecuaciones compartidas que cumplen con el intervalo de tiempo elegido la cantidad de ecuaciones solicitadas por el usuario.

## 2.3. Requerimientos no funcionales

### 2.3.1. Ingreso de ecuaciones

El ingreso de las ecuaciones se realiza por medio de una botonera la cual cuenta con las operaciones (raíz, potencias, cociente y paréntesis) y los espacios para tipear los operadores básicos (+, -, *) y números, así como la variable.

### 2.3.2. Ecuaciones iniciales

Se cuenta con un conjunto de ecuaciones incluidas por defecto en la instalación de la aplicación que facilitará el trabajo inicial del docente con los alumnos. El conjunto fue definido por el cliente para que tengan un nivel de dificultad y orden adecuado al aprendizaje de los alumnos según su experiencia.

### 2.3.3. Trabajar sin conexión a Internet

Como se comentó anteriormente siempre se pueden crear ecuaciones nuevas y resolver en el momento sin contar con conexión a Internet. También se puede resolver las ecuaciones iniciales, creadas o descargadas que fueron guardadas en otra instancia.

### 2.3.4. Interfaz de usuario

La aplicación cuenta con una interfaz atractiva e intuitiva para el usuario dado que permite adaptarse a su funcionamiento con facilidad para ingresar nuevas ecuaciones y resolverlas. Fue diseñada teniendo en cuenta que será utilizada principalmente en los equipos del Plan Ceibal, pero puede ser utilizada en otros equipos similares que cuente con Linux Ubuntu como sistema operativo y la versión correspondiente de Java (ver 2.3.6). El diseño se describe en 4.5

### 2.3.5. Idioma

Todos los textos están en idioma español para que el idioma no sea un impedimento en la comprensión del desarrollo de la resolución de la ecuación.

### 2.3.6. Hardware y Software

La aplicación corre en los equipos del Plan Ceibal, destinados a estudiantes de ciclo básico, que son conocidos como laptops Magallanes y Positivo, ambos cuentan con sistema operativo Linux Ubuntu en su versión 12.040. El promedio de memoria es de 1GB en las Magallanes y 2GB en las Positivo, mientras que el tamaño de pantalla es de 10.1' y 11.6' respectivamente. Los

requisitos mínimos de Hardware y Software sugeridos fueron determinados principalmente por las especificaciones de las laptops Magallanes que son los equipos con recursos más restringidos y la tecnología de javaFX[6] utilizada para el desarrollo de la aplicación.

Requisitos mínimos determinados:

- Procesador Intel Atom N270 1.66 Ghz.
- Memoria RAM de 1GB.
- Al menos 100MB de espacio libre en el disco.
- Pantalla de 10.1çon resolución de 1024 x 600.
- Sistema operativo Linux Ubuntu 12.040 o superior.
- Java SE Runtime Environment 8
- Java SE Development Kit 8

El cliente solicitó que la aplicación funcione en las laptops Magallanes y Positivo, pero también puede ser utilizada en otros equipos que cumplan con los requisitos mencionados anteriormente.

## 2.4. Sintaxis de ecuaciones

Como se mencionó anteriormente las ecuaciones que se deben resolver son aquellas de hasta tercer grado y con una sola ocurrencia de la variable.

A continuación vamos a describir mediante gramáticas y diagramas, las ecuaciones que se deben poder crear y resolver con la aplicación.

**Formato ecuación**

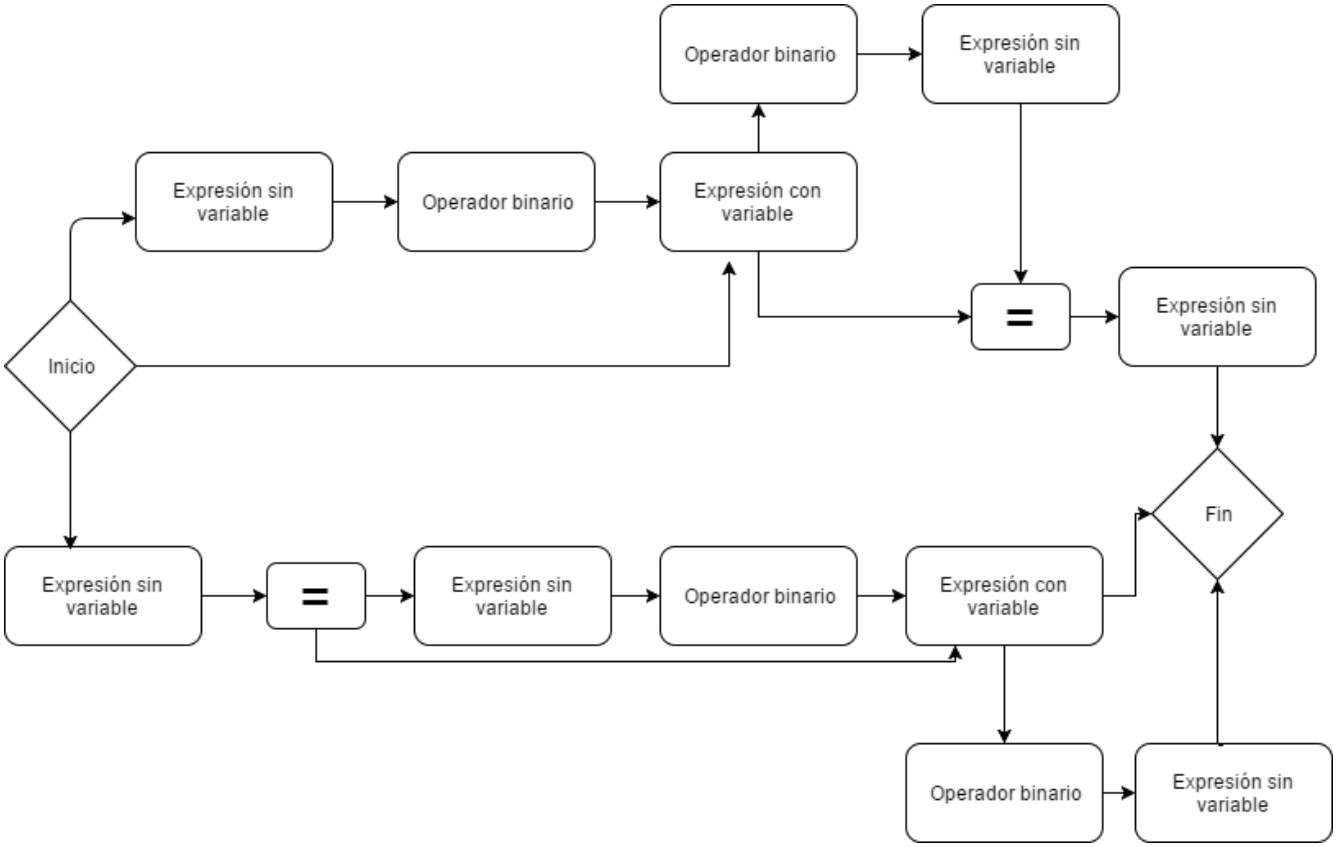



Figura 9: Diagrama de sintáxis de la ecuación

**Expresión sin variable**

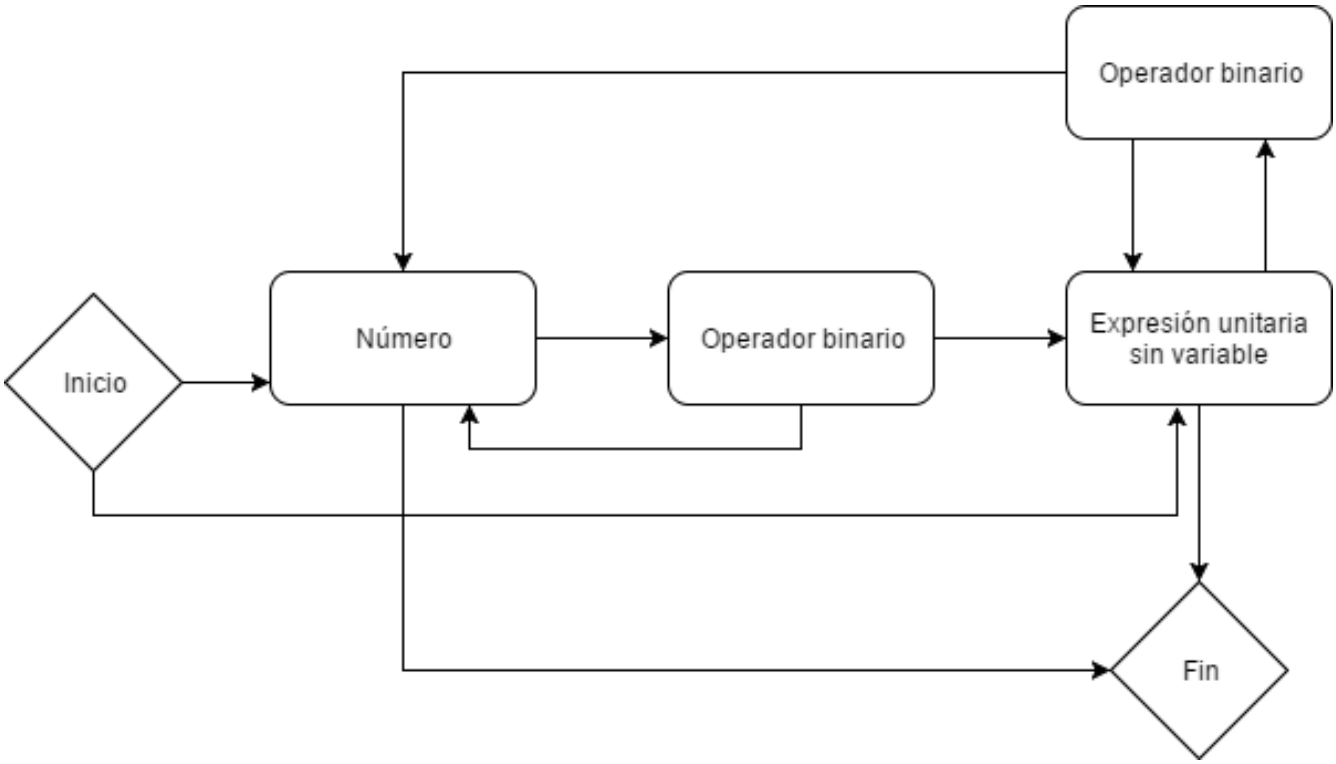



Figura 10: Diagrama de la expresión sin variable

**Expresión con variable**

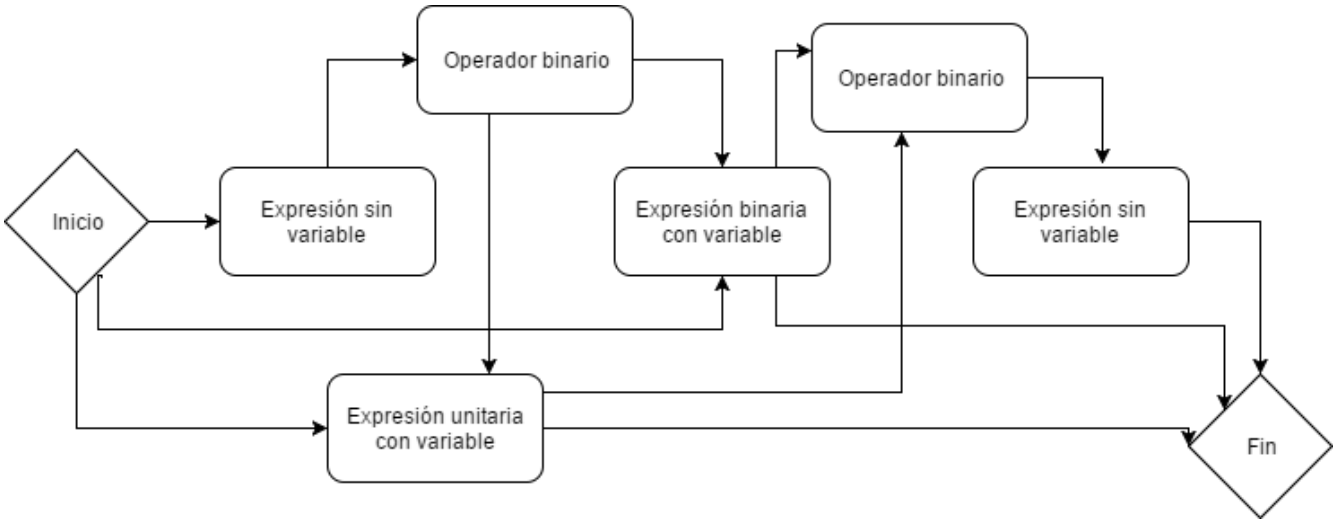



Figura 11: Diagrama de la expresión con variable

Las expresiones unitarias y binarias con variable y sin variable, se describen con gramáticas, debido a que la utilización de paréntesis no permite describir la ecuación mediante diagramas.

**Expresiones unitarias sin variable**

$S->D$

$D->E|U(EOA)|U(AOE)|U(EOE)|U(D)$

$E->U(A)$

$A->$ Expresión binaria sin variable

$U->$ Operadores unitarios

$O->$ Operador binario

**Expresiones unitarias con variable**

$S->D$

$D->U(V)|U(DOB)|U(BOD)|U(D)$

$V->$ Expresión binaria con variable

$U->$ Operadores unitarios

$B->$ Expresión binaria sin variable | Expresión unitaria sin variable

$O->$ Operador binario

**Expresión binaria sin variable**

$S->C$

$C->BOB|B$

$B->(B)|NON|N$

$N->$ Número

$O->$ Operador binario

**Expresión binaria con variable**

$S->V|VOB|BOV|(V)$

$V->BOX|XOB|X$

$X->a..z$

$B->$ Expresión sin variable

$O->$ Operador binario

**Operadores binarios**

- Suma
- Resta
- Multiplicación
- División

**Operadores unarios**

- Raíz cuadrada
- Raíz cúbica
- Potencia cuadrada
- Potencia cúbica
- Potencia a la un medio
- Potencia a la un tercio
- Potencia a la menos uno
- Potencia a la menos dos
- Potencia a la menos tres
- Opuesto

# 3. Algoritmos desarrollados

Durante la construcción del software nos encontramos con los siguientes problemas principales: la representación visual de la ecuación durante su creación; la verificación de los resultados propuestos por los estudiantes y el reconocimiento de las partes seleccionadas de la ecuación para asignarles el resultado. En esta sección, vamos a presentar algunos de los algoritmos con los que resolvimos estas dificultades separándolos en dos categorías: Front-end para los algoritmos que se encuentran en el desarrollo de la parte visual de la aplicación y Back-end para los que corresponden a la lógica de la resolución de las ecuaciones.

## 3.1. Back-end

El principal problema con el que nos enfrentamos es la verificación de las soluciones propuestas por el estudiante en los procesos intermedios, es decir, cómo saber que el valor ingresado por el estudiante es correcto para la sub ecuación seleccionada. Para solucionar este problema, optamos por tener conocidos de antemano los resultados de la ecuación, y sustituirlos en la sub ecuación para comprobar si validan el resultado ingresado. Esto implica, que al momento de la creación de cada ecuación se debe encontrar su solución o soluciones, para luego poder utilizarla durante la resolución de la misma. Por lo tanto, vamos a comenzar describiendo el algoritmo de creación de una ecuación antes de explicar el algoritmo de resolución.

```
crearEcuacion(String ecuacionInfija): Ecuacion
    listaInfija = obtenerListaInfijaDeString(ecuacionInfija)
    listaPostFija = convertirListaInfijaAPostfija(listaInfija)
    arbolEcuacion = obtenerArbolDeNotacionPostFija(listaPostFija)
    arbolDespejado = despejarVariable(arbolEcuacion)
    resultados = hallarResultados(arbolDespejado , false)
    solucion = resultadosVerificados(resultados, ecuacionInfija)
    Se retorna una ecuación que contenga los datos iniciales y el conjunto solución.
```

El algoritmo *crearEcuacion* recibe como parámetro un string que contiene la ecuación en forma infija (ver 2.4). El string tiene un formato específico para detectar fácilmente operadores y operandos, que se encuentra descrito en los anexos. Este string se parsea mediante la función *obtenerListaInfijaDeString*, que lo devuelve en el formato de lista, donde cada ítem contiene sólo un operador, la variable o un número. El algoritmo devuelve una representación de una ecuación inicializada con los datos de la ecuación y el conjunto solución (ver 4.3).

Para poder realizar los cálculos y hallar la solución de la ecuación, nosotros decidimos representar la ecuación en una estructura de árbol, debido a que en asignaturas de la carrera enfrentamos este tipo de problemas utilizando esta estructura. Para ello, es necesario tener la ecuación escrita en notación postfija (primero el operando y luego el operador), por lo tanto la lista resultado de *obtenerListaInfijaDeString* es convertida en una lista en notación postfija, mediante la función

*convertirListaInfijaAPostfija* y esta lista en el árbol mediante la función *obtenerArbolDeNotacionPostFija*.

El resultado del proceso descrito es un árbol cuyos nodos son operaciones y cuyas hojas son los números o la variable, a partir del cual realizamos los cálculos para resolver la ecuación.

Dado que el tipo de ecuaciones puede contener sólo una vez la variable, para hallar su valor optamos por despejarla, es decir, construir un árbol que sólo contenga los datos necesarios para calcular la o las solución/ones, que llamamos "árbol despejado" (sin variable). La función *despejarVariable* recibe el árbol original y retorna el "árbol despejado". En los anexos se describe este algoritmo detallado.

Teniendo el árbol que representa los cálculos a realizar para solucionar la ecuación, se utiliza la función *hallarResultados* que recibe el "árbol despejado" y un booleano para indicar los casos de raíces dobles, y retorna el resultado del cálculo de las operaciones. El parámetro booleano se utiliza para indicar cuándo es necesario que se retorne la raíz y su opuesto, o cuándo es necesario que se retorne sólo el resultado de la raíz. Por ejemplo, si la ecuación es del tipo $x^2 = 16$, en el árbol despejado la operación para calcular el resultado de la x es raíz de 16; con el parámetro booleano en false se retorna el 4 y también el -4. Si la ecuación es del tipo $x - \sqrt{16} = 0$, con el parámetro booleano en true, se retorna solamente el 4.

El siguiente paso del algoritmo es la verificación de las soluciones halladas mediante la función *resultadoVerificados*, que recibe como parámetro el conjunto de los resultados hallados y el string que representa la ecuación en forma infija, y retorna los resultados que verifican la ecuación, eliminando del conjunto los que no la verifican.

Para cada resultado obtenido se sustituye la variable en la ecuación y se calcula que los resultados de ambos lados de la igualdad sean iguales. Ésta validación se realiza de la siguiente forma: como se pasó por parámetro el string de la ecuación, para obtener los distintos lados de la igualdad se divide la ecuación a partir del símbolo de igualdad. Luego al lado de la ecuación que no contiene la variable, se le realizan las transformaciones necesarias para convertirlo a un árbol y de ahí hallar su resultado. Luego, el string del lado de la ecuación que contiene la variable, se convierte a una lista postfija, por lo tanto, por cada uno de los resultados hallados anteriormente, se sustituye el item que contiene la variable, se convierte esta lista a un árbol (ya sin la variable) y se calcula su resultado. Si el valor recién obtenido es igual a los cálculos del lado de la igualdad sin la variable, se verifica que el resultado por el cual se sustituyó verifica la ecuación.

Para realizar los cálculos de los lados de la ecuación, se utiliza la función *hallarResultados* con el parámetro booleano en true para que retorne solamente el valor positivo de la raíz, dado que en este caso se trabaja sólo con números reales.

Al trabajar con números Reales, es posible que se ingresen ecuaciones que tengan como solución el conjunto vacío, como por ejemplo la ecuación del tipo $x^2 = -1$, para detectar que esta ecuación no tiene solución nos apoyamos en la respuesta que retorna java al realizar una raíz de un número negativo. La respuesta que retorna Java es NaN (Not a Number), por lo tanto, en el caso de obtener este resultado, nosotros ya detectamos que el conjunto solución es el vacío.

El único caso en el que no permitimos dar de alta una ecuación, es en el caso en que se encuentre que hay una división con el divisor igual a cero. Para detectar este caso también nos basamos en la respuesta Infinite que retorna Java, por lo tanto si al hallar los resultados obtenemos este valor, se devuelve un mensaje de error indicando que hay una división por cero.

A continuación se describe el algoritmo con el que se valida un paso de la resolución realizado por el estudiante. Durante la resolución de una ecuación el estudiante selecciona partes de la ecuación y les asigna un valor. El siguiente algoritmo se encarga de verificar que el valor ingresado para esa selección sea correcto.

validarResolucion(idEcuacion, subEcuacionSeleccionada, respuesta, estadoBifuracion):String

Si la sintaxis de la ecuación es incorrecta se retorna sintaxis incorrecta.

Actualizar el id de la resolución en el caso de que se esté comenzando a resolver la ecuación.

Si la respuesta ingresada por el estudiante resulta un conjunto vacío

Verificar que la solución de la ecuación es el conjunto vacío y además que el paso que se seleccionó no tiene solución. Si el paso seleccionado tiene solución válida se setea en la respuesta que el paso es incorrecto.

Si la respuesta ingresada no resulta el conjunto vacío

Si la ecuación tiene solución

correcto = Determinar la correctitud de la respuesta dada

Si es correcto

Si fue seleccionada sólo la variable

esVariable = true

Verificar que no se esté repitiendo el valor de la solución. Si se repite un resultado, se asigna en la respuesta que se repitió el resultado

Si fueron ingresadas todas las soluciones, se asigna en la respuesta que la resolución fue finalizada, sino, se setea en la respuesta que faltan ingresar más soluciones.

Sino

En el caso que se esté repitiendo al paso de la bifurcación, se verifica que no se repita el resultado.

Si repite el resultado

correcto = false

Se asigna en la respuesta que el resultado es incorrecto

Sino

Si no estoy bifurcando y la selección está incluida en una potencia cuadrada.

Verificar si hay marcado un paso anterior como de bifurcación

Si no se encontró un paso anterior marcado con bifurcación

Seteo a este paso como el de la bifurcación

Asignar en la respuesta que el paso fue correcto

Sino

Setear en la respuesta que el paso fue incorrecto.

Si la ecuación no tiene solución

Verificar si el paso de la ecuación sin solución es correcta o no y asignar el resultado en la respuesta.

Actualizar el estado de la ecuación



Si los datos de la ecuación fueron modificados, actualizar los datos en la base de datos.

Loguear el paso de la resolución en la base de datos.

Retornar la respuesta de la resolución.

### 3.1.1. Comentarios generales

Como se puede ver en el algoritmo, el primer paso a realizar es la validación de la sintaxis de la ecuación, es decir, verificar que los operadores tengan la cantidad de operandos correcta, y verificar que la expresión seleccionada sea una sub expresión de la ecuación. Esta última validación se realiza, debido a que desde la interfaz gráfica se pueden seleccionar partes de la ecuación de una forma incorrecta, por ejemplo, si se tiene la ecuación $\frac{1+8}{x+3} = 1$, en la aplicación es posible seleccionar $\frac{1}{x}$, haciendo imposible al algoritmo validar la correctitud de esa selección. Por solicitud del cliente, es necesario que la aplicación guarde el trabajo realizado por los estudiantes, el cual puede resultar de resolver varias veces la misma ecuación, por lo tanto, cada ecuación debe tener asociada las distintas resoluciones que fueron realizadas de forma de poder permitir visualizar cada una de ellas y permitir retomar la última resolución iniciada pero no finalizada. El atributo `estado` de la ecuación indica el estado de la ecuación (sin trabajar, trabajada o finalizada) y en base a su valor, se incrementa o no el parámetro idEcuación. Si la ecuación está en estado trabajada, entonces se continúa con su mismo idEcuacion hasta que sea finalizada, en otro caso se incrementa.

### 3.1.2. Conjunto vacío ingresado como respuesta

El parámetro `respuesta` representa la respuesta ingresada por el estudiante, ésta puede ser un número, una expresión a calcular o una una constante que representa el conjunto vacío. Si es el conjunto vacío se verifica que la solución de la ecuación sea el conjunto vacío y que la sub ecuación seleccionada no tenga solución real. Es posible que pasos previos en la resolución tengan solución pero que no haya un valor real para la variable que solucione la ecuación. Estos casos se pueden dar en una ecuación del tipo $x^2 + 8 = -8$, por ejemplo. Si se selecciona $x^2$, el valor que verifica esta selección es -16 debido a que $-16 + 8 = -8$, pero sin embargo, los valores de $x$ que elevados al cuadrado son igual a -16 no son reales, por lo tanto el conjunto que verifica la ecuación es el vacío.

### 3.1.3. Verificar respuesta ingresada

En el caso que no se haya ingresado como respuesta el conjunto vacío, se verifica que la respuesta ingresada sea correcta. Para realizar esta verificación, se sustituye la variable de la expresión seleccionada por cada uno de los elementos del conjunto `resultados` (las soluciones de la ecuación) y se calcula el resultado. Si para alguna sustitución se obtiene el mismo resultado que el ingresado por el estudiante, se devuelve que el resultado es correcto. Para realizar los cálculos que llevan a la solución de la ecuación, se utiliza la estructura de árbol y la función *hallarResultados*, descrito en la sección anterior.

### 3.1.4. Verificación del paso anterior

La necesidad de verificación de pasos anteriores se explica a través de los siguientes ejemplos:

**Ejemplo 1**

Una vez que ya sabemos que el resultado ingresado verifica la sub expresión seleccionada, se comprueba que también se verifique el paso anterior. Esta validación es importante para corroborar que las respuestas se deducen del paso anterior, ya que si no se realiza esta validación se puede retornar como paso correcto un paso que no se corresponde con la resolución realizada. Para explicar mejor el problema, se va a presentar un ejemplo utilizando una ecuación con dos soluciones válidas, en este caso la ecuación es $(x+1)^2 - 8 = 8$ con las soluciones 3 y -5.

Suponemos la siguiente resolución:

$(x+1)^2 - 8 = 8$ $(I)$

...

$x + 1 = 4$ $(II)$

$x = -5$ Verifica $(I)$ pero no verifica $(II)$

Suponiendo que en la resolución se ingresó el resultado de la selección x+1 igual a 4 $(II)$, si luego se selecciona la variable y se le asigna la respuesta -5, el resultado del algoritmo sin la validación del paso anterior determina que es correcto debido a que el resultado -5 verifica la ecuación, ya que $(-5+1)^2 - 8$ es igual a 8, pero se puede ver que no es cierto que -5+1 es igual a 4, por lo tanto, la validación del paso anterior se utiliza para que sean consistentes los resultados de la ecuación con la secuencia de resolución realizada. En este caso, con la validación del paso anterior se indica que el paso x = -5 no es válido.

**Ejemplo 2**

En la explicación anterior, cuando mencionamos que se realiza la verificación del paso anterior, no siempre se verifica el paso inmediatamente superior, sino que depende de si ya se encontró el primer resultado de la ecuación y se está en búsqueda de otro o no. Para explicar mejor este motivo, se va mostrar un ejemplo suponiendo que se encontró el primer resultado de la ecuación a partir de la siguiente secuencia de resolución, donde la aplicación vuelve a mostrar la expresión $x + 1 = ...$ para que se ingrese el otro valor posible.

Resolución:

$(x+1)^2 - 8 = 8$

$(x+1)^2 = 16$ $(I)$

$x + 1 = 4$ (Paso de bifurcación) $(II)$

$x = 3$ $(III)$

$x + 1 = ...$

$x + 1 = -4$ No verifica el paso $(II)$ ni $(III)$. Verifica el paso $(I)$

Cuando se ingresa el valor -4 a la expresión $x + 1$, si se verifica la respuesta con el paso anterior $x = 3$ $(III)$ el resultado es incorrecto porque no es posible realizar esta validación, tampoco es correcta la respuesta con el paso $x + 1 = 4$ $(II)$, porque claramente es incorrecto, por lo tanto recién es posible verificar si el paso es consecuente con la resolución a partir de la expresión $(x + 1)^2 = 16$ $(I)$. Este paso se corresponde con el paso anterior al que se realiza la bifurcación, se detecta la bifurcación cuando la sub ecuación seleccionada tiene más de un resultado posible.

Por lo tanto, dependiendo si se está verificando el paso de bifurcación o no, se tiene que corroborar con el paso inmediatamente superior o con el anterior a la bifurcación.

### 3.1.5. Retomar resolución

El cliente planteó la necesidad de permitir al estudiante encontrar otra solución, cuando la hay, siguiendo la resolución desde un paso intuitivo.

A partir de esta resolución:

$(x + 1)^2 - 8 = 8$

$(x + 1)^2 = 16$

$x + 1 = 4$

$x = 3$

la aplicación detecta que hay otra solución además de x = 3, hay que decidir a partir de qué paso plantear la resolución para encontrarla, de modo que el estudiante no tenga que realizar cuentas mentalmente. Se decidió retomar el proceso desde el paso en que se detectó la bifurcación.

Si la resolución para encontrar la otra solución se retoma en un paso distinto al de la bifurcación, es probable que se incremente la dificultad para hallar el valor. Por ejemplo, si se retoma la resolución en un paso posterior al de la bifurcación, como puede ser en la variable $x$, el ejemplo de resolución quedaría de esta forma:

$(x + 1)^2 - 8 = 8$

$(x + 1)^2 = 16$

$x + 1 = 4$ (Paso de bifurcación)

$x = 3$

$x = ...$

Como se puede ver, a partir del paso $x = ...$ es difícil determinar su resultado, porque mentalmente es difícil detectar cuál es el paso que debe tener otro valor, para poder obtener la otra solución.

Si en cambio se elige un paso previo a la bifurcación como por ejemplo el $(x + 1)^2$, el ejemplo de resolución quedaría de la siguiente forma:

$(x + 1)^2 - 8 = 8$

$(x+1)^2 = 16$

$x+1 = 4$ (Paso de bifurcación)

$x = 3$

$(x+1)^2 = ...$

En este paso, el valor a ingresar es el mismo que el ingresado previamente (16), sin dar un indicio de que se debe ingresar un valor distinto en el paso $x+1$ para poder encontrar la otra solución.

Ahora si se retoma la resolución desde el paso en que se detecta la bifurcación:

$(x+1)^2 - 8 = 8$

$(x+1)^2 = 16$

$x+1 = 4$ (Paso de bifurcación)

$x = 3$

$x+1 = ...$

se da indicio de que $x+1$ tiene otro resultado y por lo tanto, facilita al estudiante a encontrar la otra solución.

### 3.1.6. Determinar la cantidad de soluciones ingresadas

Una vez que ya determinamos si el resultado ingresado es correcto, pasamos a dividir el algoritmo según si fue seleccionada sólo la variable o una expresión que abarca más términos. Se realiza esta partición, porque detectando si es una variable o no, se puede determinar si se está ingresando una solución a la ecuación y permite verificar si se encontraron todas las raíces de la ecuación, si se está repitiendo la solución antes ingresada o si todavía falta encontrar otro resultado. Como se puede ver en el algoritmo, cuando se detecta que se seleccionó sólo la variable, se asigna a la variable `esVariable` el valor true. Se realiza esta asignación para poder indicarle al logueo de la resolución que en este paso se ingresó una solución. El tener este dato guardado, permite identificar en la base de datos los pasos en que se ingresó una solución a la ecuación. Además, con la ayuda de la variable `correcto`(también guardada en la base de datos) se puede identificar qué resultados ingresados durante la resolución fueron correctos o no. Tener estos datos guardados resulta de utilidad, para poder contar cuántas respuestas distintas y correctas fueron ingresadas y determinar si se encontraron todos los resultados de la ecuación, si se está volviendo a ingresar el resultado de la ecuación que ya fue ingresado anteriormente, o si falta encontrar la otra solución de la ecuación.

Según los diferentes casos antes comentados, se va a retornar distintas respuestas para el manejo de la aplicación. En el caso en que se esté repitiendo la solución se va a devolver que repite el resultado, para poder indicarle al estudiante que falta encontrar otra solución. Si se encontraron todas las soluciones, se retorna que la ecuación fue finalizada, esto permite mostrar el conjunto solución de la aplicación y dar por finalizada la resolución. En el caso se haya encontrado una

solución pero todavía no se encontró la otra, se retorna que existe otra solución para que se continúe con la resolución de la ecuación.

### 3.1.7. Paso de bifurcación

En el caso que se haya seleccionado un conjunto de términos, y se esté resolviendo el paso marcado como bifurcación, se verifica que el resultado ingresado no sea el mismo que el ingresado previamente. Para poder determinar si se está resolviendo por segunda vez el paso bifurcado, se utiliza la variable pasada por parámetro `estadoBifurcacion`. Como en la presentación de la aplicación se determina si se tiene que volver a mostrar el paso de bifurcación para encontrar la otra solución, es ésta quien indica al algoritmo el valor de este estado. La necesidad de validar que no se repita la misma respuesta del paso de la bifurcación, surge para evitar que se continúe por un camino en el cual no se va a encontrar la otra solución. Esto se puede dar, porque al ya haberse encontrado una solución y volver a repetir los pasos realizados anteriormente, una vez que se seleccione la variable y se ingrese el valor de la solución, es posible que se ingrese la solución anterior y por lo tanto la aplicación va a indicar que se está repitiendo la respuesta, o se va a ingresar la otra solución de la ecuación, pero como el valor ingresado no valida la secuencia de resolución se va a indicar que el paso es incorrecto sin poder salir de esa situación. Para verificar que no se repita la respuesta ingresada anteriormente en el paso de bifurcación, la primera vez que se detecta la bifurcación se guarda en la base de datos que este paso es el de la bifurcación, por lo tanto, para validar se puede obtener este paso y obtener el valor que fue ingresado. En el caso que se haya repetido la respuesta, se devuelve que el paso es incorrecto.

Para detectar si el paso es de bifurcación, se verifica si la selección pertenece a una ecuación cuadrática, es decir, si la selección tiene dos resultados posibles que verifiquen la ecuación. Si este es el caso, entonces se verifica que no se haya detectado el paso de bifurcación anteriormente. Como se puede ver, en la ecuación $(x+1)^2-8=8$, el $x+1$ tiene dos resultados posibles, al igual que sólo la $x$, pero se continúa la resolución de la otra solución a partir de $x+1$. Cabe aclarar, que si en vez de seleccionar en un principio la expresión $x+1$ se selecciona sólo la variable, éste es el paso que queda marcado como bifurcación y luego se continuará la resolución a partir de la $x$.

Para detectar que una selección tiene más de un resultado, se sustituye la selección dentro de la ecuación original por una variable, y se procede a hallar la resolución como se explicó en un principio, si se retornan dos resultados entonces detectamos que la selección está dentro de una potencia cuadrada, como son los casos de $x+1$ y la $x$, en cambio si se selecciona $(x+1)^2$, y se sustituye esta selección en la ecuación original por una variable, quedando $z-8=8$, se obtiene un sólo resultado, confirmando que la selección realizada no se encuentra dentro de una potencia cuadrada.

En el caso que se haya detectado que la selección tiene más de un resultado posible y no se haya encontrado otro paso anterior marcado como bifurcación, se procede a indicar que éste es el paso en que se detectó la bifurcación.

### 3.1.8. Ecuación sin solución

En el caso en que la ecuación no tenga soluciones asociadas, se permite al estudiante intentar varias veces para luego comentarle que la ecuación no tiene solución real. El funcionamiento de este requisito es el siguiente, si el estudiante seleccionó una sección que no tiene solución real, las respuestas que se ingresen van a ser incorrectas, por lo tanto luego de tres intentos se muestra un mensaje preguntando si la ecuación tendrá solución real, luego se permite ingresar dos intentos más para luego indicar que la ecuación no tiene solución real. Si estando en un paso que no tiene solución se ingresa la respuesta de conjunto vacío, se devuelve que el resultado ingresado fue correcto, pero este caso se encuentra validado al inicio del algoritmo.

En todos estos casos, una vez que se finalizó la resolución de la ecuación, ya sea porque se ingresó como respuesta el conjunto vacío o se intentó ingresar otros resultados incorrectos, se muestra en la aplicación que la solución de esta ecuación es el conjunto vacío.

Para poder contar la cantidad de veces que se ingresaron respuestas a una selección sin solución, se mantiene guardado en la base de datos los pasos seleccionados que no tienen solución. Por lo tanto, se puede obtener cuántos intentos fueron realizados para poder detectar el momento en que se debe retornar las distintas respuestas.

### 3.1.9. Actualización de los datos

En la última parte del algoritmo, se realiza la actualización de los datos de la ecuación, como por ejemplo el id de la resolución actual de la ecuación, o el estado de la misma. En el caso en que se haya encontrado todas las soluciones de la ecuación se setea como finalizada, y si es el primer paso de la resolución de la ecuación se modifica el estado a trabajada. También en esta última parte, se deja registrado en la base de datos el paso ingresado por el estudiante así como los valores de otras variables necesarias para la realización del algoritmo. Por último, se retorna la respuesta del algoritmo antes asignada.

Los algoritmos antes mostrados fueron simplificados para poder concentrarse en las secciones más relevantes de los mismos, como por ejemplo, en el último, cómo se obtiene la ecuación que se desea resolver, cómo se calcula el valor de la respuesta dada (ya que ésta puede ser una expresión con operaciones), así como la utilidad de varias de las variables pasadas por parámetro que no fueron utilizadas en el mismo, y el proceso seguido en el caso de que la solución de la ecuación sea el conjunto vacío. Los algoritmos detallados se encuentran en los anexos.

## 3.2. Interfaz gráfica de usuario (Front-end)

La interfaz gráfica de usuario, (en adelante *GUI*), está diseñada con tecnología *JavaFX* que, combinada con *Java*, nos permitió lograr un diseño moderno y amigable, ya que habilita embeber lenguaje de estilo *CSS*[2] siendo sencillo personalizar aspectos de apariencia como son los colores,

[2]Son las siglas de Cascading Style Sheets, hojas de estilo en cascada.

fondos, márgenes, bordes, tipos de letras, entre otros.

Durante la etapa de implementación de la interfaz nos enfrentamos principalmente con tres situaciones críticas: creación, visualización y resolución de las ecuaciones, que fueron resueltas combinando los diferentes esquemas de distribución (*layouts*), controladores básicos (cajas de texto, botones, etc) y manejo de eventos asociados a estos últimos, como explicaremos en las siguientes secciones.

### 3.2.1. Creación de las ecuaciones

La GUI permite el ingreso de una nueva ecuación por medio de una botonera la cual cuenta con las operaciones (raíz, potencias, cociente y paréntesis) y los espacios para tipear los operadores básicos (+, -, *) y números, así como la variable. El panel para crear la ecuación está formada por una zona que contiene la botonera y otra zona para el tipeo como se puede apreciar en la siguiente imagen.

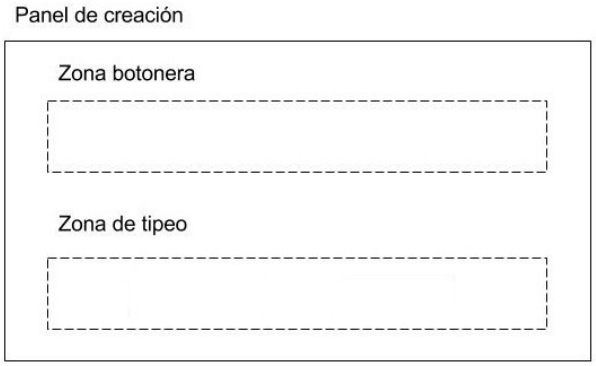



Figura 12: Estructura principal del panel de creación.

### 3.2.2. Botonera y zona de tipeo

Se implementó la clase *Botonera* encargada de resolver el diseño y manejo de lógica necesaria para la creación del *string* en notación infija de la ecuación que utiliza el algoritmo *crearEcuacion* desarrollado al comienzo del capítulo.

Para generar la zona de botones se utiliza la función *crearBotonera* que recibe como parámetro el tamaño de botón deseado y devuelve un *layout* de tipo *GridPane*, que acomodan los elementos en una cuadrícula por sus respectivas filas y columnas. En el caso de la botonera se utiliza una única fila que contiene los botones que la conforman. Los botones raíz y potencia despliegan

un menú vertical con las distintas raíces y potencias soportadas por la aplicación para que el estudiante seleccione la deseada.

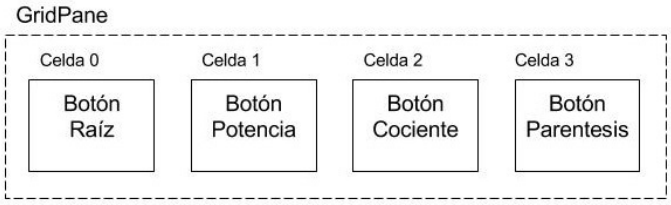



Figura 13: Estructura zona de la botonera.

La zona de tipeo se crea utilizando la función *crearZonaEditarEcuacion* que devuelve un *layout* de tipo *HBox* que acomoda los elementos de forma horizontal. En esta zona se irá formando la ecuación a medida que el estudiante la va ingresando y va cambiando su estructura a medida que se interactúa con la botonera. La instancia del *layout* devuelto por la función *crearZonaEditarEcuacion* contiene tres elementos que llamaremos zona izquierda, zona de signo igual y zona derecha de la ecuación.

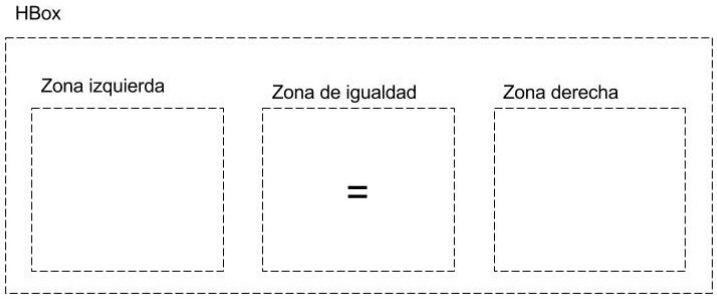



Figura 14: Estructura zona de tipeo.

La zona del signo de igual está formado por un *layout* del tipo *VBox* que contiene una *label* con el símbolo de igualdad. Este tipo de *layout* ordena los elementos de forma vertical y se le seteó el atributo de alineación para que el *label* quede centrado verticalmente dentro de la zona.

Se utilizaron *layout* del tipo *VBox* para crear las zonas izquierda y derecha de la ecuación, ambas tienen inicialmente la misma estructura formada por un *layout* del tipo *HBox* con un contenedor de caja de texto creado con la función *crearEstructuraTexto*. Esta función recibe como parámetro un número entero que será utilizado para identificar las cajas de texto dentro de la zona e interactuar con la botonera como veremos más adelante.

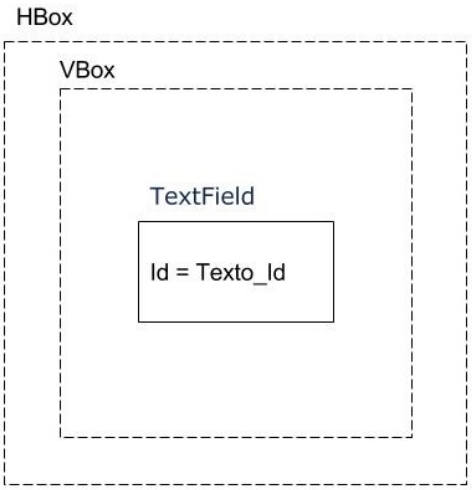



Figura 15: Estructura de texto.

Los botones de la botonera tienen asociado al evento *setOnAction*, las funciones correspondientes a cada operador. Si bien los operadores son diferentes, la idea de la acción es similar en cada función. Se tiene guardado el id de la última caja de texto seleccionada en la zona de tipeo y con ese id se obtiene el *layout* de tipo $HBox$ que es padre de esa estructura de texto, donde se agregarán dos zonas nuevas que llamaremos zona del operador y zona de afuera. La zona de afuera es una estructura de texto creada con la función $crearEstructuraTexto$ y está pensada para que el estudiante pueda ingresar nuevos términos después del operador y la zona del operador es la que varía dependiendo del mismo. Vale la pena aclarar que se utilizan imágenes para representar los paréntesis, potencias y símbolo de raíz.

Las potencias comienzan y terminan con paréntesis donde el último tiene el número del exponente asociado. Cuando se trata de una potencia la zona del operador se forma con un *layout* del tipo $VBox$ que contiene la imagen del paréntesis izquierdo, un segundo *layout* que contiene una estructura de texto creada con la función $crearEstructuraTexto$ y un último *layout* del tipo $VBox$ que contiene la imagen del paréntesis derecho junto con el exponente que corresponda. Notar que la acción del botón paréntesis se comporta similar pero se omite el exponente en el paréntesis derecho.

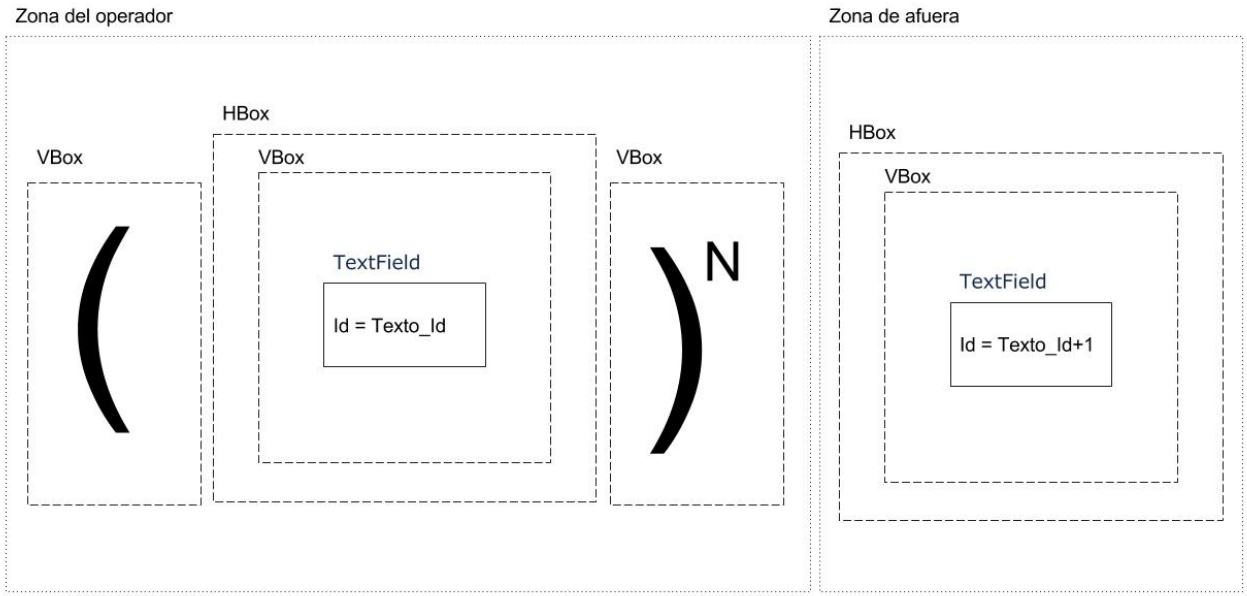



Figura 16: Ejemplo estructura acción potencia.

La zona del operador de una raíz se forma primero por un *layout* del tipo $VBox$ que contiene la imagen con el comienzo del símbolo y por un segundo *layout* que contiene una estructura de texto creada con la función $crearEstructuraTexto$. En este último, se dibuja el borde izquierdo y el borde superior para terminar de definir el símbolo e indicar que todo el contenido de la estructura de texto forma parte de la raíz.

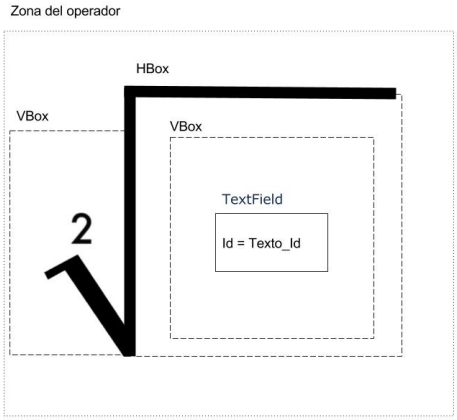



Figura 17: Ejemplo estructura de zona del operador raíz cuadrada.

Cuando se selecciona un cociente la zona del operador varía un poco más, dado que se tiene que permitir ingresar términos en el dividendo y en el divisor. Para esto se utiliza un *layout* de tipo $VBox$ que contiene dos estructuras de texto creadas con la función $crearEstructuraTexto$. Como estas están contenidas en un $VBox$ quedan alineadas verticalmente representando la zona del dividendo y divisor, para representar la línea del cociente se dibuja el borde en común entre los dos *layout*.

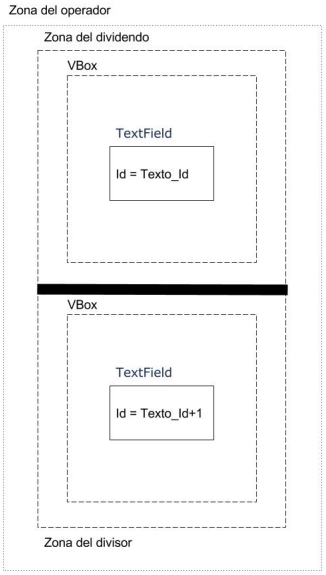



Figura 18: Ejemplo estructura de zona del operador cociente.

Como se mencionó en un comienzo de esta sección la clase *Botonera* es la encargada de la creación del *string* en notación infija de la ecuación. Para esto se mantiene una estructura en forma de lista formada por elementos del tipo $ElementoGUI$ que contiene los datos necesarios para crear dicho *string*. Estos elementos básicamente contienen el tipo de operador que representan, una lista de elementos primarios, otra de elementos secundarios para representar el divisor o para representar los términos a la derecha de la igualdad, el identificador que mencionamos al comienzo que identifica las estructuras de texto y por último el elemento siguiente.

De esta forma, cuando el estudiante selecciona un operador de la botonera, además de modificar la zona de tipeo, se debe reflejar el operador seleccionado en la lista de elementos utilizando la función $actualizarElementosStringEcuacion$, que recibe por parámetros el tipo de elemento, el identificador del padre y el último identificador utilizado.

Inicialmente la lista comienza con el elemento del tipo $IGUAL$ que contiene en su lista de elementos primarios el identificador de la caja de texto ubicada en la zona izquierda de la ecuación y en la lista de elementos secundarios el identificador de la caja de texto ubicada a la derecha. Entonces, cuando se selecciona, por ejemplo, una potencia cúbica se llama a la operación $actualizarElementosStringEcuacion$ y esta se encarga de buscar en la lista de elementos donde está el padre con el identificador pasado por parámetro y le agrega un elemento nuevo al final de la lista que lo contenga. Este elemento es del tipo $POT\_3$ con un elemento en la lista de elementos primarios del tipo $TEXTO$ e identificado igual al de la caja de texto dentro del operador y como siguiente un elemento de tipo $TEXTO$ e identificado igual al de la caja de texto ubicada en la zona de afuera del operador.

### 3.2.3. Algoritmo de creación de ecuación en notación infija

A continuación se describe el algoritmo recursivo encargado de generar el *string* en notación infija para ser utilizado en la lógica.

```
generarEcuacion(ElementoGUI elemento): String
    Definir variable ecuación igual a string vacío
    Si el elemento es distinto a vacío
        Retornar variable ecuación igual a string vacío
    Si el elemento es distinto a vacío
        Mirar qué tipo de elemento es
            Si es del tipo TEXTO
                Obtener la caja de texto con id igual al elemento.id
                Obtener el texto ingresado en la caja
                    ecuacion = ecuacion + generarEcuacion(elemento.sig)
            Si es del tipo RAIZ_2
                ecuacion = "RAIZ_2(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo RAIZ_3
                ecuacion = "RAIZ_3(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_2
                ecuacion = "POT_2(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_3
                ecuacion = "POT_3(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_1_2
                ecuacion = "POT_1_2(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_1_3
                ecuacion = "POT_1_3(" + generarEcuacion(elemento.elementosPri) +
                ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_MENOS_1
                ecuacion = "POT_MENOS_1(" + generarEcua-
                cion(elemento.elementosPri) + ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_MENOS_2
                ecuacion = "POT_MENOS_2(" + generarEcua-
                cion(elemento.elementosPri) + ")" + generarEcuacion(elemento.sig)
            Si es del tipo POT_MENOS_3
                ecuacion = "POT_MENOS_3(" + generarEcua-
                cion(elemento.elementosPri) + ")" + generarEcuacion(elemento.sig)
```

```
                Si es del tipo DIV
                    ecuacion = "((" + generarEcuacion(elemento.elementosPri) + ")/("
                    + generarEcuacion(elemento.elementosSec) + "))" + generarEcua-
                    cion(elemento.sig)
                Si es del tipo PARENTESIS
                    ecuacion = "(" + generarEcuacion(elemento.elementosPri) + ")" +
                    generarEcuacion(elemento.sig)
                Si es del tipo IGUAL
                    ecuacion = generarEcuacion(elemento.elementosPri) + "=" + gene-
                    rarEcuacion(elemento.elementosSec)
            Retornar string ecuacion
```

Antes de invocar a la lógica con el string obtenido del algoritmo se chequea la existencia de multiplicaciones implícitas y en caso de existir se explicitan, se transforman los signos de - que hacen de negación en el operador unario NEG y se cambian las ',' por '.' en los decimales.

Finalmente se valida que la sintaxis cumpla con los siguiente puntos:

- Que se ingrese algún término en ambos lados de la igualdad.
- Que no exista paréntesis vacíos.
- Que se ingrese la variable y tenga una única ocurrencia.
- Que los decimales están bien definidos.
- Que no existan operadores indefinidos del tipo +, -, o *.

### 3.2.4. Visualización de las ecuaciones

Una vez que la ecuación es guardada localmente, para poder visualizarla se genera una grilla que se rellena a partir del árbol generado en la lógica. La clase *Ecuacion*, en el ámbito de la GUI, es la encargada de generar el *layout* de tipo *GridPane*. El constructor de esta clase determina la dimensión de la grilla para poder recorrer el árbol y rellenarla.

La cantidad de filas se obtiene a partir de la máxima cantidad de cocientes anidados que tiene la ecuación multiplicando el resultado por 2 y sumándole 1. Por ejemplo, si la máxima cantidad de cocientes anidados es 1, la grilla tendrá tres filas donde en la primer fila se representa el término del dividendo, en la segunda la línea del cociente y en la tercera el término divisor. Para obtener la cantidad de columnas se utiliza el siguiente algoritmo.

```
calcularColumnas(Arbol arbol): entero
    Si el árbol es vacío
        Retornar 0
    Sino
        Obtener el tipo del nodo actual
        Si es del tipo operador unario
            Si es una Potencia
                Si el subarbol de la derecha no es un número o variable
                    (Reservar dos columnas para los parentesis + una columna
                    para el exponente + la cantidad de columnas del término in-
                    cluido dentro de la potencia)
                    Retornar 3 + calcularColumnas(arbol.getDer())
                Sino
                    (Reservar una columna para exponente + la cantidad de co-
                    lumnas que formen el número o la variable)
                    Retornar 1+ calcularColumnas(arbol.getDer())
            Si es una raíz
                (Reservar una columna para el símbolo de la raíz + la cantidad de
                columnas del término incluido dentro de la raíz)
                Retornar 1 + calcularColumnas(arbol.getDer())
            Si es una negación
                Si el lado derecho es un operador es posible que lleve paréntesis
                    Si el subárbol derecho es la multiplicación de un número con
                    la variable no lleva paréntesis
                        (Reservar una columna para el símbolo de negación +
                        la cantidad de columnas del término incluido en la ne-
                        gación)
                        Retornar 1 + calcularColumnas(arbol.getDer())
                    Sino, lleva paréntesis
                        (Reservar dos columnas para los parentesis + la columna
                        del operador de negación + la cantidad de columnas del
                        término incluido dentro de la negación)
                        Retornar 3 + calcularColumnas(arbol.getDer())
                Sino, el lado derecho es un número o variable que no lleva paréntesis
                    (Reservar una columna para el símbolo de negación + la can-
                    tidad de columnas del término incluido en la negación)
                    Retornar 1 + calcularColumnas(arbol.getDer()
```

Si es del tipo operador binario

Si es una suma o resta

(Reservar la cantidad de columnas que tiene el término a la izquierda del operador + una columna para el símbolo del operador + la cantidad de columnas del a la derecha del operador)

Retornar 1 + calcularColumnas(arbol.getIzq()) + calcularColumnas(arbol.getDer())

Si es una multiplicación

Si el subárbol izquierdo es un número y el derecha la variable se omite el operador de multiplicación

Retornar calcularColumnas(arbol.getIzq()) + calcularColumnas(arbol.getDer())

Sino

Verificar si el subárbol izquierdo lleva paréntesis por precedencia

cant = 0

Si lleva paréntesis

(Reservar dos columnas para los paréntesis)

cant = 2

Verificar si el subárbol derecho lleva paréntesis por precedencia

Si lleva paréntesis

(Reservar dos columnas para los paréntesis)

cant = cant + 2

Si cant es 0, ninguno de los subárboles lleva paréntesis de precedencia por lo que agrego el símbolo del operador

Retornar 1 + calcularColumnas(arbol.getIzq()) + calcularColumnas(arbol.getDer())

Sino, alguno de los subarboles lleva paréntesis por lo que no es necesario agregar el símbolo del operador

Retornar cant + calcularColumnas(arbol.getIzq()) + calcularColumnas(arbol.getDer())

Si es un cociente

cantIzq = Obtener la cantidad de columnas del subárbol izquierdo

cantDet = Obtener la cantidad de columnas del subárbol derecho

Si cantIzq ¿cantDer

(Reservar la cantidad de columnas que necesita el dividendo)

Retornar cantIzq

Sino

(Reservar la cantidad de columnas que necesita el divisor)

Retornar cantDer

```
        Si es la igualdad
            Retornar 1 + calcularColumnas(arbol.getIzq()) + calcularColumnas(arbol.getDer())
    Si es NUMERO
        (Reservar tantas columnas como cantidad de dígitos que contenga)
        Retornar cantidad de digitos del numero
    Si es la VARIABLE
        Retornar 1
```

Una vez que se obtiene la dimensión de la grilla se calcula la fila central desde donde se comienza a cargar las celdas a medida que se recorre el árbol asociado a la ecuación. La fila central se calcula obteniendo el dividendo que ocupa mas filas entre los cocientes que forman la ecuación. Luego, se comienza a rellenar las celdas recorriendo el árbol con el algoritmo descripto en la función *generarGrilla*.

```
generarGrilla(Arbol arbol): void
    Si el árbol no es vació
        Obtener el tipo del nodo actual
        Si es del tipo operador unario
            Si es una Potencia
                Si el subárbol de la derecha es un operador agrego paréntesis
                    Cargar paréntesis izquierdo en la celda (fila, columna)
                    columna = columna + 1
                    generarGrilla(arbol.getDer())
                    columna = columna + 1
                    Cargar paréntesis derecho en la celda (fila, columna)
                Sino
                    generarGrilla(arbol.getDer())
                Cargar exponente en la celda (fila, columna)
                columna = columna + 1
            Si es una raiz
                cantFilas = (calcularFilas(arbol.getDer()) - 1) / 2
                Cargar símbolo de raíz en la celda (fila + cantFilas, columna)
                columna = columna + 1
                //Marcar borde de las celdas que forman parte del símbolo de la raiz
                i = -cantFilas
                Mientras i ¡cantFilas
                    Marcar borde izquierdo de celda (fila - i, columna)
                    i = i + 1
                Fin mientras
                cantColumnas = calcularColumnas(arbol.getDer())
                i = 0
                Mientras i ¡cantColumnas
                    Marcar borde superior de celda (fila - cantFilas, columna + i,
                    columna)
                    i = i + 1
                Fin mientras
                generarGrilla(arbol.getDer())
```

```
                Si es una negación
                    Cargar símbolo de negación en la celda (fila, columna)
                    columna = columna + 1
                    Si el lado derecho es un operador es posible que lleve paréntesis
                        Si el subárbol derecho es la multiplicación de un número con
                        la variable no lleva paréntesis
                            generarGrilla(arbol.getDer())
                        Sino, lleva paréntesis
                            Cargar paréntesis izquierdo en la celda (fila, columna)
                            columna = columna + 1
                            generarGrilla(arbol.getDer())
                            Cargar paréntesis derecho en la celda (fila, columna)
                            columna = columna + 1
                    Sino, el lado derecho es un número o variable que no lleva paréntesis
                        generarGrilla(arbol.getDer())
            Si es del tipo operador binario
                Si es una suma o resta
                    generarGrilla(arbol.getIzq())
                    Cargar símbolo de suma o resta en la celda (fila, columna)
                    columna = columna + 1
                    generarGrilla(arbol.getDer())
                Si es una multiplicación
                    Si el subárbol izquierdo es un número y el derecha la variable se omite
                    el operador de multiplicación
                        generarGrilla(arbol.getIzq())
                        generarGrilla(arbol.getDer())
                    Sino
                        Si el subárbol izquierdo lleva paréntesis por precedencia
                            Cargar paréntesis izquierdo en la celda (fila, columna)
                            columna = columna + 1
                            generarGrilla(arbol.getIzq())
                            Cargar paréntesis derecho en la celda (fila, columna)
                            columna = columna + 1
                        Sino
                            generarGrilla(arbol.getIzq())
```

```
                    Si ninguno de los subarboles lleva paréntesis agrego el símbolo
                    de la multiplicación
                        Cargar símbolo de multiplicación en la celda (fila, co-
                        lumna)
                        columna = columna + 1
                    Si el subárbol derecho lleva paréntesis por precedencia
                        Cargar paréntesis izquierdo en la celda (fila, columna)
                        columna = columna + 1
                        generarGrilla(arbol.getDer())
                        Cargar paréntesis derecho en la celda (fila, columna)
                        columna = columna + 1
                    Sino
                        generarGrilla(arbol.getDer())
            Si es un cociente
                //Guardar fila y columna actual
                filaAux = fila
                columnaAux = columna
                //Obtener cantidad de columnas dividendo y divisor
                columnasIzq = calcularColumnas(arbol.getIzq())
                columnasDer = calcularColumnas(arbol.getDer())
                //Determinar largo linea de cociente
                Si columnasIzq ¡columnasDer
                    maxColumna = columnasDer
                else
                    maxColumna = columnasIzq
                //Cargar linea de cociente
                i = 0
                Mientras i ¡maxColumna
                    Cargar linea en celda (filaAux, columnaAux + i)
                    i = i + 1
                Fin mientras
```

```
                // Calcular fila central de la grilla para el dividendo
                lst = obtenerSubArbolesCociente(arbol.getIzq()): List¡Arbol¿
                i = 0
                fila = filaAux - 1
                Mientras lst no es vaciá
                    i = calcularFilas(a.getDer())
                    Si fila es mayor a (filaAux - 1) - i
                        fila = (filaAux - 1) - i
                Fin mientras
                columna = columnaAux + (maxColumna - columnasIzq) / 2
                generarGrilla(arbol.getIzq())
                // Calcular fila central de la grilla para el divisor
                lst = obtenerSubArbolesCociente(arbol.getDer()): List¡Arbol¿
                i = 0
                fila = filaAux + 1
                Mientras lst no es vaciá
                    i = calcularFilas(a.getIzq())
                    Si fila es menor a (filaAux + 1) + i
                        fila = (filaAux + 1) + i
                Fin mientras
                columna = columnaAux + (maxColumna - columnasDer) / 2
                generarGrilla(arbol.getDer())
                fila = filaAux
                columna = columnaAux + maxColumna
            Si es la igualdad
                generarGrilla(arbol.getIzq())
                Cargar símbolo de igual en la celda (fila, columna)
                columna = columna + 1
                generarGrilla(arbol.getDer())
        Si es NUMERO
            Por cada digito del numero
                Cargar el digito en la celda (fila, columna)
                columna = columna + 1
        Si es la VARIABLE
            Cargar variable en celda (fila, columna)
            columna = columna + 1
```

### 3.2.5. Resolución de las ecuaciones

Al comenzar la resolución se muestra al estudiante la ecuación y se le permite seleccionar la expresión que desea resolver. Internamente la GUI detecta cuando el estudiante hace *click* sobre una celda de la grilla que representa la ecuación y arrastra el *mouse* sobre el resto de las celdas de la grilla para seleccionar la expresión. Con las celdas seleccionadas identificadas se rearma el *substring* de la ecuación que representa la expresión seleccionada. Mediante una grilla auxiliar[3] que guarda los números de los nodos del árbol que corresponde a cada celda de la grilla visual de la ecuación, se obtienen los nodos seleccionados del árbol para poder generar el *substring* mediante una recorrida inorden[4]. En el ejemplo, de la figura 19, se muestra cómo se obtiene el *substring* de la expresión seleccionada en la grilla de la ecuación. La grilla de la derecha es la auxiliar que contiene los números de los nodos del arbol y a partir de los nodos seleccionados se obtiene el *substring*.

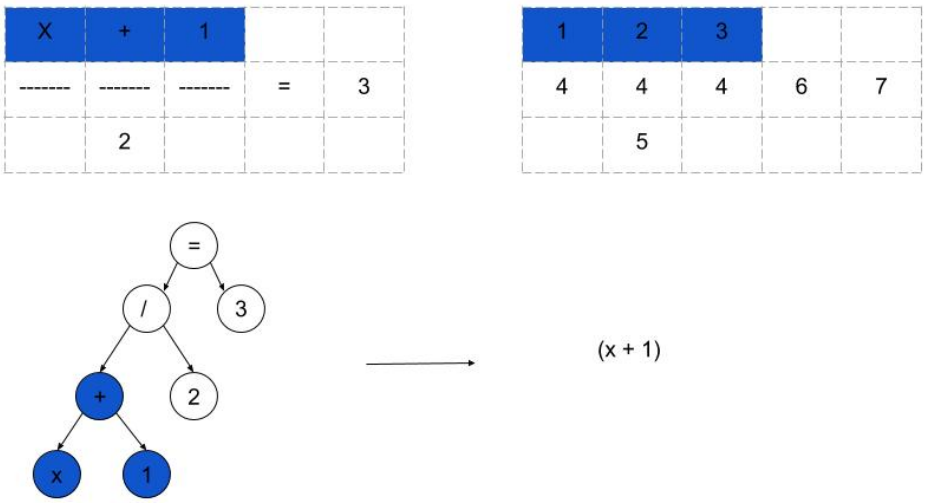



Figura 19: Ejemplo selección de expresión.

Se valida que la expresión seleccionada contenga la variable, que los operadores estén bien definidos y que no incluya el símbolo de igualdad. Si la selección es correcta se genera una nueva zona que llamaremos zona de resolución donde se muestra la expresión seleccionada y se aguarda a que el estudiante ingrese el resultado. En el caso que la selección sea incorrecta se alerta al estudiante mediante un mensaje para que el estudiante vuelva a seleccionar la expresión.

[3]Se crea al momento de generar la grilla visual de la ecuación y es de la misma dimensión.

[4]Se recorre primero los nodos del subarbol izquierdo, después la raíz y por último los nodos del subarbol derecho.

La zona de resolución se arma reutilizando la lógica de la clase *Botonera* pero con la diferencia que en esta etapa el término de la izquierda ya esta definido por la expresión seleccionada y el estudiante solo tiene que ingresar el término de la derecha. Cuando el estudiante ingresa el resultado a validar se invoca el algoritmo *validarResolucion* explicado al comienzo del capítulo. Si es un paso intermedio y es correcto, se muestra que el resultado fue correcto y se espera a que el estudiante seleccione una nueva expresión a resolver. Si es incorrecto se alerta al estudiante y se aguarda a que ingrese un nuevo resultado. Este comportamiento se repite hasta llegar al resultado de la ecuación.

# 4. Arquitectura y Diseño

En este capítulo vamos a presentar la arquitectura de la aplicación y el diseño realizado para poder implementar los requisitos solicitados por el cliente.

## 4.1. Descomposición en Subsistemas

Aquí presentaremos el diagrama de descomposición en subsistemas, en la que se utiliza una arquitectura en capas donde cada una consume los servicios de la capa inferior. El repositorio central brinda servicios que son consumidos desde la capa lógica.

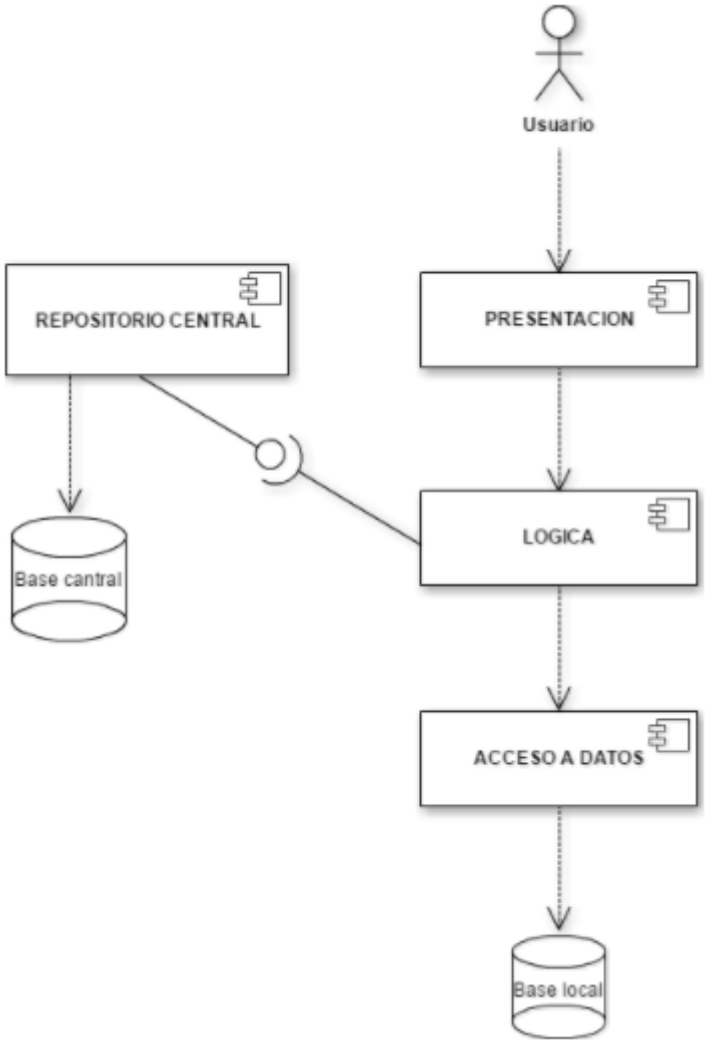



Figura 20: Diagrama de descomposición en subsistemas

La capa Presentación es la encargada implementar la interfaz gráfica del sistema que será accedida por el usuario para interactuar con la capa lógica.

La capa Lógica se encarga de implementar las reglas de negocio necesarias para brindar solución a la capa de presentación, interactuar con la capa de acceso a datos local y acceder a los servicios brindados por el repositorio central. Dentro de esta capa se tiene un controlador de ecuaciones encargado de todo el manejo de las operaciones correspondientes a la lógica de las ecuaciones.

La capa de accesos a datos de la base local, es la encargada de brindar las operaciones necesarias para el acceso y modificación de los datos que se encuentran en la base de datos local en la máquina del estudiante.

El componente Repositorio Central es un web service que provee los servicios para subir y descargar ecuaciones a la base de datos central. En la siguiente figura se detallan sus componentes.

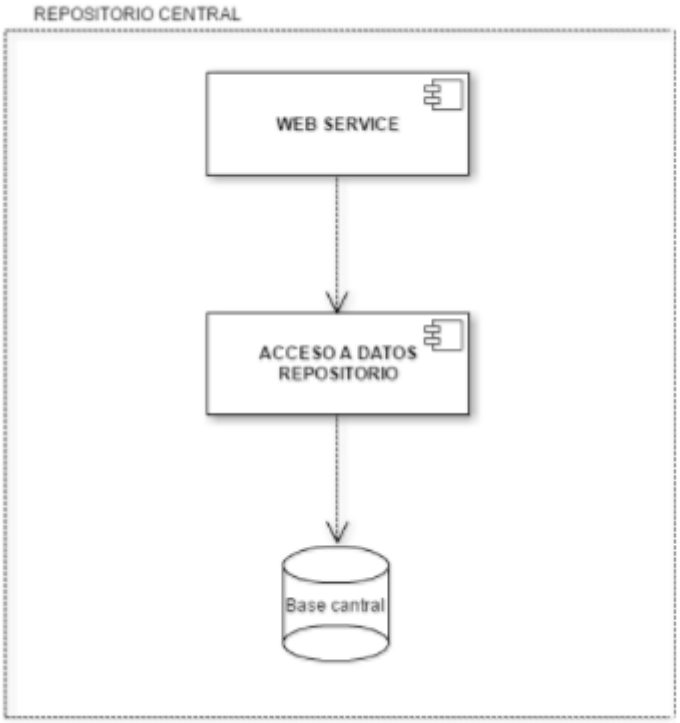



Figura 21: Repositorio Central; Diagrama de descomposición en subsistemas

## 4.2. Vista del Modelo de Distribución

### 4.2.1. Diagrama de Distribución

El modelo de distribución describe la distribución física del sistema en términos de cómo se reparten las funcionalidades entre los equipos que conforman la aplicación.

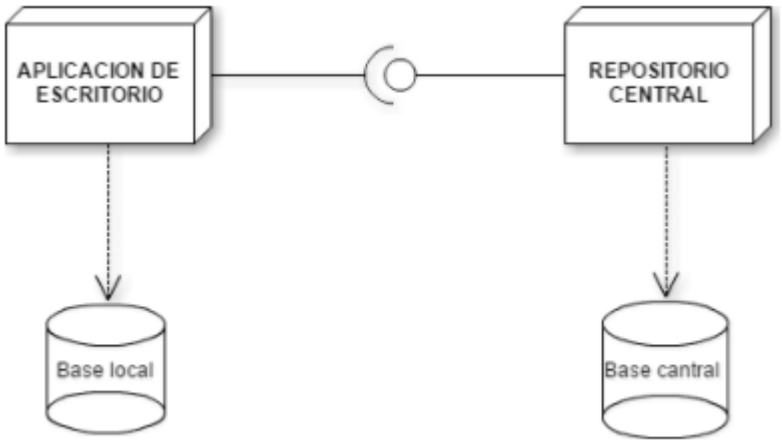



Figura 22: Diagrama de distribución

### 4.2.2. Aplicación de escritorio

En esta aplicación se debe poder crear, eliminar y resolver ecuaciones, almacenar el trabajo realizado por el estudiante en una base de datos local, además de subir y descargar las ecuaciones del repositorio central.

### 4.2.3. Base de datos local

Contiene las ecuaciones iniciales de la aplicación, las ingresadas por el estudiante y las descargadas del repositorio central.

### 4.2.4. Repositorio central

El repositorio central es un servidor que provee servicios web para subir y descargar ecuaciones a la base de datos central.

### 4.2.5. Base de datos central

Contiene las ecuaciones subidas con los servicios provistos por el servidor.

### 4.2.6. Conexión entre aplicación de escritorio y repositorio central

Como mencionamos anteriormente el repositorio central está implementado mediante un web service que brinda los servicios necesarios para la interacción entre la aplicación de escritorio y los datos que se alojan en la base central.

## 4.3. Diagrama de clases

En esta sección se presentará el diagrama de clases reducido correspondiente a las capas lógica y de acceso a datos, en el cual se presentan las clases principales con algunas de las operaciones mencionadas en los algoritmos de la sección 3. El diagrama de clases se encuentra en los anexos en el documento "Descripción de la arquitectura, modelo de diseño y datos".

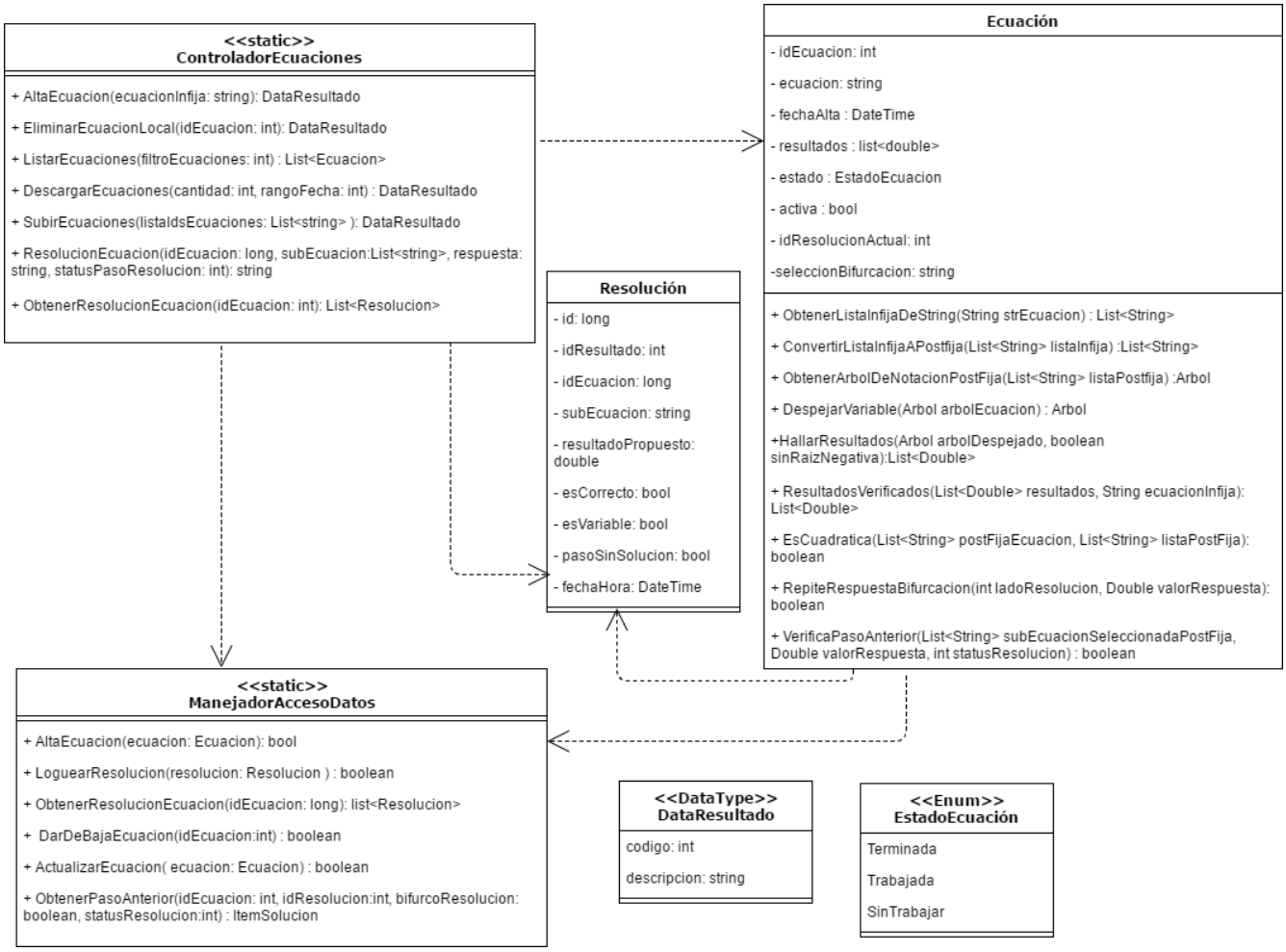



Figura 23: Diagrama de clases reducido.

La clase *ControladorEcuaciones* permite a la capa Presentación acceder a la lógica de la aplicación. Esta clase contiene las operaciones necesarias para que la capa de Presentación pueda implementar los casos de uso solicitados, por ejemplo *AltaEcuacion*, que recibe el string en forma infija y guarda una ecuación con su solución asociada.

En clase *Ecuación* se implementaron la mayoría de los algoritmos necesarios para resolver las ecuaciones. En el diagrama se puede ver que esta clase contiene varias de las funciones utilizadas en los algoritmos del capítulo 3.

*Resolución* representa los datos que se guardan durante la resolución. Cada vez que se selecciona una parte de la ecuación, se le asigna un valor y luego llama al algoritmo para validar el paso, se guarda en la base de datos los datos que contiene la clase *Resolución*. Como se puede ver, además de guardar los identificadores de la ecuación y la resolución, también se guarda si el paso es correcto, si se seleccionó la variable o si el paso no tiene solución. Estos datos son utilizados

en la implementación de los algoritmos.

La clase *ManejadorAccesoDatos* se utiliza como acceso a la capa de Datos. Esta clase permite obtener y guardar las ecuaciones y las resoluciones en la base de datos, además de dar acceso a los datos necesarios para la implementación de los algoritmos.

En el diagrama de clases también se presenta el enumerado *Estado* que define los estados Terminada, Trabajada y SinTrabajar. El estado de la ecuación eliminada se maneja con el atributo `activa` en la clase ecuación. Al crear una ecuación, ésta se asigna como activa, pero si luego de haber sido trabajada se la desea eliminar, se setea como inactiva (`activa`=false).

La mayoría de las operaciones en la clase *ControladorEcuaciones* retornan su resultado con el tipo *DataResultado*. Este clase, permite retornar un código y una descripción a la capa de Presentación. Cuando la ejecución de una función finaliza correctamente se retorna el código 0 con la descripción éxito. En el caso en que haya un error, se retorna un código de error y una descripción del error.

## 4.4. Modelo de Datos

A continuación se presentará el modelo conceptual de la base de datos local y la del Repositorio Central.

### 4.4.1. MER - Local

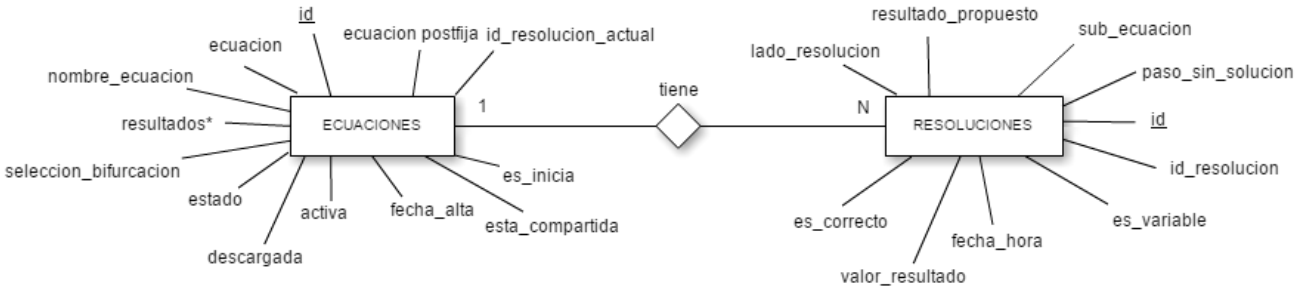



Figura 24: Modelo de entidad-relación de la base de datos local.

### 4.4.2. MER - Repositorio Central

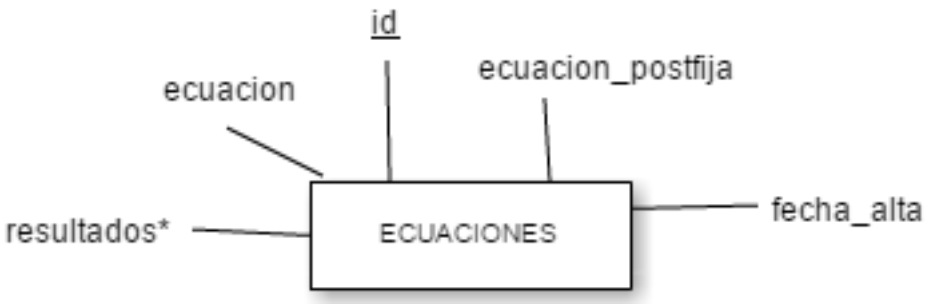



Figura 25: Modelo de entidad-relación - Repositorio central

## 4.5. Diseño de Interfaz de Usuario

Uno de los requisitos solicitado por el cliente es que la aplicación tenga una interfaz atractiva. Para poder cumplir con este requisito, se realizó un mockup para cada una de las vistas de la aplicación[5], y se le presentó al cliente para confirmar que sea de su agrado. Además, el diseño de las vistas de crear ecuación y resolver ecuación, se realizó muy similar a la aplicación *Cover up* presentada anteriormente, de forma que al cliente le resulte familiar utilizar la aplicación.

Como la vista principal de la aplicación contiene el listado de las ecuaciones, en el que cada ecuación puede tener un color distinto según su estado (Sin trabajar, Trabajada, Finalizada o Eliminada), para determinar con qué color identificar que estado, se realizó una prueba con tres grupos de ciclo básico. La prueba consistió en presentar ecuaciones con distintos colores, para que los estudiantes indiquen qué estado de la ecuación asocian con qué color. Luego se analizaron las respuestas para poder determinar los colores a asociar con cada estado, y se determinó que el color para las ecuaciones sin trabajar es el azul, el color de las ecuaciones trabajadas es el naranja y el color verde para las ecuaciones finalizadas. El color de las ecuaciones eliminadas es el gris, pero este estado no se presentó en el estudio ya que es un color muy utilizado para indicar opciones deshabilitadas, el cual era nuestro fin.

Una vez que la aplicación fue desarrollada respetando los diseños presentados y los colores seleccionados por los estudiantes, el cliente quedó satisfecho con la interfaz de la aplicación.

[5]El diseño de la interfaz de usuario se puede encontrar en los anexos.

# 5. Implementación

La aplicación solicitada fue desarrollada y probada en el sistema operativo Ubuntu 14.04 para simular lo mejor posible el sistema operativo de los dispositivos del Plan Ceibal. La implementación de la aplicación fue desarrollada en Java 8, y la interfaz gráfica fue desarrollada con JavaFX debido a que permite la creación de aplicaciones de escritorio utilizando hojas de estilo CSS con las que estamos familiarizados.

El motor de base de datos utilizado para la aplicación de escritorio es SQLite, este motor fue elegido debido que es una base de datos liviana, utilizada usualmente en dispositivos móviles, lo cual nos favorece debido a las restricciones de hardware que tienen los dispositivos del Plan Ceibal. Además tiene la ventaja de ser compatible con varias plataformas, entre las que se encuentra Java.

En el repositorio central se implementaron servicios SOAP utilizando JAX-WS. En el repositorio se crearon dos interfaces, una para subir ecuaciones y otra para descargar las ecuaciones. Para levantar la aplicación web se puede utilizar Tomcat 8 debido a que fue utilizado durante el desarrollo de la aplicación funcionando correctamente.

El sistema de base de datos que se utilizó para el repositorio central fue Postgresql y se utilizó Hibernate para facilitar la implementación de la inserción y obtención de datos desde la base de datos.

## 5.1. Instalador

El instalador crea automáticamente la carpeta donde se van a guardar todos los archivos necesarios para utilizar la aplicación. En el proceso de instalación se descarga mediante Internet el jre8 necesario para ejecutar la aplicación. En el siguiente enlace se puede encontrar el instalador en sus dos versiones, una de 64 bits y otra de 32 bits:

https://drive.google.com/drive/folders/0B7ZkXxrHN3hYXzg3dnpwdHN5VDA?usp=sharing

Para poder subir y descargar ecuaciones desde la aplicación, es necesario tener acceso al servidor del repositorio central. En el enlace mostrado anteriormente, también se puede obtener el manual de instalación, que indica el archivo que se debe modificar para realizar la comunicación.

Actualmente el repositorio central no se encuentra levantado debido a que no se pudo obtener un servidor en el cual instalarlo. Igualmente, en el caso en que se instale en un servidor público, en la misma carpeta antes compartida publicaremos la dirección del servicio al que se debe apuntar.

## 5.2. Manual de usuario

Como manual de usuario se solicitó un video explicativo que presente las distintas funcionalidades que tiene la aplicación. Este video se encuentra publicado en el mismo enlace indicado en la

sección anterior.

# 6. Testing

Las pruebas para verificar nuestro sistema se dividieron en tres etapas, la primera fue la prueba de los algoritmos, la segunda fue la prueba del sistema y la última fue las pruebas con usuarios finales.

## 6.1. Pruebas de los algoritmos

Para probar los algoritmos y el resto de las funciones de la lógica de la aplicación, se realizaron pruebas unitarias. Estas pruebas se ejecutaron durante el desarrollo de las funcionalidades con el fin de ir asegurando la correctitud de las mismas. A su vez, estas pruebas, permitieron realizar pruebas de regresión cuando se realizaban modificaciones a operaciones ya implementadas.

## 6.2. Pruebas del sistema

Se realizaron pruebas funcionales, para las cuales se diseñaron los casos de prueba para verificar los casos de uso de la aplicación. Estas pruebas se realizaron luego de desarrolladas todas las funcionalidades de la aplicación, y finalizaron cuando todos los casos de prueba fueron aprobados. En esta etapa también fueron incluídas las pruebas de integración, para verificar la comunicación entre el repositorio central y la aplicación de escritorio, y comprobar que se permita subir y descargar ecuaciones del mismo. Una vez que se aprobaron todos los casos de prueba y se encontró implementado el instalador de la aplicación, se realizó la prueba de instalación.

Esta prueba se planificó a través del cliente con autoridades del liceo de Shangrilá. Previamente se relevó el equipamiento del liceo, constatándose que las computadoras de la sala de máquinas contaban con el sistema operativo Ubuntu y eran de 32 bits al igual que las Magallanes del Plan Ceibal. Nuestro instalador fue generado para computadoras de 64 bits como las Positivo del Plan Ceibal (se contó con una de ellas durante la realización del proyecto), por lo tanto fue necesario generar un instalador para 32 bits. Una vez generado, se concurrió al liceo y se instaló la aplicación en todas las máquinas disponibles.

## 6.3. Pruebas con usuarios finales

Durante todo el desarrollo se interactuó frecuentemente con el cliente que pudo probar los avances y las funcionalidades desarrolladas. Además de ello, se realizaron varias pruebas con usuarios finales de la aplicación, desde profesores y estudiantes de profesorado de Matemática, hasta estudiantes de primero y segundo año del ciclo básico. Se describen a continuación.

### 6.3.1. CUREM

La primera prueba realizada con usuarios finales consistió en la presentación, por parte de nosotros y del cliente, de nuestro producto con el grado de desarrollo alcanzado hasta ese momento, en el CUREM 7 ($7^o$ Congreso Uruguayo de Educación Matemática), que se llevó a cabo en Montevideo el 21 de Mayo de 2017. La presentación fue en modalidad de taller, donde los profesores de matemática tuvieron la oportunidad de utilizar la aplicación para resolver distintas ecuaciones así como de discutir distintos aspectos de su utilización en el aula. Por ejemplo, surgió la necesidad de permitir al usuario ingresar la respuesta Conjunto Vacío, de forma que al detectar que la ecuación no tiene solución, los estudiantes no estén obligados a realizar los cinco intentos para finalizar la resolución. El público se notó claramente interesado en la aplicación y dispuesto a incorporarla en sus clases de Matemática.

### 6.3.2. DAED

Como uno de los estudiantes partícipes en este proyecto realizó en simultáneo el curso de Didáctica de Algoritmos y Estructuras de Datos (DAED), aprovechó a realizar la tarea final de esta asignatura utilizando la aplicación desarrollada en este proyecto de grado. Para el trabajo, se realizó una experiencia piloto con cuatro estudiantes de primero de liceo que todavía no habían trabajado con ecuaciones.

Además de cumplir con los objetivos de la asignatura, el estudio buscó comprobar la usabilidad del producto y la naturaleza de las instrucciones necesarias para que pudieran utilizar la aplicación estudiantes novatos. Se pudo ver que en todos los casos, breves explicaciones sobre algunas funcionalidades habilitaron a los estudiantes a trabajar con desenvoltura, incluso con aquellas funcionalidades que no se explicaron anteriormente.

### 6.3.3. Liceo de Shangrilá

En esta prueba se simuló el uso normal de la aplicación en clase, con un grupo de aproximadamente 20 estudiantes de segundo año del liceo Shangrilá y con su profesora presente, a pesar de que fue el cliente quién asumió el rol de profesor dado que era la primera vez que se presentaba el software. Para comenzar con la actividad se proyectó en un televisor la aplicación, con el fin de trabajar en conjunto en la creación y resolución de ecuaciones. Luego se dividió la clase en grupos de tres estudiantes, entre quienes se repartieron las computadoras del Plan Ceibal, en las que previamente se había instalado la aplicación, para que la utilizaran para resolver un conjunto de ecuaciones planteadas por el cliente. Durante el trabajo los estudiantes contaron con la asistencia tanto del cliente como de la profesora del grupo, quien a su vez aprendió a usar la aplicación [6]. La clase transcurrió según lo esperado, lo que permitió concluir que el desarrollo de la aplicación se encontraba finalizado con éxito.

[6] Actualmente se cuenta con un video como manual de usuario

### 6.3.4. CIBEM

Del 10 al 14 de julio de 2017 se llevó a cabo en Madrid (España) el VIII Congreso Iberoamericano de Educación Matemática CIBEM, que reúne cada año a profesores e investigadores en educación matemática de varios países europeos y latinoamericanos.

Las profesoras Teresa Pérez y Nora Ravaioli dictaron un taller utilizando nuestro software, titulado "UNA APUESTA AL DESARROLLO DEL PENSAMIENTO FLEXIBLE. - LA RESOLUCIÓN DE ECUACIONES" y lo presentaron en una ponencia titulada "DISEÑANDO UN SOFTWARE EDUCATIVO. UN TRABAJO COLABORATIVO ENTRE ESTUDIANTES DE INGENIERÍA EN COMPUTACIÓN Y DOCENTES DE DIDÁCTICA DE LA MATEMÁTICA". En ambas instancias recibieron elogiosos comentarios de los participantes. El taller se puede encontrar el resumen en el libro de resúmenes del congreso con el número 153 y la ponencia en el programa con el número 192 de (http://www.cibem.org/index.php/es/)

# 7. Conclusiones y trabajo a futuro

## 7.1. Conclusiones

El objetivo de nuestro proyecto fue la creación de un software que permitiera resolver ecuaciones utilizando la estrategia cover up y resolviera las especificaciones del cliente (mencionadas en la sección 1.2). Para cumplir con los objetivos se realizó un análisis para diseñar la solución al sistema y determinar las herramientas más adecuadas a utilizar en el desarrollo de la aplicación. Además se realizaron mockups de la interfaz de usuario para asegurarnos el cumplimiento del requisito de amigabilidad por parte del usuario. Para lograr la conformidad del cliente, durante el desarrollo de la aplicación se realizaron distintas presentaciones con el fin de realimentar nuestro producto, y además, se realizaron distintas pruebas para asegurar que la aplicación desarrollada cumpliera con los casos de uso especificados y satisficiera las expectativas del cliente.

A su vez, el haber obtenido el interés de los profesores de Matemática, nos hace concluir que el producto fue bien logrado, debido a que la intención de incluir nuestro producto en sus clases, nos da a entender que que el producto satisface las expectativas y constituye un aporte didáctico de calidad.

## 7.2. Trabajo a futuro

Uno de los requisitos solicitados por el cliente fue el permitir almacenar el trabajo del estudiante para que el docente pueda acceder al mismo. Actualmente, el trabajo queda guardado en la computadora del estudiante, es decir, los docentes pueden ver la resolución en la computadora del mismo o se puede generar una captura de la pantalla en la cual se puede visualizar la resolución. Una mejora a realizar es que los docentes no necesiten solicitar una copia de la captura a los estudiantes o acercarse a su computadora, sino ver los trabajos desde la suya. Esto se puede realizar desarrollando una nueva aplicación que permita visualizar el trabajo realizado por los estudiantes, y aprovechar el repositorio central para guardar las ecuaciones asociadas con sus resoluciones. Para esto, también es necesario que la nueva aplicación permita el registro de usuarios, de forma que las resoluciones estén asociadas a un estudiante, y los docentes puedan visualizar los trabajos de cada uno.

Además, en nuestra aplicación, la descarga de las ecuaciones se realiza en forma aleatoria, indicando la cantidad y un rango de fechas determinado se descarga la cantidad especificada de ecuaciones, que se seleccionan aleatoriamente dentro del rango de fechas especificado. Una nueva funcionalidad que se puede agregar, es poder elegir desde la aplicación de escritorio las ecuaciones a descargar, ya sea listando las ecuaciones del repositorio central o permitir descargar un grupo de ecuaciones antes nombrado. Esta funcionalidad se puede realizar ampliando la aplicación de escritorio, con el fin de permitir nombrar un conjunto de ecuaciones a subir. Por lo tanto, si el repositorio central cuenta con conjuntos de ecuaciones nombrados, se podría realizar la descarga de las ecuaciones asociadas a un nombre, y por ende, desde la aplicación de escritorio se podría

buscar a partir de los nombres los grupos de ecuaciones a descargar. Esta funcionalidad puede resultar de utilidad a los docentes, ya que les permitiría crear tareas con anticipación y solicitar a los estudiantes que descarguen un conjunto de ecuaciones armado para la actividad.

Otra utilidad de desarrollar una nueva aplicación que permita la visualización de las resoluciones, es el facilitar la incorporación de tareas de administración que aporten valor a los docentes a la hora de utilizar la aplicación.

# 8. Referencias

1. Las estrategias aritméticas como recurso en el primer acercamiento a la resolución de ecuaciones: la transición de la aritmética al álgebra - Nora Ravaioli y Teresa Pérez - Diciembre 2014

2. WolframAlpha: https://www.wolframalpha.com/ - 01/07/2017

3. Mathway: https://www.mathway.com/Algebra - 01/07/2017

4. Ecuation Solving: https://maple.cloud/doc=16205006 - 01/07/2017

5. PAT2Math: http://pat2math.unisinos.br/ - 30/07/2017

6. Plan Ceibal: http://www.ceibal.edu.uy/es - 06/07/2017

7. JavaFX: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm - 06/07/2017

# Índice de figuras

# 9. Anexos

## 9.1. Anexo A

- Formato de operaciones
- Algoritmos para la resolución de la ecuación
- Pseudocódigos de casos de uso
- Mensajes de interacción con el usuario

## 9.2. Anexo B

- Prototipo Electron
- Prototipo SQLite
- Interfaz de usuario
- Documento de riesgos
- Descripción de la arquitectura, modelo de diseño y datos
- Especificación de requisitos de software
- Estandar de implementación
- Plan de SQA
- Plan de configuración
- Plan de verificación y validación
- Alcance del sistema
- Modelo de casos de prueba
- Modelo de casos de uso
- Plan de proyecto
- Glosario
- Prueba de performance
- Acta de reunión de requisitos 1

- Acta de reunión de requisitos 2
- Acta de reunión de requisitos 3
- Acta de reunión de requisitos 4
- Acta de reunión de requisitos 5
- Acta de reunión de requisitos 6
- Acta de reunión de requisitos 7

# Anexos

Martín Poli - Camila Rojí

**Tutores**

Sylvia da Rosa - Federico Gómez

**Clientes**

Nora Ravaioli - Teresa Pérez

UdelaR-InCo Montevideo, Uruguay

Agosto 2017

# Anexo A

Formato de operaciones

Algoritmos para la resolución de la ecuación

Pseudocódigos de casos de uso

Mensajes de interacción con el usuario

# Formato de operaciones
# Versión 1.0

En este documento vamos a describir mediante gramáticas y diagramas, las ecuaciones que se deben poder crear y resolver.

Los operadores binarios se describen con los siguientes caracteres:

Suma : "+"
Resta :  "-"
Cociente: "/"
Multiplicación: "*"
Igual:  "="

Los operadores unarios se codifican de la siguiente forma dentro del string de la ecuación:

Raíz cuadrada : "raiz2"
Raíz cúbica: "raiz3"
Potencia cuadrada:  "pot2"
Potencia cúbica: "pot3"
Potencia a la un medio: "pot1_2"
Potencia a la un tercio: "pot1_3"
Potencia a la menos uno:  "pot_1"
Potencia a la menos dos: "pot_2"
Potencia a la menos tres: "pot_3"
Opuesto: "neg"

Las expresiones unitarias y binarias con variable y sin variable, se describen con gramáticas, debido a que la utilización de paréntesis no permite describir la ecuación mediante diagramas.

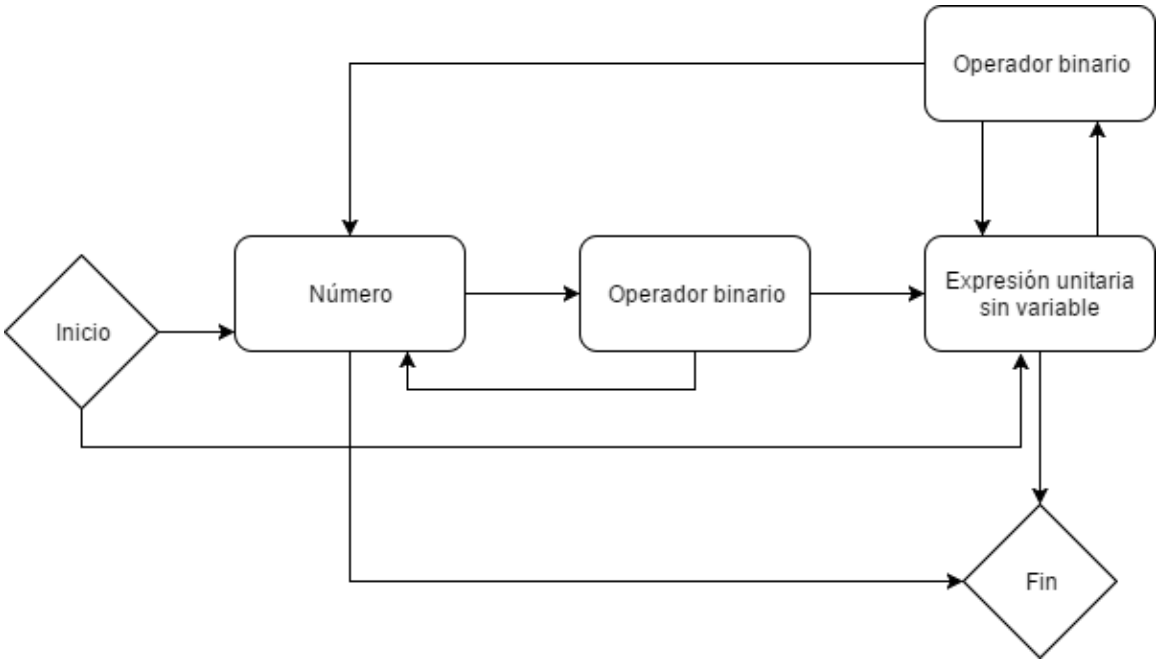
**Expresiones unitarias sin variable:**
S-> E
E-> U(A) | U(E)
A -> Expresión binaria sin variable
U -> Operadores unitarios

**Expresiones unitarias con variable**
S-> E
E -> U(V) | U(E)
V -> Expresión binaria con variable
U -> Operadores unitarios

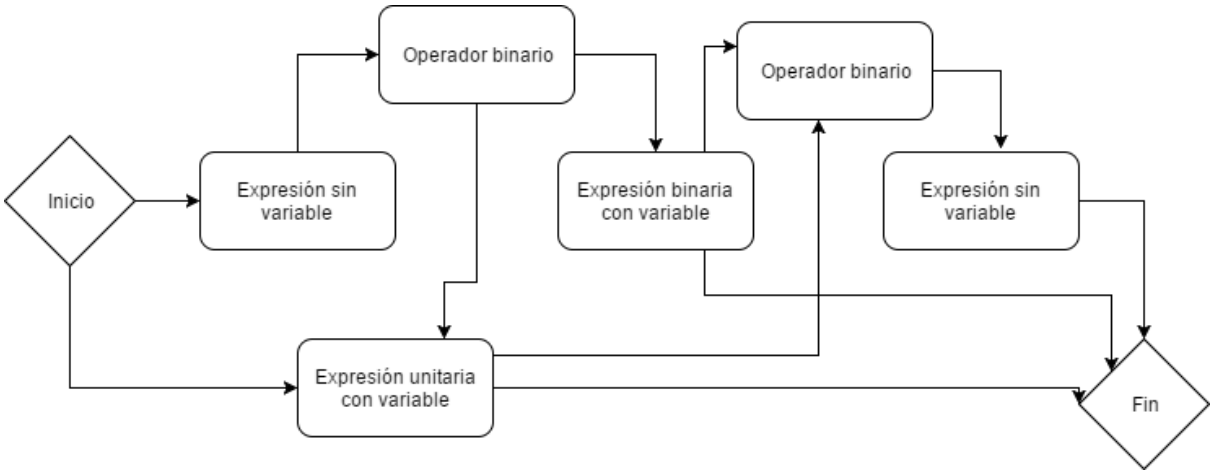
**Expresión binaria sin variable:**
S-> C
C -> BOB | B
B -> (B) | NON| N
N-> número
O-> operador binario

**Expresión binaria con variable**
S->V
V->BOX | XOB| X
X->a..z
B->Expresión sin variable
O-> operador binario

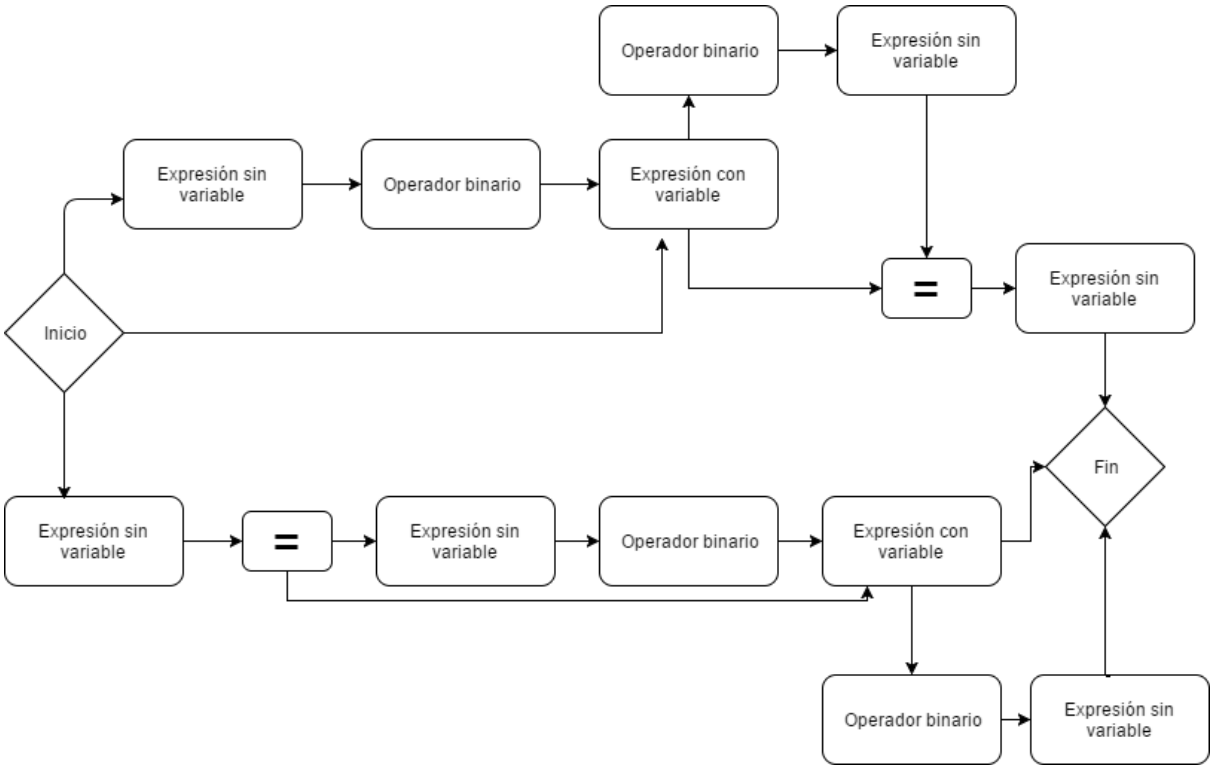
**Expresión sin variable**




**Expresión con variable**




**Formato ecuación**



# Algoritmos para la resolución de la ecuación

# Versión 1.0

Cuando un estudiante crea la ecuación, el sistema la resuelve previamente a guardarla de forma de ya tener conocimiento sobre si tiene solución o no, y cuántas soluciones tiene. Para resolver la ecuación se realizan una secuencia de distintos algoritmos. Primero, teniendo como entrada el texto de la ecuación que creò el estudiante en notación infija, se modifica la notación a postfija para luego generar el árbol que representa a la ecuación. Decidimos generar el árbol a partir de la notación postfija debido a que investigando encontramos un algoritmo que nos permite crear el árbol fácilmente utilizando dicha notación.
Una vez teniendo la ecuación representada en un árbol se despeja la variable, quedando un árbol que sólo contiene la variable y otro árbol que contiene las operaciones a resolver y de esta forma determinar el resultado de la variable.

A continuación especificamos los operadores que se pueden utilizar para crear ecuaciones con el orden de precedencia (de mayor a menor). Se realiza esta especificación porque es necesaria para la ejecución de los algoritmos de resolución. Los operadores que cuenten con la misma precedencia se especificarán en la misma línea.

**Operadores binarios:**
Potencia
Multiplicación, Cociente
Suma, Resta
Igualdad

**Operadores unarios:**
Raíz cuadrada, Raíz cúbica
Negación

En las siguientes secciones vamos a especificar los algoritmos antes mencionados, “Conversión de notación infija a postfija”, “Traducción de notación postfija a árbol binario”,

### Conversión de notación infija a postfija

El siguiente algoritmo en pseudocódigo traduce una expresión en notación infija a notación postfija. Este algoritmo fue tomado de la referencia [1]

**Entrada:** Una lista que contiene los términos de la ecuación en notación infija.
**Salida:** Una lista que contiene los términos de la ecuación en notación postfija.

**INICIO**
Crear pila y la lista de salida, inicialmente vacías.
**MIENTRAS** lista de entrada no esté vacía y no se ha encontrado ningún error **HACER**

```
        Extraer el primer término de la lista (E)
        SEGÚN-SEA E
                CASO E es número:
                        Insertar E al final de la lista de salida
                CASO E es la variable:
                        Insertar E al final de la lista de salida
                CASO E es un paréntesis izquierdo:
                        Insertar E en la pila
                CASO E es un paréntesis derecho :
                        MIENTRAS La pila no esté vacía y su cima no sea un paréntesis
izquierdo HACER
                                Extraer elemento de la pila
                                Insertarlo al final de la lista de salida
                        FIN-MIENTRAS
                        SI Encontramos el paréntesis izquierdo ENTONCES
                                Extraerlo de la pila y destruirlo
                        SINO
                                Se ha detectado un ERROR
                        FIN-SI
                        Destruir E
                CASO E es un operador :
                        MIENTRAS La pila no esté vacía y su cima sea un operador de
precedencia mayor o igual que la de E HACER
                                Extraer elemento de la pila
                                Insertarlo al final de la lista de salida
                        FIN-MIENTRAS
                        Insertar E en la pila
        FIN-SEGÚN-SEA
      FIN-MIENTRAS

   MIENTRAS Pila no esté vacía HACER
        Extraer elemento de la pila
        Insertarlo al final de la lista de salida
   FIN-MIENTRAS
   Destruir pila
FIN
```

**Traducción de notación postfija a árbol binario**

El siguiente algoritmo traduce de una notación infija a un árbol binario. Al igual que el algoritmo anterior fue tomado de la referencia [1] pero se le realizaron modificaciones para agregarle los operadores unarios ya que este algoritmo sólo consideraba los operadores binarios.

**Entrada:** La lista obtenida en el algoritmo anterior, que contiene los términos de la ecuación en notación postfija.
**Salida:** Un árbol binario que representa la ecuación.

**INICIO**
Crear pila y árbol, inicialmente vacíos.
**MIENTRAS** lista de entrada no esté vacía y no se ha encontrado ningún error **HACER**
Extraer el primer término de la lista (lo llamaremos **E**)
**SEGÚN-SEA** E
**CASO** E es número**:**
Insertar E en la pila
**CASO** E es la variable**:**
Insertar E en la pila
**CASO** E es una expresión (un árbol)**:**
Insertar E en la pila
**CASO** E es un paréntesis izquierdo**:**
Se ha detectado un **ERROR**
**CASO** E es un operado binario**:**
**SI** La pila tiene menos de dos elementos **ENTONCES**
Se ha detectado un **ERROR**
**SINO**
Extraer elemento de la pila (lo llamaremos A2)
Extraer elemento de la pila (lo llamaremos A1)
Crear un árbol donde la raíz contenga al operador E, el hijo izquierdo sea A1 y el hijo derecho sea A2
Insertar el árbol en la pila
**FIN-SI**
**CASO** E es un operado unario**:**
**SI** La pila tiene menos de un elemento **ENTONCES**
Se ha detectado un **ERROR**
**SINO**
Extraer elemento de la pila (lo llamaremos A1
Crear un árbol donde la raíz contenga al operador E, el hijo izquierdo sea Null y el hijo derecho sea A1
Insertar el árbol en la pila

**FIN-SEGÚN-SEA**

**FIN-MIENTRAS**

**SI** pila vacía o con más de un elemento **ENTONCES**
Se ha detectado un **ERROR 3**
**SINO**
Extraer elemento de la pila (lo llamaremos E)
**SI** Elemento no es una expresión (un arbol) **ENTONCES**
Convertir E en un arbol con hijo izquierdo y derecho vacíos
**FIN-SI**
El resultado del algoritmo (el arbol de salida) es E
**FIN-SI**
**{** Borrado de la pila, si se ha producido error **}**
**MIENTRAS** pila no esté vacía **HACER**
Extraer elemento de la pila
Destruir elemento
**FIN-MIENTRAS**
Destruir pila
**FIN**

**Algoritmo para despejar la variable**

El siguiente algoritmo despeja la variable de la ecuación, generando un árbol que contiene la variable y otro que contiene las operaciones a resolver. Este algoritmo fue realizado por los integrantes del equipo analizando con distintas ecuaciones de forma de ir verificando los pasos planteados.

Nos paramos en la raíz del àrbol y nos fijamos de que lado està la x
Creamos dos àrboles, uno con el sub arbol que contiene la x (A), y otro con el sub arbol que no la tienen(B).
MIENTRAS la raíz no sea la variable
Nos paramos en la raìz del arbol que contiene la x (A)
Si la raíz es un operador binario
Nos fijamos de que lado està la x y tomamos el sub arbol que no la tiene (A1)
Creamos un nuevo àrbol con el operador opuesto en la raíz y asociamos A1 al sub àrbol derecho. y al sub àrbol izquierdo le asociamos B
Asignamos el àrbol recièn creado a B.
Asigmanos el sub àrbol derecho de A, que serìa (A2) a A.
Si la raìz es un operador unario
Si es raiz cuadrada
Creamos un àrbol con la raìz potencia
Asigmanos 2 al sub arbol derecho
Asignamos B al sub arbol izquierdo
Si es raíz cubica
Creamos un àrbol con la raìz potencia
Asigmanos 3 al sub arbol derecho

Asignamos B al sub arbol izquierdo

Si es el símbolo negativo

Cramos un árbol con + en la raíz

Asignamos el sub arbol derecho de A a al sub àrbol derecho del àrbol recién creado

Asignamos al sub arbol izquierdo del arbol recién creado B

Asignamos el árbol recien creado a B

Asigmanos el sub àrbol derecho de A, que serìa (A2) a A.

FIN MIENTRAS

**Resoluciòn del árbol B para hallar la variable**

El siguiente algoritmo recorre el árbol que contiene las operaciones a resolver y devuelve una lista con los resultados que satisfacen la ecuación. En este algoritmo se retorna una lista debido a que la raíz cuadrada tiene como resultado la raíz y el opuesto.

```
try
{
Si la raíz no es null
        a = Recursión sobre el àrbol izquierdo
        op = Obtengo el valor de la raíz
        if(op es numérico)
                listRespuestas.Add(op)
                return listRespuestas
        else
                switch(op)

                        +: lista = Recursiòn sobre el àrbol derecho
                                foreach( b in lista)
                                        listRespuestas.add(a + b)
                                  return listRespuestas.
                        -: lista = Recursiòn sobre el àrbol derecho
                                foreach( b in lista)
                                        listRespuestas.add(a - b)
                                  return listRespuestas.
                        *: lista = Recursiòn sobre el àrbol derecho
                                foreach( b in lista)
                                        listRespuestas.add(a * b)
                                  return listRespuestas.
                        /: lista = Recursiòn sobre el àrbol derecho
                                foreach( b in lista)
                                        listRespuestas.add(a / b)
                                  return listRespuestas.
                        potencia: lista = Recursiòn sobre el àrbol derecho
                                foreach( b in lista)
                                        if(b = 0.5)
                                {
                                        p = a pot b
                                        listRespuestas.add(p)
                                        listRespuestas.add(-p)

                                }
                                else
                                {
                                        listRespuestas.add(a pot b)
```

```
                    }
                    return listRespuestas.
            raiz cuadrada : b =  Recursiòn sobre el àrbol derecho
                    r = raiz cuadrada de b
                    return [r, -r]
            raíz cúbica :
                    lista = Recursiòn sobre el àrbol derecho
                    foreach( b in lista)
                            listRespuestas.add(raíz cubica de b)
                      return listRespuestas.
            negativo: lista = Recursiòn sobre el àrbol derecho
                    foreach( b in lista)
                            listRespuestas.add(0 - b)
                      return listRespuestas.



}
catch(MathException e)
{
        “La ecuación no tiene solución.”
}
```

**Referencias**

[1] http://www.infor.uva.es/~cvaca/asigs/AlgInfPost.htm - 21/09/2016

# Pseudocódigos de casos de uso

# Versión 1.0

```
String Alta ecuación(string ecuacionInfija)
{
	String mensajeRespuesta = “Ocurrió un error dando de alta la ecuación”;
	Ecuacion ecuacion = new Ecuación(ecuacionInfija);
	//Se verfica que la ecuación haya quedado bien formada
	if(ecuacion.getCodigoRespuesta() != 0)
	{
		return ecuacion.getDescripcionRespuesta();
	}
	double[] resultados = ecuacion.ObtenerResultados();
	Xml xmlArbol = ecuacion.ObtenerXMLArbolEcuacion();
	Date fechaAlta = Date.Today;
	String estado = EstadoEcuacion.SinTrabajar;
	bool activa = true;

	bool ok = ManejadorAccesoDatos.AltaEcuacion(ecuacionInfija, resultados, xmlArbol,
	fechaAlta, estado, activa);

	if(ok)
	{
		mensajeRespuesta = “Ecuación guardada correctamente”;
	}
	return mensajeRespuesta;
}

List<Ecuacion> ListarEcuaciones(bool sinTrabajar)
{
	List<Ecuacion> listadoEcuaciones =
	ManejadorAccesoDatos.ObtenerListadoEcuaciones(sinTrabajar);
	return listadoEcuaciones;
}

String EliminarEcuacionLocal(long idEcuacion)
{
	Respuesta respuesta = ManejadorAccesoDatos.EliminarEcuacionLocal(idEcuacion);
	if(respuesta.getCodigo() == 0)
	{
		return “Ecuación eliminada correctamente.”;
	}
	else
```

```
        {
                return respuesta.getDescripcion();
        }
}


String ResoluciónEcuacion(int idEcuacion, string subEcuacionStr, double respuesta)
{
        //Las respuestas de esta funcion son:
        // PC : La respuesta del paso es correcta. La subEcuacion no es una variable (Paso
        correcto)
        // RC: La respuesta es correcta y la subEcuacion es una variable pero quedan mas
        soluciones. (Resultado correcto)
        // SS: La ecuacion no tiene solucion y se superaron la cantidad de intentos. (sin
        solucion)
        // PI : La respuesta del paso es incorrecta.(Paso incorrecto)
        // RT: Se ingresaron todas las soluciones de la ecuación correctas. (Resolución
        terminada)
        // SI: Sintaxis inválida
        // RR: Resultado correcto, pero ya fue ingreasdo. (Repite resultado)

        String resultadoRetornar = "";
        Ecuacion subEcuacion = new Ecuacion(subEcuacionStr);
        if(ecuacion.getCodigoRespuesta() != 0)
        {
                return "SI";
        }

        bool pasoSinSolucion = false;
        bool correcto = false;
        Ecuacion ecuacion = ManejadorAccesoDatos.ObtenerEcuacion(idEcuacion);
        //Tiene soluciones
        double[] resultados = ecuacion.getResultado();
        if( resultados != null && resultados .count > 0)
        {
                foreach(double var in resultados)
                {
                        double res = subEcuacion.Evaluar(var);
                        if (res == respuesta)
                        {
                                correcto == true;
                                break;
                        }
                }
                if(subEcuacion.length() == 1)
                {
```

```
    if(correcto)
    {
        //Es una variable
        double[] solucionesLogueadas =
        ManejadorAccesoDatos.ObtenerSolucionesLogueadasDistintas(idEcu
        acion, ecuacion.getIdResolucion());
        if(!solucionesLogueadas.contains(respuesta))
        {
                if(solucionesLogueadas.count() +  1 == resultados.count())
                {
                        //Encontrò todas las soluciones
                        resultadoRetornar = “RT”;
                }
                else
                {
                        //Faltan encontrar màs soluciones
                        resultadoRetornar = “RC”;
                }
        }
        else
        {
                resultadoRetornar = “RR”;
        }
        }
     else
    {
                resultadoRetornar = “RI”;
    }
}
else
{
        //Selecciona conjunto de terminos
        foreach(double var in resultados)
        {
                double res = subEcuacion.Evaluar(var);
                if (res == respuesta)
                {
                        correcto == true;
                        break;
                }
        }
        if(correcto)
        {
                resultadoRetornar = “PC”;
        }
        else
```

```
                {
                        resultadoRetornar = “PI”;
                }
        }
}
else
{
        //Caso que la ecuación no tiene solución
        Arbol arbolAux = ecuacion.SusutituirSubArbol(subEcuacion, “ZZ”);
        double[] resultadosAValidar = arbolAux.ObtenerRresultados();
        if(resultadosAValidar.Count() > 0)
        {
                if(resultadosAValidar.Contains(respuesta))
                {
                        correcto = true;
                        resultadoRetornar = “PC”;
                }
                else
                {
                        resultadoRetornar = “PI”;
                }
        }
        else
        {
                pasoSinSolucion = true;
                int cantIntentos =
                ManejadorAccesoDatos.ObtenerCantidadPasosSinSolucion(idEcuaci
                on, ecuacion.getIdResoluconActual());
                if(cantIntentos + 1 == 3)
                {
                        resultadoRetornar = “SS”;
                }
                else
                {
                        resultadoRetornar = “PI”;
                }
        }
}
        ManejadorAccesoDatos.LoguearResolucion(idEcuacion, subEcuacion, respuesta,
DateTime.Now, ecuacion.getIdResolucion(), subEcuacion.lenght() == 1, pasoSinSolucion);
}

Lista<Resolucion> ObtenerResolucionEcuacion(idEcuacion)
{
        Lista<Resolucion> listaResolucion =
        ManejadorAccesoDatos.ObtenerResolucion(idEcuacion);
```

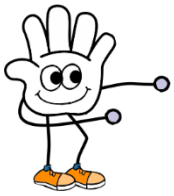
```
        return listaResolucion;
}



String DescargarEcuaciones(int cantidad, IntervaloFechas intervalo)
{
        try
        {
                Servicio serv = new Servicio();
                        Lista<Ecuacion> ecuacionesDescargadas =
                ser.DescargarEcuaciones(cantidad, intervalo);
                foreach(Ecuacion e in ecuacionesDescargadas)
                {
                        ManejadorAccesoDatos.AltaEcuacion(e);
                }
                return “La descarga fue realizada con éxito.”;
        }
        catch(Exception e)
        {
                return “Ocurrió un error descargando las ecuaciones. Inténtelo más tarde.”;
        }
}

String SubirEcuaciones(List<Ecuacion> listaEcuaciones)
{
        try
        {
                Servicio serv = new Servicio();
                Respuesta res = serv.SubirEcuaciones(listaEcuaciones);
                return res.getDescripcion();
        }
        catch(Exception e)
        {
                return “Ocurrió un error de conexión. Inténtelo más tarde.”;
        }
}
```

# Mensajes de interacción con el usuario
# Versión 1.0

En este documento se especifican los textos definidos por el cliente para los mensajes de interacción con el usuario en la resolución de ecuaciones. Además, nos brindaron las imágenes que les gustaría que aparecieran asociada a cada mensaje.

## Mensajes

1. Cuando hay más de una solución o más de una opción en un paso de bifurcación, mostrar el siguiente mensaje:

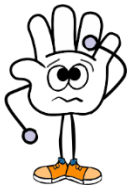


**“Existe otro número que también sirve”**

2. Cuando existe más de una solución y el usuario vuelve a poner el mismo valor, mostrar:

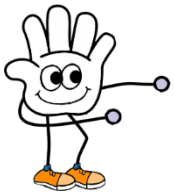
**”Otro número, no el mismo”**

3. Cuando selecciona una expresión inadecuada para resolver pero que contenga la incógnita, mostrar:



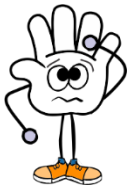
**”¿Mmm…. Estas segur@?”**

4. Cuando selecciona una expresión inadecuada pero que no incluya la incógnita, mostrar:



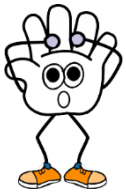
**“Tu selección debe incluir el número desconocido”**

5. Cuando la solución es vacía y el usuario intentó ingresar 3 valores posibles, mostrar el siguiente mensaje:



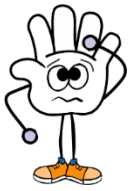
**“¿Mmm….  Existirá algún número?”**¨

Si el usuario no se da cuenta que debe elegir la opción de S={} mostra el siguiente mensaje después de haber probado 2 valores más:



**“¡No hay ningún número real que sirva!”**

6. Si elige la opción de S={} cuando existe solución distinta a la vacía, mostrar:



**”¿Mmmmm…. Estas segur@?”**

# Anexo B

Prototipo Electron

Prototipo SQLite

Interfaz de usuario

Documento de riesgos

Descripción de la arquitectura, modelo de diseño y datos

Especificación de requisitos de software

Estandar de implementación

Plan de SQA

Plan de configuración

Plan de verificación y validación

Alcance del sistema

Modelo de casos de prueba

Modelo de casos de uso

Plan de proyecto

Glosario

Prueba de performance

Acta de reunión de requisitos 1

Acta de reunión de requisitos 2

Acta de reunión de requisitos 3

Acta de reunión de requisitos 4

Acta de reunión de requisitos 5

Acta de reunión de requisitos 6

Acta de reunión de requisitos 7

# Prototipo Electron
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 13/10/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

# ÍNDICE

## Introducción

En este documento se especificará el motivo de la realización del prototipo, cómo se realizó el prototipo y las conclusiones tomadas.

## Motivo

Electron es un framework que permite realizar aplicaciones desktop permitiendo utilizar HTML y CSS para realizar el diseño de interfaz de usuario.
La intención de utilizar esta herramienta es realizar una interfaz atractiva al estudiante como fue solicitado por el cliente. Consideramos que esta opción facilitaba realizar una aplicación más moderna a la que están acostumbrados los usuarios de esta época al contrario de las herramientas de diseño que provee java, ya que a nuestro entender son más anticuadas.

## Prototipo

Al realizar el prototipo no pudimos realizar la instalación de la herramienta debido a las distintas versiones que Electron, Node y otros complementos que eran necesarios en Linux para implementar. Además investigamos en internet la posibilidad de importar una librería desarrollada en Java que contenga la lógica de la aplicación, pero no encontramos esa opción disponible. Por lo tanto, al notar que los lenguajes que teníamos que utilizar para desarrollar nuestra aplicación eran Node o Java Script los cuales ninguno de los integrantes tenía un conocimiento suficiente sobre los lenguajes como para realizar el desarrollo de toda la aplicación. Encontramos información de que Electron tenía la posibilidad de acceder a web services, pero al tener la restricción de que la aplicación funcione sin internet, la posibilidad de realizar un servidor que provea servicios que realicen la lógica quedaba descartada. Tampoco quisimos que la misma máquina del estudiante levantara el servicio porque aumentaría la probabilidad de las fallas al levantar el mismo generando insatisfacción por parte del usuario y de los clientes. El principal uso de la aplicación es realizar una clase con el apoyo en el software a desarrollar, y la posibilidad de que la aplicación no corra en el momento en que se está dictando la clase consideramos que se tiene que minimizar.

Por lo antes mencionado, se tomó la decisión de buscar otra alternativa para realizar el diseño de la interfaz de usuario. Las alternativas que encontramos disponibles para realizar la interfaz de usuario eran utilizar Swing o JavaFX. Para tomar la decisión sobre estas dos opciones, investigamos diseños de aplicaciones disponibles en internet y tomamos la decisión de utilizar JavaFX.
El motivo de esta decisión fue que los diseños en JavaFX tenían un estilo más moderno que los diseños realizados con Swing. Además encontramos que JavaFX tiene la posibilidad de utilizar CSS al cual ya estamos más adaptados para realizar el diseño de los elementos de la interfaz.

# Prototipo SQLite
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 13/10/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

# ÍNDICE

## Introducción

En este documento se especifica el motivo de la realización del prototipo, como se realizó el prototipo y las conclusiones tomadas.

## Descripción

Al inicio del proyecto se había tomado la decisión, en conjunto con los tutores, de utilizar archivos para guardar los datos en forma local en la computadora de los estudiantes, de forma de asegurar que los estudiantes pudieran trabajar sin conexión a internet. Luego analizando los recursos de hardware y con el comentario del cliente que indicaba que la computadora de los estudiantes era lenta debido al uso personal que ellos realizaban, se planteó la preocupación del desempeño que podía tener la aplicación al utilizar distintos archivos y teniendo que realizar operaciones de lectura y parseo de la información que estos contenían. Por este motivo, se tomó la iniciativa de probar SQLite, ya que es una base de datos liviana utilizada usualmente en dispositivos móviles y que es compatible con varias plataformas, incluida java.

Además, notamos que utilizar una base de datos nos simplificaría la implementación de la persistencia y recuperación de los datos guardados en distintos archivos.

Uno de los requisitos del cliente consistía en ver la resolución de una ecuación realizada por un alumno. Nuestro planteo para resolver ese requisito fue mostrarle el archivo guardado que contiene la resolución de todos los intentos que había realizado el estudiante indicando en qué fecha fue realizada la operación. Utilizando una base de datos, nosotros podemos tener un historial de las resoluciones y permitir al usuario una mejor visibilidad de las resoluciones logueadas.

## Prototipo realizado

Para verificar que era posible utilizar SQLite con el lenguaje java, se realizó un desarrollo que consistía en crear una base de datos, insertar datos y luego consultarlas utilizando el jar de SQLite compatible con java.

Como se pudo realizar estas operaciones con facilidad se decidió utilizar SQLite para realizar la persistencia en la computadora del estudiante.

# Interfaz de usuario
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 01/10/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

# ÍNDICE

## Introducción

En este documento se presenta el diseño de los bosquejos realizados para el desarrollo de la interfaz de usuario. Se buscó lograr un diseño amigable e intuitivo para los estudiantes de ciclo básico, pensando en la navegabilidad entre los distintos casos de uso.

## Bosquejos

Los bosquejos fueron diseñados considerando un tamaño de pantalla de 10’ siendo este el tamaño promedio de las laptops Magallanes.

Para aprovechar el espacio decidimos no utilizar un menú general para acceder a los distintos casos de uso y en su lugar le brindamos acceso mediante pequeños botones como se observa a continuación.

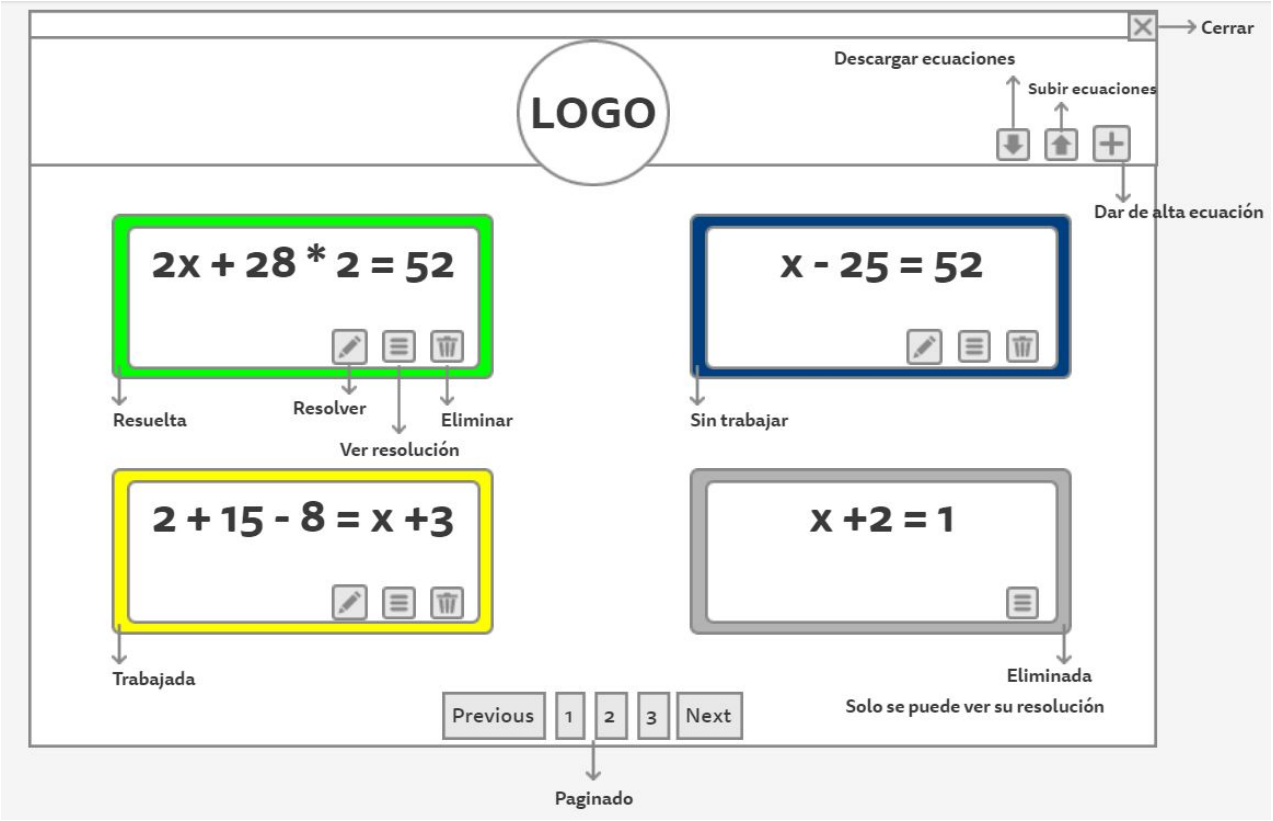
### Pantalla principal

La pantalla principal contará con un cabezal formado por el logo y tendrá en el lateral izquierdo botones para acceder a la descarga, subida y alta de ecuación respectivamente.

En el cuerpo de la pantalla principal se mostrará el listado de ecuaciones donde estarán clasificadas por un color que representa su estado. Se colorean de verde las ecuaciones resueltas, en azul las que no se comenzó a trabajar en su resolución, en amarillo las que se comenzaron a resolver y en gris las ecuaciones que fueron eliminadas localmente pero poseen un historial de resolución para ver.

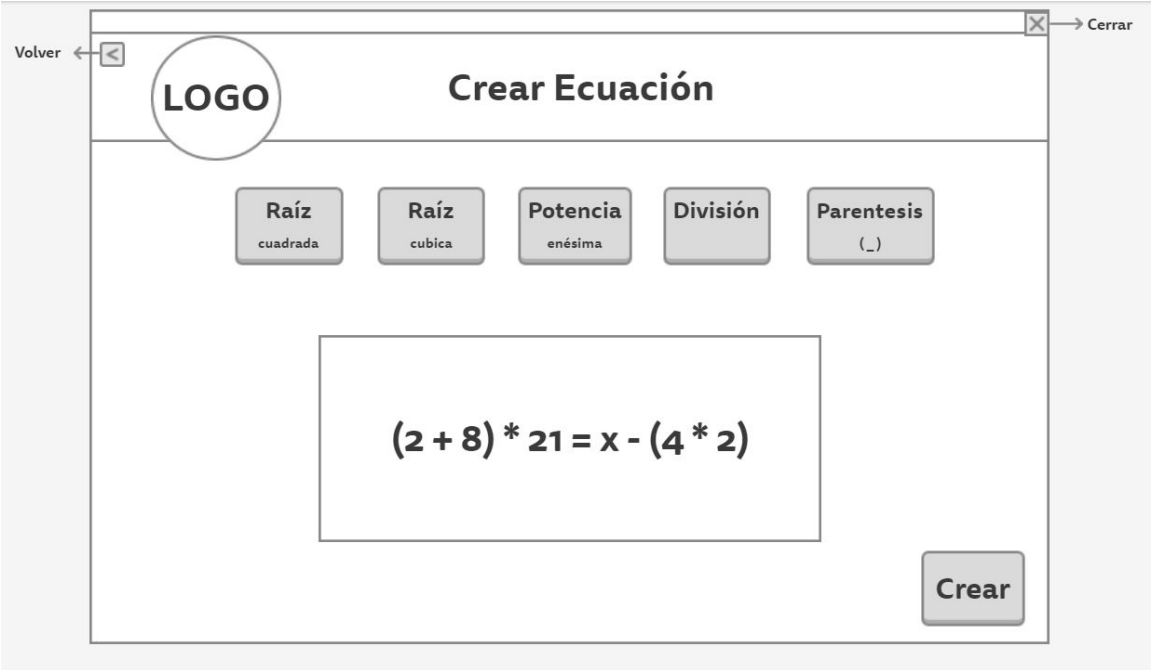
Como se mencionaba anteriormente se brinda acceso a los casos de uso de resolución, ver resolución y eliminación local mediante pequeños botones contenidos en cada ecuación que forma parte del listado.

A continuación se observa el diseño de la pantalla principal.




**Pantalla crear ecuación**

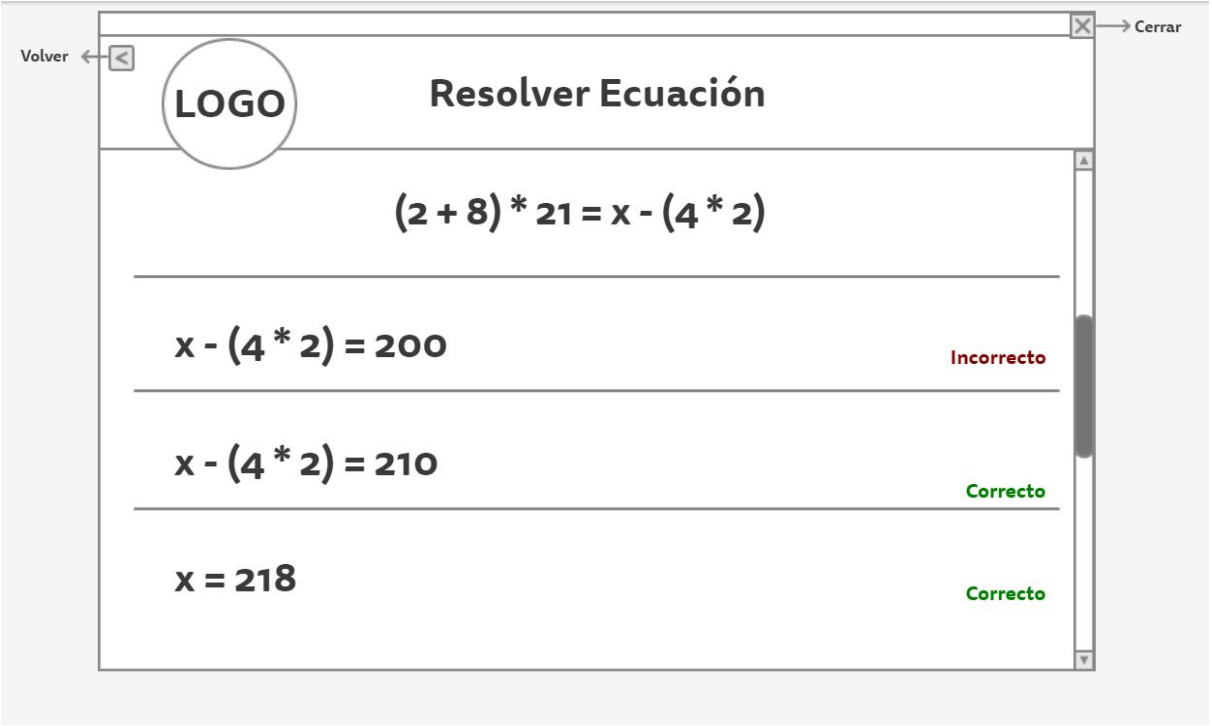
La pantalla para crear ecuaciones permite dar de alta una ecuación localmente mediante una botonera que facilita el uso de los operadores de raíz cuadrada y cubica, potencia división y el uso de paréntesis como se observa en la figura. El resto de la ecuación es tipeada como se definió en el documento de requerimiento.



### Pantalla resolver ecuación

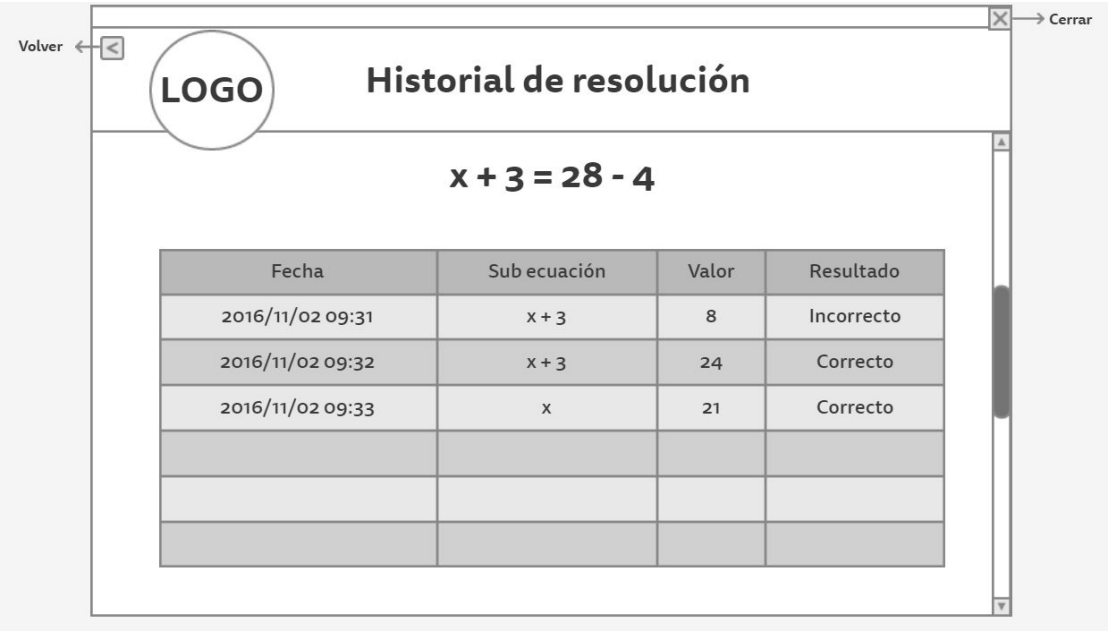
La pantalla de resolución tiene un comportamiento similar a la pantalla del software que actualmente usan los alumnos para que sea más intuitivo su usos dado que están familiarizados con su uso.

Cuando el alumno selecciona una sub ecuación para resolver esta se copia debajo en una nueva línea y se solicita que ingrese el resultado. La aplicación muestra en cada paso si el resultado ingresado es correcto o incorrecto.



### Pantalla historial de resolución de ecuación

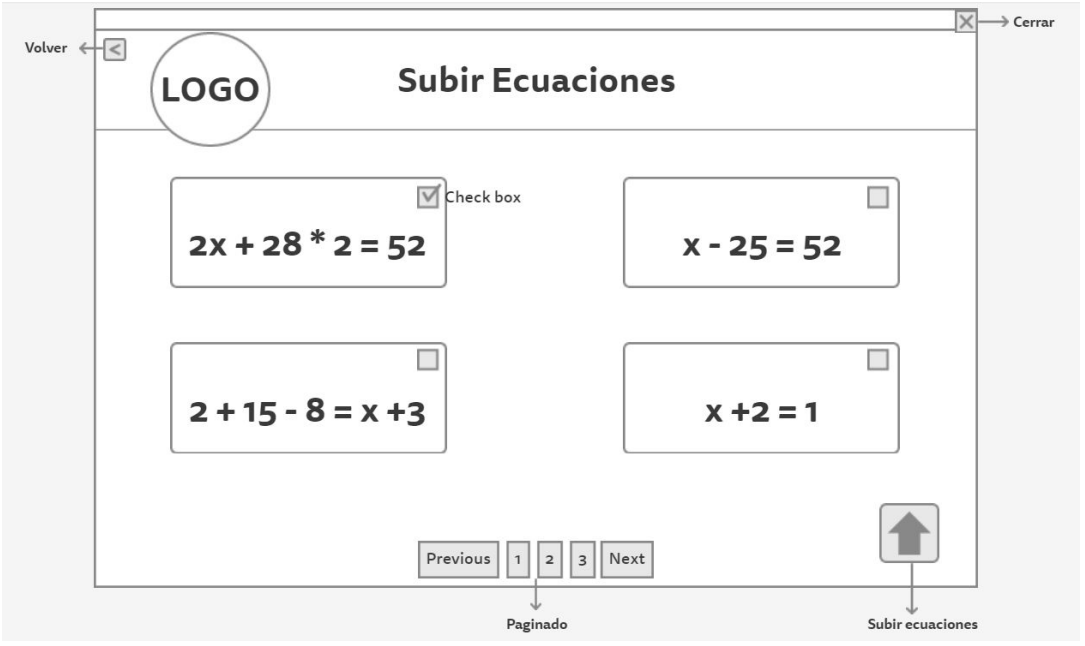
En esta pantalla se mostrará todo el historial guardado localmente asociado a la ecuación solicitada.




### Pantalla subir ecuación

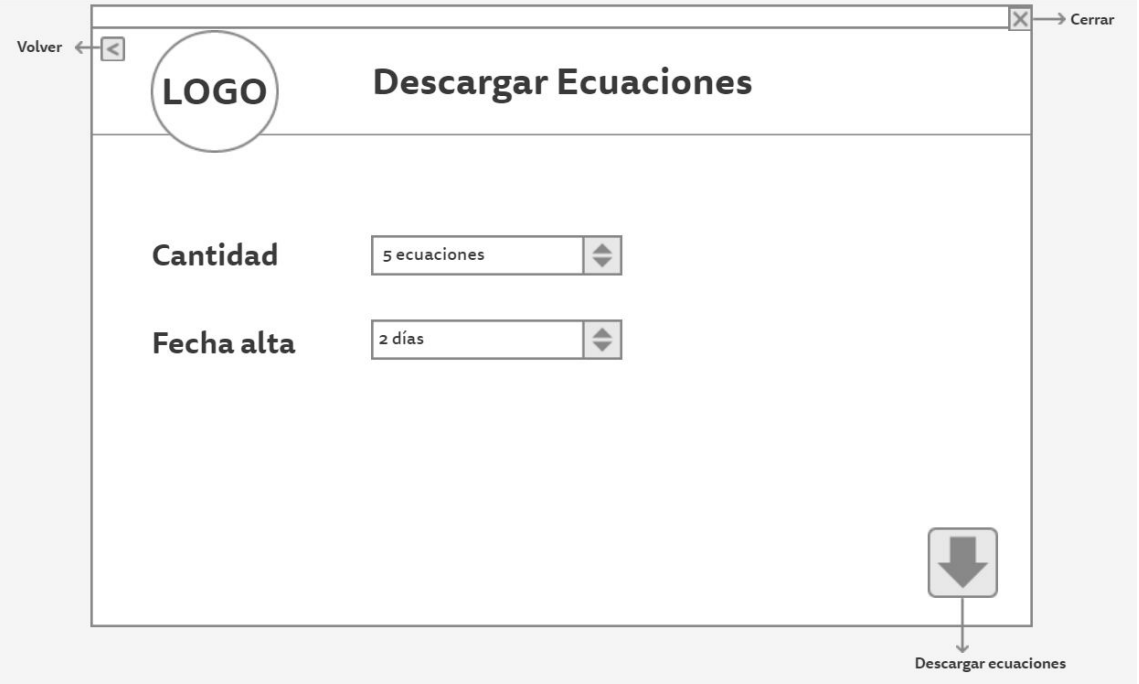
Cuando se accede a esta pantalla, mediante el botón subir ecuación que se encuentra en el cabezal de la pantalla principal, se visualiza la lista de ecuaciones que fueron dadas de alta localmente y todavía no fueron subidas al repositorio central.

Permitiendo al usuario seleccionar las ecuaciones que desea subir mediante checkbox.



### Pantalla descargar ecuación

A esta pantalla se accede mediante el botón descargar ecuación que se ubica en el cabezal de la pantalla principal y le permite al usuario seleccionar las opciones para descargar ecuaciones del repositorio central.



# Documento de Riesgos
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 16/09/2016 | 1.0 | Inicial | Martín Poli, Camila Rojí |
| | | | |
| | | | |
| | | | |

# ÍNDICE

## Lista de Riesgos identificados

Se enumeran y describen los Riesgos del proyecto, en general en forma descendente de acuerdo a su gravedad, así como la mitigación, monitoreo, y contingencia de los mismos.

### Riesgo de enfermedad o ausencia de algún integrante del equipo

#### Gravedad

Medio

#### Descripción

La ausencia de uno de los integrantes del equipo por enfermedad o estudio para otras materias en un período de tiempo no muy prolongado.

#### Probabilidad de ocurrencia

Alta

#### Impacto

El impacto de este riesgo es atrasos en la planificación o sobrecarga de tareas en tiempos acotados.

### Riesgo de alta curva de aprendizaje

#### Gravedad

Alta

#### Descripción

El riesgo consiste en tener dificultad para entender y aprender la tecnología.

#### Probabilidad de ocurrencia

Media

#### Impacto

El impacto de este riesgo es atrasos en la planificación, sobrecarga de tareas en tiempos acotados y desmotivación de los integrantes del equipo.

### Riesgo de hardware de las máquinas de los estudiantes

#### Gravedad

Alta

#### Descripción

El riesgo consiste en que el producto realizado no corra en las laptops Magallanes de los estudiantes correctamente.

#### Probabilidad de ocurrencia

Baja

### Impacto

El impacto de este riesgo es que el producto no pueda se utilizado y quede desconforme el cliente.

## Riego de interfaz gráfica

### Gravedad

Media

### Descripción

Este riesgo consiste en la no adaptación al uso de la aplicación debido a una interfaz poco intuitiva y al no agrados de los usuarios con respecto a las interfaces gráficas de la aplicación.

### Probabilidad de ocurrencia

Media

### Impacto

El impacto de este riesgo los usuarios no quieran utilizar la aplicación por su dificultad para utilizarla.

## Riesgo de mala planificación y malas mediciones

### Gravedad

Alta

### Descripción

Este riesgo consiste en la falla al tomar las mediciones para estimar el producto y que derive en una mala planificación llevando a un alcance difícil de alcanzar en el plazo determinado.

### Probabilidad de ocurrencia

Media

### Impacto

El impacto que tiene este riesgo es tener que solicitar prórroga para extender el tiempo de realización del proyecto.

## Riesgo de tener un análisis erróneo del sistema

### Gravedad

Alta

### Descripción

El riesgo consiste en la mala interpretación de las necesidades del cliente y realizar un producto inadecuado para el mismo.

### Probabilidad de ocurrencia

Baja

### Impacto

El impacto de este riego es generar disconformidad en el cliente o el re trabajo por parte de los integrantes del equipo y tener atraso en la planificación del proyecto.

### Riesgos tecnológicos

#### Gravedad

Alta

#### Descripción

Este riesgo consiste en el mal funcionamiento de las tecnologías elegidas para el desarrollo del producto y dificultad de utilización.

#### Probabilidad de ocurrencia

Media

#### Impacto

El impacto de este riesgo es el atraso en la planificación del proyecto y tener que realizar un cambio de tecnología una vez comenzada la etapa de desarrollo.

## Estrategia de Mitigación

Se describen las decisiones tomadas y las acciones previstas por el equipo para reducir la probabilidad de ocurrencia de cada Riesgo en el proyecto.

### Riesgo de enfermedad o ausencia de algún integrante del equipo

Para mitigar este riesgo se tendrá en cuenta en las planificaciones de las iteraciones los meses en que hay más probabilidad de tener que rendir parciales o exámenes de forma de no sobrecargar esos meses con trabajo difícil de realizar.

### Riesgo de alta curva de aprendizaje

Para mitigar este riesgo se realizarán prototipos previo a la fase de desarrollo para ir conociendo y aprendiendo la tecnología con la que se va a trabajar.

### Riesgo de hardware de las máquinas de los estudiantes

Para mitigar este riesgo se probará en alguna laptop Magallanes disponible para verificar el correcto funcionamiento de lo implementado hasta el momento.

### Riego de interfaz gráfica

Para mitigar este riesgo se realizará un diseño de la interfaz gráfica y se le mostrarán bosquejos al cliente para confirmar que sea de su agrado. Además se respetará la forma de utilización de la aplicación actual para que les sea más intuitiva de utilizar.

### Riesgo de mala planificación y malas mediciones

Antes de realizar las mediciones se analizarán en profundidad la forma en la que se implementará la resolución de las ecuaciones de forma de saber con certeza la dificultad de la tarea. Además se agregarán colchones de tiempo en la planificación considerando las dificultades en la realización de los requisitos.

### Riesgo de tener un análisis erróneo del sistema

Al utilizar un modelo de desarrollo iterativo incremental vamos a ir obteniendo el feedback del cliente en etapas tempranas de forma de detectar requisitos mal interpretados.

### Riesgos tecnológicos

Para mitigar este riesgo se realizarán prototipos de forma de verificar que la tecnología es adecuada para el desarrollo del proyecto.

## Monitoreo

Los riesgos mencionados anteriormente no son posibles de identificar anteriormente, varios de los riesgos los detectaremos en la mitigación de los mismos. Por ejemplo, realizando los prototipos para mitigar la curva de aprendizaje y el riesgo tecnológico. Así como las reuniones periódicas con los clientes para corroborar que el análisis sea correcto así como que la interfaz gráfica es de su agrado.

## Plan de Contingencia

Se describen las acciones que tomará el equipo en el caso que alguno de los Riesgos identificados se presente, como por ejemplo dar una solución alternativa, establecer reducción en la funcionalidad del Sistema, etc.

### Riesgo de enfermedad o ausencia de algún integrante del equipo

En el caso de que un integrante se enferme sorpresivamente, el otro tomará las tareas del compañero que éste ya tenía comenzadas de forma de respetar la planificación y minimizar el retraso del proyecto.

### Riesgo de alta curva de aprendizaje

En el caso de que ocurra este riesgo, si el integrante lo cree necesario, buscará la forma de dedicarle más horas diarias de las que tenía planificadas.

### Riesgo de hardware de las máquinas de los estudiantes

Se solicitará ayuda a los tutores para pedir prórroga y alargar el tiempo para solucionar el problema.

### Riego de interfaz gráfica

Capacitar más detalladamente a los usuarios para que entiendan puedan utilizar la aplicación de forma más fluída.

### Riesgo de mala planificación y malas mediciones

Se solicitará ayuda a los tutores para pedir prórroga y alargar el tiempo para finalizar las tareas.

### Riesgo de tener un análisis erróneo del sistema

En caso de que ocurra este riesgo, los integrantes del equipo buscarán la forma de dedicarle más horas diarias de las que tenían planificadas de forma de realizar el requisito correcto. Si el cambio solicitado implica mucho desarrollo se solicitará ayuda a los tutores para obtener más tiempo en la realización del proyecto.

### Riesgos tecnológicos

Se solicitará ayuda a los tutores para solicitar prórroga y obtener más tiempo para realizar el desarrollo del proyecto.

# Descripción de la arquitectura, modelo de diseño y datos.

# Versión 2.2

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 05/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| 16/09/2016 | 1.1 | Reestructuración del documento | Camila Rojí - Martín Poli |
| 13/10/2016 | 2.0 | Especificación de la arquitectura | Camila Rojí - Martín Poli |
| 17/10/2016 | 2.1 | Se agregan la descomposición en subsistemas, el modelo de diseño y el modelo de Datos | Camila Rojí - Martín Poli |
| 23/10/2016 | 2.2 | Se agregan correcciones y mejoras sugeridas por los tutores | Camila Rojí - Martín Poli |

# ÍNDICE

# Introducción

En este documento se presenta la arquitectura de la aplicación, mostrando la visión física y lógica de la misma. También se detallaran los casos de uso que son relevantes a la arquitectura.

### Propósito

Este documento proporciona una apreciación global y comprensible de la arquitectura del sistema usando diferentes puntos de vista para mostrar distintos aspectos del mismo. Intenta capturar y llegar a las decisiones de arquitectura críticas que han sido hechas en el sistema.

# Vista del Modelo de Casos de Uso

## Descripción de Casos de Uso relevantes a la Arquitectura

En esta sección se detallan los Casos de Uso y actores más significativos para definir la Arquitectura del sistema.

Estos casos de uso son:

- los que ayudan a mitigar los riesgos más serios
- los más importantes para los usuarios y clientes
- los que ayudan a cubrir toda la funcionalidad importante.

### Alta de ecuación

El sistema debe permitir dar de alta ecuaciones de hasta tercer grado (lineales, cuadráticas y cúbicas) racionales y con radicales o combinaciones de estas, con una única variable, pudiendo estar en cualquier lado de la igualdad. Las cuales se construirán mediante una botonera que cuenta con las operaciones raiz, potencia, cociente y paréntesis, y se tipearán los operadores básicos (+, -, *), números, el símbolo de igualdad y la variable.

La ecuación se mantendrá guardada localmente y opcionalmente se podrá subir al repositorio central.

### Subir ecuación a repositorio

El sistema debe permitir subir las ecuaciones ingresadas por los usuarios al repositorio central cuando este lo solicite. Las ecuaciones se subirán en el caso que se cuente con conexión a Internet sino la hay se deberá mostrar un mensaje que indique que lo intente mas tarde.

Las ecuaciones que se van a subir son las que fueron creadas localmente y no fueron subidas anteriormente.

### Descargar ecuación de repositorio

El sistema debe permitir al usuario descargar del repositorio central ecuaciones aleatorias. El usuario deberá indicar la cantidad deseada desde un combo con opciones predeterminadas.

Las ecuaciones descargadas serán almacenadas localmente para su posterior resolución.

### Resolución de ecuación

El sistema debe permitir al usuario ir resolviendo en etapas la ecuación mediante métodos aritméticos y culminar la resolución cuando el estudiante encontró todas las soluciones posibles. En el caso en que la ecuación no tenga solución, luego de una determinada cantidad de intentos se deberá mostrar un mensaje indicando que la ecuación no tiene solución.

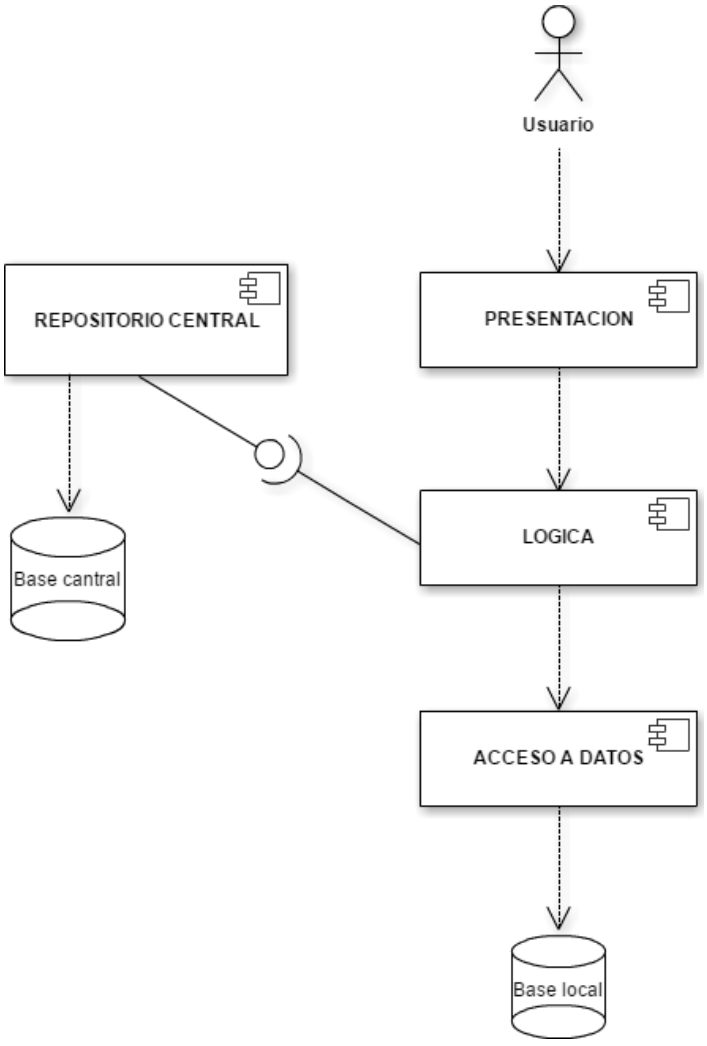
En cada paso el sistema deberá ir guardando la solución del estudiante en un archivo local, lo que le permitirá retomar la resolución de una ecuación comenzada anteriormente.

La ecuación a resolver podrá ser toma de las ecuaciones ingresada o descargada previamente. En el caso que la ecuación se haya comenzado a resolver anteriormente el usuario puede volver a comenzar la resolución o continuar con la misma.

## Vista del Modelo de Diseño

### Descomposición en Subsistemas

En esta sección se presenta el diagrama de descomposición en subsistemas. Se utiliza una arquitectura en capas en la que cada una consume los servicios de la capa inferior. El repositorio central brinda servicios que serán consumidos desde la capa lógica.



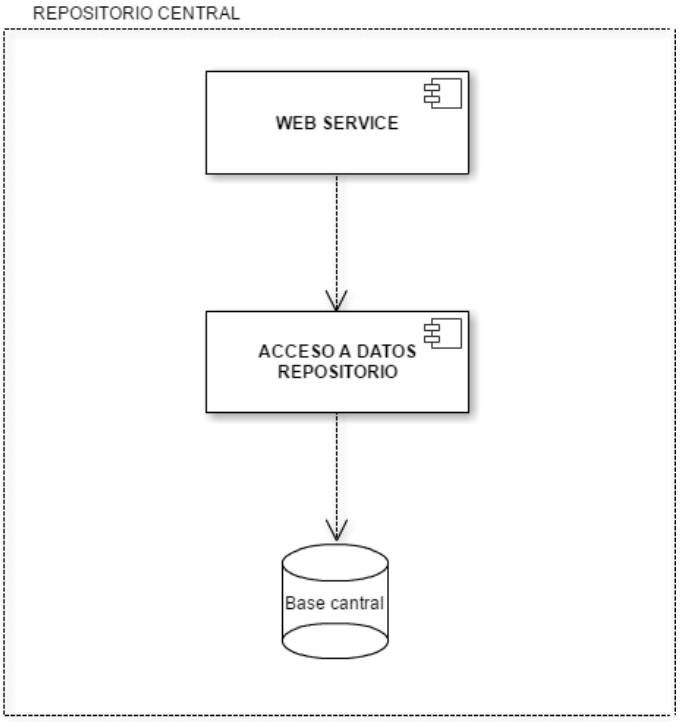

La capa Presentación es la encargada implementar la interfaz gráfica del sistema que será consumida por el usuario e interactuar con la capa lógica.

La capa lógica se encarga de implementar las reglas de negocio necesarias para brindar solución a la capa de presentación, interactuar con la capa de acceso a datos local y/o acceder a los servicios brindados por el repositorio central. Dentro de la capa lógica tendremos un controlador de ecuaciones encargado de todo el manejo de las operaciones correspondientes a la lógica de las ecuaciones.

Dado que este proyecto es la base para un futuro proyecto de grado, en dicha capa, se podrá tener un controlador de usuarios, por ejemplo, para toda la manejo administrativo de los docentes.

La capa de accesos a datos, de la base local, es la encargada de brindar todas las operaciones necesarias para el acceso y modificación de los datos que se encuentran en la base de datos local a la máquina del estudiante.

El componente Repositorio Central es un web service que provee los servicios para subir ecuaciones al repositorio y la descarga de las mismas. En la siguiente figura se detallan sus componentes.

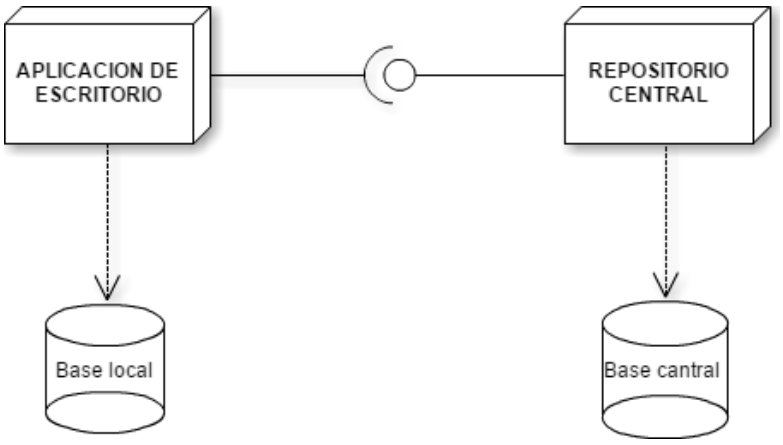



La capa de acceso a datos, del repositorio central, se encarga de obtener y modificar datos que se encuentran en la base de datos del repositorio.

# Vista del Modelo de Distribución

### Diagrama de Distribución

El modelo de distribución describe la distribución física del sistema en términos de cómo se reparten las funcionalidades entre los equipos que conforman la aplicación.




### Aplicación de escritorio

En esta aplicación se debe permitir crear, eliminar y resolver ecuaciones. Además de loguear el trabajo realizado por el estudiante en una base de datos local, debe permitir subir y descargar las ecuaciones del repositorio central.

### Base de datos local

Contiene las ecuaciones ingresadas por el estudiante o que fueron descargadas del repositorio central.

### Repositorio central

El repositorio central es un servidor que provee servicios para subir y descargar ecuaciones a la base de datos central.

### Base de datos central

Contiene las ecuaciones ingresadas por los estudiantes y que fueron subidas con los servicios provistos por el servidor.

### Conexión entre aplicación de escritorio y repositorio central

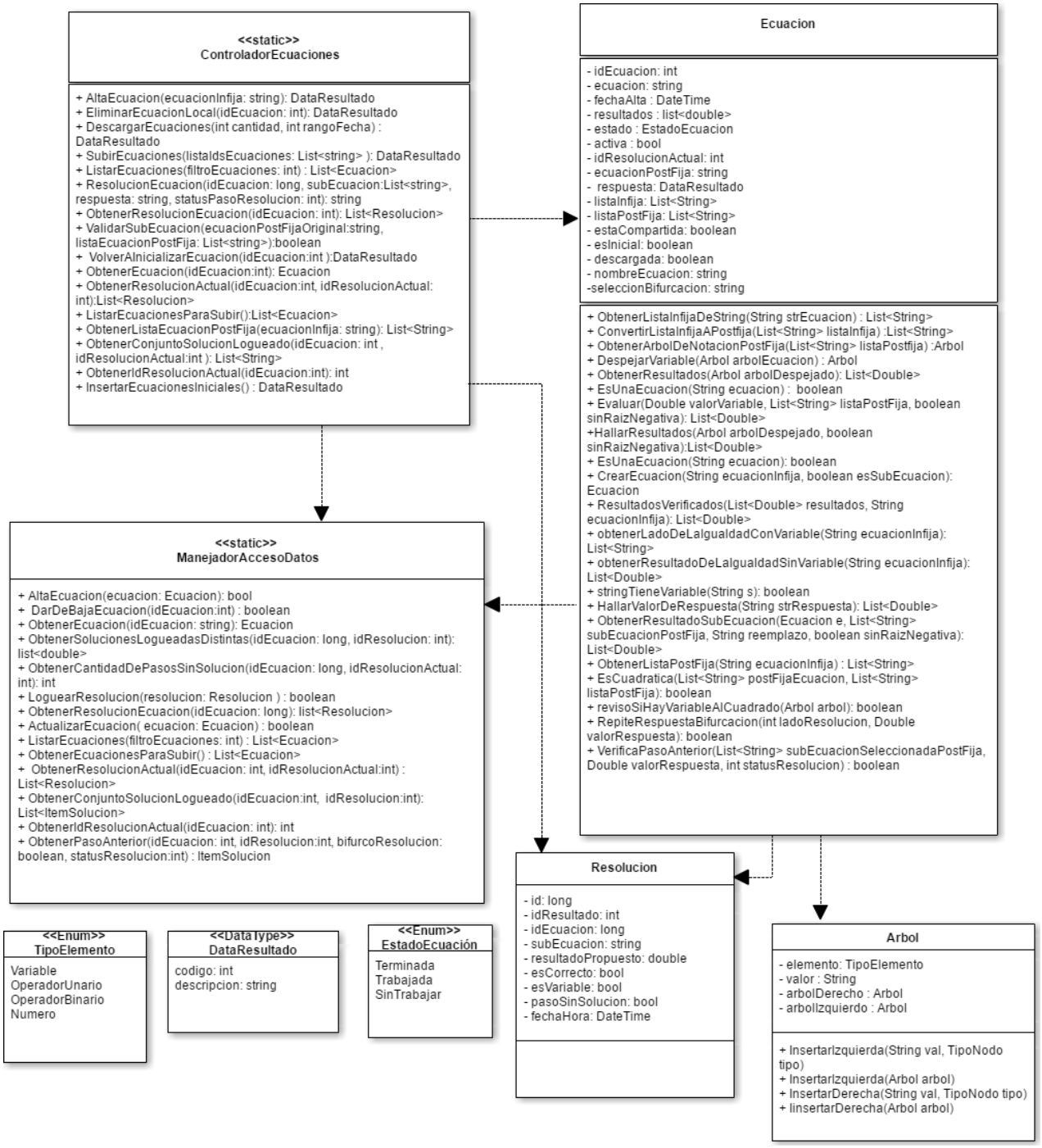
Como mencionamos anteriormente el repositorio central estará implementado mediante un web service que brinda los servicios necesarios para la interacción entre la aplicación de escritorio y los datos que se alojan en la base central.

## Diagrama de Clases

En esta sección se especifica el diagrama de clases correspondiente al subsistema que representa la aplicación de escritorio analizado anteriormente que permite al estudiante dar de alta, baja, resolver ecuaciones, etc.

El controlador de ecuaciones será la puerta de entrada a la lógica desde la capa gráfica, mientras que el manejador de acceso a datos es la puerta de entrada al manejo de datos local.

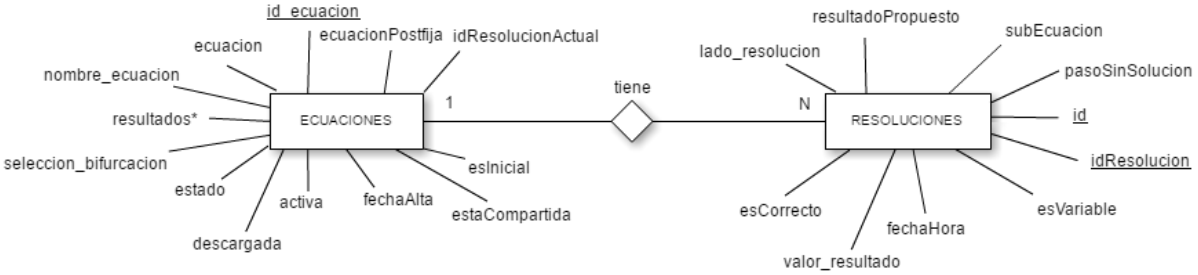
Se tendrá una clase principal Ecuación que permite todo el manejo relacionado a la ecuación, a su vez, la ecuación estará representada por un Árbol que nos facilitar el manejo y resolución de la ecuación.


<<static>>
ControladorEcuaciones
+ AltaEcuacion(ecuacionInfija: string): DataResultado
+ EliminarEcuacionLocal(idEcuacion: int): DataResultado
+ DescargarEcuaciones(int cantidad, int rangoFecha) : DataResultado
+ SubirEcuaciones(listaIdsEcuaciones: List<string> ): DataResultado
+ ListarEcuaciones(filtroEcuaciones: int) : List<Ecuacion>
+ ResolucionEcuacion(idEcuacion: long, subEcuacion:List<string>, respuesta: string, statusPasoResolucion: int): string
+ ObtenerResolucionEcuacion(idEcuacion: int): List<Resolucion>
+ ValidarSubEcuacion(ecuacionPostFijaOriginal:string, listaEcuacionPostFija: List<string>):boolean
+ VolverAInicializarEcuacion(idEcuacion:int ):DataResultado
+ ObtenerEcuacion(idEcuacion:int): Ecuacion
+ ObtenerResolucionActual(idEcuacion:int, idResolucionActual: int):List<Resolucion>
+ ListarEcuacionesParaSubir():List<Ecuacion>
+ ObtenerListaEcuacionPostFija(ecuacionInfija: string): List<String>
+ ObtenerConjuntoSolucionLogueado(idEcuacion: int , idResolucionActual:int ): List<String>
+ ObtenerIdResolucionActual(idEcuacion:int): int
+ InsertarEcuacionesIniciales() : DataResultado
Ecuacion
- idEcuacion: int
- ecuacion: string
- fechaAlta : DateTime
- resultados : list<double>
- estado : EstadoEcuacion
- activa : bool
- idResolucionActual: int
- ecuacionPostFija: string
- respuesta: DataResultado
- listaInfija: List<String>
- listaPostFija: List<String>
- estaCompartida: boolean
- esInicial: boolean
- descargada: boolean
- nombreEcuacion: string
-seleccionBifurcacion: string
+ ObtenerListaInfijaDeString(String strEcuacion) : List<String>
+ ConvertirListaInfijaAPostfija(List<String> listaInfija) :List<String>
+ ObtenerArbolDeNotacionPostFija(List<String> listaPostfija) :Arbol
+ DespejarVariable(Arbol arbolEcuacion) : Arbol
+ ObtenerResultados(Arbol arbolDespejado): List<Double>
+ EsUnaEcuacion(String ecuacion) : boolean
+ Evaluar(Double valorVariable, List<String> listaPostFija, boolean sinRaizNegativa): List<Double>
+HallarResultados(Arbol arbolDespejado, boolean sinRaizNegativa):List<Double>
+ EsUnaEcuacion(String ecuacion): boolean
+ CrearEcuacion(String ecuacionInfija, boolean esSubEcuacion): Ecuacion
+ ResultadosVerificados(List<Double> resultados, String ecuacionInfija): List<Double>
+ obtenerLadoDeLalgualdadConVariable(String ecuacionInfija): List<String>
+ obtenerResultadoDeLalgualdadSinVariable(String ecuacionInfija): List<Double>
+ stringTieneVariable(String s): boolean
+ HallarValorDeRespuesta(String strRespuesta): List<Double>
+ ObtenerResultadoSubEcuacion(Ecuacion e, List<String> subEcuacionPostFija, String reemplazo, boolean sinRaizNegativa): List<Double>
+ ObtenerListaPostFija(String ecuacionInfija) : List<String>
+ EsCuadratica(List<String> postFijaEcuacion, List<String> listaPostFija): boolean
+ revisoSiHayVariableAlCuadrado(Arbol arbol): boolean
+ RepiteRespuestaBifurcacion(int ladoResolucion, Double valorRespuesta): boolean
+ VerificaPasoAnterior(List<String> subEcuacionSeleccionadaPostFija, Double valorRespuesta, int statusResolucion) : boolean
<<static>>
ManejadorAccesoDatos
+ AltaEcuacion(ecuacion: Ecuacion): bool
+ DarDeBajaEcuacion(idEcuacion:int) : boolean
+ ObtenerEcuacion(idEcuacion: string): Ecuacion
+ ObtenerSolucionesLogueadasDistintas(idEcuacion: long, idResolucion: int): list<double>
+ ObtenerCantidadDePasosSinSolucion(idEcuacion: long, idResolucionActual: int): int
+ LoguearResolucion(resolucion: Resolucion ) : boolean
+ ObtenerResolucionEcuacion(idEcuacion: long): list<Resolucion>
+ ActualizarEcuacion( ecuacion: Ecuacion) : boolean
+ ListarEcuaciones(filtroEcuaciones: int) : List<Ecuacion>
+ ObtenerEcuacionesParaSubir() : List<Ecuacion>
+ ObtenerResolucionActual(idEcuacion: int, idResolucionActual:int) : List<Resolucion>
+ ObtenerConjuntoSolucionLogueado(idEcuacion:int, idResolucion:int): List<ItemSolucion>
+ ObtenerIdResolucionActual(idEcuacion: int): int
+ ObtenerPasoAnterior(idEcuacion: int, idResolucion:int, bifurcoResolucion: boolean, statusResolucion:int) : ItemSolucion
Resolucion
- id: long
- idResultado: int
- idEcuacion: long
- subEcuacion: string
- resultadoPropuesto: double
- esCorrecto: bool
- esVariable: bool
- pasoSinSolucion: bool
- fechaHora: DateTime
Arbol
- elemento: TipoElemento
- valor : String
- arbolDerecho : Arbol
- arbolIzquierdo : Arbol
+ InsertarIzquierda(String val, TipoNodo tipo)
+ InsertarIzquierda(Arbol arbol)
+ InsertarDerecha(String val, TipoNodo tipo)
+ IinsertarDerecha(Arbol arbol)
<<Enum>>
TipoElemento
Variable
OperadorUnario
OperadorBinario
Numero
<<DataType>>
DataResultado
codigo: int
descripcion: string
<<Enum>>
EstadoEcuación
Terminada
Trabajada
SinTrabajar

## Modelo de Datos

En esta sección se presenta el modelo conceptual de las bases de datos local y la del Repositorio Central, así como sus respectivos diseños lógicos.

### MER - Local




### Diseño Lógico - Base de datos Local

Las ecuaciones serán almacenadas en la base en una tabla de nombre ECUACIONES donde contendrá las siguientes columnas:

- id: identificador numérico dentro de la base local
- ecuación: string con la notación infija de la ecuación
- estado: indica en qué estado está la ecuación (sin trabajar, trabajada, etc.)
- activa: indica si la ecuación no fue borrada lógicamente de la base
- fechaAlta: indica la fecha en que fue dada de alta localmente o la fecha de alta que se encuentra almacenada en el repositorio central para el caso de las ecuaciones descargadas
- xmlArbol: es la representación del árbol que forma la ecuación en formato xml
- idResolucionActual: indica cual es el identificador de resolución actual, este valor se incrementa con cada comienzo nuevo de resolución solicitado por el estudiante.

Las resultados de la resolución de la ecuación serán representados en una tabla de nombre RESULTADOS. Donde los resultados de una ecuación se obtendrá por el id de ecuación. A su vez, la tabla contrendar un id que lo identifica y una columna valorResultado que contiene el resultado.

Por último, la base de datos local tendrá una tabla RESOLUCIONES que almacena el historial de resoluciones realizado por el estudiante. Esta tabla contendrá las siguientes columnas:

- id: identificador interna de la resolución
- idEcuacion: identificador de la ecuación
- idResolucion: identificador del intento de resolución actual
- subEcuacion: sección de la ecuación que fue resuelta por el estudiante en un determinado paso
- resultadoPropuesto: resultado dado por el estudiante para evaluar la subEcuacion
- fechaHora: fecha y hora en que fue realizado dicho paso
- esCorrecto: indica si el resultado propuesto por el alumno para la sub ecuación es válido.
- pasoSinSolucion: indica si en el paso actual se detectó que la ecuación no tiene solución
- esVariable: indica si la sub ecuación seleccionada es una variable

En resumen el diseño lógico de la base de datos local es el siguiente:

ECUACIONES(id, ecuacion, estado, activa, fechaAlta, xmlArbo, idResolucionActuall)

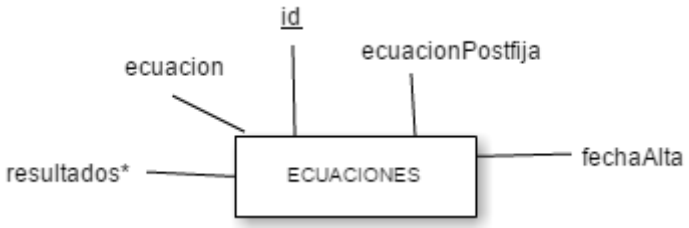
RESULTADOS(id, idEcuacion, valorResultado)

RESOLUCIONES(id, idResolucion, idEcuacion, subEcuacion, resultadoPropuesto, fechaHora, esCorrecto)

$\Pi$ idEcuacion (RESULTADOS) $\subseteq$ $\Pi$ id (ECUACIONES)

$\Pi$ idEcuacion (RESOLUCIONES) $\subseteq$ $\Pi$ id (ECUACIONES)

### MER - Repositorio Central




### Diseño Lógico - Repositorio Central

Al igual que el repositorio local la ecuaciones y los resultados serán almacenados en las tabla de nombre ECUACIONES y RESULTADOS.

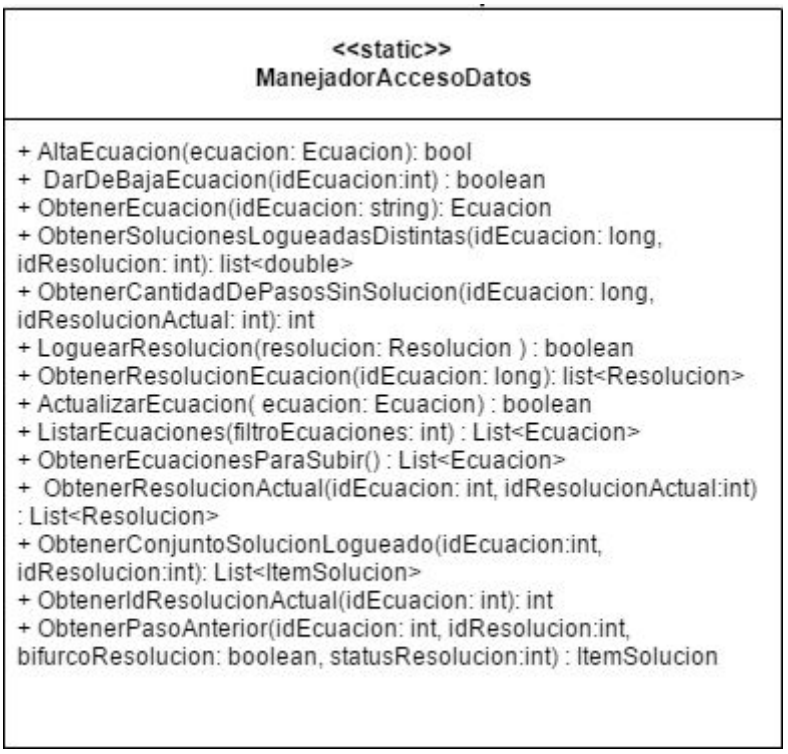
ECUACIONES(id, ecuacion, fechaAlta, xmlArbol)

RESULTADOS(idResultado, idEcuacion, valorResultado)

$\Pi$ idEcuacion (RESULTADOS) $\subseteq$ $\Pi$ id (ECUACIONES)

**Diagrama de clases ampliado**


+ ObtenerListaInfijaDeString(String strEcuacion) : List<String>
+ ConvertirListaInfijaAPostfija(List<String> listaInfija) :List<String>
+ ObtenerArbolDeNotacionPostFija(List<String> listaPostfija) :Arbol
+ DespejarVariable(Arbol arbolEcuacion) : Arbol
+ ObtenerResultados(Arbol arbolDespejado): List<Double>
+ EsUnaEcuacion(String ecuacion) : boolean
+ Evaluar(Double valorVariable, List<String> listaPostFija, boolean sinRaizNegativa): List<Double>
+HallarResultados(Arbol arbolDespejado, boolean sinRaizNegativa):List<Double>
+ EsUnaEcuacion(String ecuacion): boolean
+ CrearEcuacion(String ecuacionInfija, boolean esSubEcuacion): Ecuacion
+ ResultadosVerificados(List<Double> resultados, String ecuacionInfija): List<Double>
+ obtenerLadoDeLalgualdadConVariable(String ecuacionInfija): List<String>
+ obtenerResultadoDeLalgualdadSinVariable(String ecuacionInfija): List<Double>
+ stringTieneVariable(String s): boolean
+ HallarValorDeRespuesta(String strRespuesta): List<Double>
+ ObtenerResultadoSubEcuacion(Ecuacion e, List<String> subEcuacionPostFija, String reemplazo, boolean sinRaizNegativa): List<Double>
+ ObtenerListaPostFija(String ecuacionInfija) : List<String>
+ EsCuadratica(List<String> postFijaEcuacion, List<String> listaPostFija): boolean
+ revisoSiHayVariableAlCuadrado(Arbol arbol): boolean
+ RepiteRespuestaBifurcacion(int ladoResolucion, Double valorRespuesta): boolean
+ VerificaPasoAnterior(List<String> subEcuacionSeleccionadaPostFija, Double valorRespuesta, int statusResolucion) : boolean


<<static>>
ManejadorAccesoDatos
+ AltaEcuacion(ecuacion: Ecuacion): bool
+ DarDeBajaEcuacion(idEcuacion:int) : boolean
+ ObtenerEcuacion(idEcuacion: string): Ecuacion
+ ObtenerSolucionesLogueadasDistintas(idEcuacion: long, idResolucion: int): list<double>
+ ObtenerCantidadDePasosSinSolucion(idEcuacion: long, idResolucionActual: int): int
+ LoguearResolucion(resolucion: Resolucion ) : boolean
+ ObtenerResolucionEcuacion(idEcuacion: long): list<Resolucion>
+ ActualizarEcuacion( ecuacion: Ecuacion) : boolean
+ ListarEcuaciones(filtroEcuaciones: int) : List<Ecuacion>
+ ObtenerEcuacionesParaSubir() : List<Ecuacion>
+ ObtenerResolucionActual(idEcuacion: int, idResolucionActual:int) : List<Resolucion>
+ ObtenerConjuntoSolucionLogueado(idEcuacion:int, idResolucion:int): List<ItemSolucion>
+ ObtenerIdResolucionActual(idEcuacion: int): int
+ ObtenerPasoAnterior(idEcuacion: int, idResolucion:int, bifurcoResolucion: boolean, statusResolucion:int) : ItemSolucion

# Especificación de Requisitos de Software
# Versión 1.3

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 29/08/2016 | 1.0 | Inicial | Camila Rojì, Martín Poli |
| 05/09/2016 | 1.1 | Se agregan requisitos y se amplía la descripción de los mismos. | Camila Rojì, Martín Poli |
| 16/09/2016 | 1.2 | Corrección de redacción | Camila Rojì, Martín Poli |
| 23/09/2016 | 1.3 | Actualización de cambios de requisitos | Camila Rojì, Martín Poli |

# ÍNDICE

# Introducción

En este documento se busca especificar los requisitos solicitados por el cliente en la reunión inicial del proyecto.

### Propósito

Se busca definir los requisitos del sistema a desarrollar con el objetivo de colaborar en etapas posteriores, así como ayudar a obtener un mejor diseño y construcción del sistema a desarrollar.

También se busca poder validar estos requisitos en futuras reuniones con el cliente.

### Alcance

Se desea desarrollar un software que apoye el desarrollo de las etapas iniciales de exploración basada en las competencias aritméticas de los alumnos y orientado a la consolidación de la noción de ecuación que contribuya al desarrollo de la fluidez procedimental a partir de la comprensión conceptual.

Se propone la construcción de un software que dada una ecuación, permite al usuario resolverla mediante métodos de resolución aritméticos.

### Referencias

Propuesta para proyecto de investigación - Sylvia da Rosa, Federico Gómez - 2016

# Descripción general

Se desea contar con un software para trabajar sobre resolución de ecuaciones que permita, tener una base de datos con ecuaciones disponibles y que se pueda ingresar ecuaciones al sistema. La aplicación se debe poder utilizar con o sin conexión a Internet y permitir guardar el trabajo realizado por el alumno, de modo que el docente pueda acceder a ello.

La aplicación debe proveer una interfaz atractiva y debe ser siempre el usuario quien proponga la solución en cada caso.

# Interfaces de usuario

La interfaz de usuario va a estar enfocada en las laptop Ceibal Magallanes y Positivo que son utilizadas por los estudiantes.

Se realizarán prototipos para validar la creación y resolución de las ecuaciones.

# Interfaces con hardware

La aplicación deberá correr en Linux Ubuntu versión 12.04 que es el que contiene todas las versiones de laptops Magallanes.

# Interfaces con software

Se utilizará Postgres SQL para el manejo de la base de datos en el servidor central.

El servidor central debe contar con sistema operativo Linux y debe ser accesible mediante web services.

## Interfaces de comunicación

Se tendrá en cuenta que el funcionamiento de la aplicación está pensada para ser utilizada sin conexión a Internet. Y el usuario podrá subir las ecuaciones creadas cuando dispongo de conexión a Internet.

## Restricciones de memoria

La mayoría de las Magallanes cuentan con una memoria de 1 GB, por ese motivo contamos con una restricción de memoria de ese tamaño. En el anexo se muestra las características de hardware de la Magallanes.

## Funciones del producto

Las principales funcionalidades que tendrá el producto son las siguientes:

- Dar de alta ecuaciones mediante una botonera que provea los operadores raíz, potencia, cociente y paréntesis.
- Resolver las ecuaciones previamente ingresadas o descargadas donde las soluciones de la ecuación serán provistas por el usuario en cada paso.
- Permitir subir las ecuaciones ingresadas por el usuario que no fueron subidas previamente, al repositorio central.
- Permitir al usuario descargar ecuaciones del repositorio central indicando la cantidad deseada o un intervalo de tiempo.
- Permitir guardar el trabajo realizado por el estudiante paso a paso mientras realiza la resolución.

## Características de los usuarios

El sistema está enfocado en los estudiantes de ciclo básico que cuenten con una Magallanes.

## Requerimientos funcionales

### Alta ecuación

El sistema debe permitir dar de alta ecuaciones de hasta tercer grado (lineales, cuadráticas y cúbicas), racionales, con radicales o combinaciones de estas, con una única variable, pudiendo estar en cualquier lado de la igualdad. Las ecuaciones se construirán mediante una botonera que cuente con las operaciones mencionadas anteriormente y el usuario deberá tipear los operadores básicos (+, -, *) y numeros, así como el símbolo de igualdad y la variable.

La ecuación se mantendrá guardada localmente y opcionalmente se podrá subir al repositorio central.

### Eliminar ecuación local

El sistema debe permitir dar de baja una ecuación que fue creada o descargada previamente. La eliminación de la ecuación será solo en la computadora del estudiante sin

modificar el repositorio central, ni los archivos de logueo locales. No se eliminarán las ecuaciones cargadas por defectos en la primer instalación.

### Resolución de ecuación

El sistema debe permitir al usuario ir resolviendo en etapas la ecuación mediante métodos aritméticos y culminar la resolución cuando el estudiante encontró todas las soluciones posibles. En el caso en que la ecuación no tenga solución, luego de una determinada cantidad de intentos se deberá mostrar un mensaje indicando que la ecuación no tiene solución.

En cada paso el sistema deberá ir guardando la solución del estudiante en un archivo local, lo que le permitirá retomar la resolución de una ecuación comenzada anteriormente.

La ecuación a resolver podrá ser tomada de las ecuaciones ingresadas o descargadas previamente. En el caso que la ecuación se haya comenzado a resolver anteriormente el usuario puede volver a comenzar la resolución o continuar con la misma.

### Listado de ecuaciones locales

El sistema debe listar las ecuaciones creadas y/o descargadas por el usuario, y proveer las opciones para eliminar o resolver las mismas.

### Subir ecuaciones a repositorio central

El sistema debe permitir subir las ecuaciones ingresadas por los usuarios al repositorio central cuando éste lo solicite. Las ecuaciones se subirán en el caso de que se cuente con conexión a Internet, sino la hay se deberá mostrar un mensaje que indique que lo intente mas tarde.

Las ecuaciones que se van a subir son las que fueron creadas localmente, no fueron subidas anteriormente y fueron seleccionadas por el usuario.

### Descargar ecuaciones

El sistema debe permitir al usuario descargar del repositorio central ecuaciones elegidas en forma aleatoria donde el usuario deberá indicar la cantidad deseada desde un combo con opciones predeterminadas, las cuales determinarán la cantidad máxima posible por descarga. Además, podrá indicar un intervalo de tiempo de las ecuaciones a descargar. Las ecuaciones descargadas serán almacenadas localmente para su posterior resolución.

### Logueo del trabajo del estudiante

El sistema debe permitir guardar los pasos en la resolución de una ecuación en un archivo local, para que el trabajo del estudiante pueda ser consultado en un futuro por los docentes. El logueo se realizará en el momento en que se realiza la resolución y dicho logueo se almacenará en un archivo asociado a la ecuación, donde se generará un archivo por ecuación resuelta.

### Listado ecuaciones terminadas y/o trabajadas

El sistema debe listar las ecuaciones que ya fueron comenzadas a resolver y/o finalizadas por el usuario y proveer la opción de ver la resolución de la ecuación.

### Ver resolución de la ecuación

El sistema debe permitir al usuario ver la resolución de una ecuación eligiendola desde un menú que lista las ecuaciones que ya fueron resueltas.

## Requerimientos no funcionales

### Ingreso de ecuaciones

El ingreso de las ecuaciones se debe realizar con botonera la cual debe contar con las operaciones (raiz, potencias, cociente y paréntesis) y los espacios para tipear los operadores básicos (+, -, *) y numeros, así como la variable.

### Trabajo local

El sistema debe permitir al usuario trabajar sin conexión a Internet, utilizando las ecuaciones guardadas previamente en la computadora y/o creadas en el momento.

### Interfaz de usuario

La interfaz debe ser atractiva al usuario, es decir, que permita con facilidad el ingreso de ecuaciones y la resolución de las mismas.

A su vez, se le consultara al cliente para corroborar si la interfaz es de su agrado.

### Ecuaciones iniciales

El sistema debe contar con ecuaciones pre cargadas y accesibles en la aplicación.

### Idioma

Los textos del sistema deben estar en Español.

### Hardware

El sistema debe correr en las laptop Ceibal Magallanes, la especificaciones técnicas están detalladas en el anexo.

# Requerimientos de documentación

## Material de capacitación

Se entregará a el cliente una presentación con las características generales del producto y ejemplos de uso.

## Guías de instalación, configuración y archivo Léame.

La aplicación se descargará desde una página web a definir, donde se indicará también los pasos para la instalación.

## Anexo

En la siguiente tabla se mostrará el resumen del hardware de los distintos tipos de Magallanes, esta información fue tomada de la página del plan ceibal. [1]

| Equipo | CPU | Memoria | Display |
|---|---|---|---|
| Magallanes MG1 | Intel Atom N270 1,66 Ghz | 1 Gbyte DDR2 | 8.9 pulgadas LCD 1024 x 600 |
| Magallanes MG2 | Intel Atom N450 1,66 Ghz | 1 Gbyte DDR2 | 10.1 pulgadas LCD 1.024 x 600 |
| Magallanes MG3 | Intel Atom N455 1,66 Ghz | 1 Gbyte DDR3 | 10.1 pulgadas LCD 1024 x 600 |
| Magallanes MG4 | Intel Celeron 847 1.10GHz | 1GB DDR3 1333 | 1024*600 (WSVGA) LCD LED |
| Magallanes MG6 | Intel Celeron 847 1.1 GHz, doble núcleo | SO-DIMM 2GB PC3-12800 | LCD LED 10.1" HD (1366x768) |

## Referencias

[1] Plan Ceibal
http://www.ceibal.edu.uy/art%C3%ADculo/noticias/estudiantes/Especificaciones-tecnicas

# Estándar de Implementación
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 30/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

# ÍNDICE

# Introducción

En este documento se presenta los estándares básicos de codificación elegidos para dar formato y escribir el código de la aplicación basado en el lenguaje Java.

El uso de estándar de implementación facilita la lectura y la comprensión del código permitiendo depurar y continuar con el desarrollo más cómodamente.

# Convenciones

Cada archivo Java contiene una única public class o interface. Si una private class o una interface están asociadas a una public class, se las puede poner en el mismo archivo Java. La public class debe ser la primera clase o interface en el archivo. Los archivos Java tienen el siguiente orden:

- Comentarios al principio
- Paquetes e Importaciones
- Declaraciones de clases e interfaces

## Paquete e importaciones

La primera línea no comentada de la mayoría de los archivos Java indica el paquete al que pertenece el archivo. Luego vienen las importaciones.

Por ejemplo:

*package java.awt;*

*import java.awt.peer.CanvasPeer;*

El prefijo de un paquete siempre se escribe en letras minúsculas comúnmente son com, edu, gov, mil, net, org, etc.

Debería evitarse usar wildcards para las importaciones (import somepackage.*), ya que tiende a llenar de elementos innecesarios el namespace o namescope y tampoco permite identificar qué clases realmente son las que están siendo usadas.

Las importaciones deben agruparse por el paquete del que se las importa, y también deberían ordenarse por el propósito de los paquetes; es decir, paquetes de propósito general primero y paquetes más específicos a lo último.

## Clases e interfaces

Para la declaración de clases e interfaces se deben seguir las siguientes reglas:

- No dejar espacios entre el nombre del método y el paréntesis "(" que empieza la lista de parámetros.
- Abrir la llave "{" al final de la línea de declaraciones de parámetros con un espacio de separación.
- Cerrar la llave "}" empieza una nueva línea y debe estar indentado al mismo nivel que la operación, excepto cuando es un método vacío en el cual la llave debe estar a continuación de la llave que abre.

Los nombres de clases e interfaces deben ser en su mayoría sustantivos, donde la primera letra de cada palabra del nombre de la clase va en mayúsculas. Tratar de mantener los

nombres simples y descriptivos. Tratar de usar acrónimos y abreviaciones a menos que genere mucha confusión.

## Indentación

La indentación debe ser escrita usando caracteres TAB. Los espacios no deben ser usados para indentar. Se recomienda que las líneas no superen los 130 caracteres.

Cuando una expresión no entra en una sola línea, es posible separarla en varias de acuerdo a ciertos principios generales:

- Partir después de una coma.
- Partir antes de un operador.
- Alinear la nueva línea con el comienzo de la expresión al mismo nivel que la línea previa.

## Declaraciones

Una declaración por línea es recomendable, ya que permite realizar comentarios sobre cada declaración en particular.

Como regla general hacer declaraciones al principio de cada bloque (se considera bloque como una sección de código entre llaves {}). En situaciones similares es conveniente retrasar la declaración hasta la primera vez que se usa

## Métodos

Los Métodos suelen ser verbos, con la primera letra del nombre en minúscula y la primera letra de las palabras internas del nombre en mayúsculas.

Los métodos setters y getters deben tener el prefijo set y get seguido del nombre de la variable a retornar siguiendo las reglas para dar nombres a los métodos. Existe una excepción con los getters, y es cuando devuelven valores booleanos, en ese caso se cambia el prefijo get por *es*.

## Constantes

Las constantes deben estar todas en mayúsculas, y sus palabras estar separadas por “_”.

## Variables

El nombre de las variables van mezcladas como los métodos, primera palabra del nombre en minúsculas y luego la primer letra de cada palabra en mayúscula. Las variables no deberían empezar con los signos _ o $, aunque están permitidos.

Deben ser cortos pero tener significado. Se deben evitar las variables de un solo carácter excepto para las temporales. Sus nombre más comunes son i, j, k, m, y n para enteros; c, d, y e para caracteres. Tratar de inicializar las variables cuando son definidas, a menos que dependa de otros cálculos.

Tratar de inicializar las variables cuando son definidas, a menos que dependa de otros cálculos. Evitar declaraciones de variables locales con el mismo

nombre que otras de mayor nivel o visibilidad.

### Comentarios de implementación

Son delimitados por /*...*/ y //. Están pensados para comentar cosas particulares de la implementación o comentar nuestro código.

### Sentencias if, if-else, if else-if else

La sentencia *if-else* tiene la siguiente forma:

```
if (condition) {
        statements;
}

if (condition) {
        statements;
} else {
        statements;
}

if (condition) {
        statements;
} else if (condition) {
        statements;
} else {
        statements;
}
```

### Sentencias for

Una sentencia *for* debe tener la siguiente forma:

```
for (initialization; condition; update) {
        statements;
}
```

Una sentencia *for* vacía (en la que se define la inicialización, condición y actualización únicamente) debe ser de la siguiente forma:

```
for (initialization; condition; update);
```

### Sentencias while

Una sentencia *while* debe ser de la siguiente forma:

```
while (condition) {
        statements;
}
```

Una sentencia while vacia debe ser:

```
while (condition);
```

### Sentencias switch

Una sentencia *switch* debe ser:

```
switch (condition) {
        case ABC:
                statements; /* falls through */
        case DEF:
                statements;
                break;
        case XYZ:
                statements;
                break;
        default:
                statements;
                break;
}
```

Cada sentencia *switch* debe incluir el caso *default*.

### Sentencias try-catch

Una sentencia *try-catch* debe ser:

```
try {
        statements;
} catch (ExceptionClass e) {
        statements;
```

```
}
```

Un *try-catch* puede también estar seguido de un *finally*, el cual se ejecuta sin importar que pase con el bloque de código del *try*.

```
try {
        statements;
} catch (ExceptionClass e) {
        statements;
} finally {
        statements;
}
```

# Plan de SQA

# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 27/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

# ÍNDICE

# Propósito

Esta sección contiene el propósito y alcance del Plan de Calidad.

Se especificará el uso que se le dará al software que se está desarrollando y se deben listar los elementos del software que serán cubiertos por el Plan.

Además se especificará la porción del ciclo de vida del software que será cubierta por el Plan.

# Gestión

La gestión de la calidad será realizada por los integrantes del grupo siguiendo lo indicado en este documento.

## Organización

Los integrantes del equipo se encargarán de todas las líneas (requisitos, análisis, diseño, implementación, testing, calidad e implantación) de desarrollo del proyecto de grado, respetando estándares y pautas de calidad previamente definidos por ellos mismos.

## Actividades

### Ciclo de vida del software cubierto por el Plan

Las actividades que cubre el Plan de Calidad van desde la implementación hasta la documentación, realizando revisiones para verificar que los estándares especificados se hayan cumplido.

### Actividades de calidad a realizarse

Las tareas a ser llevadas a cabo deberán reflejar las evaluaciones a realizar, los estándares a seguir, los productos a revisar, los procedimientos a seguir en la elaboración de los distintos productos y los procedimientos para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

Las actividades que se realizarán son:

- Revisar cada producto
- Revisar el ajuste al proceso
- Revisión técnica formal (RTF)
- Asegurar que las desviaciones son documentadas.

### Revisar cada producto

En esta actividad se revisan los productos que se definieron como claves para verificar en el Plan de calidad.

Se debe verificar que no queden correcciones sin resolver en los informes de revisión previos, si se encuentra alguna no resuelta, debe ser incluida en la siguiente revisión. Se revisan los productos contra los estándares, utilizando la checklist definida para el producto.

Se debe identificar, documentar y seguir la pista a las desviaciones encontradas y verificar que se hayan realizado las correcciones.

### Revisar el ajuste al proceso

En esta actividad se revisan los productos que se definieron como claves para verificar el cumplimiento de las actividades definidas en el proceso. Con el fin de asegurar la calidad en el producto final del desarrollo, se deben llevar a cabo revisiones sobre los productos durante todo el ciclo de vida del software.

Esta información se obtiene de el Plan del Proyecto, Plan de la iteración, Plan de Verificación.

Antes de comenzar, se debe verificar en los informes de revisión previos que todas las desviaciones fueron corregidas, si no es así, las faltantes se incluyen para ser evaluadas.

### Realizar Revisión Técnica Formal (RTF)

El objetivo de la RTF es descubrir errores en la función, la lógica ó la implementación de cualquier producto del software, verificar que satisface sus especificaciones, que se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas. Es un proceso de revisión riguroso, su objetivo es llegar a detectar lo antes posible, los posibles defectos o desviaciones en los productos que se van generando a lo largo del desarrollo.

La revisión será realizada por ambos integrantes del grupo dejando documentado los errores encontrados.

### Asegurar que las desviaciones son documentadas

Las desviaciones encontradas en las actividades y en los productos deben ser documentadas y ser manejadas de acuerdo a un procedimiento establecido.

### Relaciones entre las actividades de SQA y la planificación

Las revisiones de código serán realizadas al final de cada iteración por el integrante contrario al que escribió el código.

La revisión del ajuste del proceso se realizará al comienzo de cada iteración por ambos integrantes del equipo.

Se verificará que las correcciones que quedaban por resolver al final de cada iteración por ambos integrantes del grupo, de forma de comenzar con un entorno estable en la próxima iteración.

## Documentación

### Propósito

Identificación de la documentación relativa a desarrollo, Verificación & Validación, uso y mantenimiento del software.

Establecer como los documentos van a ser revisados para chequear consistencia: se confirman criterio e identificación de las revisiones.

### Documentación mínima requerida

La documentación mínima es la requerida para asegurar que la implementación logrará satisfacer los requisitos.

## Especificación de requisitos del software

El documento de especificación de requisitos deberá describir, de forma clara y precisa, cada uno de los requisitos esenciales del software además de las interfaces externas.

El cliente deberá obtener como resultado del proyecto una especificación adecuada a sus necesidades en el área de alcance del proyecto, de acuerdo al compromiso inicial del trabajo y a los cambios que este haya sufrido a lo largo del proyecto, que cubra aquellos aspectos que se haya acordado detallar con el cliente.

La especificación debe:

- Ser completa :
  a. Externa, respecto al alcance acordado.
  b. Internamente, no deben existir elementos sin especificar.
- Ser consistente, no pueden haber elementos contradictorios.
- Ser no ambigua, todo término referido al área de aplicación debe estar definido en un glosario.
- Ser verificable, debe ser posible verificar siguiendo un método definido, si el producto final cumple o no con cada requisito.
- Estar acompañada de un detalle de los procedimientos adecuados para verificar si el producto cumple o no con los requisitos.
- Incluir requisitos de calidad del producto a construir.

Los requerimientos de calidad del producto a construir son considerados dentro de atributos específicos del software que tienen incidencia sobre la calidad en el uso y se detallan a continuación:

*Funcionalidad*

a. adecuación a las necesidades
b. precisión de los resultados
c. interoperabilidad

*Confiabilidad*

a. tolerancia a faltas
b. recuperabilidad

*Usabilidad*

a. comprensible
b. aprendible
c. operable
d. atractivo

*Eficiencia*

a. comportamiento respecto al tiempo
b. utilización de recursos

*Mantenibilidad*

a. analizable

b. modificable
c. estable, no se producen efectos inesperados luego de modificaciones
d. verificable

*Portabilidad*

a. instalable

Cada uno de estos atributos debe cumplir con las normas y regulaciones aplicables a cada uno.

## Descripción del diseño del software

El documento de diseño específica como el software será construido para satisfacer los requerimientos.

Deberá describir los componentes y subcomponentes del diseño del software, incluyendo interfaces internas. Este documento deberá ser elaborado primero como Preliminar y luego será gradualmente extendido hasta llegar a obtener el Detallado.

El cliente deberá obtener como resultado del proyecto el diseño de un producto de software que cubra aquellos aspectos que se haya acordado con el cliente.

El diseño debe:

- Corresponder a los requisitos a incorporar:
  a. Todo elemento del diseño debe contribuir a algún requisito
  b. La implementación de todo requisito a incorporar debe estar contemplada en por lo menos un elemento del diseño.
- Ser consistente con la calidad del producto

## Plan de Verificación & Validación

El Plan de V & V deberá identificar y describir los métodos a ser utilizados en:

- La verificación de que:
  a. los requisitos descritos en el documento de requisitos han sido aprobados entre el cliente y el equipo.
  b. los requisitos descritos en el documento de requisitos son implementados en el diseño expresado en el documento de diseño.
  c. el diseño expresado en el documento de diseño está implementado en código.
- Validar que el código, cuando es ejecutado, se adecúa a los requisitos expresados en el documento de requisitos.

## Reportes de Verificación & Validación

Los resultados de la ejecución de los procesos descritos en el Plan de V & V se encontrarán registrados en la herramienta web Trello.

### Plan de Gestión de configuración

El Plan de gestión de configuración debe contener métodos para identificar componentes de software, control e implementación de cambios, y registro y reporte del estado de los cambios implementados.

## Estándares, prácticas, convenciones y métricas

### Estándar de documentación

El estándar de documentación se encuentra especificado en el documento de Configuración, el cual contiene las fuentes y tamaños a utilizar en los documentos del proyecto.

### Estándar de verificación y prácticas

Se utilizan las prácticas definidas en el Plan de Verificación y Validación.

### Estándar Implementación

Se utilizan las prácticas definidas en el documento “Estándares de Implementación.”

### Pautas para la interfáz de usuario

Se utilizarán las pautas definidas en el documento “Pautas para la interfaz de usuario.”

## Reporte de problemas y acciones correctivas

Los problemas encontrados serán reportados en la herramienta Trello y se planificaran las acciones para corregir los errores encontrados.

# Plan de Configuración
# Versión 2.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 23/08/2016 | 1.0 | Inicial | Martín Poli - Camila Rojí |
| 30/08/2016 | 1.1 | Se agrega la sección del repositorio del código fuente y documentación | Martín Poli - Camila Rojí |
| 23/09/2016 | 1.2 | Se incorpora el documento glosario a la linea base | Martín Poli - Camila Rojí |
| 17/10/2016 | 2.0 | Se quitan de los documentos el Modelo de Diseño y el Modelo de Datos | Martín Poli - Camila Rojí |

# ÍNDICE

# Introducción

## Propósito

Este documento describe las actividades de gestión de configuración de software que deben ser llevadas a cabo durante el proceso de desarrollo del proyecto. Aquí se definen tanto los productos que se pondrán bajo control de configuración como los procedimientos que deben ser seguidos por los integrantes del equipo de trabajo.

## Alcance

El Plan de configuración está basado en algunos supuestos que se detallarán:

- El tiempo de duración del proyecto está estimado entre 8 a 10 meses, por lo tanto se busca una rápida respuesta a los cambios, tratando que este procedimiento sea lo menos burocrático posible.
- El Modelo de Proceso se basa en un desarrollo incremental, dado por las distintas iteraciones. Resulta importante tener control sobre cada una de las iteraciones y fases, de los productos generados en estas y de los cambios surgidos, evaluados y aprobados.

## Actividades de Gestión de Configuración del Software

Identifica todas las actividades y tareas que se requieren para el manejo de la configuración del sistema. Estas deben ser tanto actividades técnicas como de gestión de SCM, así como las actividades generales del proyecto que tengan implicancia sobre el manejo de configuración.

# Identificación de la configuración

## Elementos de configuración

- Documento de riesgos
- Descripción de la arquitectura
- Especificación de requisitos
- Estándar de implementación
- Plan de calidad
- Plan de configuración
- Plan de verificación y validación
- Prototipo (Riesgos técnicos)
- Pautas para la interfaz de usuario
- Alcance del sistema
- Gestión de cambios
- Modelo de casos de prueba
- Modelo de casos de uso
- Plan de proyecto
- Material para capacitación
- Plan de implantación
- Código fuente

- Glosario
- Scripts de base de datos
- Informe del proyecto de grado
- Acta de reunión

## Nomenclatura de Elementos

En esta sección se especifican la identificación y descripción única de cada elemento de configuración.

Además se especifica como se distinguirán las diferentes versiones de cada elemento.

Para todos los elementos de configuración del tipo documento, se les deberá agregar, después del nombre del mismo, la versión del mismo.

El formato para esta nomenclatura es: **NombreDocumento_v**Y.X.extensión, donde:

· Y indica la versión del elemento de configuración o entregable.

. X indica la cantidad de modificaciones que tuvo el documento en la fase especificada.

· Extensión indica la extensión del elemento de configuración o entregable.

La versión del elemento se corresponde con la fase del plan de desarrollo y con la cantidad de veces que se modificó el documento. Es decir la versión 2.1 indica que fue modificado en la segunda fase y se realizó una modificación al documento.

Ejemplo: PlanDeCalidad_v2.0.doc, es como se deberá llamar el entregable "Plan de Calidad" cuya versión del documento es la 2.0.

Para los elementos, se deberá identificar a que Fase e iteración corresponden en forma manual. Esto es: para los elementos bajo control de configuración se los almacenará de forma que se puedan recuperar dada la Fase e iteración a la que corresponden.

## Elementos de la Línea Base del Proyecto

| **FASE: Inicial** | | |
|---|---|---|
| **Elemento** | **Descripción** | **Disciplina** |
| EspecificaciónDe requisitos | Especificación de requisitos | Requisitos |
| AlcanceSistema | Alcance del sistema | Requisitos |
| Riesgos | Documento de riesgos | Gestión de proyecto |
| PlanDeCalidad | Plan de calidad | Gestión de calidad |
| PlanDeConfiguración | Plan de configuración | Gestión de configuración |
| PlanDeVerificaciónValidación | Plan de verificación y validación | Verificación |
| PlanDeProyecto | Plan de proyecto | Gestión de proyecto |
| Estándar Implementación | Estándar de implementación | Gestión de calidad |
| Informe | Informe final del proyecto | |

| | | |
|---|---|---|
| Glosario | | |

En la fase de especificación se agrega a la línea base los siguientes documentos:

| **FASE: Especificación** | | |
|---|---|---|
| **Elemento** | **Descripción** | **Disciplina** |
| DescripciónArquitectura | Descripción de la arquitectura | Diseño |
| ModeloDeDatos | Modelo de datos | Diseño |
| ModeloDeDiseño | Modelo de diseño | Diseño |
| ModeloCasosDePrueba | Modelo de casos de prueba | Verificación |
| ModeloCasosDeUso | Modelo de casos de uso | Diseño |
| InterfazDeUsuario | Pautas para la interfaz de usuario | Diseño |
| Prototipo | Prototipos de riesgos técnicos | Implementación |

En la fase de construcción se agrega a la línea base:

| **FASE: Construcción** | | |
|---|---|---|
| **Elemento** | **Descripción** | **Disciplina** |
| GestiónDeCambios | Gestión de cambios | Gestión de la configuración y control de cambios |

Finalmente la fase de transición incorpora a la línea base los siguientes documentos:

| **FASE: Transición** | | |
|---|---|---|
| **Elemento** | **Descripción** | **Disciplina** |
| Capacitación | Material para capacitación | Implantación |
| PlanDeImplantación | Plan de implantación | Implantación |

## Configuración del repositorio de código fuente

El repositorio en el que se guardará el código es Bitbucket. En este repositorio se tendrá una rama principal que tendrá el código liberado y si fuera necesario una rama develop en la cual se desarrollará durante las iteraciones. En dicho caso, una vez finalizada una iteración se mergea el código con la rama principal para que ésta tenga el código actualizado.

Los commits realizados durante el desarrollo deberán tener un breve comentario descriptivo de la funcionalidad, a su vez se deberá agregar el identificador del caso de uso que se está desarrollando.

## Configuración del repositorio de documentos

El repositorio de documentos es Sharelatex y se encontrarán respaldados en Google Drive.

Se contarán con dos carpetas, una con los documentos generados durante el proceso de desarrollo y otra que contiene el informe final.

A su vez, dentro de la carpeta de documentos se encontrarán sub carpetas que contienen la línea base de fase.

Los nombres de los documentos deberán respetar la nomenclatura antes mencionada.

# Plan de Verificación y Validación
# Versión 1.1

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 26/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| 03/10/2016 | 1.1 | Se agrega la planificación para las fases de especificación y construcción | Camila Rojí - Martín Poli |
| | | | |
| | | | |

# ÍNDICE

# Introducción

## Propósito

Este Plan de Verificación para el proyecto de grado soporta los siguientes objetivos:

Identificar la información de proyecto existente y los componentes de software que deben ser verificados.

Enumerar los requerimientos recomendados para verificar.

Recomendar y describir las estrategias de verificación que serán usadas.

Identificar los recursos necesarios y proporcionar una estimación de esfuerzo para realizar la verificación.

Enumerar los entregables del proyecto de verificación.

## Punto de partida

Esta sección contiene una breve descripción del destino de la verificación (componentes, aplicación, sistema, etc.) y sus objetivos.

## Alcance

En esta sección describe los tipos de prueba que se realizarán a lo largo del proyecto.

Pruebas unitarias: Las pruebas unitarias se realizarán por la misma persona que implementó el módulo, de forma de verificar las funciones implementadas para desarrollar el requisito.

Pruebas de integración: el objetivo es verificar que los componentes ya verificados, trabajen juntos como un sistema integrado.

Pruebas funcionales: Se verificará que el sistema cumpla con los requisitos funcionales.

Pruebas de aceptación: Se verificará que el sistema cumpla con los requisitos del cliente.

Pruebas de instalación: Se probará que el sistema se instale y corra correctamente en el ambiente de producción.

# Requerimientos para verificar

Se verificarán todos los requisitos que se encuentran detallados en el documento “Especificación de requisitos”.

# Estrategia de Verificación

Esta sección presenta el enfoque recomendado para la verificación. Describe cómo se verificarán los elementos.

Se indicarán las técnicas usadas y el criterio para saber cuando una prueba se completó (criterio de aceptación).

## Tipos de pruebas

### Prueba de integridad de los datos y la base de datos

#### *Objetivo de la prueba*

Asegurar que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos.

#### *Técnica*

Invoque cada método o proceso de acceso a la base de datos con datos válidos y no válidos.

Inspeccione la base de datos para asegurarse de que se han guardado los datos correctos, que todos los eventos de la base de datos ocurrieron correctamente, o repase los datos devueltos para asegurar que se recuperaron datos correctos por la vía correcta.

#### *Criterio de aceptación*

Todos los métodos y procesos de acceso a la base de datos funcionan como fueron diseñados y sin datos corruptos.

#### *Consideraciones especiales*

La prueba requiere un entorno de administración de DBMS o controladores para ingresar o modificar información directamente en la base de datos.

Los procesos deben ser invocados manualmente.

Se deben usar bases de datos pequeñas para aumentar la facilidad de inspección de los datos para verificar que no sucedan eventos no aceptables.

### Prueba de Funcionalidad

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos o funciones y reglas del negocio. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio. Este tipo de prueba se basa en técnicas de caja negra, que consisten en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interfase de usuario y analizar los resultados obtenidos.

#### *Objetivo de la prueba*

Asegurar la funcionalidad apropiada del objeto de prueba, incluyendo la navegación, entrada de datos, proceso y recuperación.

#### *Técnica*

Ejecute cada caso de uso, flujo de caso de uso, o función usando datos válidos y no válidos, para verificar lo siguiente:

- Se obtienen los resultados esperados cuando se usan datos válidos.

- Cuando se usan datos no válidos se despliegan los mensajes de error o advertencia apropiados.
- Se aplica apropiadamente cada regla del negocio.

***Criterio de aceptación***

Todas las pruebas planificadas se realizaron. Todos los defectos encontrados han sido debidamente identificados.

## Prueba de Interfase de Usuario

Esta prueba verifica que la interfaz de usuario proporcione al usuario el acceso y navegación a través de las funciones apropiada.

***Objetivo de la prueba***

Verificar que: la navegación a través de los elementos que se están probando reflejen los requerimientos, incluyendo manejo de ventanas, campos y métodos de acceso.

***Técnica***

Crear o modificar pruebas para cada ventana verificando la navegación y los estados de los objetos para cada ventana de la aplicación y cada objeto dentro de la ventana.

***Criterio de aceptación***

Cada ventana ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.

## Prueba de Instalación

La Prueba de Instalación tiene dos propósitos. Uno es asegurar que el software puede ser instalado en diferentes condiciones (como una nueva instalación, una actualización, y una instalación completa o personalizada) bajo condiciones normales y anormales. Condiciones anormales pueden ser insuficiente espacio en disco, falta de privilegios para crear directorios, etc. El otro propósito es verificar que, una vez instalado, el software opera correctamente. Esto significa normalmente ejecutar un conjunto de pruebas que fueron desarrolladas para Prueba de Funcionalidad.

***Objetivo de la prueba***

Verificar que el software objeto de prueba se instala correctamente en cada configuración de hardware requerida bajo las siguientes condiciones:

- instalación nueva, una nueva máquina, nunca instalada previamente.
- actualización, máquina previamente instalada con la misma versión
- actualización, máquina previamente instalada con una versión anterior.

***Técnica***

Manualmente o desarrollando programas, para validar la condición de la máquina destino (nueva, nunca instalado, misma versión, versión anterior ya instalada).

Realizar la instalación.

Ejecutar un conjunto de pruebas funcionales ya implementadas para la Prueba de Funcionalidad.

***Criterio de aceptación***

Las pruebas de funcionalidad se ejecutan exitosamente sin fallas.

### Herramientas

PgAdmin: PostgresSQL

Reporte de bugs y administración del proyecto: Trello

Pruebas unitarias: JUnit

### Sistema

En la siguiente tabla se establecen los recursos de sistema necesarios para realizar la verificación.

| Recurso | Nombre/Tipo |
|---|---|
| PC Cliente para pruebas | Laptop Magallanes/Positivo |
| Requerimientos especiales | Memoria ram 1GB |

## Hitos del proyecto de Verificación

### Fase Especificación

Como en esta fase sólo se realizarán prototipos y no se implementará caso de uso, sólo se realizará testing unitario para los algoritmos de resolución de la ecuación y para la formación del string de la ecuación generada por botonera.

### Fase Construcción

En todas las fases se realizará testing unitario correspondientes a los casos de uso especificado en la planificación del Alcance del sistema.

Al finalizar cada iteración se realizará testing del sistema para verificar que los casos de uso satisfagan los requisitos planteados por el cliente además verificar que se cumpla con los criterios de aceptación mencionados anteriormente para los distintos tipos de prueba.

En la última iteración se coordinará con el cliente para realizar pruebas con estudiantes para verificar la amigabilidad y que la interfaz sea intuitiva.

# Apéndice

## Niveles de gravedad de error

En muchas actividades del proceso de verificación se deben clasificar los errores según su nivel de gravedad. Se asigna un nivel de gravedad a los errores para poder capturar de alguna manera su impacto en el sistema. Además para poder evaluar la verificación y el sistema.

A continuación se da una sugerencia de cuatro niveles diferentes de gravedad de error:

- **Catastrófico**: un error cuya presencia impide el uso del sistema.
- **Crítico**: un error cuya presencia causa la pérdida de una funcionalidad crítica del sistema. Si no se corrige el sistema no satisfará las necesidades del cliente.
- **Marginal**: un error que causa un daño menor, produciendo pérdida de efectividad, pérdida de disponibilidad o degradación de una funcionalidad que no se realiza fácilmente de otra manera.
- **Menor**: un error que no causa perjuicio al sistema, pero que requiere mantenimiento o reparación. No causa pérdida de funcionalidades que no se puedan realizar de otra manera.

## Niveles de aceptación para lo elementos verificados

Se debe establecer un nivel de aceptación para los elementos verificados para poder establecer el estado en el que se encuentra el proyecto.

En esta sección defina niveles de aceptación y los criterios de pertenencia a cada nivel.

Como ejemplo de niveles de aceptación:

- **No aprobado**: el elemento verificado tiene errores catastróficos (uno o varios) que impiden su uso o tiene errores críticos (uno o varios) que hacen que el elemento verificado no sea confiable. El usuario no puede depender de él para realizar el trabajo.
- **Aprobado con Observaciones**: el elemento verificado no tiene errores catastróficos, ni errores críticos, pero tiene errores marginales (uno o varios) que hacen que el elemento de software se degrade en algunas situaciones.
- **Aprobado**: el elemento verificado no tiene errores o tiene errores menores que no afectan el normal funcionamiento del elemento.

# Alcance del Sistema
# Versión 2.2

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 03/10/2016 | 1.0 | Inicial | Martín Poli - Camila Rojí |
| | 2.0 | Justificación utilización de puntos de función para las mediciones. | Martín Poli - Camila Rojí |
| 19/12/2016 | 2.1 | Se ajusta el cronograma para recuperar atrasos | Martín Poli - Camila Rojí |
| 02/07/2017 | 2.2 | Se agrega la sección atrasos en la planificación | Martín Poli - Camila Rojí |

# ÍNDICE

### Introducción

En este documento se presenta las mediciones realizadas y la estimación para la realización del proyecto, así como la definición del alcance para el proyecto de grado.

### Propósito

Este documento describe el alcance del sistema en términos de los Casos de Uso que se implementarán a lo largo de todo el ciclo del proyecto y en qué fase e iteración está previsto que se implementen.

### Referencias

Los documentos que utilizamos para la realización de las mediciones y la determinación del alcance fueron “Especificación de requisitos”, “Modelo de casos de uso” y “Plan de proyecto”

## Mediciones

Las mediciones para el proyecto de grado fueron realizadas con puntos de función. Se midió cada caso de uso que se encontraba especificado y se calcularon los puntos de función con la herramienta COCOMO II.

A continuación mostramos la figura que contiene los cálculos realizados.

roject Name: ProyectoGrado

Project Notes

| Module Name | Module Size | LABOR Rate ($/month) | EAF | Language | NOM Effort DEV | EST Effort DEV | PROD |
|---|---|---|---|---|---|---|---|
| Alta ecuacion | F:530 | 0.00 | 1.00 | JAVA | 1.9 | 1.9 | 284.5 |
| ListadoEcuacio | F:530 | 0.00 | 1.00 | JAVA | 1.9 | 1.9 | 284.5 |
| EliminarEcuaci | F:530 | 0.00 | 1.00 | JAVA | 1.9 | 1.9 | 284.5 |
| ResolucionEcua | F:1219 | 0.00 | 1.00 | JAVA | 4.3 | 4.3 | 284.5 |
| ListadoResuelt | F:530 | 0.00 | 1.00 | JAVA | 1.9 | 1.9 | 284.5 |
| VerResolucion | F:530 | 0.00 | 1.00 | JAVA | 1.9 | 1.9 | 284.5 |
| DescargarEcuac | F:1219 | 0.00 | 1.00 | JAVA | 4.3 | 4.3 | 284.5 |
| Subir ecuacion | F:901 | 0.00 | 1.00 | JAVA | 3.2 | 3.2 | 284.5 |

| | |
|---|---|
| Total Lines of Code: | 5989 |
| Hours/PM: | 152.00 |

| Estimated | Effort | Sched | PROD | COST | INST | Staff | RISK |
|---|---|---|---|---|---|---|---|
| Optimistic | 14.1 | 8.5 | 424.7 | 0.00 | 0.0 | 1.7 | |
| Most Likely | 21.0 | 9.7 | 284.5 | 0.00 | 0.0 | 2.2 | 0.0 |
| Pessimistic | 31.6 | 11.0 | 189.7 | 0.00 | 0.0 | 2.9 | |

La columna Sched que se muestra en la figura indica la cantidad de meses del tiempo de desarrollo para un solo recurso. Se puede ver, que la estimación optimista es de 8.5 meses, las más probable es 9.7 y la pesimista duraría 11 meses.

Al ser 2 recursos para implementar, la cantidad de meses se reduciría a la mitad dando como resultado las siguiente estimaciones.

Caso optimista: 4.25 meses

Caso más probable: 4.85

Caso pesimista: 5.5

Para cumplir con estos tiempos es necesario que los integrantes del equipo, dediquen 5 horas diarias a la implementación.

Considerando que las tres estimaciones están dentro del plazo especificado en el documento "Plan de proyecto", es posible realizar todos los requisitos especificados en el documento "Especificación de requisitos."

## Planificación para lograr el Alcance

La planificación es en base a los Casos de Uso , mencionando su nombre y el nombre de la funcionalidad, e indicando en qué fase e iteración serán incluidos en el sistema.

### **Fase Especificación** - Octubre, Enero 2017

**Iteración 1 -** Duración 1 mes

- Diseño de la interfaz gráfica.

I**teración 2 -** Duración 3 meses

- Implementación de la botonera.
- Generar string de la ecuación a partir de la botonera
- Implementación de algoritmos para la resolución de la ecuación.

### **Fase Construcción -** De Febrero 2017 a Abril 2017

En las siguientes iteraciones se van a listar los casos de uso por orden de prioridad (de mayor a menor) y precedencia, de forma de ir implementando los requisitos en orden.

**Iteración 1 -** Duración 1 mes

Los casos de uso a implementar en esta iteración son:

- Alta ecuación
- Resolución de ecuación

**Iteración 2 -** Duración 15 días

Los casos de uso a implementar en esta iteración son:

- Listado de ecuaciones locales
- Listado de ecuaciones resueltas
- Ver resolución de ecuación

**Iteración 3** - Duración 1 mes

Los casos de uso a implementar en esta iteración son:

- Descarga de ecuación
- Subir ecuación a repositorio central
- Eliminar ecuación local

**Iteración 4 -** Duración 15 días

En esta iteración se realizará testing funcional a la aplicación además de las pruebas con el cliente

## Atrasos en la planificación

El desarrollo de la aplicación de la aplicación se atrasó un mes más de lo planificado finalizando a fines de mayo. Esto se debió a que las profesoras clientas de nuestro software solicitaron unos cambios con respecto a la resolución de la ecuación. En ese momento, nosotros al resolver ecuaciones que tienen como solución más de una raíz, indicábamos que había más de una solución una vez que ingresaban el resultado de una variable, siendo éste correcto, e inmediatamente reescribíamos la x para que ellos ingresaran el otro resultado. Como muchas veces el paso que puede tener más de una solución es antes de llegar a la variable, por ejemplo en la ecuación (x+1)^2 + 10 = 26, el paso que tiene dos resultados es la selección de x+1, que puede valer 4 o -4. Entonces, como al llegar al final de la ecuación se solicita que se ingrese el otro valor de la x, es bastante difícil de razonar, ya que mentalmente tienes que ir para atrás y detectar que otro paso tenía otro valor posible. Por eso, nos solicitaron que en vez de preguntar por el otro valor de la x, preguntamos por el otro valor que tiene el paso que tiene más de un resultado.

Como consideramos que la petición era bastante coherente y necesaria para la utilización del software decidimos realizar el cambio, teniendo que prolongar un poco más la etapa de desarrollo.

El testing funcional de la aplicación fue realizando en las siguientes dos semanas de la finalización del desarrollo, en la que la prueba de aceptación fue realizada el 16 de junio del 2017 y dimos por finalizado el desarrollo del software en esa fecha.

# Modelo de Casos de Prueba
# Versión 1.1

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 10/05/2017 | 1.0 | Inicial | Camila Rojí |
| 10/06/2017 | 1.1 | Se agregan casos de prueba a la resolución de ecuaciones y al listado de ecuaciones | Camila Rojí |
| | | | |
| | | | |

# ÍNDICE

# Casos y procedimientos de Pruebas del Sistema

Se detallan los Casos de pruebas del Sistema que se deben verificar para satisfacer los requisitos del sistema y los casos de uso especificados.

Escenarios

Se plantean los casos de prueba separando por cada caso de uso.

## Caso de uso Alta ecuación

### Caso de prueba 1

Título: Caso común

Descripción: Se debe ingresar una ecuación sintácticamente correcta y guardar con éxito.

Pasos:

1. En la vista de crear ecuación se debe ingresar una ecuación sintácticamente correcta.
2. Cliquear en el botón para dar de alta la ecuación
3. Se debe mostrar un mensaje indicando que la ecuación fue creada con éxito.
4. Se debe comprobar que en el listado se muestra la ecuación creada anteriormente.

### Caso de prueba 2

Título: Probar botonera

Descripción: Se debe verificar que los botones de la botonera ingresan el símbolo que representan.

Pasos:

1. En la vista de crear ecuación se debe ingresar una ecuación que contenga los operadores potencia cuadrada, raíz cuadrada, cociente, paréntesis y el opuesto. La ecuación debe ser sintácticamente válida.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que la ecuación fue creada con éxito.
4. Se debe verificar que la ecuación antes creada se encuentra en el listado.

### Caso de prueba 3

Título: Probar botones potencia y raíz cúbica

Descripción: Se debe verificar que los botones de potencia y raíz cúbica ingresan el símbolo que representan.

Pasos:

1. En la vista para crear una ecuación se debe ingresar una ecuación que contenga los operadores potencia cúbica y raíz cúbica. La ecuación debe ser sintácticamente válida.
2. Comprobar que la ecuación ingresada muestra los símbolos correctamente.
3. Cliquear en el botón para dar de alta la ecuación.
4. Comprobar que se muestra un mensaje indicando que la ecuación fue creada con éxito.
5. Comprobar que en el listado se muestra la ecuación recién creada.

### Caso de prueba 4

Título: Probar potencia a la ½

Descripción: Probar que el botón de potencia a la ½ muestre el símbolo correctamente

y que luego de crear la ecuación, en el listado se muestra raíz de dos en vez de la potencia a la ½.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga el símbolo de potencia a la ½. La sintaxis de la ecuación debe ser correcta.
2. Ver que la ecuación creada contiene el símbolo de potencia a la ½ dibujado correctamente.
3. Cliquear en el botón para dar de alta la ecuación.
4. Comprobar que se muestra un mensaje indicando que la ecuación fue dada de alta correctamente.
5. Verificar que en el listado se muestra la ecuación, pero en vez de la potencia a la un medio se muestra la raíz de dos.

### Caso de prueba 5

Título: Botón potencia a la ⅓

Descripción: Probar que el botón de potencia a la ⅓ se muestre correctamente y se guarde como raíz cúbica.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga el símbolo de potencia a la ⅓ y que contenga una sintaxis correcta.
2. Comprobar que en la vista se muestra correctamente el símbolo de potencia a la ⅓
3. Cliquear en el botón para dar de alta la ecuación.
4. Ver que se muestra un mensaje indicando que la ecuación fue dada de alta correctamente.
5. Verificar que en el listado de las ecuaciones se muestra la ecuación recién creada, pero en vez de que esté el símbolo de potencia a la ⅓ se encuentre el símbolo de raíz cúbica.

### Caso de prueba 6

Título: Botón potencia a la menos uno

Descripción: Verificar que al cliquear el botón de potencia a la menos uno se muestra el símbolo correctamente y que la ecuación queda guardada como 1/a

Pasos:

1. En la vista para dar de alta la ecuación insertar una ecuación que contenga el símbolo potencia a la menos uno y que la sintaxis de la ecuación sea correcta.
2. Comprobar que en la vista se muestra correctamente el símbolo de potencia a la menos uno
3. Cliquear en el botón para dar de alta la ecuación.
4. Ver que se muestra un mensaje indicando que la ecuación fue dada de alta correctamente.
5. Verificar que en el listado de las ecuaciones se muestra la ecuación recién creada, pero en vez de que esté el símbolo de potencia a la un medio se encuentre el símbolo de 1/a donde a es elemento elevado a la menos uno.

### Caso de prueba 7

Título: Botón potencia a la menos dos

Descripción: Verificar que al cliquear el botón de potencia a la menos dos se muestra el símbolo correctamente y que la ecuación queda guardada como 1/a^2

Pasos:

1. En la vista para dar de alta la ecuación insertar una ecuación que contenga el símbolo potencia a la menos dos y que la sintaxis de la ecuación sea correcta.

2. Comprobar que en la vista se muestra correctamente el símbolo de potencia a la menos dos
3. Cliquear en el botón para dar de alta la ecuación.
4. Ver que se muestra un mensaje indicando que la ecuación fue dada de alta correctamente.
5. Verificar que en el listado de las ecuaciones se muestra la ecuación recién creada, pero en vez de que esté el símbolo de potencia a la menos dos se encuentre el símbolo de 1/a^2 donde a es elemento elevado a la menos dos.

### Caso de prueba 8

Título: Botón potencia a la menos tres

Descripción: Verificar que al cliquear el botón de potencia a la menos tres se muestra el símbolo correctamente y que la ecuación queda guardada como 1/a^3

Pasos:

1. En la vista para dar de alta la ecuación insertar una ecuación que contenga el símbolo potencia a la menos tres y que la sintaxis de la ecuación sea correcta.
2. Comprobar que en la vista se muestra correctamente el símbolo de potencia a la menos tres
3. Cliquear en el botón para dar de alta la ecuación.
4. Ver que se muestra un mensaje indicando que la ecuación fue dada de alta correctamente.
5. Verificar que en el listado de las ecuaciones se muestra la ecuación recién creada, pero en vez de que esté el símbolo de potencia a la menos tres se encuentre el símbolo de 1/a^3 donde a es elemento elevado a la menos tres.

### Caso de prueba 9

Título: Comenzar con el símbolo de resta al inicio

Descripción: Verificar que al crear una ecuación con el símbolo de resta al inicio se muestra un mensaje de error indicando que falta un operando en la ecuación.

Pasos:

1. En la vista para dar de alta la ecuación insertar una ecuación que comience con el símbolo de resta.
2. Cliquear en el botón para dar de alta la ecuación.
3. Ver que se muestra un mensaje indicando que falta un operando a la ecuación.

### Caso de prueba 9

Título: Ecuación con cociente pero sin numerador

Descripción: Probar que se muestra un mensaje de error cuando se intenta crear una ecuación con un cociente sin numerador.

Pasos:

1. En la vista de alta ecuación crear una ecuación con el operador de cociente y no insertarle el numerador.
2. Cliquer en el botón para dar de alta la ecuación
3. Ver que se muestra un mensaje indicando que falta un operando a la ecuación.

### Caso de prueba 10

Título: Ecuacion con cociente pero sin denominador

Descripción: Probar que se muestra un mensaje de error cuando se intenta crear una ecuación con un cociente sin denominador.

Pasos:

1. En la vista de alta ecuación crear una ecuación con el operador de cociente y

no insertarle el denominador.
2. Cliquer en el botón para dar de alta la ecuación
3. Ver que se muestra un mensaje indicando que falta un operando a la ecuación.

### Caso de prueba 11

Título: Ecuación con cociente 0

Descripción: Probar que se muestra un mensaje de error cuando se intenta crear una ecuación con un cociente con valor cero.

Pasos:

1. En la vista para dar de alta una ecuación, crear una que tenga cociente con valor cero.
2. Cliquear en el botón para dar de alta la ecuación.
3. Ver que se muestra un mensaje indicando que hay una división por cero.

### Caso de prueba 12

Título: Ecuación con operador de suma pero un solo operando

Descripción: Probar que al intentar crear una ecuación que tenga el operador suma pero le falte un sumando se muestre un mensaje de error.

Pasos:

1. En la vista para dar de alta una ecuación crear una que contenga el símbolo de suma pero le falte un sumando.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje que indique que falta un operando.

### Caso de prueba 13

Título: Ecuación con operador opuesto pero sin operando

Descripción: Verificar que al crear un ecuación con el operador opuesto pero sin operando se muestra un mensaje de error indicando que falta un operando.

Pasos:

1. En la vista para dar de alta una ecuación, crear una que contenga el operador de opuesto pero sin el operando.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que falta un operando en la ecuación.

### Caso de prueba 14

Título: Ecuación con operador paréntesis pero sin operando

Descripción: Verificar que al crear una ecuación con paréntesis sin nada dentro siendo éste un operando de otro operador, se muestra un mensaje de error indicando que falta un operando.

Pasos:

1. En la vista para dar de alta la ecuación, crear una que contenga como operando de un operador al operador paréntesis sin nada dentro.
2. Cliquer en el botón para dar de alta la ecuación
3. Se debe mostrar un mensaje de error indicando que falta un operando.

### Caso de prueba 15

Título: Ecuación con solución vacía

Descripción: Verificar que se puede crear una ecuación cuyo conjunto solución sea vacío.

Pasos:

1. En la vista para dar de alta una ecuación crear una que contenga como solución un número imaginario.
2. Cliquear en el botón para dar de alta la ecuación.
3. Verificar que se muestra un mensaje indicando que se pudo dar de alta la ecuación correctamente.
4. Verificar que en el listado de ecuaciones se encuentra la ecuación creada recientemente.

### Caso de prueba 16

Título: Ecuación con dos ocurrencias de la variable

Descripción: Verificar que se muestra mensaje de error cuando se intenta crear una ecuación con más de una ocurrencia de la variable.

Pasos:

1. En la vista para dar de alta una ecuación, crear una que tenga más de una vez la variable.
2. Cliquer en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis de la ecuación es incorrecta.

### Caso de prueba 17

Título: Ecuación con dos símbolos de igual

Descripción: Verificar que al intentar crear una ecuación con más de un símbolo de igual se muestra un mensaje de error indicando que la sintaxis de la ecuación no es correcta.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una que contenga más de un símbolo de igual.
2. Cliquer en el botón para dar de alta la ecuación.
3. Ver que se muestra un mensaje indicando que la sintaxis de la ecuación no es correcta.

### Caso de prueba 18

Título: Ecuación con símbolo de producto implícito al lado de variable

Descripción: Verificar que se puede crear una ecuación correctamente que contenga un símbolo de multiplicación implícito al lado de la variable.

Pasos:

1. En la vista para dar de alta la ecuación, ingresar una que contenga la variable pegada a un número.
2. Cliquer en el botón para dar de alta la ecuación.
3. Ver que se muestra un mensaje indicando que se pudo dar de alta la ecuación correctamente.
4. Verificar que la ecuación recién creada se encuentra en el listado de ecuaciones.

### Caso de prueba 19

Título: Ecuación con símbolo de producto implícito al lado de paréntesis izquierdo.

Descripción: Verificar que se puede crear una ecuación con el símbolo de multiplicación implícito al lado del paréntesis izquierdo.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una que contenga un número antes de un paréntesis que abre.

2. Cliquer en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que se pudo dar de alta la ecuación correctamente.
4. Verificar que la ecuación recién creada se encuentra en el listado de ecuaciones.

### Caso de prueba 20

Título: Ecuación con símbolo de producto implícito al lado del paréntesis derecho

Descripción: Verificar que se puede crear una ecuación con el símbolo de producto implícito al lado de un paréntesis derecho.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una que contenga un número o una variable luego de un paréntesis derecho.
2. Cliquer en el botón para dar de alta una ecuación.
3. Se debe mostrar un mensaje indicando que se dió de alta exitosamente
4. Verificar que la ecuación recién creada se encuentra en el listado de ecuaciones.

### Caso de prueba 21

Título: Ecuación con simbolo de producto implícito entre paréntesis que cierra y paréntesis que abre.

Descripción: Verificar que se puede crear una ecuación con el producto de dos factores rodeados por paréntesis.

Pasos:

1. En la vista para dar de alta una ecuación crear una ecuación del tipo (expresión)(expresión).
2. Cliquer en el botón para dar de alta la ecuación
3. Ver que se muestra un mensaje indicando que la ecuación se pudo crear con éxito.
4. Verificar que en el listado se muestra la ecuación recién creada.

### Caso de prueba 22

Título: Crear ecuación con variable del lado derecho de la igualdad

Descripción: Verificar que se puede crear una ecuación con la variable del lado derecho de la igualdad.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga la variable en el lado derecho de la igualdad.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que la ecuación fue dada de alta exitosamente.
4. Verificar que la ecuación recién creada se encuentra en el listado de ecuaciones.

### Caso de prueba 23

Título: Crear ecuación con símbolo de multiplicación pero sin un operador

Descripción: Verificar que al intentar crear una ecuación con el operador de multiplicación pero sin un operador, se muestra un mensaje de error.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una que contenga el símbolo de multiplicación y le falte un operador.

2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error que indique que falta un operando en la ecuación.

### Caso de prueba 24

Título: Crear ecuación sin variable

Descripción: Verificar que al intentar crear una ecuación sin variable se muestra un mensaje de error.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una pero sin la variable.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis de la ecuación no es correcta.

### Caso de prueba 25

Título: Crear ecuación vacía

Descripción: Verificar que al intentar crear una ecuación sin haber insertado nada en la vista, se muestra un mensaje de error indicando que la sintaxis es incorrecta.

Pasos:

1. Entrar a la vista para dar de alta una ecuación.
2. Cliquer en el botón para dar de alta una ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis de la ecuación no es correcta.

### Caso de prueba 26

Título: Crear ecuación con una solución

Descripción: Verificar que se puede crear una ecuación que contenga sólo una solución.

Pasos:

1. En la vista para dar de alta una ecuación, insertar una que contenga una sola solución.
2. Cliquer en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que se dió de alta correctamente.
4. Verificar que la ecuación recién creada se muestra en el listado de las ecuaciones.

### Caso de prueba 27

Título: Crear ecuación con raíz doble

Descripción: Verificar que se puede crear una ecuación que tenga como solución a un raíz doble.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que tenga como solución una raíz doble.
2. Cliquer en el botón para dar de alta una ecuación.
3. Se debe mostrar un mensaje indicando que la ecuación se dió de alta correctamente.
4. Verificar que la ecuación se encuentra en el listado de las ecuaciones.

### Caso de prueba 28

Título: Crear ecuación con dos raíces distintas.

Descripción: Verificar que se puede crear una ecuación que contenga dos raíces distintas.

Paso:

1. En la vista para dar de alta una ecuación, insertar una que contenga como solución dos raíces distintas.
2. Cliquear en el botón para dar de alta una ecuación.
3. Se debe mostrar un mensaje que indica que la ecuación fue dada de alta correctamente.
4. Verificar que en el listado de las ecuaciones se encuentra la ecuación recién creada.

### Caso de prueba 29

Título: Crear ecuación con operador sin operandos

Descripción: Verificar que se muestra mensaje de error cuando se intenta crear una ecuación con un operador que no tiene ningún operando.

Pasos:

Para cada operador

1. En la vista de crear ecuación, ingresar una ecuación con el operador sin ningún operando.
2. Cliquer en el botón para dar de alta la ecuación
3. Se debe mostrar un mensaje de error indicando que la sintaxis de la ecuación es incorrecta.

### Caso de prueba 30

Título: Crear ecuación con resta sin el segundo operando

Descripción: Verificar que se muestra mensaje de error cuando se intenta crear una ecuación con el símbolo de resta sin el segundo operando.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación con el símbolo de resta pero sin el segundo operando.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis no es correcta.

### Caso de prueba 31

Título: Crear ecuación con potencia pero sin operando

Descripción: Verificar que cuando se intenta crear una ecuación con una potencia sin operando se muestra un mensaje de error.

Pasos:

Para cada operador de potencia

1. En la vista para dar de alta una ecuación, ingresar una ecuación con el operador de potencia sin operando.
2. Cliquear en el botón para dar de alta la ecuación.
3. Ver que se muestra un mensaje indicando que falta un operador.

### Caso de prueba 32

Título: Crear ecuación con el operador raíz sin operando

Descripción: Verificar que cuando se intenta dar de alta una ecuación con un operador raíz sin operando se muestra un mensaje de error.

Pasos:

Para cada operador de raíz

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga el símbolo de raíz sin operador.
2. Cliquer en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que falta un operador.

### Caso de prueba 33

Título: Crear ecuación con un lado de la igualdad vacía.

Descripción: Verificar que al intentar crear una ecuación con un lado de la igualdad vacía se muestra un mensaje de error.

Pasos:

1. En la vista para dar de alta una ecuación, insertar una que contenga un lado de la ecuación vacía.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis de la ecuación es incorrecta.
4. Repetir los primeros tres puntos con el otro lado de la ecuación.

### Caso de prueba 34

Título: Crear ecuación con dos variables distintas.

Descripción: Verificar que se muestra un mensaje de error cuando se intenta crear una ecuación con dos variables distintas.

Paso:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga dos variables distintas.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje de error indicando que la sintaxis no es correcta.

### Caso de prueba 35

Título: Crear una ecuación con una variable de la forma xx

Descripción: Verificar que se muestra un mensaje de error cuando se intenta crear una ecuación con una variable de la forma xx.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una que contenga la variable de la forma xx.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que la sintaxis no es correcta.

### Caso de prueba 36

Título: Crear una ecuación en que la variable no sea una x

Descripción: Verificar que se puede crear una ecuación con una variable distinta a la x.

Pasos:

1. En la vista para dar de alta ecuación, crear una ecuación con la variable distinta a la letra x.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje indicando que la ecuación se creó correctamente.
4. Verificar que en el listado se encuentra la ecuación creada anteriormente.

### Caso de prueba 37

Título: Crear una ecuación con la variable del lado izquierdo de la igualdad.

Descripción: Verificar que es posible crear una ecuación con la variable del lado

izquierdo de la igualdad.

Pasos:

1. En la vista para dar de alta una ecuación, ingresar una ecuación que contenga la variable del lado izquierdo de la ecuación.
2. Cliquear en el botón para dar de alta la ecuación.
3. Se debe mostrar un mensaje que indique que se dió de alta la ecuación correctamente.
4. Verificar que la ecuación antes creada se encuentra en el listado de ecuaciones.

### Caso de prueba 37'

Título: Deshacer operaciones de botonera

Descripción: Verificar que las operaciones agregadas con botonera se puedan deshacer.

Pasos:

1. En la vista para crear ecuaciones, agregar una operación mediante la botonera.
2. Apretar el botón para deshacer.
3. Verificar que la operación fue borrada.
4. Agregar una operación por cada botón de la botonera.
5. Cliquear el botón de deshacer, y verificar que se van deshaciendo de la última operación agregada hasta la primera.
6. Crear una operación e ingresarle los operandos.
7. Cliquer en el botón de deshacer
8. Ver que se borra la operación, incluído sus operandos.

## Caso de uso Listado de ecuaciones locales

### Caso de prueba 38

Título: Listado por defecto

Descripción: Verificar que el listado muestra por defecto todas iniciales cargadas por el programa..

Pasos:

1. En la vista inicial en la que se encuentra el listado comprobar que están todas las ecuaciones que se crean al iniciar por primera vez la aplicación.
2. Verificar que nos se muestran las opciones para eliminar las ecuaciones.

### Caso de prueba 39

Título: Listado con todas las ecuaciones

Descripción: Verificar que seleccionar el checkbox para que se muestren todas las ecuaciones, se muestren también las ecuaciones dada de baja.

Pasos:

1. En la vista en la que se encuentra el listado de ecuaciones, seleccionar el checkbox para que se muestren todas las ecuaciones.
2. Cliquear en el botón de eliminar en una ecuación que contenga una resolución logueada.
3. Verificar que la ecuación recién dada de baja se muestra en el listado.

### Caso de prueba 40

Título: No mostrar botón de eliminar ecuación, en ecuaciones iniciales.

Descripción: Verificar que las ecuaciones iniciales no se pueden eliminar mediante la

aplicación.

Pasos:

1. Borrar la base de datos
2. Iniciar la aplicación para que se carguen las ecuaciones iniciales.
3. Verificar que las ecuaciones iniciales no contienen el botón para eliminar la ecuación.
4. Verificar que las ecuaciones contienen el botón para resolver la ecuación.

### Caso de prueba 41

Título: Ecuaciones sin logueo de resolución

Descripción: Verificar que las ecuaciones sin logueo de resolución no muestran el botón para ver la resolución.

Pasos:

1. Dar de alta una ecuación.
2. Ver que en el listado se encuentra la ecuación antes creada y no muestra el botón para ver la resolución de la ecuación.
3. Ver que se muestra el botón para resolver la ecuación.
4. Ver que se muestra el botón para eliminar la ecuación.

### Caso de prueba 42

Título: Ecuaciones con resolución logueada

Descripción: Verificar que las ecuaciones con resolución logueada, se pueden volver a resolver, eliminar y ver su resolución.

Pasos:

1. Resolver una ecuación que se encuentre en el listado.
2. Verificar que la ecuación resuelta muestra los botones para volver a resolver, eliminar y ver su resolución.

### Caso de prueba 43

Título: Ecuación sin trabajar

Descripción: Verificar que la ecuación sin trabajar muestra los botones para resolverla y eliminar y que se muestra con el fondo de color azul.

Pasos:

1. Crear una ecuación
2. Ver que ésta se encuentra en el listado y contiene los botones para resolverla y eliminarla.
3. Ver que la ecuación tiene el fondo de color azul.

### Caso de prueba 44

Título: Ecuación trabajada

Descripción: Verificar que las ecuaciones trabajadas pero no terminadas se muestran en color naranja en el listado y que se muestran los botones para resolverla, eliminarla y ver su resolución.

Pasos:

1. Ingresar a la vista para resolver una ecuación sin trabajar.
2. Ingresar el resultado de un sub string de la ecuación, pero que no sea solo la variable.
3. Cliquear en el botón para ir para atrás
4. Ver que la ecuación comenzada a resolver se muestra con el fondo de color naranja.
5. Ver que la ecuación contiene los botones para resolverla, eliminarla y ver su

resolución.

### Caso de prueba 45

Título: Ecuación terminada

Descripción: Verificar que las ecuaciones terminadas se muestran con el fondo de color verde y que muestran los botones para resolver, eliminar y ver su resolución.

Pasos:

1. Resolver una ecuación del listado que estaba sin trabajar.
2. Ver que en el listado se muestra la ecuación resulta con el fondo de color verde.
3. Ver que se muestran los botones para resolverla, eliminar y ver su resolución.

### Caso de prueba 46

Título: Ecuaciones terminadas y vueltas a inicializar

Descripción: Verificar que una ecuación que fue resuelta, pero se volvió a inicializar su resolución se muestra en color azul en el listado, y contiene los botones para resolverla, eliminar y ver su resolución.

Pasos:

1. Cliquear en el botón de resolver en una ecuación ya resuelta. Si no hay ecuaciones resueltas entonces resolver una y volver al listado de las ecuaciones..
2. Cliquer en el botón para resolver la ecuación, y cliquear en el botón ok cuando se muestra un mensaje indicando si se quiere volver a resolver.
3. Cliquear en el botón de volver sin haber comenzado a resolver la ecuación.
4. Comprobar que en el listado se muestra la ecuación antes mencionada en color azul y contiene los botones para eliminar, resolver y ver su resolución.

### Caso de prueba 47

Título: Ecuación dada de baja

Descripción: Verificar que una ecuación dada de baja se muestra en color gris y sólo contiene el botón par ver su resolución.

Pasos:

1. En el listado para ver las ecuaciones, seleccionar el checkbox para que se muestren todas las ecuaciones.
2. Dar de baja una ecuación trabajada o terminada. Si no hay ninguna de estas resolver una.
3. Cliquear en el botón para eliminar la ecuación.
4. Ver que la ecuación se encuentra en el listado con el color de fondo gris.
5. Ver que la ecuación no contiene los botones para resolverla ni para eliminarla.
6. Ver que contiene el botón para ver la resolución de la ecuación.

### Caso de prueba 79:

Título: Listado de ecuaciones descargadas

Descripción: Verificar que en el listado de ecuaciones descargadas sólo se muestran las ecuaciones que fueron descargadas.

Pasos:

1. Seleccionar en el combo de los listados, el listado de ecuaciones descargadas.
2. Ver que no tiene ninguna ecuación y si tiene alguna eliminarla.
3. Ir a la vista para descargar ecuaciones y descargar 3 ecuaciones.

4. Verificar que en el listado de ecuaciones descargadas se encuentran 3 ecuaciones nuevas.

### Caso de prueba 80:

Título: Listado de ecuaciones eliminadas.

Descripción: Verificar que en el listado de ecuaciones eliminadas sólo se muestran las ecuaciones que fueron eliminadas.

Pasos:

1. Entrar al listado de las ecuaciones eliminadas.
2. Ver si hay ecuaciones que estén en color gris.
3. Si en el paso anterior no se encontró ninguna ecuación en el listado, entonces ir a el listado de todas las ecuaciones y eliminar una que haya sido previamente trabajada.
4. Volver a entrar al listado de las ecuaciones eliminadas.
5. Verificar que la ecuación recién eliminada se muestra en el listado.

### Caso de prueba 81:

Título: Listado de ecuaciones dadas de alta.

Descripción: Verificar que en el listado de las ecuaciones dadas de alta, solo se muestran las ecuaciones dada de alta

Pasos:

1. Entrar al listado de ecuaciones creadas
2. Ver si hay ecuaciones en el listado.
3. Si hay ecuaciones comprobar que las ecuaciones existentes todas tengan el botón para eliminar.
4. Dar de alta una ecuación
5. Verificar que en el listado se encuentra la ecuación recién creada.

## Caso de Uso Eliminar ecuación local

### Caso de prueba 48

Título: Eliminar ecuación sin trabajar

Descripción: Verificar que cuando se elimina una ecuación que no fue trabajada se elimina del sistema.

Pasos:

1. Si no hay una ecuación sin trabajar en el listado de las ecuaciones, entonces crear una.
2. Cliquer en el botón de eliminar en la ecuación que no fue trabajada.
3. Se debe mostrar un cartel preguntando si se desea eliminar la ecuación.
4. Cliquear en el botón de ok
5. Se debe mostrar un cartel indicando que la ecuación fue eliminada correctamente.
6. Seleccionar el checkbox para mostrar todas las ecuaciones.
7. Verificar que la ecuación no se encuentra en el listado.

### Caso de prueba 49

Título: Eliminar ecuación ya trabajada.

Descripción: Verificar que cuando una ecuación ya trabajada se da de baja, la ecuación sigue existiendo en el sistema pero se muestra en giris en el listado.

Pasos:

1. Si no hay una ecuación ya trabajada en el listado de ecuaciones, crear una.
2. Cliquear en el botón para eliminar la ecuación, en una ecuación ya trabajada.
3. Se debe mostrar un cartel consultando si está seguro de eliminar la ecuación.
4. Cliquear en el botón de ok.
5. Se debe mostrar un mensaje indicando que la ecuación fue eliminada correctamente.
6. En el listado de las ecuaciones, seleccionar el checkbox para mostrar todas.
7. Ver que la ecuación recién eliminada se encuentra en color gris.

### Caso de prueba 50

Título: Eliminar ecuación ya trabajada y vuelta a inicializar.

Descripción: Verificar que una ecuación ya trabajada y vuelta a inicializar, cuando se elimina sigue estando en el sistema coloreada en gris.

Pasos:

1. Si no hay una ecuación ya trabajada y vuelta a inicializar. Volver a inicializar una ecuación ya trabajada.
2. Cliquear en el botón para eliminar la ecuación.
3. Se debe mostrar un cartel indicando si está seguro de eliminar la ecuación.
4. Cliquear en el botón de ok.
5. Se debe mostrar un cartel que la ecuación fue eliminada correctamente.
6. Seleccionar el checkbox para indicar que se muestren todas las ecuacione.
7. Ver que la ecuación recién eliminada se encuentra en el listado de color gris.

## Caso de Uso Resolución de ecuación

### Caso de prueba 51

Título:Resultado correcto de una variable

Descripción: Verificar que cuando se pone el resultado correcto de una ecuación con una solución se devuelve la respuesta de que fue correcto y queda logueada la resolución correctamente.

Pasos:

1. Cliquear en el botón para resolver una ecuación que todavía no fue trabajada. Si no existe, crear una.
2. Seleccionar la variable.
3. Ver que se escribe debajo la variable y se deja un espacio para que se ingrese la solución.
4. Ingresar el valor de la solución
5. Ver que se muestra un tic indicando que el resultado es correcto
6. Ver que se muestra el conjunto solución con el resultado previamente ingresado.
7. Regresar al listado de las ecuaciones
8. Cliquear en el botón para ver la resolución de la ecuación antes trabajada.
9. Ver que se muestra la variable y el resultado ingresado.

### Caso de prueba 52

Título: Resultado correcto de una sub expresión

Descripción: Verificar que al ingresar el resultado correcto de una sub expresión se muestra un tic indicando que es correcto.

Pasos:

1. Cliquear en el botón para resolver una ecuación sin trabajar.
2. Seleccionar una sub expresión de la ecuación que contenga la variable.

3. Ver que se muestra la sub expresión seleccionada debajo, y se muestra un lugar para ingresar el resultado.
4. Ingresar el resultado correcto de la sub expresión.
5. Se debe mostrar un tic al lado de la expresión, indicando que el resultado es correcto.
6. Volver al listado de las ecuaciones y entrar el la resolución de la ecuación.
7. Ver que se muestra la sub expresión antes seleccionada con el resultado ingresado.

### Caso de prueba 53

Título: Resultado correcto de una variable con más de una solución

Descripción: Verificar que cuando se ingresa el resultado de una ecuación, cuando tiene más de una, se solicita que se continúe con la solución.

Pasos:

1. Cliquear en el botón para resolver una ecuación que está sin trabajar y que tenga más de una solución.
2. Seleccionar la variable de la ecuación.
3. Se debe mostrar la variable escrita abajo con un espacio para ingresar el resultado.
4. Ingresar una solución de la ecuación.
5. Se debe mostrar un mensaje indicando que hay más soluciones.
6. Se debe mostrar un tic al lado del resultado ingresado.
7. Verificar que el resultado recién ingresado quedó logueado.

### Caso de prueba 54

Título: Ingreso todos los resultados de la ecuación.

Descripción: Verificar que una vez que se ingresan todos los valores válidos de la solución, se muestra el conjunto solución de la ecuación.

Pasos:

1. Cliquear en el botón para resolver una ecuación que está sin trabajar y que tenga más de una solución.
2. Seleccionar la variable de la solución
3. Se debe mostrar la variable escriba abajo con un espacio para ingresar el valor de la solución.
4. Ingresar un valor de la solución
5. Se debe mostrar un cartel indicando que hay más soluciones para la ecuación.
6. Se debe mostrar un tic al lado del resultado recién ingresado
7. Se debe mostrar nuevamente la variable escrita abajo, con un espacio para ingresar el resultado.
8. Ingresar el otro valor de la solución y cliquear en el botón validar.
9. Se debe mostrar un tic al lado del resultado recién ingresado.
10. Se debe mostrar el conjunto solución de la ecuación.

### Caso de prueba 55

Título: Repito resultado correcto

Descripción: Verificar que cuando una ecuación tiene dos soluciones distintas y ya se ingresó un valor de la solución correcto, cuando se vuelve repetir el mismo resultado se muestra un mensaje de error.

Pasos:

1. Cliquear en el botón para resolver una ecuación que está sin trabajar y que tenga más de una solución.
2. Seleccionar la variable de la solución
3. Se debe mostrar la variable escriba abajo con un espacio para ingresar el valor

de la solución.
4. Ingresar un valor de la solución
5. Se debe mostrar un cartel indicando que hay más soluciones para la ecuación
6. Se debe mostrar un tic al lado del resultado recién ingresado
7. Se debe mostrar nuevamente la variable escrita abajo, con un espacio para ingresar el resultado.
8. Ingresar nuevamente el resultado antes colocado
9. Se debe mostrar un cartel indicando que ese resultado ya fue ingresado.

### Caso de prueba 56

Título: Resultado de variable incorrecto

Descripción: Verificar que cuando se ingrese un resultado de la ecuación incorrecto se muestre una cruz indicando que es incorrecto y comprobar que quedó logueado.

Pasos:

1. Cliquear en el botón para resolver una ecuación sin resolver.
2. Seleccionar la variable
3. Ingresar un resultado incorrecto.
4. Se debe mostrar una cruz al lado del resultado recién ingresado.
5. Verificar que el valor antes ingresado y el resultado quedó logueado.

### Caso de prueba 57

Título: Resultado de sub expresión incorrecto

Descripción: Verificar que cuando se ingresa un valor incorrecto de una sub expresión se muestra una cruz indicando que es incorrecto, y lo antes ingresado queda logueado junto con su resultado.

Pasos:

1. Cliquear en el botón para resolver una ecuación, en una ecuación sin trabajar.
2. Seleccionar una sub expresión que contenga la variable e ingresar un resultado incorrecto.
3. Se debe mostrar una cruz roja al lado del resultado indicando que fue incorrecto.
4. Verificar que lo antes ingresado quedó logueado.

### Caso de prueba 58

Título: Ecuación con solución vacía

Descripción: Verificar que cuando una ecuación tiene solución vacía, a los tres intentos de ingresar el resultado se muestra un mensaje preguntando si la ecuación tiene solución

Pasos:

1. Cliquear en el botón para resolver una ecuación que tenga como solución el conjunto vacío.
2. Seleccionar la variable e ingresar un resultado
3. Se debe mostrar una cruz al lado del resultado.
4. Repetir los pasos 2 y 3 dos veces más
5. Se debe mostrar un mensaje indicando consultando si la ecuación tendrá solución real.
6. Verificar que quedaron logueados los intentos antes realizados.

### Caso de prueba 59

Título: Ecuación con raíz doble o triple

Descripción: Verificar que cuando una ecuación tiene solución vacía se muestra en el conjunto solución una vez sóla la variable.

Pasos:

1. Cliquear en el botón para resolver una ecuación que tenga como solución una raíz doble.
2. Seleccionar la variable e ingresar su solución
3. Se debe mostrar un tic al lado del resultado.
4. Se debe mostrar el conjunto solución donde sólo se encuentra una vez la raíz.
5. Verificar que lo ante realizado quedó logueado correctamente.

### Caso de prueba 60

Título: Selección de sub expresión inválida

Descripción: Verificar que al seleccionar una sub expresión inválida de la ecuación se muestra un mensaje de error indicando que la sub expresión no está bien formada.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Seleccionar una sub expresión de la ecuación inválida, por ejemplo que finalice en más y le falte un operando. Probar todos los operadores.
3. Se debe mostrar un cartel indicando que sub expresión no está bien formada.

### Caso de prueba 61

Título: Selección de sub expresión sin variable

Descripción: Verificar que al seleccionar una sub expresión sin variable se muestra un mensaje de error indicando que faltó seleccionar la variable.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Seleccionar una sub expresión que no contenga la variable
3. Se debe mostrar un mensaje indicando que faltó seleccionar la variable.

### Caso de prueba 62

Título: No ingresar respuesta al resolver la ecuación

Descripción: Verificar que al cliquear el botón para validar sin haber ingresado un valor de respuesta se muestra un mensaje indicando que faltó ingresar el resultado.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Seleccionar una sub expresión de la ecuación que contenga la variable.
3. Cliquear en el botón para validar el resultado sin haber ingresado un valor de respuesta.
4. Se debe mostrar un mensaje indicando que faltó ingresar la respuesta.

### Caso de prueba 63

Título: Ingresar letras en el resultado

Descripción: Verificar que no se permite ingresar letras como resultado de la expresión seleccionada.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Seleccionar una sub expresión de la ecuación
3. Ingresar letras en el resultado
4. Ver que no se permiten ingresar.

### Caso de prueba 64

Título: Salir de la resolución

Descripción: Verificar que se puede salir de la resolución sin haber terminado de resolverla.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Comenzar a resolverla
3. Cliquear en el botón de volver
4. Verificar que se volvió al listado de las ecuaciones.

### Caso de prueba 65

Título: Continuar resolución

Descripción: Verificar que una ecuación que fue comenzada a resolver pero no fue finalizada se puede continuar resolviendo.

Pasos:

1. Cliquear en el botón para resolver una ecuación que está de color naranja.
2. Se debe mostrar los pasos realizados en el intento anterior.
3. Verificar que se puede seguir resolviendo la ecuación.

### Caso de prueba 66

Título: Volver a comenzar la resolución

Descripción: Verificar que una ecuación comenzada a resolver se puede volver a comenzar.

Pasos:

1. Cliquear en el botón para resolver una ecuación en estado trabajada.
2. Ver que se muestran todos los pasos realizados anteriormente
3. Cliquear en el botón para volver a comenzar la resolución.
4. Ver que se limpia la vista y se sacan los pasos realizados anteriormente.

### Caso de prueba 66’

Título: Ingresar resultado con botonera.

Descripción: Verificar que se pueden ingresar las respuestas utilizando las operaciones de la botonera.

Pasos:

1. Cliquear en el botón para resolver una ecuación.
2. Seleccionar una sub expresión.
3. Ingresar el resultado correcto utilizando operadores.
4. Se debe mostrar un tic al lado del resultado.

### Caso de prueba 66’’

Título: Ecuación con decimales en los valores de la solución

Descripción: Verificar que las ecuaciones que tienen valores con decimales en la solución como por ejemplo raiz de 2, raiz de 3, etc, se muestran en el conjunto solución dibujadas como en los listados.

Pasos:

1. Cliquear en el botón para resolver una ecuación que tiene como solución raiz cuadrada o raíz cúbica.
2. Resolver la ecuación
3. Verificar que en el conjunto solución se encuentra la raiz cuadrada o cúbica dibujada como se muestra en los listados.

### Caso de prueba 82

Título: Ecuación sin solución luego de 5 intentos

Descripción: Verificar que una ecuación sin solución luego de intentar ingresar el resultado de la variable 5 veces, se muestra un mensaje indicando que la ecuación no tiene solución real.

Pasos:

1. Cliquear para resolver una ecuación sin solución. Si no existe, crear una ecuación que no tenga solución.
2. Seleccionar la variable e ingresar un resultado
3. Ver que se muestra una cruz indicando que es incorrecto
4. Repetir los dos pasos anteriores dos veces más.
5. Se debe mostrar un mensaje consultado si la ecuación tiene resultado.
6. Ingresar dos veces más el resultado incorrecto.
7. Se debe mostrar un mensaje indicando que la ecuación no tiene solución real.

### Caso de prueba 83

Título: Ecuación sin solución, se indica que no tiene solución

Descripción: Verificar que cuando a una ecuación que no tiene solución, cliquean al botón para indicar que no tiene solución, Se muestra que la respuesta fue correcta y el conjunto solución vacío.

Pasos:

1. Entrar a resolver una ecuación que no tiene solución real.
2. Seleccionar la variable
3. Cliquear en el botón de solución vacía.
4. Se debe mostrar un tic al lado de su respuesta y se debe escribir S={}

### Caso de prueba 84

Título: En ecuación con solución, se indica que la ecuación no tiene solución.

Descripción: Verificar que si se cliquea en el botón para indicar que no tiene solución en una ecuación que sí la tiene se muestra una cruz al lado de su resultado.

Pasos:

1. Cliquear en el botón para resolver una ecuación que tenga solución.
2. Seleccionar la variable y cliquear en el botón para indicar que no tiene solución
3. Se debe mostrar una cruz al lado de la respuesta.

### Caso de prueba 85

Título: Ecuación con dos raíces y bifurcación en la variable

Descripción: Verificar que cuando ingresa el resultado correcto a la variable, se muestra un mensaje indicando que hay más soluciones y se vuelve a ingresar la variable para que ingresen otro resultado.

Pasos:

1. Entrar a resolver una ecuación que tenga dos raíces y se bifurque en la variable.
2. Resolver la ecuación hasta seleccionar solo la variable e ingresar un paso correcto.
3. Se debe mostrar un mensaje indicando que hay más soluciones y escribirse abajo nuevamente la variable para que ingresen un nuevo resultado
4. Se ingresa el otro resultado
5. Se debe mostrar el conjunto solución compuesto por las dos raíces antes ingresadas.

### Caso de prueba 86

Título: Ecuación con dos raíces y bifurcación en un paso que no es solo la variable

Descripción: Verificar que luego de ingresar el resultado correcto de una variable se muestra el paso en que tiene más de un resultado que valida la ecuación.

Pasos:

1. Entrar a resolver una ecuación que se bifurca en un paso que no es solo la variable.
2. Resolver la ecuación hasta ingresar el primer resultado de la variable.
3. Ver que se muestra nuevamente el paso que tiene más de un resultado para que ingresen otro.

### Caso de prueba 87

Título: Ecuación con dos raíces, se repite paso bifurcado en sub expresión.

Descripción: Verificar que cuando se vuelve a repetir el paso para que se ingrese otro resultado, si se vuelve a poner el mismo valor que se ingresó antes, se muestre un mensaje indicando que tiene que ingresar otro resultado.

Pasos:

1. Entrar a resolver una ecuación que tiene dos raíces y se bifurca en un paso antes que la variable.
2. Resolver la ecuación hasta que te vuelve a solicitar el resultado para un paso bifurcado.
3. Ingresar el mismo valor que se ingresó previamente.
4. Verificar que se muestra un mensaje indicando que repitió el resultado.

### Caso de prueba 88

Título: Ecuación con dos raíces, ingresa mal resultado de paso bifurcado.

Descripción: Verificar que cuando se ingresa mal el resultado del paso bifurcado se muestra una cruz al lado del paso y se vuelve a solicitar que ingrese otro resultado.

Pasos:

1. Ingresar a resolver una ecuación que tiene dos raíces y se bifurca en un paso anterior a la variable.
2. Resolver la ecuación hasta que se vuelve a solicitar el resultado para el paso bifurcado.
3. Ingresar un resultado incorrecto en el paso bifurcado.
4. Se debe mostrar una cruz al lado de la respuesta y volver a solicitar el paso bifurcado.

## Caso de uso Ver resolución de la ecuación

### Caso de prueba 67

Título: Ver resolución de una ecuación con una sola resolución

Descripción: Verificar que se muestra correctamente la resolución de una ecuación.

Pasos:

1. Cliquear en el botón para ver la resolución de una ecuación que tiene sólo una resolución.
2. Ver que se muestra la resolución tal y como fue resuelta previamente.

### Caso de prueba 68

Título: Ver resolución de una ecuación con varias resoluciones

Descripción: Verificar que cuando una ecuación tiene más de una resolución logueada

se muestran, las dos resoluciones debidamente identificadas.

Pasos:

1. Cliquear en el botón para ver la resolución de una ecuación que tenga más de una resolución logueada.
2. Ver que se muestran las resoluciones separadas que representan correctamente cada una de las resoluciones.

## Caso de Uso Descargar ecuaciones

### Caso de prueba 69

Título: Descargar ecuaciones

Descripción: Verificar que se descarga la cantidad adecuada de ecuaciones en el rango de fechas especificado y quedan guardadas localmente.

Pasos:

1. En la vista de subir ecuaciones, subir 5 ecuaciones al repositorio central
2. Entrar a la vista para descargar ecuaciones
3. Seleccionar para descargar 5 ecuaciones, con rango desde ayer.
4. Cliquear en el botón para descargar ecuaciones.
5. Se debe mostrar un cartel indicando que se descargaron las ecuaciones correctamente.
6. Verificar en el listado que se agregaron 5 nuevas ecuaciones.

### Caso de prueba 70

Título: No hay conexión con el servidor

Descripción: Verificar que se muestre un cartel de error cuando se intentan descargar ecuaciones y el servidor no está disponible.

Pasos:

1. Apagar el servidor
2. En la vista de descargar ecuaciones, cliquear el botón para descargar ecuaciones.
3. Se debe mostrar un cartel indicando que no hay conexión con el servidor.

### Caso de prueba 71

Título: No hay ecuaciones en el servidor

Descripción: Verificar que el sistema no da error cuando no hay ecuaciones en el servidor.

Pasos:

1. Borrar las ecuaciones del servidor
2. En la vista para descargar ecuaciones, cliquear en el botón para descargar ecuaciones.
3. Se debe mostrar un cartel indicando que no hay ecuaciones disponibles para descargar.

### Caso de prueba 72

Título: La cantidad solicitada a descargar es mayor a la disponible

Descripción: Verificar que cuando se solicita descargar una cantidad mayor a las ecuaciones que se encuentran disponibles se descarga la cantidad disponible.

Pasos:

1. Subir tres ecuaciones al repositorio central
2. Seleccionar para descargar 5 ecuaciones con rango desde ayer.

3. Cliquear en el botón para descargar ecuaciones.
4. Se debe mostrar un cartel indicando que la descarga fue exitosa.
5. Verificar en el listado de ecuaciones, que hay tres ecuaciones nuevas.

### Caso de prueba 73

Título: Volver para atrás

Descripción: Verificar que se puede volver a la pantalla anterior sin haber descargado ecuaciones.

Pasos:

1. Entrar a la vista de descargar ecuaciones.
2. Cliquear en el botón para volver.
3. Verificar que se volvió al listado de las ecuaciones.

## Caso de uso Subir ecuaciones

### Caso de prueba 74

Título: Subir ecuación

Descripción: Verificar que se puede subir una ecuación al repositorio central.

Pasos:

1. Entrar a la vista para subir ecuaciones.
2. Seleccionar una ecuación a subir.
3. Cliquear en el botón para subir las ecuaciones.
4. Se debe mostrar un mensaje indicando que la ecuación se subió exitosamente.
5. Verificar que la base de datos del repositorio central contiene la ecuación recién subida.
6. Volver a la vista para subir la ecuación y verificar que esa ecuación ya no se muestra en el listado de las ecuaciones a subir.

### Caso de prueba 75

Título: Subir más de una ecuación

Descripción: Verificar que se puede subir más de una ecuación al repositorio central.

Pasos:

1. Entrar a la vista para subir ecuaciones.
2. Seleccionar más de una ecuación a subir.
3. Cliquear en el botón para subir las ecuaciones.
4. Se debe mostrar un cartel que las ecuaciones fueron subidas exitosamente.
5. Verificar que las ecuaciones antes seleccionadas se encuentran en la base de datos del repositorio central.
6. Volver a entrar a la vista para subir ecuaciones.
7. Verificar que las ecuaciones antes subidas no se muestran más en el listado de ecuaciones a subir.

### Caso de prueba 76

Título: Seleccionar el botón para subir ecuaciones sin haber seleccionado ninguna

Descripción: Verificar que se muestra un mensaje de error cuando no se selecciona ninguna ecuación para subir y se cliquea en el botón para subir las ecuaciones.

Pasos:

1. Entrar a la vista para subir las ecuaciones.
2. Cliquear en el botón para subir las ecuaciones sin haber seleccionado ninguna.
3. Se debe mostrar un cartel indicando que se debe seleccionar alguna ecuacion.

### Caso de prueba 77

Título: No hay conexión con el servidor

Descripción: Verificar que cuando se intenta subir ecuaciones y no hay conexión con el servidor, se muestra un mensaje de error indicando que no hay conexión.

Pasos:

1. Apagar el servidor del repositorio central.
2. Entrar a la vista para subir ecuaciones.
3. Seleccionar ecuaciones a subir.
4. Cliquear en el botón para subir las ecuaciones.
5. Se debe mostrar un mensaje indicando que no hay conexión con el repositorio central.

### Caso de prueba 78

Título: Ecuaciones iniciales

Descripción: Verificar que las ecuaciones iniciales no se encuentran en el listado para subir las ecuaciones.

Pasos:

1. Borrar todas las ecuaciones de la base local.
2. Prender el programa.
3. Ver que se cargaron todas las ecuaciones iniciales.
4. Entrar a la vista para subir las ecuaciones.
5. Verificar que el listado de las ecuaciones está vacío.
6. Crear una nueva ecuación.
7. Volver a entrar a la vista para subir las ecuaciones.
8. Ver que la ecuación recién creada se encuentra en el listado de las ecuaciones a subir.
9. Seleccionar la ecuación y subirla
10. Volver a entrar a la vista de subir ecuaciones.
11. Verificar que el listado nuevamente está vacío.

## Requisitos No funcionales

Cómo requerimiento no funcional se solicitó que la aplicación fuera amigable.

### Procedimiento de prueba

Realizar una lista de tareas a realizar y solicitarle a estudiantes del ciclo básico que realicen las mismas.

Comprobar si los estudiantes logran realizar las tareas sin necesidad de ayuda externa o si luego de una breve explicación pueden realizar las acciones exitosamente.

# Modelo de Casos de Uso
# Versión 3.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 12/09/2016 | 1.0 | Inicial. | Camila Rojí - Martín Poli |
| 16/09/2016 | 1.1 | Correcciones flujos alternativos. | Camila Rojí - Martín Poli |
| 23/09/2016 | 1.2 | Modificación de los casos de uso. | Camila Rojí - Martín Poli |
| 29/10/2016 | 2.0 | Se modifica el uso de archivos locales por base local. | Martín Poli |
| 12/12/2016 | 2.1 | Se actualiza el caso de uso listado de ecuación para que contemple filtro por estado. | Martin Poli |
| 30/06/2017 | 3.0 | Se actualizan los casos de uso para que reflejen los cambios solicitados por el cliente. | Camila Rojí - Martín Poli |

# ÍNDICE

# Actores

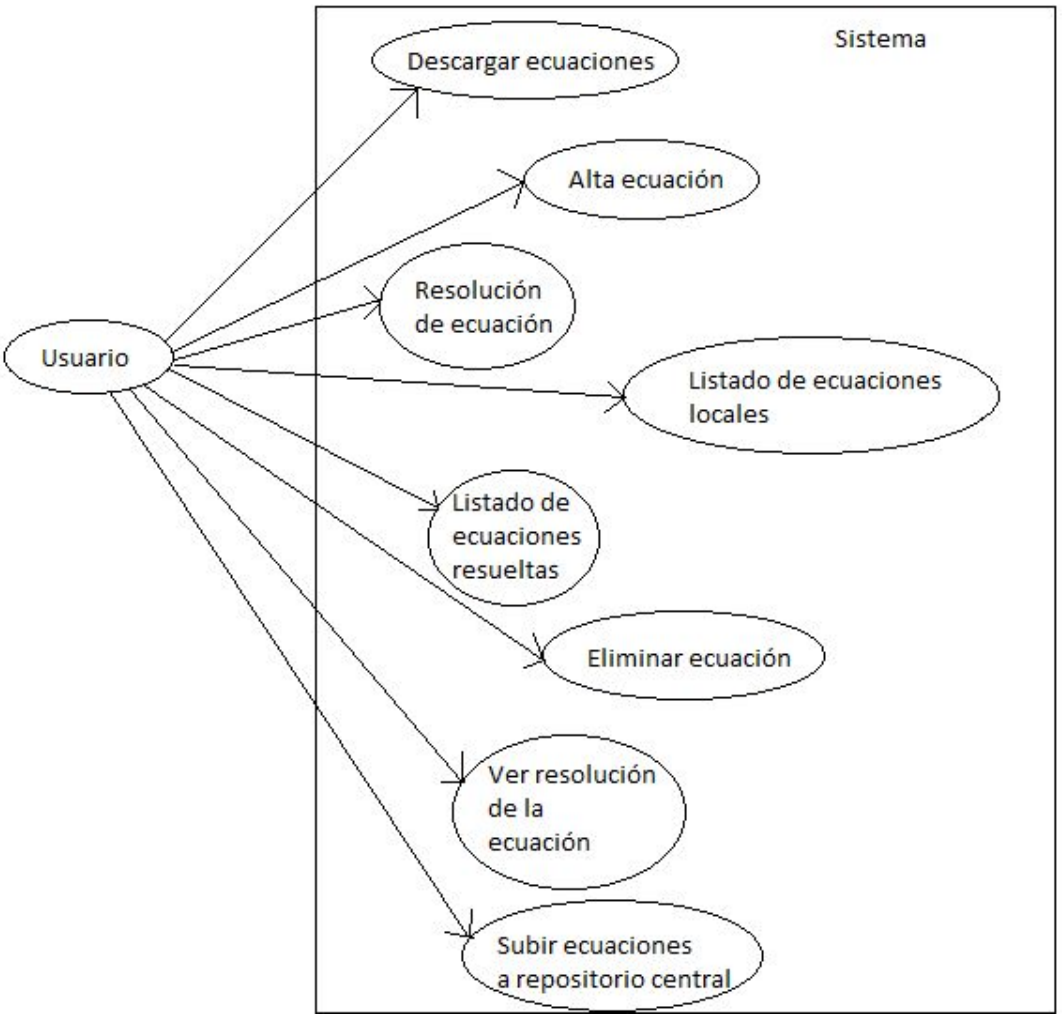
## Estudiante

Representa al estudiante de ciclo básico con acceso a una laptop Ceibal Magallanes.

# Casos de Uso

## Diagrama de Casos De Uso

Aquí se presentan los Diagramas de Casos de Uso del Sistema, para mostrar la interacción entre los Actores y los Casos de Uso del Sistema.




## Alta ecuación

### Descripción

El caso de uso comienza cuando el estudiante desea dar de alta una ecuación. El sistema muestra una nueva pantalla que contiene botones con los operadores raíz, cociente, potencia y paréntesis. El usuario va seleccionando los operadores y

tipeando los números, los operadores básicos (+, -,*) y la variable para crear la ecuación.

En cuanto el estudiante desee guardar la ecuación, el sistema valida que la ecuación ingresada sea válida y la da de alta en el sistema.

### Pre-condiciones

No hay.

### Flujo de eventos principal

1. El estudiante solicita dar de alta una ecuación.
2. El sistema solicita que el usuario ingrese los miembros de ambos lados de la igualdad.
3. El estudiante ingresa los términos que forman la ecuación.
4. El sistema verifica que la ecuación ingresada sea de hasta tercer grado, que contenga una sola ocurrencia de la variable y esté sintácticamente correcta.
5. El sistema guarda localmente la ecuación ingresada.
6. El sistema muestra el listado de ecuaciones creadas.
7. Fin de caso de uso.

### Flujos de eventos alternativos

#### 3.A Estudiante cancela

1. El estudiante cancela el alta de ecuación.
2. Fin de caso de uso.

#### 3.B Deshacer acción botonera

3. El estudiante deshace la última acción ingresada por botonera.
4. Fin de caso de uso.

#### 4.A Ecuación ingresada no válida

1. El sistema verifica que la ecuación ingresada no cumple con la validación del punto 4 del flujo principal.
2. El sistema muestra un mensaje indicando el error ocurrido.
3. Se vuelve al punto 3 del flujo principal

### Post-condiciones

La ecuación ingresada queda guardada en una base local con la lista de las ecuaciones creadas por el usuario.

## Listado de ecuaciones locales

### Descripción

El caso de uso comienza cuando el estudiante desea ver el listado de ecuaciones almacenadas localmente. Pudiendo elegir listar las ecuaciones por defecto, creadas,

descargadas, eliminadas o todas. El sistema muestra las ecuaciones existentes y las opciones de resolver, ver historial y eliminar en caso que corresponda.

### Pre-condiciones

No hay.

### Flujo de eventos principal

1. El estudiante solicita ver el listado de un tipo de ecuaciones almacenadas localmente.
2. El sistema lista las ecuaciones almacenadas localmente del tipo seleccionado ordenadas por el estado y fecha de las mismas.
3. Fin del caso de uso.

### Flujos de eventos alternativos

#### 2.A Estudiante selecciona una ecuación para eliminar

1. El estudiante indica que desea eliminar una determinada ecuación.
2. Incluye el caso de uso “Eliminar ecuación local”.

#### 2.B Estudiante selecciona una ecuación para resolver

1. El estudiante indica que desea resolver una determinada ecuación.
2. Incluye el caso de uso “Resolución de ecuación”.

#### 2.C No existen ecuaciones almacenadas localmente

1. El sistema muestra la pantalla vacía
2. Fin del caso de uso

## Eliminar ecuación local

### Descripción

El caso de uso comienza cuando el usuario indica que desea eliminar una determinada ecuación. El sistema borra la ecuación seleccionada sin modificar el repositorio central ni el logueo de la resolución de las ecuaciones.

### Pre-condiciones

Existe una ecuación almacenada localmente en el sistema que no es del tipo por defecto.

### Flujo de eventos principal

1. El estudiante solicita eliminar una ecuación almacenada localmente.

2. El sistema elimina la ecuación almacenada localmente.
3. Fin de caso de uso.

### Flujos de eventos alternativos

#### 1.A Estudiante cancela

1. El estudiante cancela el alta de ecuación.
2. Fin de caso de uso.

### Post-Condiciones

La ecuación eliminada se da de baja en la base local pero no se modifica el logueo de la resolución de la ecuación en el caso de que exista ni tampoco se modifica el repositorio central. Pudiendo el estudiante consultar el historial de resolución en cualquier momento.

## Resolución de ecuación

### Descripción

El caso de uso comienza cuando el usuario indica que desea resolver una determinada ecuación. La ecuación se muestra en una nueva pantalla de resolución donde el estudiante puede ir seleccionando conjuntos de términos de la ecuación que va a ir resolviendo. Cuando un estudiante realiza la selección, el sistema escribe los términos seleccionados esperando que el estudiante ingrese el resultado de dicha selección. Una vez que el usuario indica el resultado de la sub-ecuación seleccionada, se loguea en la base local el dato ingresado por el estudiante asociada a la ecuación, y se indica si el resultado ingresado es correcto o no.

El proceso continúa hasta que el estudiante indique todas las soluciones de la ecuación.

### Pre-condiciones

Existe una ecuación almacenada localmente en el sistema.

### Flujo de eventos principal

1. El estudiante ingresa a la pantalla de resolución de una determinada ecuación.
2. El sistema muestra la ecuación en pantalla.
3. El estudiante selecciona con el mouse la variable de la ecuación a resolver.
4. El sistema muestra en una nueva línea la variable seleccionada y espera el ingreso del resultado de la variable por el estudiante.
5. El estudiante ingresa el resultado de la variable.
6. El sistema valida que el resultado ingresado sea correcto y loguea el dato ingresado por el estudiante en la base local.
7. El sistema indica que el resultado es correcto
8. Fin del caso de uso.

## Flujos de eventos alternativos

### 3.A El estudiante selecciona un conjunto de términos

1. El estudiante selecciona un conjunto de términos de la ecuación.
2. El sistema muestra en una nueva línea los términos seleccionados por el usuario.
3. El usuario ingresa el resultado de la sub-ecuación seleccionada.
4. El sistema valida que el resultado sea correcto
5. El sistema le indica al usuario que el resultado es correcto
6. Se vuelve al punto 3 del flujo principal

### 3.B El estudiante selecciona un conjunto de términos sintacticamente invalido

1. El estudiante selecciona un conjunto de términos sintácticamente incorrecto.
2. El sistema le indica al usuario que seleccionó un conjunto de términos sintácticamente incorrecto.
3. Se vuelve al punto 3 del flujo principal.

### 5.A El estudiante selecciona solución vacía en una ecuación sin solución

1. El estudiante selecciona solución vacía.
2. El sistema loguea el resultado ingresado en la base local.
3. El sistema le indica al usuario que el resultado es correcto.
4. Fin del caso de uso.

### 5.B El estudiante selecciona solución vacía en una ecuación con solución

1. El estudiante selecciona solución vacía.
2. El sistema loguea el resultado ingresado en la base local.
3. El sistema le indica al usuario que el resultado ingresado es incorrecto.
4. Se vuelve al paso 5 del flujo principal.

### 6.A El resultado ingresado es incorrecto

1. El sistema valida que el resultado ingresado no es correcto.
2. El sistema loguea el resultado ingresado en la base local.
3. El sistema le indica al usuario que el resultado ingresado es incorrecto.
4. Se vuelve al paso 5 del flujo principal.

### 6.A.1.A La ecuación no tiene solución

1. El sistema loguea el resultado ingresado en la base local.
2. El sistema le indica al usuario que el resultado ingresado es incorrecto.
3. El estudiante ingresa un nuevo valor.
4. El sistema detecta que el estudiante intentó 3 veces ingresar el resultado.
5. El sistema muestra un mensaje indicando que la ecuación no tiene solución.

6. Fin del caso de uso.

**7.A La ecuación tiene más de una solución diferente**

1. El sistema sistema indica que el resultado es correcto y le vuelve a mostrar el paso donde se produjo la bifurcación en la resolución para que ingrese la nueva resolución.
2. Se vuelve al paso 5 del flujo principal.

**Estudiante cancela**

1. El estudiante cancela la resolución de la ecuación.
2. El sistema sale de la pantalla de resolución y muestra el listado de ecuaciones.
3. Fin de caso de uso.

**1.A Continuar resolución**

1. El usuario selecciona una ecuación previamente comenzada.
2. El sistema lista todos los pasos realizado previamente por el estudiante.
3. Se va al paso 3 del flujo principal.

**1.A.3.A Volver a comenzar la resolución**

1. El usuario solicita volver a comenzar la resolución de la ecuación
2. Se vuelve al paso 2 del flujo principal.

### Post-condiciones

Se modifica la base local almacenando todos los pasos realizados por el estudiante durante la resolución asociado a la ecuación.

## Listado ecuaciones resueltas

### Descripción

El caso de uso comienza cuando el estudiante solicita ver el listado de ecuaciones que ya fueron resueltas y/o comenzadas a trabajar y el sistema muestra en pantalla el listado de las mismas.

### Pre-condiciones

No hay.

### Flujo de eventos principal

1. El estudiante solicita ver el listado de ecuaciones resueltas.
2. El sistema lista las ecuaciones resueltas y/o comenzadas a trabajar.
3. Fin del caso de uso.

## Flujos de eventos alternativos

### 2.A Estudiante selecciona una ecuación para ver la resolución

1. Incluye el caso de uso “Ver resolución de la ecuación”.

### 2.B El sistema no tiene ecuaciones resueltas y/o comenzadas a trabajar

1. El sistema muestra la pantalla vacía.
2. Fin del caso de uso

### Estudiante cancela

1. El estudiante cancela la vista del listado de ecuaciones resueltas
2. El sistema vuelve a la pantalla anterior.
3. Fin de caso de uso.

# Ver resolución de la ecuación

## Descripción

El caso de uso comienza cuando el usuario desea ver la resolución de una ecuación determinada. El sistema muestra en una nueva pantalla todos los pasos realizados durante el proceso de resolución.

## Pre-condiciones

La ecuación fue resuelta o comenzada a resolver previamente por el estudiante.

## Flujo de eventos principal

1. El estudiante solicita ver el historial con los pasos realizados para la resolución de la ecuación.
2. El sistema muestra los pasos realizados durante la resolución.
3. Fin de caso de uso.

## Flujos de eventos alternativos

## 2.A Estudiante cancela

1. El estudiante cancela el alta de ecuación.
2. El sistema muestra el listado de las ecuaciones resueltas.
3. Fin del caso de uso.

## Descargar ecuaciones

### Descripción

El caso de uso comienza cuando el usuario desea descargar ecuaciones del repositorio central, para esto el usuario indica la cantidad deseada desde un combo con las cantidades predeterminadas y/o el rango de fecha en que las ecuaciones fueron ingresadas al mismo y el sistema descarga la cantidad indicada de ecuaciones elegidas en forma aleatoria dentro del rango de fecha indicado. Las ecuaciones descargadas quedan guardadas localmente.

### Pre-condiciones

Existen ecuaciones en el repositorio central.

### Flujo de eventos principal

1. El estudiante solicita la descarga de ecuaciones del repositorio central.
2. El sistema le solicita al estudiante la cantidad de ecuaciones a descargar y el rango de fechas.
3. El estudiante selecciona la cantidad deseada y/o el rango de fechas.
4. El sistema descarga las ecuaciones.
5. El sistema almacena localmente las ecuaciones descargadas.
6. El sistema indica que la descarga fue realizada con éxito.
7. Fin de caso de uso.

### Flujos de eventos alternativos

#### 4.A No hay conexión a internet

1. El sistema sistema no se puede conectar con el repositorio central
2. El sistema muestra un mensaje indicando que lo intente más tarde.
3. Fin del caso de uso

#### 4.B La cantidad de ecuaciones a descargar es mayor a la disponible

1. El sistema descarga menos ecuaciones de las solicitadas por el usuario.
2. Se va al punto 5 del flujo principal

#### 4.C No hay ecuaciones para el rango de fecha solicitado

1. El sistema muestra un mensaje indicando que no existen ecuaciones para el rango de fecha solicitado.
2. Se va al punto 3 del flujo principal.

#### Estudiante cancela

1. El estudiante cancela el alta de ecuación.
2. Fin de caso de uso.

#### Post-condiciones

Las ecuaciones descargadas se encuentran almacenadas localmente.

## Subir ecuaciones a repositorio central

### Descripción

El caso de uso comienza cuando el usuario desea subir sus ecuaciones creadas al repositorio central. El sistema sube las ecuaciones que fueron creadas localmente al repositorio y le indica al estudiante que la operación fue realizada con éxito.

### Pre-condiciones

Deben existir ecuaciones que fueron creadas por el estudiante.

### Flujo de eventos principal

1. El usuario indica que desea subir sus ecuaciones creadas al repositorio central.
2. El sistema sube las ecuaciones que fueron creadas por el estudiante y no fueron subidas previamente.
3. El sistema le indica al usuario que la operación fue realizada con éxito.
4. Fin del caso de uso

### Flujos de eventos alternativos

#### 2.A No hay conexión a internet

1. El sistema detecta que no hay conexión a internet
2. El sistema le indica al usuario que lo intente más tarde.
3. Fin del caso de uso

#### 1.A Estudiante cancela

1. El estudiante cancela la operación.
2. Fin de caso de uso.

### Post-condición

Se actualiza el estado de las ecuaciones que fueron subidas al repositorio central, para no volver a subirlas posteriormente.

# Plan de Proyecto
# Versión 1.1

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 22/08/2016 | 1.0 | Inicial | Martìn Poli, Camila Rojí |
| 30/08/2016 | 1.1 | Se modifica la cantidad de meses del proyecto y se justifica la elección del modelo de desarrollo | Martín Poli, Camila Rojí |
| | | | |
| | | | |

# ÍNDICE

# Introducción

Esta sección contiene una visión general del proyecto y el producto a desarrollar, una lista de los entregables del proyecto y la estrategia de evolución del Plan.

## Alcance del Proyecto

A definir

## Entregables del Proyecto

Lista de todos los entregables para el Cliente, las fechas de entrega, lugar de entrega y condiciones de satisfacción.

| Identificación Entregable | Descripción Entregable | Fecha de entrega | Lugar de entrega |
|---|---|---|---|
| Especificación de requisitos | Se describen los requisitos del sistema relevados | Mediados de setiembre | A definir |
| Alcance del sistema | Se describe el alcance del sistema | Fines de setiembre | A definir |
| Pautas para la interfaz de usuario | Contiene los bosquejos de las interfaces (gráficas) de las aplicaciones | Fines de octubre | Presencial |
| Material para capacitación | Contiene la presentación para capacitar a los usuarios. | Fines de mayo | Presencial |

# Organización del Proyecto

Esta sección contiene la especificación del modelo de proceso del Proyecto, descripción de la estructura organizacional del proyecto, identificación de interfaces e interacciones y definición de responsables.

## Modelo de Proceso

Debido la importancia en el proceso de la resolución de ecuaciones se opta por el modelo de desarrollo iterativo incremental para poder tener más retroalimentación del cliente, de forma de poder realizar una aplicación intuitiva y didáctica para los estudiantes.

La retroalimentación del cliente nos permite detectar tempranamente las preferencias de los profesores y nos facilita realizar el cambio, ya que en etapas futuras nos resultaría más costoso.

Dado el entusiasmo del cliente, este model permite involucrarlo más en el desarrollo del sistema al ir viendo el avance del mismo.

El modelo que proponemos se compone de cuatro fases, Inicial, Especificación, Construcción y Transición.

La primera fase tiene una duración de un mes y los objetivos son los siguientes:

- Identificar los requerimientos relevantes para definir el Alcance y la Arquitectura.
- Especificar los requerimientos.
- Priorizar los Requerimientos
- Definir el Alcance del Sistema.
- Identificar riesgos, planificación de mitigación y contingencia de los mismos.
- Realización de los planes Plan de Calidad, Plan de Configuración, Plan de
- Verificación y Validación, Plan de Proyecto.
- Preparar el ambiente de desarrollo.

La segunda fase tiene una duración de 2 meses y los objetivos son los siguientes.

- Definir la Arquitectura del sistema.
- Especificación del diseño del sistema
- Diseño y validación de interfaz
- Comenzar con el desarrollo de la resolución de ecuaciones
- Ajustar la planificación del proyecto

La fase de construcción tiene una duración de 5 meses y los objetivos son los siguientes:

- Obtener la construcción completa del software.
- Documentación técnica.
- Validaciones con el usuario
- Testing

La fase de transición tiene una duración de 2 meses y los objetivos son los siguientes:

- Armar la presentación del proyecto
- Informe de la tesis completo
- Instalación en el ambiente del cliente
- Solución de bugs
- Capacitación a los usuarios

### Interfaces e Interacciones

En esta sección se describen los procedimientos administrativos y de gestión entre el proyecto y: el Cliente, Gestión de configuración, Gestión de calidad y Verificación.

| Actividad | Procedimiento | Involucrados |
|---|---|---|
| Comunicación con el cliente | Se coordinará una reunión junto con los tutores del proyecto. | Tutores, clientes e integrantes del grupo de proyecto. |

## Proceso de Gestión

### Objetivos y prioridades de gestión

La Gestión de proyecto se encarga de planificar, administrar recursos y definir alcance para intentar lograr los objetivos previstos.
El propósito de la Gestión de Proyecto es:

- Realizar un seguimiento del avance del proyecto
- Identificar y seguir los riesgos.
- Planificar y administrar los recursos
- Estimar las dimensiones del proyecto

### Condiciones asumidas, dependencias y restricciones

La realización del proyecto se realizará durante un período de 10 meses.

### Mecanismos para la gestión de proyecto

Se realizarán reuniones semanales con los tutores e integrantes del equipo.

### Recursos

Los integrantes realizarán el análisis, diseño, desarrollo y testeo de la aplicación, además de la gestión del proyecto.

## Proceso técnico

### Procedimientos técnicos, herramientas y tecnologías

- IDE de desarrollo Netbeans.
- Lenguaje principal Java.
- Sistema operativo Linux Ubuntu.
- Motor de base de datos Postgres.
- Repositorio para el codigo Bitbacket.
- SubVersion a definir.
- Se usarán los estándares definidos en el documento “Estándar de implementación”.

### Documentación de software

El formato de documentos será:

- Letra título documento: Arial 18 negrita
- Letra del índice: Arial 10
- Letra títulos de secciones: Arial 13 negrita
- Letra subtítulos de secciones: Arial 11 negrita
- Letra del texto: Arial 10

## Líneas de trabajo

### Descripción

| **Descripción de Línea de trabajo** | **Descripción de actividades correspondientes** |
|---|---|
| Diseño | ● Diseñar casos de uso<br>● Describir la arquitectura<br>● Diseñar base de datos<br>● Diseñar prototipos<br>● Diseño de interfaces gráficas |
| Gestión de calidad | ● Plan de calidad<br>● Revisar las entregas<br>● Evaluar la calidad del producto |
| Gestión de la configuración y control de cambios | ● Planificar la configuración<br>● Definir la línea base del proyecto<br>● Control de cambios |
| Gestión de proyecto | ● Planificar el proyecto<br>● Seguimiento del proyecto<br>● Estimaciones y mediciones<br>● Gestión de riesgos<br>● Ajustar y controlar el desarrollo<br>● Evaluar y ajustar el plan de proyecto<br>● Realizar el informe final del proyecto<br>● Preparar el cierre del proyecto |
| Implantación | ● Planificar la implantación<br>● Elaborar la presentación del sistema para el cliente<br>● Producir la versión del producto a liberar<br>● Puesta en producción<br>● Verificar la versión del producto a liberar |
| Implementación | ● Definir estándares de documentación técnica<br>● Corregir la implementación<br>● Implementar<br>● Verificación unitaria de módulo<br>● Investigar la Herramienta de Desarrollo<br>● Definir el Modelo de Implementación |
| Requisitos | ● Reunion de requisitos<br>● Especificar requisitos<br>● Especificar casos de uso |

| | |
|---|---|
| | ● Priorizar casos de uso<br>● Validación con el cliente<br>● Definir pautas para la interfaz de usuario<br>● Definir alcance del sistema |
| Verificación | ● Planificar la verificación<br>● Especificar los casos de prueba<br>● Generar entorno de prueba<br>● Ejecutar las pruebas<br>● Ejecutar Pruebas No Funcionales |

# Glosario
# Versión 1.3

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 23/09/2016 | 1.0 | Inicial | Martín Poli - Camila Rojí |
| 27/09/2016 | 1.1 | Se agrega Trello al documento | Martín Poli - Camila Rojí |
| 30/09/2016 | 1.2 | Se agrega introducción | Martín Poli - Camila Rojí |
| 14/07/2017 | 1.3 | Se agregan definicones utilizadas en el informe | Camila Rojí |
| 02/08/2017 | 1.4 | Se agregan definicones utilizadas en el informe | Martín Poli |

# Introducción

En este documento se especifican las palabras claves para el entendimiento del proyecto.

# Glosario del informe del proyecto de grado.

### JavaFX

La tecnología JavaFX es un conjunto de productos de Oracle Corporation, que permite la creación de aplicaciones de escritorio con características y estilo visual similar a aplicaciones web, ya que permite embeber hojas de estilo en cascada.

### Mockup

Denotamos por mockup al diseño realizado en papel de las vistas de la aplicación.

### Notación infija

La notación infija  es la notación común de fórmulas aritméticas y lógicas, en la cual se escriben los operadores entre los operandos en que están actuando (ej. 2 + 2). En la notación de infijo, a diferencia de las notaciones de prefijo o posfijo, es necesario rodear entre paréntesis a los grupos de operandos y operadores, para indicar el orden en el cual deben ser realizadas las operaciones. En la ausencia de paréntesis, ciertas reglas de prioridad determinan el orden de las operaciones.

### Notación postfija

Se dice notación postfija cuando cada operador está antes de sus operandos

### SOAP

SOAP, las siglas de Simple Object Access Protocol, es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

### Tompcat

Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Implementa las especificaciones de los servlets y de JavaServer Pages(JSP) de Oracle Corporation.

# Glosario de los documentos del proyecto.

### Interfaz atractiva

Una interfaz es atractiva si es agradable para la vista de los usuarios… (completar con los libros)

### Interfaz intuitiva

Una interfaz es intuitiva si permite con facilidad el ingreso de ecuaciones y la resolución de la mismas.

### Magallanes

Las Magallanes son laptops que otorga el Plan Ceibal a los estudiantes de ciclo básico.

### Métodos aritméticos

Entendemos por resolución de métodos aritméticos aquellos que evidencian como sustancial el uso de conocimientos aritméticos relacionados con las operaciones involucradas en la ecuación.

### Positivos

Las Positivos son laptops que otorga el Plan Ceibal a los estudiantes de ciclo básico, son equipos más modernos y con mejores recursos en comparación a las Magallanes.

### Repositorio central

El repositorio central es el servidor donde se almacenarán las ecuaciones creadas por los estudiantes estudiantes además de las ecuaciones iniciales precargadas en el sistema. El repositorio central es accesible por todas las aplicaciones de los estudiantes que contengan conexión a Internet.

### Trello

Herramienta que contiene tableros y etiquetas permitiendo gestionar el proyecto en varias áreas, como la planificación, registro de errores y plan de casos de prueba.

## Referencias

Notación infija: https://es.wikipedia.org/wiki/Notaci%C3%B3n_de_infijo

# Prueba de performance

# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 30/05/2017 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

## Prueba de performance

La prueba de performance se realizó en un equipo Positivo y se utilizaron cuatro bases con 200, 400, 600 y 800 ecuaciones respectivamente. Las ecuaciones son del tipo "creadas" por el usuario para simular un posible escenario. La prueba se enfocó en medir los tiempos de respuesta para volúmenes grandes de datos y en la siguiente tabla mostramos los tiempos obtenidos:

| Cantidad de ecuaciones creadas | Tiempo de carga del listado (segundos) |
|---|---|
| 200 | 12 |
| 400 | 22 |
| 600 | 39 |
| 800 | 49 |

De los resultados obtenidos podemos concluir que para grandes cantidades de ecuaciones sería apropiado contar con un paginado en el listado de ecuaciones para agilizar la carga. Como alternativa se recomienda al usuario ir eliminando las ecuaciones que no va a volver a resolver para quitarlas del listado de ecuaciones creadas y de esta forma tener una cantidad más acotada de ecuaciones creadas.

# Acta de Reunión de Requisitos 1
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 29/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 12/08/2016

**Participantes:**
Sylvia da Rosa (Tutor)
Federico Gómez (Tutor)
Teresa Pérez (Cliente)
Nora Ravaioli (Cliente)
Camila Rojí(Estudiante)
Martín Poli (Estudiante)

# ÍNDICE

## Introducción

En esta primera reunión con el cliente se relevaron los requisitos generales planteados por el mismo.

## Temas Tratados

1. Requisitos del proyecto.

## Requisitos

### Ecuaciones

1. Dar de alta ecuaciones de hasta tercer grado, lineales, cuadráticas y cúbicas, racionales y con radicales o combinaciones de estas, con una única variable, pudiendo estar en cualquier lado de la igualdad.
2. El alta de las ecuaciones pueden ser realizadas por los estudiantes y por los profesores.
3. El alta de las ecuaciones se realizan con botonera.
4. Permitir a los profesores dar de baja ecuaciones.
5. El sistema debe permitir a los profesores ingresar material didáctico asociados a las ecuaciones
6. Se debe tener un botón de ayuda que contenga material didáctico.
7. Permitir resolver las ecuaciones ingresadas mediante métodos aritméticos y que culmine la resolución cuando encontró todas las soluciones posibles.
8. Si la ecuaciones no tiene solución real luego de x intentos se mostrará un mensaje al usuario indicando que no tiene solución.

### Usuarios

1. El sistema debe permitir el logueo de usuarios por cédula
2. El sistema debe permitir dar de alta, baja y modificar estudiantes.
3. El ABM de los estudiantes lo pueden hacer los profesores.
4. Un estudiantes se puede dar de alta y modificar a sí mismo.
5. El sistema debe permitir a los estudiantes inscribirse sin estar asociados a un profesor.
6. Un profesor se puede dar de alta y modificar a sí mismo.
7. Un profesor puede asignarse estudiantes.

### Otras funcionalidades

1. Un profesor puede agrupar ecuaciones en tareas.
2. El sistema debe permitir a los estudiantes descargar ecuaciones por nombre de tarea.
3. El sistema debe permitir a los estudiantes descargar ecuaciones por una cantidad especificada.
4. El sistema debe permitir descargar ecuaciones aleatorias.
5. El sistema debe guardar el historial de resolución de un alumno para determinada ecuación

6. El sistema debe permitir a los profesores consultar el historial de resolución de sus estudiantes
7. El sistema debe permitir a un profesor borrar un historial de un estudiante asociado a él.
8. Sincronizar los datos del usuario (fórmula e historial de resolución) de forma periódica y cuando el usuario lo indique.
9. El sistema debe permitir descargar la aplicación desktop de los estudiantes desde la aplicación web

## Requisitos no funcionales

1. El sistema debe permitir trabajar offline a los estudiantes
2. Los textos de la aplicación deben estar en español
3. Se debe ofrecer una interfaz atractiva
4. Se debe tener una base inicial con 20 ecuaciones guardadas.
5. La aplicación de escritorio de los estudiantes debe correr en las Laptops Ceibal Magallanes con sistema operativo Ubuntu.

# Acta de Reunión de Requisitos 2
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 30/09/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 21/09/2016

**Participantes:**

| | |
|---|---|
| Sylvia da Rosa | (Tutor) |
| Federico Gómez | (Tutor) |
| Teresa Pérez | (Cliente) |
| Camila Rojí | (Estudiante) |
| Martín Poli | (Estudiante) |

# ÍNDICE

## Introducción

En esta reunión con el cliente se validó el documento de Especificación de requisitos, en el cual el cliente aceptó las sugerencias propuestas por los estudiantes, así como se hayan eliminado los requisitos correspondientes a la administración por parte de los profesores.

## Temas Tratados

1. Requisitos seleccionados para el proyecto.
2. Modificaciones de los requisitos

## Requisitos seleccionados para el proyecto

La descripción de estos requisitos se encuentra en el documento Especificación de requisitos, a continuación listamos los requisitos seleccionados:

### Requisitos funcionales

1. Alta ecuación
2. Eliminar ecuación local
3. Resolución de ecuación
4. Listado de ecuaciones locales
5. Subir ecuaciones al repositorio central
6. Descargar ecuaciones
7. Logueo del trabajo del estudiante
8. Listado de ecuaciones resueltas
9. Ver resolución de la ecuación

### Requisitos no funcionales

1. Ingreso de ecuaciones
2. Trabajo local
3. Interfaz de usuarios
4. Ecuaciones iniciales
5. Idioma
6. Hardware

## Modificaciones de los requisitos

Una de las solicitudes que realizó el cliente, es que las ecuaciones que se cargan por defecto, no se puedan borrar.
Un cambio que planteó el tutor, pero no el cliente, es agregar un filtro por fecha al momento de descargar ecuaciones del repositorio. Este requisito nació debido a la intención del cliente de utilizar una aplicación del estudiante para crear ecuaciones al igual que ellos, y al poder descargar por fecha se puede asegurar que los estudiantes la descarguen.
El cliente también solicitó poder diferenciar las ecuaciones que están terminadas de las que están comenzadas a trabajar sin resolver.
En la reunión también se discutió sobre en qué etapa de la resolución es conveniente indicarle al estudiante que la ecuación no tiene solución, en el caso de que ésta no la tenga. Esto no quedó completamente definido, pero se prefiere que se indique en el paso que no se puede resolver, por más que no esté despejada la variable.

# Acta de Reunión de Requisitos 3
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 12/12/2016 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 23/12/2016

**Participantes:**

| | |
|---|---|
| Sylvia da Rosa | (Tutor) |
| Federico Gómez | (Tutor) |
| Nora Ravaioli | (Cliente) |
| Teresa Pérez | (Cliente) |
| Camila Rojí | (Estudiante) |
| Martín Poli | (Estudiante) |

# ÍNDICE

## Introducción

En esta reunión con el cliente se validó el diseños de los bosquejos asociados a las interfaces gráficas para contemplar los casos de usos definidos a partir de los requisitos del proyecto.

## Temas Tratados

1. Validación de bosquejos de interfaz gráfica.

## Modificaciones de los bosquejos

1. Se acordó el uso de scroll en lugar de paginado dado que el cliente considera que es más práctico e intuitivo para el alumno.
2. Se define criterio para visualizar las ecuaciones en el listado de ecuaciones en la pantalla principal por el estado.
3. En la pantalla de creación de ecuaciones se incorpora a la botonera el botón opuesto (+/-) que permite definir los números negativos.
4. Se acordó tener en la botonera un botón genérico raíz y al seleccionarlo se despliega la posibilidad de seleccionar la raíz cuadrada o cúbica.
5. Para la potencia también se tendrá un botón genérico que al seleccionarlo se dará la posibilidad de seleccionar potencia de ½, ⅓, 2, 3, -1. -2 o -3.
6. Para la variable se utilizará la x y en lo posible en forma cursiva.
7. Al finalizar la resolución de una ecuación se debe mostrar el conjunto S con las soluciones obtenidas. Ejemplo S = {2, 3}.
8. El historial de resolución con los pasos ingresados por el usuario para solucionar la ecuación será mostrado en forma ascendente por la fecha de ingreso.

# Acta de Reunión de Requisitos 4
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 22/03/2017 | 1.0 | Inicial | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 23/12/2016

**Participantes:**

| | |
|---|---|
| Sylvia da Rosa | (Tutor) |
| Federico Gómez | (Tutor) |
| Nora Ravaioli | (Cliente) |
| Teresa Pérez | (Cliente) |
| Camila Rojí | (Estudiante) |
| Martín Poli | (Estudiante) |

# ÍNDICE

## Introducción

En esta reunión con el cliente se validó la forma de ingresar las ecuaciones en la aplicación así como se muestran las ecuaciones luego de creadas.

## Temas Tratados

1. Validación de la creación de ecuaciones
2. Validación del listado de ecuaciones

## Modificaciones solicitadas

1. Cambiar el color de los textbox que se crean automáticamente al ingresar un operador para que tengan el mismo color del fondo, y una vez que se seleccionen se pongan en blanco.
2. En el listado de las ecuaciones diferenciar más los estados de las ecuaciones. Se van a ver distintas opciones en las que se pinta el fondo o cambios de color para que los estados sean más diferenciables.
3. Las potencias elevadas a la ½ o ⅓ se deben mostrar como raíz cuadrada o raíz cúbica.
4. Quitar los paréntesis redundantes.
5. Agregar botón para deshacer el último cambio realizado con la botonera en la creación y resolución de ecuaciones.
6. Se quita el botón opuesto (+/-) de la botonera dado que es redundante y se aprovecha el lugar para mostrar el botón deshacer cambio realizado.

# Acta de Reunión de Requisitos 5
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 20/05/2017 | 1.0 | Preparación del CUREM. | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 20/05/2017

**Participantes:**

| | |
|---|---|
| Nora Ravaioli | (Cliente) |
| Teresa Pérez | (Cliente) |
| Camila Rojí | (Estudiante) |
| Martín Poli | (Estudiante) |

# ÍNDICE

## Introducción

En esta acta se van a documentar las solicitudes realizadas por Nora y Teresa al trabajar con la aplicación.

## Temas Tratados

1. Le mostramos la aplicación a Nora para que pruebe la resolución de las ecuaciones.
2. Planificamos la charla del CUREM

## Modificaciones solicitadas

1. Se solicitó que haya listados independientes para las ecuaciones descargadas, las ecuaciones creadas, las ecuaciones por defecto y las ecuaciones eliminadas.
2. Se solicitó que la bifurcación de las ecuaciones se pueda detectar en el primer paso en que la sub ecuación seleccionada tenga más de una solución. Se pidió que en ese momento se abriera otra rama al costado que ponga como paso inicial el paso en que se notó la bifurcación y se solicite que se ingrese la otra opción.
3. Actualmente cuando se termina de resolver una ecuación, se muestra un pop up indicando que la resolución de la ecuación fue finalizada. Se solicitó que no se mostrará ese cartel ya que con que se muestre el conjunto solución debajo es suficiente.
4. Se solicitó que las ecuaciones por defecto aparezcan enumeradas en el listado de ecuaciones de la pantalla principal.

# Acta de Reunión de Requisitos 6
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 22/05/2017 | 1.0 | Presentación en el CUREM | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 22/05/2017

**Participantes:**
Sylvia da Rosa (Tutor)
Nora Ravaioli (Cliente)
Teresa Pérez (Cliente)
Camila Rojí (Estudiante)
Martín Poli (Estudiante)
Profesores y estudiantes de profesorados asistentes al congreso CUREM

# ÍNDICE

## Introducción

En la presentación en el curem se explicó la utilidad del programa y se realizaron pruebas con profesores de matemática y estudiantes de profesorado de matemática.

## Temas Tratados

1. Presentación del producto a los asistentes de la charla
2. Taller con los profesores para que probaran la aplicación

## Modificaciones solicitadas

1. Cuando una ecuación no tiene solución real, actualmente se deja que se intente tres veces. Los profesores asistentes a la charla sugirieron que los estudiantes pudieran ingresar que la ecuación no tiene solución, para los casos en que el estudiante se da cuenta antes de que se muestre el aviso de que la ecuación tiene conjunto solución vacío.

# Acta de Reunión de Requisitos 7
# Versión 1.0

## Resolución de ecuaciones por métodos aritméticos en enseñanza media

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 22/05/2017 | 1.0 | Reunión de proyecto de grado luego del CUREM | Camila Rojí - Martín Poli |
| | | | |
| | | | |
| | | | |

**Fecha reunión:** 22/05/2017

**Participantes:**
Sylvia da Rosa (Tutor)
Federico Gomez (Tutor)
Nora Ravaioli (Cliente)
Teresa Pérez (Cliente)
Camila Rojí (Estudiante)
Martín Poli (Estudiante)

# ÍNDICE

## Introducción

1. En esta reunión se informó a nuestros tutores de los cambios que nos habían solicitado Nora y Teresa en la preparación de la charla del CUREM.

## Temas Tratados

1. Nuestros tutores nos plantearon la necesidad de que les especificaremos los requisitos necesarios para el servidor del repositorio de ecuaciones. Nosotros pactamos en hacer una prueba instalando el tomcat y el resto de las aplicaciones necesarias para pasarles con seguridad los requisitos necesarios.
2. Informamos a los tutores sobre los cambios solicitados por el cliente unos días antes.
3. Se coordinó para ir a probar el instalador al liceo de Shangrila en la semana de entre el 2 y el 9 de junio.
4. Se coordinó la prueba final en el liceo de Shangrilá para el 16 de junio y se pactó que si en esa prueba no se encontraba ningún error entonces el producto quedaba finalizado.
5. Nos pusimos de acuerdo en entregar el informe final a fines de Julio.
6. Nora y Teresa dieron algunas ideas para el logo y el nombre de la aplicación.

## Modificaciones solicitadas

1. En la resolución de ecuaciones sin solución real, se solicitó que en vez de mostrar un mensaje indicando que no hay solución real a los 3 pasos que fuera a los 5. Y se confirmó en agregar un botón para indicar que la ecuación no tiene solución como lo solicitaron los profesores en el CUREM.