

# Centro de Enseñanza Técnica Industrial

Organismo Público Descentralizado Federal

## PWA para la interpretación y solución de ecuaciones diferenciales

---

PWA for ODE

### Integrantes

Andrés Huerta Vásquez

Daniel Tejeda Saavedra

David Alejandro López Torres

### Asesores

Ángel Emmanuel Brambila Santamaría

Carlos Molina Martínez



Guadalajara, Jal.

Agosto - Diciembre 2021



# Resumen

---

En este proyecto se abordará el diseño de una PWA (Aplicación web progresiva por sus siglas en ingles), la cual presentará al usuario una manera sencilla, eficaz y accesible de resolver varios tipos de ecuaciones diferenciales. Presentaremos las herramientas, tecnologías y metodologías que se siguieron para el diseño de ambas partes principales del proyecto (el lado del cliente y el lado del servidor). En el lado del cliente mostramos el desarrollo de una ampliación que presenta al usuario resultados en LaTeX y así como mecanismos para ingresar una ecuación diferencial que el sistema entienda y pueda resolver. En el lado del servidor presentamos los distintos métodos para formatear la información enviada y recibida del cliente, así como las distintas implementaciones para la resolución paso a paso de las ecuaciones y el manejo de errores o excepciones.

# Tabla de contenido

---

Índices particulares	8
Nomenclatura	10
Definición del problema	14
Delimitación del problema	15
Justificación	16
<b>Capítulo I: Solución de ecuaciones diferenciales a través del tiempo</b>	<b>17</b>
1. Panorama histórico	18
1.1. Origen del estudio de las ODEs	18
1.2. Python y Matlab: Papel de la informática en las matemáticas	20
1.3. Reconocimiento de texto en imágenes por CNN	21
1.4. Aplicaciones multiplataforma en el siglo XXI	22
2. Panorama teórico	23
2.1. Algoritmos de solución de ODEs	23
2.2. Softwares para la solución de ODEs	31
<b>Capítulo II: Diseño de un ODE Solver</b>	<b>36</b>
1. Alternativas	37
2. Evaluación y selección	38
3. Maquetado e interfaces	39
<b>Capítulo III: Descripción del ODE Solver</b>	<b>55</b>
1. Describiendo el proyecto	56
1.1. Panorama general	56
1.2. Interacción modular del cliente	56
1.3. Módulo de recepción	59
1.4. Módulo de interpretación	59
1.5. Módulo de formateo	62
1.6. Módulo de despacho	63
1.7. Módulo de retención	63
1.8. Módulo de despliegue	66
1.9. Módulo de presentación	66

1.10. Introducción al servidor	67
1.11. Estableciendo la escala de dificultad	68
1.12. Pasos	69
1.13. Paso algebraico	71
1.14. Módulo de clasificación	72
1.15. Módulo de control	77
1.16. Paso integral	77
1.17. Árbol de caminos y módulo de integración	82
1.18. Módulo traductor y serializador	87
1.19. Interacción modular del servidor	88
1.20. Recuento de excepciones del servidor	89
1.21. API Vision	90
1.22. Base de datos	91
2. Requerimientos funcionales	93
2.1. Inicio de sesión	93
2.2. Configuración de cuenta	95
2.3. Historial	96
2.4. Base de datos	98
2.5. Módulo de recepción	99
2.6. Módulo de interpretación	103
2.7. Módulo de formateo	104
2.8. Módulo de despacho	105
2.9. Módulo de despliegue	107
2.10. Módulo de presentación	108
2.11. Módulo de serialización	110
2.12. Módulo de clasificación	111
2.13. Módulo de control	115
2.14. Módulo de Integración	124
2.15. Módulo de Traducción	130
3. Requerimientos no funcionales	131
4. Costo y materiales	143
4.1. Proyección de gastos con modelo COCOMO	143
4.2. Materiales para el desarrollo	144
5. Actividades	145
<b>Capítulo IV: Codificación del servidor</b>	<b>154</b>
1. Configuración de la API	155
1.1. Introducción	155

1.2. Imports	155
1.3. Estructura del archivo api.py	157
1.4. Configuración de CORS	157
1.5. Rutas y métodos	157
1.6. Ruta para obtener solución de ecuación	159
1.7. Ruta para obtener LaTeX de ecuación	161
1.8. Ruta para obtener texto de imagen	162
1.9. Ruta para obtener PDF de solución	163
1.10. Ruta para actualizar catálogo atómico	164
1.11. Ruta para actualizar catálogo recursivo	165
1.12. Iniciar el servidor	166
2. Controlador	167
2.1. Introducción	167
2.2. Imports	167
2.3. Estructura del archivo controller.py	167
2.4. Interpretación	170
2.5. Clasificación	170
2.6. Dificultad	171
2.7. Llamada a los solvers	171
2.8. Manejo de excepciones	173
3. Solucionadores	175
3.1. Solver de ODEs Separables de Primer Orden	175
3.2. Solver de ODEs Homogéneas de Primer Orden	184
3.3. Solver de ODEs Lineares de Primer Orden	191
3.4. Solver de ODEs Reducibles a Lineares de Primer Orden (Bernoulli)	201
3.5. Solver de ODEs Exactas de Primer Orden	204
3.6. Solver de ODEs de Orden Superior con Coeficientes Constantes	212
4. Pasos algebraicos	223
5. Pasos analíticos	228
6. Anomalías del servidor	230
6.1. Anomalía de clasificación	230
6.2. Anomalía de completitud	231
7. Clasificador	232
7.1. Introducción	232
7.2. Clasificador de ODEs Separables de Primer Orden	233
7.3. Clasificador de ODEs Homogéneas de Primer Orden	234
7.4. Clasificador de ODEs Lineares de Primer Orden	235
7.5. Clasificador de ODEs Reducibles a Lineares de Primer Orden	236
7.6. Clasificador de ODEs Exactas de Primer Orden	237
7.7. Clasificador ODEs de Orden Superior Coeficientes Constantes	238

7.8. Clasificación de respaldo con SymPy	239
8. Compilación a LaTeX	240
9. Integrador	241
9.1. Introducción	241
9.2. Nivel de profundidad	244
9.3. Caso de Integrando Constante	245
9.4. Caso de Integrando Constante por Función	246
9.5. Caso de Integrando Atómico	247
9.6. Caso de Integrando Recursivo	247
9.7. Caso de Integrando en Suma de Funciones	249
9.8. Caso de Integración por Partes Forzada	250
9.9. Exportación del módulo	252
9.10. Comparación de integrales	253
9.11. Actualización de los catálogos	256
10. Intérpretes	257
11. Temporizadores	260
12. Utilidades	261
12.1. Constantes	261
12.2. Permisos de Firebase	262
<b>Capítulo V: Codificación del cliente</b>	263
1. Archivos públicos	264
1.1. Punto de entrada	264
1.2. Punto de entrada offline	265
1.3. Registro de cache de PWA	266
2. Alertas	267
2.1. Dialogo Base	267
2.2. Dialogo de Espera	267
3. Anomalías	268
3.1. Dialogo de Anomalia de Clasificación	268
3.2. Dialogo de Anomalia de Completitud	268
3.3. Dialogo de Anomalia Generica	269
3.4. Dialogo de Anomalia de interpretación	269
3.5. Dialogo de Anomalia de Tiempo	270
4. Pantallas principales	271
4.1. Pantalla de Inicio de sesión	271
4.2. Pantalla de Crear cuenta	273
4.3. Componente de Barra de navegación	276
4.4. Pantalla de Bienvenida	277
4.5. Componente de Ecuacion	278

4.6. Pantalla de Historial	280
4.7. Pantalla de Configuración	283
5. Pantallas de entrada	289
5.1. Componente de Teclado	289
5.2. Componente de Canvas	296
5.3. Pantalla de Dibujo	300
5.4. Pantalla de Teclado	300
5.5. Pantalla de Fotografía	301
6. Pantalla de solución	305
<b>Capítulo VI: Pruebas y Evaluación</b>	313
1. ODEs separables	314
2. ODEs homogéneas	319
3. ODEs Lineales	322
4. ODEs Reducibles a Linear	325
5. ODEs Exactas	327
6. ODEs Orden Superior	329
<b>Conclusiones</b>	331
1. Situación del proyecto	332
2. Futuro del proyecto	333
3. Futuras líneas	334
4. Lecciones del proyecto	335
5. Lecciones sobre el desarrollo	336
<b>Referencias</b>	337
1. Referencias de Python	338
2. Referencias de React	340
3. Referencias de ODE & Vision	341
<b>Apéndices</b>	343
1. Manual de Usuario	344
1.1. Creación de Cuenta	344
1.2. Inicio de Sesión	345
1.3. Pantalla de Bienvenida	346
1.4. Pantalla de Dibujo	347
1.5. Pantalla de Fotografía	348
1.6. Pantalla de Teclado	349
1.7. Pantalla de Vista Previa y Solución	350
1.8. Pantalla de Historial	352



1.9. Pantalla de Configuración	353
1.10. Pantalla de Información	354
2. Mantenimiento	355
3. Administración	356
<b>Anexos</b>	357
1. Integrales Atómicas	358
2. Integrales Recursivas	391
3. Actualizador de Integrales	403
4. Escritor de catálogos	404

# Índices particulares

---

Figura 1: Maquetado de Pantalla de inicio	39
Figura 2: Maquetado de Pantalla de inicio de sesión	40
Figura 3: Maquetado de Pantalla de crear cuenta	41
Figura 4: Maquetado de Pantalla de menú principal	42
Figura 5: Maquetado de Pantalla de resolver ecuación	43
Figura 6: Maquetado de Pantalla de ingresar imagen	44
Figura 7: Maquetado de Pantalla de dibujo	45
Figura 8: Maquetado de Pantalla de teclado	46
Figura 9: Maquetado de Pantalla de configuración	47
Figura 10: Maquetado de Pantalla de cambio de contraseña	48
Figura 11: Maquetado de Pantalla de historial	49
Figura 12: Maquetado de Pantalla de inicio (escritorio)	50
Figura 13: Maquetado de Pantalla de inicio de sesión (escritorio)	50
Figura 14: Maquetado de Pantalla de crear cuenta (escritorio)	51
Figura 15: Maquetado de Pantalla de menú principal (escritorio)	51
Figura 16: Maquetado de Pantalla de dibujo (escritorio)	52
Figura 17: Maquetado de Pantalla de teclado (escritorio)	52
Figura 18; Maquetado de Pantalla de imagen (escritorio)	53
Figura 19: Maquetado de Pantalla de configuración (escritorio)	53
Figura 20: Maquetado de Pantalla de historial (escritorio)	54
Figura 21: Diagrama de interacción Usuario - Cliente - Servidor	56
Figura 22: Diagrama de módulos del cliente	58
Figura 23: Diagrama de fase 1 de clasificación	73
Figura 24: Diagrama de fase 2 de clasificación	74
Figura 25: Diagrama de ejemplo de árbol de caminos	83
Figura 26: Diagrama ejemplo de árbol de caminos y pasos añadidos	86
Figura 27: Módulo de integración	87
Figura 28: Módulos del servidor	88
Figura 29: Demostración de Pantalla de crear cuenta	345
Figura 30: Demostración de Pantalla de inicio de sesión	345
Figura 31: Demostración de Pantalla inicio	346
Figura 32: Demostración de Pantalla de dibujo	348

Figura 33: Demostración de Pantalla de fotografía	349
Figura 34: Demostración de Pantalla de teclado	350
Figura 35: Demostración de Pantalla de vista previa	350
Figura 36: Demostración de Pantalla de solución	351
Figura 37: Demostración de Botones para exportar solución	351
Figura 38: Demostración de Pantalla de historial	353
Figura 39: Demostración de Pantalla de configuración	353
Figura 40: Demostración de Pantalla de información	354

# Nomenclatura

---

**Red neuronal:** Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

**Neurona:** Es el elemento básico de computación (modelo de neurona). Recibe un input desde otras unidades o de una fuente externa de datos.

**Entrenar la Red:** las redes neuronales son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. En el lenguaje propio, encontrar la combinación que mejor se ajusta.

**Sigmoide:** La función sigmoide es la función de activación de las neuronas y se define como  $\sigma(x) = 1/(1+e^{-x})$  donde "e" denota la constante exponencial, que es aproximadamente igual a 2,71828.  $\sigma(z)$  actúa como una especie de función "aplastadora", comprimiendo cualquier entrada de números reales a una salida con un rango de 0 a 1

**Neurona sigmoide:** Una neurona que utiliza el sigmoide como función de activación se le llama neurona sigmoide.

**Back-Propagation (Propagación hacia atrás):** Es un método que bajo ciertas suposiciones aproxima rápida y eficazmente los pesos para minimizar el error de entrenamiento de una red neuronal.

**Deep Learning (aprendizaje profundo):** Es un tipo de aprendizaje automático (machine Learning) en el que un modelo aprende a realizar tareas de clasificación.

**Perceptrón:** El perceptrón es la red neuronal más básica que existe de aprendizaje supervisado

**CNN (Red Neuronal Convolutacional):** Es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual de un cerebro biológico. Este tipo de red es una variación de un perceptrón multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión

artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

**Error:** Hay 2 tipos de error el primero evalúa cómo se ajusta la salida de la red neuronal al conjunto de datos de que disponemos, y que se denomina término de error, y otro que se denomina término de regularización, y que se utiliza para evitar el sobre aprendizaje por medio del control de la complejidad efectiva de la red neuronal.

**Descenso del Gradiente:** Es el algoritmo de entrenamiento más simple y también el más extendido y conocido para la optimización del error.

**Método de Newton:** Es uno de los algoritmos conocidos como de segundo orden, ya que hace uso de la Hessiana. Tiene como objetivo encontrar las mejores direcciones de variación de los parámetros haciendo uso de las derivadas segundas de la función de error.

**Algoritmo de Levenberg-Marquardt:** Es un algoritmo que optimiza la función de error, también conocido como método de mínimos cuadrados amortiguado, ha sido diseñado para trabajar específicamente con funciones de error que se expresan como suma de errores. Funciona sin calcular la matriz Hessiana exacta.

**Sistema OCR:** es un sistema computarizado de análisis que permite el reconocimiento de la máquina de caracteres de texto impreso.

**Ecuación Diferencial:** Una ecuación diferencial (ED) es una ecuación que relaciona de manera no trivial a una función desconocida y una o más derivadas de esta función desconocida con respecto a una o más variables independientes.

**Ecuación diferencial en derivadas parciales:** Una E.D.P. es una ecuación diferencial en la que aparecen derivadas parciales de una o más variables dependientes respecto a más de una variable independiente.

**Ecuación Diferencial Ordinaria:** Una EDO es una ecuación en qué las incógnitas son una o varias funciones que dependen de una variable independiente. Además, para evaluar la ecuación en un punto sólo nos hace falta conocer el valor de las funciones incógnitas y sus derivadas en ese punto.

**Orden (ED):** El orden de la derivada más alta en una ecuación diferencial se denomina orden de la ecuación diferencial.

**Grado (ED):** Es la potencia de la derivada de mayor orden que aparece en la ecuación, siempre y cuando la ecuación esté en forma polinómica, de no ser así se considera que no tiene grado. Es la potencia de la derivada de mayor orden que aparece en la ecuación, siempre y cuando la ecuación esté en forma polinómica, de no ser así se considera que no tiene grado.

**Métodos numéricos:** Un método numérico es un algoritmo que intenta resolver una operación matemática compleja en un ordenador. Los motivos por los que se usa un método numérico en vez de intentar una solución analítica pueden ser varios: El problema es muy complejo, y no se puede encontrar una solución analítica en la práctica, el problema no tiene solución analítica conocida, pero puede resolverse de manera numérica, el tamaño de la solución lo hace impracticable para resolver a mano

**Convergencia:** Se dice que un método numérico es convergente si la solución numérica se aproxima a la solución exacta cuando el tamaño de paso  $h$  tiende a 0

**Método de Euler:** Consiste en un método para encontrar iterativamente la solución de una ecuación diferencial de primer orden y valores iniciales conocidos para un rango de valores.

**Método de Euler hacia atrás:** Es un método para la aproximación de la solución de ecuaciones diferenciales. Es similar al método de Euler, pero se diferencia en que es un método implícito. El método de Euler hacia atrás tiene un orden en el tiempo.

**Método de Runge-Kutta:** En análisis numérico, los métodos de Runge-Kutta son un conjunto de métodos genéricos iterativos, explícitos e implícitos, de resolución numérica de ecuaciones diferenciales.

**Método de Rayleigh-Ritz:** Es un método numérico para encontrar aproximaciones a las ecuaciones de valor propio que son difíciles de resolver analíticamente, particularmente en el contexto de la resolución de problemas de valor límite físico que pueden expresarse como ecuaciones diferenciales de matriz.

**Python:** Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

**Kivy:** Es un framework para Python de código abierto y multiplataforma que permite desarrollar aplicaciones con funcionalidades complejas, interfaz de usuarios amigables y con propiedades multitáctiles, todo esto desde una herramienta intuitiva, orientada a generar prototipos de manera rápida y con diseños eficiente que ayuden a tener códigos reutilizables y de fácil despliegue.

**Matlab:** es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

**SciPy:** Es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos.

**NumPy:** Es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

**TensorFlow:** Es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer las necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

**Keras:** Es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano. Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de aprendizaje profundo.

# Definición del problema

---

## Perspectiva General

En la vida universitaria constantemente se afrontan problemas de todo tipo debido a que se trata de una de las etapas de la vida estudiantil que representa un mayor estrés para los alumnos, especialmente para aquellos que estudian una ingeniería. La constante demanda académica e intelectual que exigen las diversas ingenierías se ha vuelto detonante para una serie de situaciones problemáticas en la vida del estudiante estresado, tales como: insomnio, malos hábitos alimenticios, irritabilidad, falta de concentración, constante malestar físico y una tendencia a enfermarse con mayor facilidad.

Haciendo un análisis cuanto menos superficial del tema, sale a la luz que una de las materias responsables de aportar tal estrés en la vida universitaria es la de matemáticas, particularmente aquellas relacionadas con cálculo avanzado: La gran destreza y técnica que se requiere para resolver los ejercicios que día a día se presentan en la ingeniería se ha vuelto no solo una lucha para aquellos que sueñan con graduarse, también lo ha sido para aquellos que, por desgracia, han tenido que abandonar sus estudios por simplemente “no dar el ancho” con estas materias.

Profundizando aún más en la problemática resulta que, en su mayoría, los estudiantes de ingeniería aseguran “no contar con las habilidades ni conocimientos necesarios para poder resolver problemas de las materias de cálculo superior”, esto debido a una gran cantidad de factores que no serán nuestro objeto de estudio. Tomando esta problemática como referente, se ha decidido desarrollar una aplicación capaz de resolver una gran gama de problemas complejos de cálculo superior (ecuaciones diferenciales, en este caso) costando solo una simple fotografía al ejercicio en cuestión; lo que sugiere la siguiente pregunta de investigación:

¿En qué medida una aplicación, capaz de resolver ecuaciones diferenciales por medio de métodos numéricos y empleando redes neuronales para un lector óptico, reduce el estrés académico en la vida universitaria de los estudiantes?



# Delimitación del proyecto

---

## Proponiendo Objetivos

De manera general, se dice que el proyecto busca evaluar el desempeño de una aplicación, que resuelve ecuaciones diferenciales por medio de métodos numéricos y que implementa redes neuronales para generar un lector óptico, como reductor del estrés académico de los universitarios.

Siendo más concretos, es posible desglosar el proyecto en los siguientes objetivos específicos:

- Concretar un algoritmo numérico capaz de resolver ecuaciones diferenciales de manera simbólica.
- Implementar una red neuronal para la detección de texto en imágenes que represente una ecuación diferencial.
- Traducir el texto obtenido en elementos matemáticos por medio de un algoritmo para identificar el tipo de ecuación diferencial al que se hace referencia.
- Diseñar una interfaz gráfica por medio de algún framework de desarrollo web para que el usuario y el sistema intercambien información de manera clara y natural.

Los límites del proyecto se explorarán más adelante en la descripción del proyecto.

# Justificación

---

El desarrollo de este proyecto favorece a mejorar el desempeño académico, forzando así una creciente mejora en el sector académico de los estudiantes universitarios de aquellas carreras que implican el uso directo de ecuaciones diferenciales. El hecho de que esta aplicación sea tan versátil en el tipo de ecuaciones diferenciales que puede resolver permite que pueda ser utilizada en diversos contextos de ingeniería y disciplinas asociadas con las ciencias exactas.

Para el desarrollo científico también resulta una herramienta útil, ya que en varias ocasiones se presentan este tipo de problemas matemáticos que obstruyen llegar a conclusiones rápidamente, ya que su resolución puede tornarse muy compleja o en ocasiones hasta tediosa. La facilidad de uso de la aplicación gracias a su interfaz gráfica es un pilar importante del proyecto, ya que esto garantiza que está al alcance de todos los estudiantes, profesores e investigadores sin necesidad de un riguroso sistema para insertar la ecuación.

Esta herramienta representa además una puerta para nuevos estudios de fenómenos que ocurren en la vida cotidiana, pues el estudio de ecuaciones diferenciales es requerido día con día por diferentes científicos e ingenieros, de modo que esta aplicación les brinda a todos ellos una manera sencilla de estudiar el comportamiento de dichos fenómenos por medio de ecuaciones diferenciales que, de ser results, podrían brindarles información para concluir rotundamente con sus investigaciones. Con todo esto, el desarrollo de la aplicación impacta no solo a la comunidad estudiantil, sino que envuelve todo un entorno de mejoras en la comunidad científica de la sociedad.

# Capítulo I:

## Solución de ecuaciones diferenciales a través del tiempo

# 1. Panorama histórico

---

## 1.1. Origen del estudio de las ODEs

Las ecuaciones diferenciales ordinarias comienzan con el nacimiento del cálculo de Isaac Newton (1643-1727) y Gottfried Wilhelm Leibniz (1646-1716), quienes iniciaron el estudio del problema inverso de la diferenciación: dada una relación entre dos cantidades y sus diferenciales (o fluxiones), cómo encontrar una relación entre las cantidades (o flujos). Sin embargo, este problema analítico de la integración de ecuaciones diferenciales de orden corresponde a un problema geométrico formulado con anterioridad: el método inverso de tangentes; esto es, cómo encontrar una curva caracterizada por una propiedad dada de sus tangentes.

Utilizando expansiones de expresiones en series de potencias, Newton mostró que el problema inverso de las tangentes era totalmente soluble. Leibniz, sin embargo, expresando su deseo de lograr soluciones dando la naturaleza de las curvas, no estaba satisfecho con el sistemático uso de series y pensaba que, hablando de forma general, no había suficiente conocimiento todavía acerca del método inverso de las tangentes. Su procedimiento fue esencialmente cambiar variables para intentar transformar la ecuación diferencial dada en una ecuación con variables separables:

$$f(x)dx = g(y)dy$$

pues su solución se obtenía inmediatamente por cuadraturas. Incluso, antes de comenzar el siglo XVIII, los trabajos de, especialmente, Gottfried Wilhelm Leibniz, Jacob Bernoulli (1654-1705) y Johann Bernoulli (1667-1748) llevaron hacia la integración (reducción a cuadraturas) de ecuaciones diferenciales homogéneas y de ecuaciones diferenciales lineales de primer orden.

Sin embargo, incluso habiendo logrado tal separación de variables, aunque no siempre es el caso, continúa el problema de reducir las cuadraturas a otras más simples. Además, Johann Bernoulli destaca en su "Lectiones mathematicae" en 1691, que la separación de variables puede ocultar la naturaleza del problema. Por ejemplo,  $ydx = 2xdy$  escrita como variables separables involucra, en apariencia curvas logarítmicas cuando, en realidad, la solución es algebraica:  $y^2 = Kx$ .

Varios problemas geométricos y mecánicos provocaron que los matemáticos comenzaran a pensar acerca de las ecuaciones diferenciales de orden mayor que uno. Este es el caso de Jacopo Riccati (1676-1754) quien presentó en 1723 la ecuación que lleva su nombre:  $x^m d^2x = d^2y + dy^2$  resuelta por Daniel Bernoulli

(1700-1782) y Leonhard Euler. Las bases de la teoría general de la ecuación diferencial lineal de orden  $n$  con coeficientes variables fueron desarrolladas en 1765 por Joseph Louis Lagrange (1736-1813) y Jean le Rond D'Alembert (1717-1783). Usando dos métodos diferentes, mostraron que  $n$  integrales particulares de la ecuación homogénea determinan la integral completa de la ecuación no homogénea a través de  $n$  cuadraturas. En 1776, Lagrange nota que este resultado puede también ser demostrado usando el método de variación de la constante, que se convirtió en el método general más utilizado.

En 1715, Brook Taylor (1685-1731) ya se había encontrado con una solución en el caso de las ecuaciones de segundo grado, y notado su carácter singular. En 1758, Euler enfatizó la paradoja dual de tales soluciones singulares en el cálculo integral. Estas soluciones son obtenidas no por integración, sino por diferenciación de ecuaciones diferenciales. A medida que se comienzan a estudiar sistemas físicos más complejos, por ejemplo, en la astronomía, se requiere resolver sistemas de ecuaciones diferenciales ordinarias.

El problema del movimiento de dos cuerpos bajo atracción de la fuerza de gravedad fue resuelto geoméricamente por Newton en 1687, pero no es hasta 1734 que Daniel Bernoulli resuelve el problema de los dos cuerpos de forma analítica. El llamado problema de los  $n$  cuerpos es una generalización de este que no puede ser resuelta de la misma manera y es ampliamente estudiado hasta la fecha. Aparecen, para casos muy particulares, resultados de Newton, Euler y en especial de Lagrange (1772). Este mismo problema, condujo al desarrollo de la teoría del cálculo de perturbaciones para encontrar soluciones aproximadas, donde destacan Clairaut en 1747, Euler en 1748, Lagrange en 1774-1775 y Pierre-Simon Laplace (1749-1827) de 1772 a 1780 aproximadamente.

El estudio y resolución de sistemas de ecuaciones diferenciales tuvo un gran impulso con las ideas de Lagrange a quien se le debe la aplicación del método de variación de parámetro a un sistema de tres ecuaciones de segundo orden en 1808. No fue hasta inicios del siglo XIX cuando el estudio de las ecuaciones diferenciales ajenas a fenómenos físicos y cuestiones geométricas comenzó a tomar importancia, esto con los avances en la rama aportados por Carl Friedrich Gauss y Bernhard Riemann. A partir de este punto, la solución de algunos tipos de ecuaciones diferenciales se volvió de particular interés para la entonces naciente informática. La llegada de las guerras mundiales del siglo XX fue un factor detonante para el crecimiento de la informática en modelos de predicción y de localización enemiga. Esta demanda de algoritmos provocó a su vez la evolución de sistemas electrónicos, volviéndolos más compactos y con una gran velocidad para realizar operaciones matemáticas.

El aumento de la velocidad de proceso de los computadores dio lugar a lo que hasta ese entonces se había quedado como una concepción teórica: Los métodos numéricos [34]. Los métodos numéricos permitieron resolver algunas ecuaciones diferenciales de las cuales no se tenía una solución algebraica en ese entonces. La supremacía de los ordenadores en la segunda mitad del siglo XX llegó a su apogeo durante la Guerra Fría; y con la llegada del hombre a la luna se reafirmó el poder que podrían llegar a tener los ordenadores para resolver problemas matemáticos.

A finales del siglo XX la magnitud de la creciente informática impulsó al desarrollo de nuevos lenguajes de programación orientados a facilitar el proceso de codificación de las nuevas generaciones. Cada uno de estos lenguajes de programación nacientes fue diseñado para atender una necesidad específica del campo informático que hasta ese entonces se encontraba muy fragmentado y lejos de ser comprensible por la mayoría de la población. Fue bajo este contexto que se desarrollaron lenguajes como Python y Matlab, dos pilares del tratamiento informático de nuestra era.

## 1.2. Python y Matlab: Papel de la informática en las matemáticas modernas

Python [1] fue diseñado en primer lugar como un lenguaje que buscaba conservar la esencia de la programación orientada a objetos que lenguajes anteriores a él poseían, pero apostó por mejorar la sintaxis y reducir las líneas de código para facilitar su lectura. Sin embargo, la facilidad de código permitió rápidamente anexar otros paradigmas y plataformas al entonces primitivo Python. La incorporación del sistema de importación de paquetes de Python fue crucial para comenzar con la etapa de conexión de Python con otros sistemas e incluso lenguajes de programación.

Por otro lado, Matlab es un lenguaje de programación que fue diseñado para el trabajo con numérico de los matemáticos. Su parecido al lenguaje C le hizo adquirir cierta popularidad en la década de los 90's, con lo que comenzó a recibir un mayor mantenimiento y actualización. Rápidamente Matlab se posicionó en la cabecera de los programas favoritos de los matemáticos, pues se comenzaron a añadir funciones cada vez más complejas que permitían resolver problemas de la misma dificultad.

Para inicios del siglo XXI el mundo ya contaba con un fuerte repertorio de métodos numéricos para resolver ecuaciones diferenciales de todos los colores y sabores. Sin embargo, estos métodos numéricos solían ser complejos de codificar y requerían de

una exhaustiva validación para clasificar las ecuaciones. Fue bajo este contexto que Matlab desarrolló una librería que implementa una serie de métodos capaces de resolver ecuaciones diferenciales y, lo que hasta ese momento parecía imposible, llegar a una solución algebraica concreta.

Fue tal el éxito de la librería que otros lenguajes de programación y sistemas informáticos intentaron diseñar sus propios métodos; muchos tuvieron éxito, pero fueron más quiénes fracasaron en el intento. El ya igualmente famoso lenguaje de programación Python desarrolló a mediados del 2016 una librería capaz de resolver una amplia gama de ecuaciones diferenciales; dicha librería recibió el nombre de ODEINT [5], y actualmente se utiliza para resolver ecuaciones diferenciales simples que se pueden presentar en algunas implementaciones de Python.

No conforme con ello, los desarrolladores de Python llegaron a un acuerdo de trabajo colaborativo con Matlab para la importación de funciones de Matlab vía paquetería al código de Python. Esta idea permitió aprovechar lo mejor de ambos lenguajes de programación: la facilidad de código y de asociación con otros lenguajes de Python, así como las poderosas herramientas matemáticas de Matlab.

## 1.3. Reconocimiento de imágenes en texto por CNN

Otro de los grandes retos que la informática ha abordado a lo largo del siglo XXI ha sido el reconocimiento de imágenes. Durante las últimas dos décadas, este campo ha sido enfrentado por las compañías informáticas más importantes de todo el mundo, generando algunos productos y algoritmos para llevar a cabo esta tarea. Los esfuerzos por llevar la detección de imágenes a un nivel más alto impulsaron al desarrollo de lo que hasta ese entonces se no se había dado un papel tan relevante en la informática moderna: Las Redes Neuronales Convolucionales (CNN) [24]. Las CNN permitieron implementar sistemas de detección y clasificación de imágenes y texto en imágenes mucho más sofisticados a los antes desarrollados por medios como Random Forest.

La codificación de las CNN fue una ardua labor para aquellos que lo intentaron, constituyendo uno de los retos aún vigentes de la informática. Pesé a los esfuerzos, el diseño teórico de las CNN no era compatible con varios de los paradigmas que poseen muchos lenguajes de programación. Frente a este reto informático, Python fue el primer lenguaje que permitió codificar de manera exitosa una CNN con todas sus etapas. La implementación de las CNN en lenguaje Python abrió la puerta a la

gestión y control de imágenes por medio de lenguajes de programación de alto nivel, dejando por un lado las complejas matemáticas que posee una CNN detrás.

## 1.4. Aplicaciones multiplataforma en el siglo XXI

Finalmente, el último gran problema informático abordado en el siglo XXI sobre el cuál hablaremos es el creciente número de dispositivos móviles a nivel global: El siglo XXI llegó con una revolución de lo que llevamos en el bolsillo. La codificación para dispositivos móviles fue en inicios todo un desafío, debido a que rápidamente se creó un mercado alrededor de los teléfonos celulares que impidió el desarrollo de la codificación universal. Sin embargo, la llegada del sistema operativo Android ayudó a la estandarización del código interno de los móviles, a excepción de aquellos pertenecientes a marcas que poseen su propio sistema operativo.

Esta destacable división en el mercado de los móviles volvió una tarea complicada desarrollar aplicaciones compatibles para diferentes sistemas operativos ya que era necesario codificar la misma aplicación múltiples veces utilizando los distintos entornos de desarrollo de apps que nos ofrecen los propios sistemas operativos. Esta barrera de marcas fue vencida por un equipo de desarrolladores de Python con su nuevo Framework: Kivy[31].

Kivy[35] brinda la oportunidad a los programadores de Python dar a sus códigos elementos visuales propios de una aplicación como botones y campos de texto. Pero lo realmente sorprendente de esta nueva herramienta es su capacidad para generar soluciones multiplataforma; esto es, la aplicación desarrollada vía Python puede correrse en dispositivos con Android, iOS, Windows, Linux, entre otros.



## 2. Panorama teórico

---

### 2.1. Algoritmos de solución de ODEs

Con base al estudio de las ecuaciones diferenciales desde sus orígenes se han concretado algoritmos y metodologías concretas para la solución de ciertos tipos de ecuaciones diferenciales. Estas metodologías resultan particularmente útiles desde el punto de vista informático ya que presentan una secuencia general de solución para el tipo de ODE una vez que haya sido identificado (que suele ser una tarea más simple que resolver la ecuación en sí).

A continuación, se presentan algunos de los algoritmos más conocidos para la solución de algunos tipos de ecuaciones diferenciales.

#### Primer Orden Separable

Se resuelve mediante un despeje de variables e integración directa a partir de la forma buscada:

$$p(x) - q(y)y' = 0$$

$$\Rightarrow p(x) = q(y) \frac{dy}{dx}$$

$$\Rightarrow p(x)dx = q(y)dy$$

$$\Rightarrow \int p(x)dx = \int q(y)dy$$

$$\Rightarrow P(x) = Q(y)$$

Siendo la última expresión la **solución implícita** de la ecuación diferencial. En caso de que sea posible despejar  $y(x)$  de  $Q(y)$ , entonces esta sería la **solución explícita**. El servidor buscará la solución explícita; en caso de encontrarla en un tiempo establecido también será mostrada al usuario como un paso extra.

## Primer Orden Linear

Se utiliza un cambio de variable que cumpla la siguiente propiedad:

$$v(x)y' + p(x)v(x)y = \frac{d(v(x)y)}{dx}$$

Desarrollando y despejando  $v(x)$ :

$$\Rightarrow v(x)y' + p(x)v(x)y = v'(x)y + v(x)y'$$

$$\Rightarrow p(x)v(x)y = v'(x)y$$

$$\Rightarrow p(x)v(x) = v'(x)$$

$$\Rightarrow p(x)v(x) = \frac{dv}{dx}$$

$$\Rightarrow \frac{dv}{v} = p(x)dx$$

$$\Rightarrow \int \frac{dv}{v} = \int p(x)dx$$

$$\Rightarrow \ln(v) = \int p(x)dx$$

$$\Rightarrow v(x) = \exp\left(\int p(x)dx\right)$$

Partiendo de la ecuación original en la forma buscada:

$$q(x) + p(x)y + y' = 0$$

Multiplicando por  $v(x)$  y utilizando la definición impuesta:

$$\Rightarrow v(x)q(x) + v(x)p(x)y + v(x)y' = 0$$

$$\Rightarrow v(x)q(x) + \frac{d(v(x)y)}{dx} = 0$$

$$\Rightarrow \frac{d(v(x)y)}{dx} = -v(x)q(x)$$

$$\Rightarrow d(v(x)y) = -v(x)q(x)dx$$

$$\Rightarrow v(x)y = -\int v(x)q(x)dx$$

$$\Rightarrow y(x) = -\frac{1}{v(x)} \int v(x)q(x)dx$$

Utilizando el valor obtenido para  $v(x)$  y resolviendo la integral tendríamos la solución explícita de la ecuación diferencial.

## Primer Orden Reducible a Linear (Bernoulli)

Dividiendo la forma general buscada entre  $y^n$ :

$$q(x)y^n + p(x)y + y' = 0$$

$$\Rightarrow q(x) + p(x)y^{1-n} + y'y^{-n} = 0$$

Utilizando el cambio de variable:

$$u = y^{1-n} \Rightarrow \frac{du}{dx} = (1-n)y^{-n} \frac{dy}{dx} = (1-n)y'y^{-n}$$

$$\Rightarrow \frac{1}{1-n} u' = y'y^{-n}$$

La ecuación toma la forma:

$$\Rightarrow q(x) + p(x)u + \frac{1}{1-n} u' = 0$$

$$\Rightarrow (1-n)q(x) + (1-n)p(x)u + u' = 0$$

$$\Rightarrow q_u(x) + p_u(x)u + u' = 0$$

Que es de la forma linear definiendo  $p_u(x) = (1-n)p(x)$  y  $q_u(x) = (1-n)q(x)$ . Al obtener la solución para la ecuación, tendríamos el valor de  $u(x)$ , por lo que solo bastaría deshacer el cambio de variable original para tener la solución explícita:

$$\Rightarrow y(x) = u^{n-1}(x)$$

## Primer Orden Homogénea

Haciendo el cambio de variable  $y = vx \Rightarrow y' = v'x + v$  en la forma general buscada y conociendo que ambas funciones  $p(x, y)$ ,  $q(x, y)$  son homogéneas del mismo grado:

$$q(x, y) + p(x, y)y' = 0$$

$$\Rightarrow q(x, vx) + p(x, vx)(v'x + v) = 0$$

La cuál es una ecuación diferencial en variables separables. Podemos ver esto con el siguiente desarrollo (el servidor lo resolverá desde el paso anterior mandando a llamar a la función para variables separables, lo siguiente solo se muestra para fines demostrativos de que siempre será separable).

$$\Rightarrow xq(1, v) + xp(1, v)(v'x + v) = 0$$

$$\Rightarrow xq_v(v) + xp_v(v)(v'x + v) = 0$$

$$\Rightarrow q_v(v) + p_v(v)(v'x + v) = 0$$

$$\Rightarrow q_v(v) + v'xp_v(v) + vp_v(v) = 0$$

$$\Rightarrow v'xp_v(v) = -(vp_v(v) + q_v(v))$$

$$\Rightarrow \frac{dv}{dx}xp_v(v) = -(vp_v(v) + q_v(v))$$

$$\Rightarrow -\frac{p_v(v)}{vp_v(v) + q_v(v)}dv = \frac{dx}{x}$$

$$\Rightarrow -\int \frac{p_v(v)}{vp_v(v) + q_v(v)}dv = \int \frac{dx}{x}$$

$$\Rightarrow F(v) = \ln(x)$$

$$\Rightarrow F\left(\frac{y}{x}\right) = \ln(x)$$

Que sería la solución implícita de la ecuación diferencial. Note que sobre la marcha se definió  $q_v(v) = q(1, v)$  y que  $p_v(v) = p(1, v)$ .

## Primer Orden Exacta

Sabiendo que la ecuación diferencial es exacta, se busca una función  $F(x, y)$  que cumpla que:

$$q(x, y) = \frac{\partial F}{\partial x}; p(x, y) = \frac{\partial F}{\partial y}$$

Vemos que bajo estas condiciones en la ecuación original tenemos que:

$$q(x, y) + p(x, y)y' = 0$$

$$\Rightarrow q(x, y)dx + p(x, y)dy = 0$$

$$\Rightarrow \frac{\partial F}{\partial x}dx + \frac{\partial F}{\partial y}dy = 0$$

$$\Rightarrow F(x, y) = k$$

Con  $k$  una constante. Utilizando la primera definición impuesta:

$$F(x, y) = \int q(x, y)dx + g(y) \Rightarrow F(x, y) = Q(x, y) + g(y)$$

Derivando y utilizando la segunda definición impuesta:

$$\Rightarrow \frac{\partial F}{\partial y} = \frac{\partial Q}{\partial y} + \frac{dg}{dy}$$

$$\Rightarrow p(x, y) = \frac{\partial Q}{\partial y} + \frac{dg}{dy}$$

$$\Rightarrow dg = \left( p(x, y) - \frac{\partial Q}{\partial y} \right) dy$$

$$\Rightarrow g(y) = \int \left( p(x, y) - \frac{\partial Q}{\partial y} \right) dy$$

Finalmente, la solución implícita de la ecuación diferencial será:

$$Q(x, y) + g(y) = k$$

Utilizando las definiciones de  $Q(x, y)$  y  $g(y)$  obtenidas.

$$\begin{aligned}
&\Rightarrow xq(1, v) + xp(1, v)(v'x + v) = 0 \\
&\Rightarrow xq_v(v) + xp_v(v)(v'x + v) = 0 \\
&\Rightarrow q_v(v) + p_v(v)(v'x + v) = 0 \\
&\Rightarrow q_v(v) + v'xp_v(v) + vp_v(v) = 0 \\
&\Rightarrow v'xp_v(v) = -(vp_v(v) + q_v(v)) \\
&\Rightarrow \frac{dv}{dx}xp_v(v) = -(vp_v(v) + q_v(v)) \\
&\Rightarrow -\frac{p_v(v)}{vp_v(v) + q_v(v)}dv = \frac{dx}{x} \\
&\Rightarrow -\int \frac{p_v(v)}{vp_v(v) + q_v(v)}dv = \int \frac{dx}{x} \Rightarrow F(v) = \ln(x) \\
&\Rightarrow F\left(\frac{y}{x}\right) = \ln(x)
\end{aligned}$$

## Orden Superior con Coeficientes Constantes

La ecuación diferencial the n-ésimo orden más general toma la forma:

$$P_n(t)y^n + P_{n-1}(t)y^{n-1} + \dots + P_1(t)y' + P_0(t)y = G(t)$$

Donde  $y^m = \frac{d^m y}{dx^m}$

Muchas de las ideas de resolución requieren que  $y^n$  tenga coeficiente 1, por lo que dividiendo entre  $P_n(t)$  obtenemos

$$y^n + p_{n-1}(t)y^{n-1} + \dots + p_1(t)y' + p_0(t)y = g(t)$$

Para dicha ecuación diferencial requeriremos de n condiciones iniciales dadas por:

$$y^{n-1}(t_0) = y_{n-1}, \dots, y'(t_0) = y_1, y(t_0) = y_0$$

Con las 2 condiciones anteriores podemos encontrar una solución única dado el siguiente teorema:

### Teorema 1

Supongamos que las funciones  $p_0, p_1, \dots, p_{n-1}$  y  $g(t)$  son continuas en un intervalo abierto  $I$  que contiene a  $t_0$ , luego existe una solución única para la ecuación diferencial dadas las ecuaciones 2) y 3) y la solución existirá para toda  $t$  en  $I$ .

Ahora debemos movernos a la discusión de la solución de la respectiva ecuación homogénea:

$$y^n + p_{n-1}(t)y^{n-1} + \dots + p_1(t)y' + p_0(t)y = 0$$

Supongamos que sabemos que  $y_{n-1}(t), \dots, y_1(t), y_0(t)$  son soluciones a la ecuación 4), luego por extensión del principio de superposición sabemos que:

$$y(t) = c_1 y_1(t) + c_2 y_2(t) + \dots + c_n y_n(t)$$

En orden de que esta sea una solución general debemos encontrar los coeficientes constantes  $c_1, c_2, \dots, c_n$  para cualquier elección de  $t_0$  en el intervalo  $I$  del teorema 1, y cualquier elección de  $y_1, y_2, \dots, y_n$ . En otras palabras debemos encontrar  $c_1, c_2, \dots, c_n$  tal que:

$$\begin{aligned} c_1 y_1(t_0) + c_2 y_2(t_0) + \dots + c_n y_n(t_0) &= y_0 \\ c_1 y_1'(t_0) + c_2 y_2'(t_0) + \dots + c_n y_n'(t_0) &= y_1 \\ &\vdots \\ c_1 y_1^{n-1}(t_0) + c_2 y_2^{n-1}(t_0) + \dots + c_n y_n^{n-1}(t_0) &= y_{n-1} \end{aligned}$$

Para resolver este sistema podemos utilizar la regla de Cramer y el denominador de cada una de las respuestas será el determinante de cada una de las matrices  $n \times n$ :

$$\begin{array}{cccc} y_1 & y_2 & \cdots & y_n \\ y_1' & y_2' & \cdots & y_n' \\ \vdots & \vdots & \ddots & \vdots \\ y_1^{n-1} & y_2^{n-1} & \cdots & y_n^{n-1} \end{array}$$

Dicha matriz puede ser definida como el Wroksiano y se denota como:

$$W(y_1, y_2, \dots, y_n)(t) = \begin{vmatrix} y_1 & y_2 & \cdots & y_n \\ y_1' & y_2' & \cdots & y_n' \\ \vdots & \vdots & \ddots & \vdots \\ y_1^{n-1} & y_2^{n-1} & \cdots & y_n^{n-1} \end{vmatrix}$$

Debido a que el Wroksiano es el denominador en la solución de cada  $c_i$  es claro que la solución obtenida no es cero para cualquier valor de  $t = t_0$  en la que elijamos evaluar el Wroksiano. Esto queda resumido en el siguiente teorema:

## Teorema 2

Suponga que las funciones  $p_0, p_1, \dots, p_{n-1}$  son continuas en el intervalo abierto  $I$  y suponga que  $y_1(t), y_2(t), \dots, y_n(t)$  son soluciones de 4). Si  $W(y_1, y_2, \dots, y_n)(t) \neq 0$  para cada  $t$  en  $I$  entonces  $y_1(t), y_2(t), \dots, y_n(t)$  forman un conjunto fundamental de soluciones y la solución general es:

$$y(t) = c_1 y_1(t) + c_2 y_2(t) + \dots + c_n y_n(t)$$

Tome en cuenta que si un conjunto de soluciones de un conjunto de soluciones fundamentales por tanto serán también un conjunto de funciones lineares independientes.

Dada la incrementada complejidad de encontrar en la práctica soluciones para ecuaciones homogéneas con coeficientes no constantes omitiremos dichas ecuaciones y nos concentraremos en la resolución de ecuaciones con coeficientes constantes.

Sea la ecuación homogénea de la forma:

$$c_n y^n + c_{n-1} y^{n-1} + \dots + c_1 y' + c_0 y = 0$$

Asumiendo que la solución de dicha ecuación diferencial es de la forma  $y(t) = e^{rt}$  y sustituyendo dicha solución en la ecuación previa obtenemos:

$$e^{rt}(c_n r^n + c_{n-1} r^{n-1} + \dots + c_1 r + c_0) = 0$$

Puesto que  $e^{rt} \neq 0$ , el polinomio debe ser 0, y el problema se reduce a encontrar las raíces.

En caso de que tengamos solamente raíces diferentes las distintas soluciones serán

$$e^{r_1 t}, e^{r_2 t}, \dots, e^{r_k t}$$

En caso de que exista una raíz de multiplicidad  $l$

$$e^{rt}, t e^{rt}, \dots, t^{l-1} e^{rt}$$

Tomando en cuenta que  $r$  es complejo podemos expresar  $r$  como  $r = \lambda \pm \mu i$  entonces el conjunto de soluciones anterior se puede describir como:

$$e^{\lambda t} \cos(\mu t), e^{\lambda t} \sin(\mu t), t e^{\lambda t} \cos(\mu t), t e^{\lambda t} \sin(\mu t), \dots, t^{l-1} e^{\lambda t} \cos(\mu t), t^{l-1} e^{\lambda t} \sin(\mu t)$$



## Método de variación de parámetros

Dada la ecuación diferencial inicial

$$y^n + p_{n-1}(t)y^{n-1} + \dots + p_1(t)y' + p_0(t)y = g(t)$$

Asumimos que existe un conjunto de soluciones para la ecuación homogénea asociada. Nosotros sabemos que dicha solución complementaria está dada por:

$$Y(t) = c_1 y_1(t) + c_2 y_2(t) + \dots + c_n y_n(t)$$

El método de variación de parámetros consta en encontrar un conjunto de nuevas funciones  $u_1(t), u_2(t), \dots, u_n(t)$  tal que:

$$Y(t) = u_1(t)y_1(t) + u_2(t)y_2(t) + \dots + u_n(t)y_n(t)$$

Para determinar si esto es posible y encontrar las  $u_i(t)$ , requeriremos de un total de  $n$  ecuaciones que involucren a las funciones desconocidas, que podamos resolver.

Para encontrar dichas ecuaciones comencemos derivando la ecuación anterior:

$$Y'(t) = u_1 y_1' + u_2 y_2' + \dots + u_n y_n' + u_1' y_1 + u_2' y_2 + \dots + u_n' y_n$$

De aquí en adelante puesto que podemos decidir cuáles son las condiciones de nuestra solución particular, podemos asumir condiciones convenientes que simplifiquen nuestra ecuación en este caso asumiremos que:

$$u_1' y_1 + u_2' y_2 + \dots + u_n' y_n = 0$$

## 2.2. Softwares para la solución de ODEs

En la actualidad, se han desarrollado distintas implementaciones en busca de resolver ecuaciones diferenciales por medio de aplicaciones, otras cuantas aplicaciones de detección de ecuaciones por medio de redes neuronales; pero en general ha habido pocos avances en el desarrollo de aplicaciones que resuelvan ecuaciones diferenciales por medio de una imagen (esto es, conectar ambas partes). A lo largo de este apartado describiremos algunas de las alternativas que se han implementado para cumplir la labor.

Podemos observar el trabajo de aplicaciones que resuelven ecuaciones diferenciales realizado sobre el lenguaje Python. En 2017 Python desarrolló una nueva librería para

la solución de ecuaciones diferenciales a medida de reemplazo de la ya mencionada ODEINT. Esta nueva librería incluye la capacidad de llegar a soluciones algebraicas de algunas ecuaciones diferenciales. Este módulo llamado ODE que pertenece a la librería SYMPY funciona mediante métodos recursivos de clasificación e iteración controlada por algoritmos definidos por los métodos numéricos que plantea desarrollar. ODE representa una buena opción para hacer proyectos elaborados en Python que tienen que trabajar en la solución de ecuaciones diferenciales.

Si hablamos de aplicaciones para el reconocimiento de ecuaciones escritas podemos apreciar la labor desarrollada con los sistemas OCR [32]. El desarrollo de sistemas de OCR (Reconocimiento de caracteres) por medio de CNN montados en aplicaciones móviles se vuelve una tarea compleja debido a que el procesamiento de imagen de un dispositivo móvil es por mucho más lento y reducido al que nos brindaría un computador cualquiera con una CNN. Los sistemas OCR tienen dos categorías: en línea, en la que la información de entrada se obtiene a través de sensores de escritura en tiempo real; y fuera de línea, en el que la información de entrada se obtiene a través de información estática (imágenes). Dentro de la categoría fuera de línea, se reconoce el texto mecanografiado y manuscrito.

Durante muchos años, los sistemas HTR han utilizado los modelos ocultos de Márkov (HMM) para la tarea de transcripción, pero recientemente, a través del aprendizaje profundo, el enfoque de redes neuronales recurrentes convolucionales (CRNN) se ha utilizado para superar algunas limitaciones de HMM.

El flujo de trabajo se puede dividir en 3 pasos.

- Paso 1: la imagen de entrada se alimenta a las capas CNN para extraer características. La salida es un mapa de características.
- Paso 2: a través de la implementación de la memoria a corto plazo (LSTM), el RNN puede propagar información a distancias más largas y proporcionar funciones más sólidas para el entrenamiento.
- Paso 3: con la matriz de salida RNN, la Clasificación temporal conexionista (CTC) calcula el valor de pérdida y también decodifica en el texto final.

Otro de los trabajos que se abordan acerca del reconocimiento de imágenes por redes neuronales sugiere utilizar herramientas incorporadas en sistemas operativos como Windows en el reconocimiento de ecuaciones por medio de los recursos de Wolfram Alpha.

Wolfram System utiliza el reconocedor matemático de Microsoft integrado en Windows 7 y superior para reconocer expresiones matemáticas escritas a mano. Esto

le permite ingresar una notación matemática estandarizada manuscrita en una computadora portátil Wolfram System en la forma tradicional. El panel de entrada de escritura a mano matemática fue diseñado para usarse con un bolígrafo digital en dispositivos compatibles, pero puede usarlo con cualquier dispositivo de entrada, como una pantalla táctil, un digitalizador externo o incluso un ratón.

Una de las herramientas más útiles para incorporar en un equipo de Android una red neuronal es por medio de lo que se conoce como TensorFlow. El propósito completo de TensorFlow es tener un llamado gráfico computacional que se pueda ejecutar de manera mucho más eficiente que si los mismos cálculos se realizaran directamente en Python. TensorFlow puede ser más eficiente que NumPy porque TensorFlow conoce todo el gráfico de cálculo que debe ejecutarse, mientras que NumPy solo conoce el cálculo de una sola operación matemática a la vez.

TensorFlow también puede calcular automáticamente los gradientes necesarios para optimizar las variables del gráfico a fin de que el modelo funcione mejor. Esto se debe a que el gráfico es una combinación de expresiones matemáticas simples, por lo que el gradiente de todo el gráfico se puede calcular utilizando la regla de la cadena para derivadas.

TensorFlow también puede aprovechar las CPU de varios núcleos y las GPU, e incluso Google ha creado chips especiales solo para TensorFlow, que se denominan TPU (unidades de procesamiento de tensor) y son incluso más rápidos que las GPU. Un gráfico de TensorFlow consta de las siguientes partes:

- Variables de marcador de posición utilizadas para ingresar datos en el gráfico.
- Variables que se van a optimizar para que la red convolucional funcione mejor.
- Las fórmulas matemáticas para la red convolucional.
- Una medida de costo que puede usarse para guiar la optimización de las variables.
- Un método de optimización que actualiza las variables.

Profundizando aún más en lo que envuelve a una red neuronal, llegamos a que, en un nivel superior, las redes neuronales son codificadores, decodificadores o una combinación de ambos. Los codificadores encuentran patrones en datos sin procesar para formar representaciones compactas y útiles. Los decodificadores generan datos nuevos o información útil de alta resolución a partir de esas representaciones. El Deep Learning descubre formas de representar el mundo para que podamos razonar al respecto. El resto son métodos inteligentes que ayudan a utilizar la información

visual, el lenguaje, el sonido e incluso actuar en un mundo basado en esta información y recompensas ocasionales.

Finalmente, podemos hablar acerca de la posibilidad de crear aplicaciones de Android por medio de Kivy: Es posible crear un paquete para Android usando el proyecto python-for-android. También es posible empaquetarse aplicación para que Kivy Launcher ejecute programas Kivy sin compilarlos. Otra forma de crear una aplicación de Android en Kivy es con Buildozer.

Buildozer es una herramienta que automatiza todo el proceso de construcción. Descarga y configura todos los requisitos previos para python-for-android, incluidos el SDK y NDK de Android, luego crea un apk que se puede enviar automáticamente al dispositivo. Buildozer actualmente solo funciona en Linux, y es una versión alfa, pero ya funciona bien y puede simplificar significativamente la compilación de apk.

Kivy está diseñado para operar de manera idéntica en todas las plataformas y, como resultado, toma algunas decisiones de diseño claras. Incluye su propio conjunto de widgets y, de forma predeterminada, crea un APK con todas las dependencias y bibliotecas principales necesarias.

De este modo podemos confirmar lo que comentamos al inicio de la sección: existe un amplio trabajo acerca de los procedimientos de detección de imagen y de resolución de ecuaciones diferenciales, pero no abundan demasiados ejemplos en la actualidad de lo que podría ser un sistema que acople ambas partes.

Muchos estudiantes de carreras de ingeniería o ciencias exactas suelen verse en la necesidad de resolver ecuaciones diferenciales para una gran parte de sus tareas, lo cual es una carga adicional de estrés en la vida diaria de los estudiantes, se ven frustrados porque ante ciertos problemas que involucran la resolución de ecuaciones diferenciales se sienten incapaces de dar con una solución correcta.

Actualmente solo algunas plataformas y sistemas complejos son capaces de resolver una gran gama de ecuaciones diferenciales que podrían presentarse en la carrera de los estudiantes de ingeniería o ciencias exactas, así como algunos investigadores. El acceso a estas plataformas y sistemas suele llegar a representar una barrera para dar con la solución de la ecuación diferencial.

La propuesta se desenvuelve en el campo académico ya sea visto desde el lado del estudiante o del investigador. Por la diferencia entre estos dos usuarios, el sistema se modelaría para adaptarse al conocimiento y necesidad de cada usuario (estudiantes e investigadores). En cuanto a los recursos necesarios, se tienen las siguientes notas:

El sistema está pensado para funcionar como una PWA (Progressive Web App), de modo que el usuario final puede acceder desde cualquier dispositivo móvil (al menos la mayoría), ordenador de escritorio o a una versión web. Para el uso adecuado del sistema se requiere que el usuario tenga conexión a internet durante el procesamiento de la ecuación.

La PWA será capaz de interpretar ecuaciones diferenciales desde múltiples entradas (imagen, pantalla o texto) y presentar una solución por pasos de estas.

Algunas de las aplicaciones similares que más utilizan los estudiantes son:

**Photomath:** Es la aplicación número 1 para el aprendizaje de las matemáticas; puede leer y resolver problemas que van desde la aritmética al cálculo instantáneamente usando la cámara en su dispositivo móvil.

Se utiliza para abordar los problemas de matemáticas a través de pasos animados e instrucciones detalladas o revisión de tarea para detectar cualquier problema impreso o manuscrito.

**Wolfram Alpha:** Es un buscador online que responde a preguntas y realiza cálculos de manera inmediata. Sus respuestas son detalladas y específicas a los conceptos introducidos en su motor de búsqueda en lugar de proporcionar una lista de documentos o páginas web como hacen otro tipo de buscadores.

Nuestro proyecto es similar a estas aplicaciones, pero dando solución a ecuaciones diferenciales utilizando la aplicación desde cualquier dispositivo.

# Capítulo II:

## Diseño de un ODE Solver

# 1. Alternativas

---

Con el fin de resolver la problemática expuesta, fue necesario decidir cuáles serían las tecnologías que pudieran ser de utilidad en el desarrollo. En particular, la naturaleza del proyecto plantea el desarrollo de dos partes: una que procese las soluciones de las ecuaciones diferenciales (la cual se nombra **servidor**), y otra capaz de brindar al usuario ordinario interacción con este procesamiento (la cual se nombre **cliente**). Dependiendo de las tecnologías y arquitecturas de ambas partes, se obtienen diferentes alterativas para la resolución de la problemática.

Una primera alternativa sería el desarrollo del servidor con base a las tecnologías de manejo simbólico que ya existen en la actualidad y que en su momento fueron pioneros en la resolución de ecuaciones diferenciales a través de Software (Como Matlab y Wolfram). Para esto, sería necesario adquirir las licencias de estos programas y aprender a manejarlos de tal manera que sea posible integrarlos con una aplicación cliente desarrollada de manera externa. Una segunda opción para el servidor podría ser utilizar algunas de las tecnologías Open Source que existen (como es el caso de SymPy) y desarrollar el servidor de tal forma que funcione como una API que sea accesible para una aplicación cliente, cualquiera que se decidiera desarrollar.

Para el desarrollo de la aplicación cliente existen diferentes enfoques que pueden suplir la parte de accesibilidad que requiere este sistema. El desarrollo de aplicaciones clientes para cada plataforma (Android, IOS, Windows, Linux, entre otros) es una posible solución para garantizar la accesibilidad de todos los usuarios y permite que se puedan utilizar los recursos de estas tecnologías en la mejor medida posible. Por otro lado, el desarrollo de una aplicación multiplataforma (ya sea una Web App, alguna de sus variantes como PWA, o bien una solución entregada por un Framework como Kivy) también es una opción que garantiza la accesibilidad del usuario al sistema siempre y cuando posea conexión a internet.

## 2. Evaluación y Selección

---

En cuanto al uso de las herramientas de Software para el manejo simbólico, son buenas soluciones de software que nos permitirían emular el comportamiento actual del servidor. Estas opciones sin embargo presentan algunos problemas el más evidente de ellos es la naturaleza de código cerrado del sistema y su licenciamiento. Esto trae las consecuencias de que no sabemos cómo estos programas están manejando la información tras bambalinas por lo que su comunicación y formateo de información para su uso en otras aplicaciones podría ser limitada y difícil de compatibilizar. Además, debido a su tipo de licenciamiento, hay que pagar una cierta suma para trabajar con ambos softwares, y puesto que el servidor por ahora estará alojado en línea eso volvería el uso del software ya mencionado aún más complicado.

Una aplicación nativa en Android, cuyo desarrollo correría en paralelo con el de la página web. Esto habría implicado programar 2 veces el lado del cliente, lo cual tomaría más tiempo y recursos, sin embargo, la experiencia para los usuarios móviles sería más amena y diseñada especialmente para este dispositivo.

Con esta idea en mente, se decidió desarrollar una PWA para la aplicación cliente, aprovechando sus características multiplataforma y la implementación en algún futuro del proyecto de operaciones offline. La comunicación de esta aplicación se da con un servidor en tipo API que utiliza SymPy para el manejo de expresiones simbólicas y con ello se construyen las soluciones de la ecuación diferencial.



## 3. Maquetado e Interfaces

---

### Welcome - mobile screen

Cuenta con una imagen (logo), 2 botones (iniciar sesión y crear sesión) y 2 textos planos.



Figura 1: Maquetado de Pantalla de inicio

## Sign in - mobile screen

Cuenta con 2 campos de texto (para ingresar usuario/email y contraseña) estos campos de texto son del tipo especificado para que solo se pueda poner un correo y la contraseña se vea en asteriscos, también cuenta con un botón para confirmar los datos y un botón para regresar.

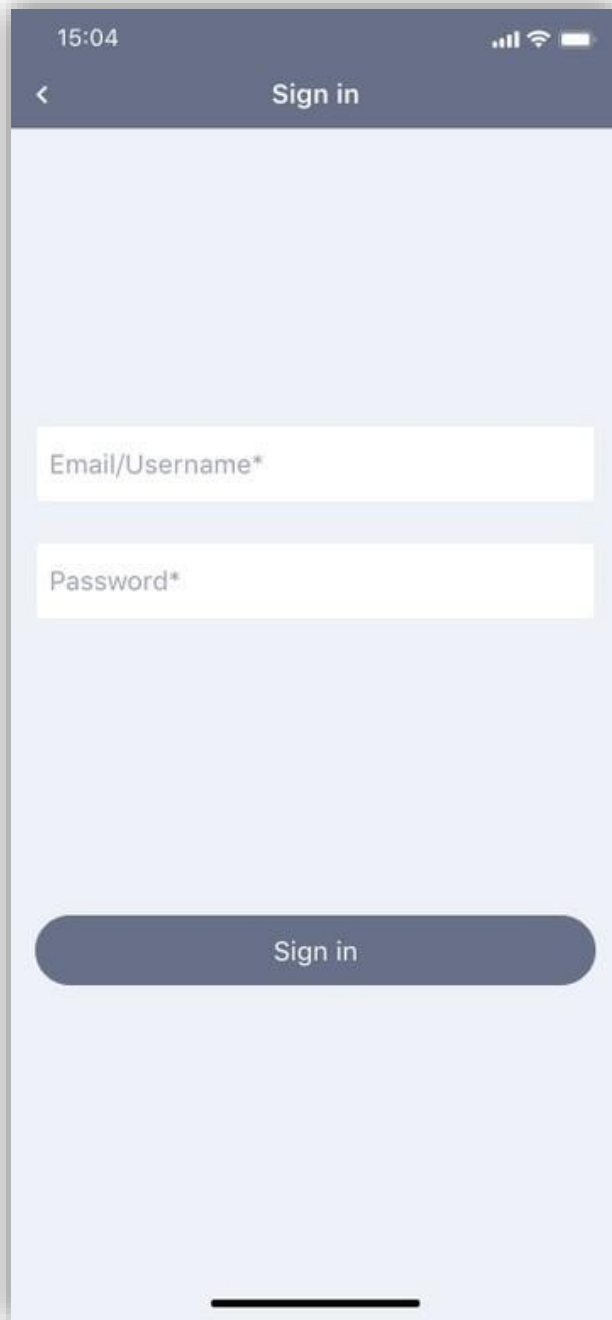
A mobile screen mockup of a 'Sign in' page. The screen has a light blue background. At the top, there is a dark blue header bar with a back arrow on the left, the text 'Sign in' in the center, and status icons (time 15:04, signal, Wi-Fi, battery) on the right. Below the header, there are two white text input fields. The first field is labeled 'Email/Username\*' and the second is labeled 'Password\*'. Below the input fields, there is a dark blue rounded rectangular button with the text 'Sign in' in white. At the very bottom of the screen, there is a thin black horizontal line representing the home indicator.

Figura 2: Maquetado de Pantalla de inicio de sesión

## Sign up - mobile screen

Cuenta con 4 campos de texto (usuario, email, contraseña, verificar contraseña) cada uno de su debido tipo, un scroll con las opciones de estudiante o investigador, una checkbox para aceptar recibir correos, un botón para confirmar los datos y otro para regresar.

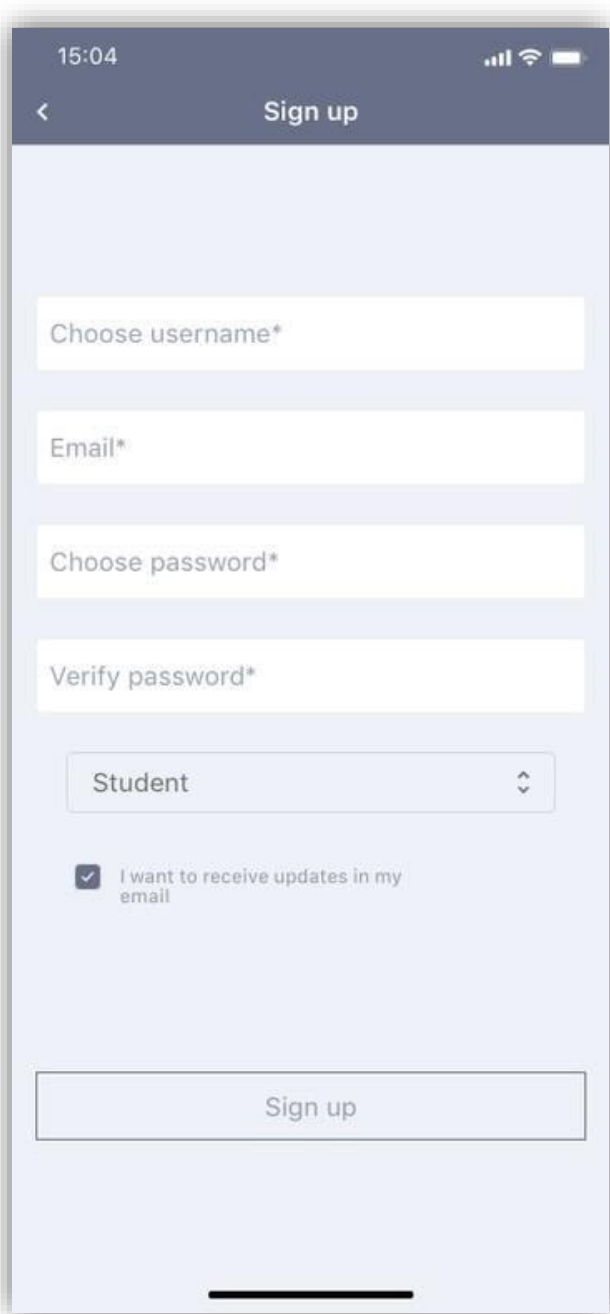
A mobile app wireframe for a 'Sign up' screen. The screen has a dark blue header with a back arrow, the title 'Sign up', and status bar icons (time 15:04, signal, Wi-Fi, battery). Below the header are four white text input fields with labels: 'Choose username\*', 'Email\*', 'Choose password\*', and 'Verify password\*'. Below these is a dropdown menu currently showing 'Student'. Under the dropdown is a checked checkbox with the text 'I want to receive updates in my email'. At the bottom is a large white button with the text 'Sign up'. The entire form is set against a light blue background.

Figura 3: Maquetado de Pantalla de crear cuenta

## Menu - mobile screen

Se sobrepone a las vistas de ecuación, configuración e historial, cuenta con una imagen (foto de perfil), 2 textos (nombre y correo), y 3 textos que actúan como botones (ecuación, settings, historial) además de un botón para cerrar sesión.

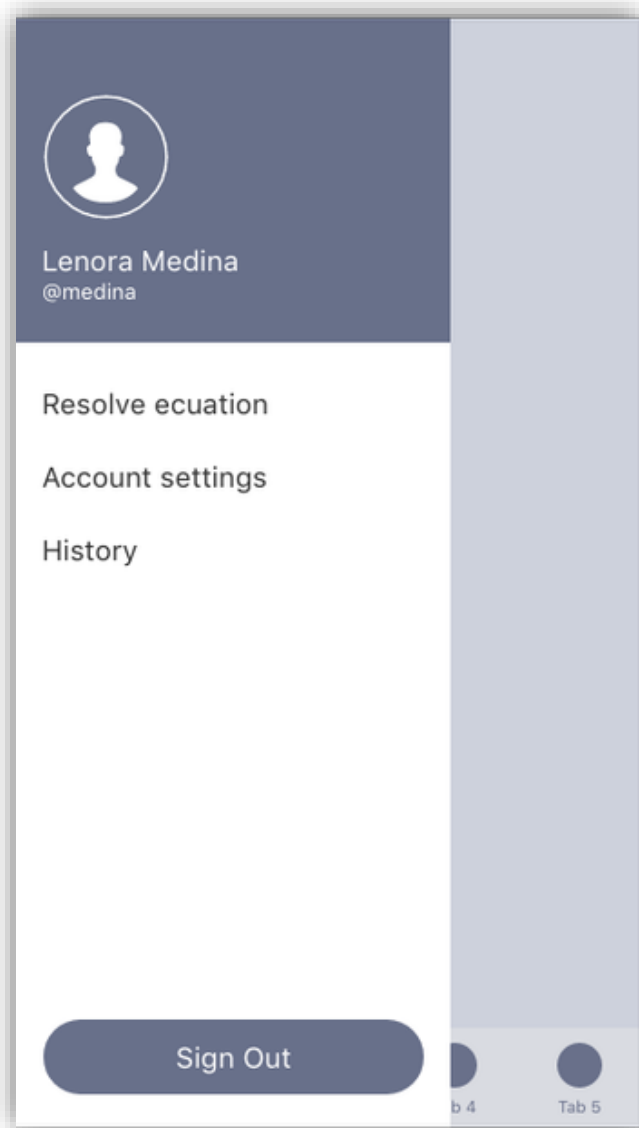


Figura 4: Maquetado de Pantalla de menú principal

## Equation – mobile screen

Cuenta con 3 botones (imagen, dibujar, teclado) y un botón para regresar.



Figura 5: Maquetado de Pantalla de resolver ecuación

## Image – mobile screen

Cuenta con un botón remarcado (imagen) que se puede seleccionar para salir volver a elegir otro método, y 2 botones (cámara y galería) además de otro botón para volver.

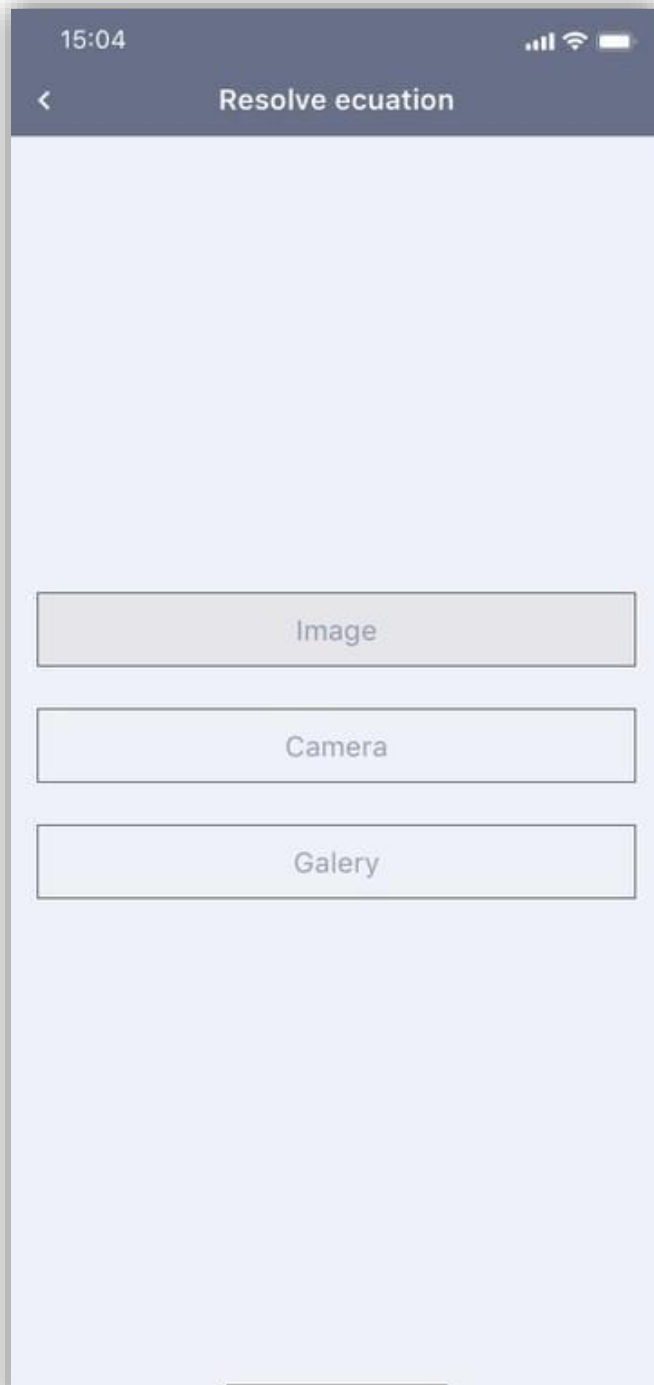


Figura 6: Maquetado de Pantalla de ingresar imagen

## Draw – mobile screen

Cuenta con 4 botones (dibujar, anterior paso, cancelar y aceptar) además de un espacio para dibujar con el dedo

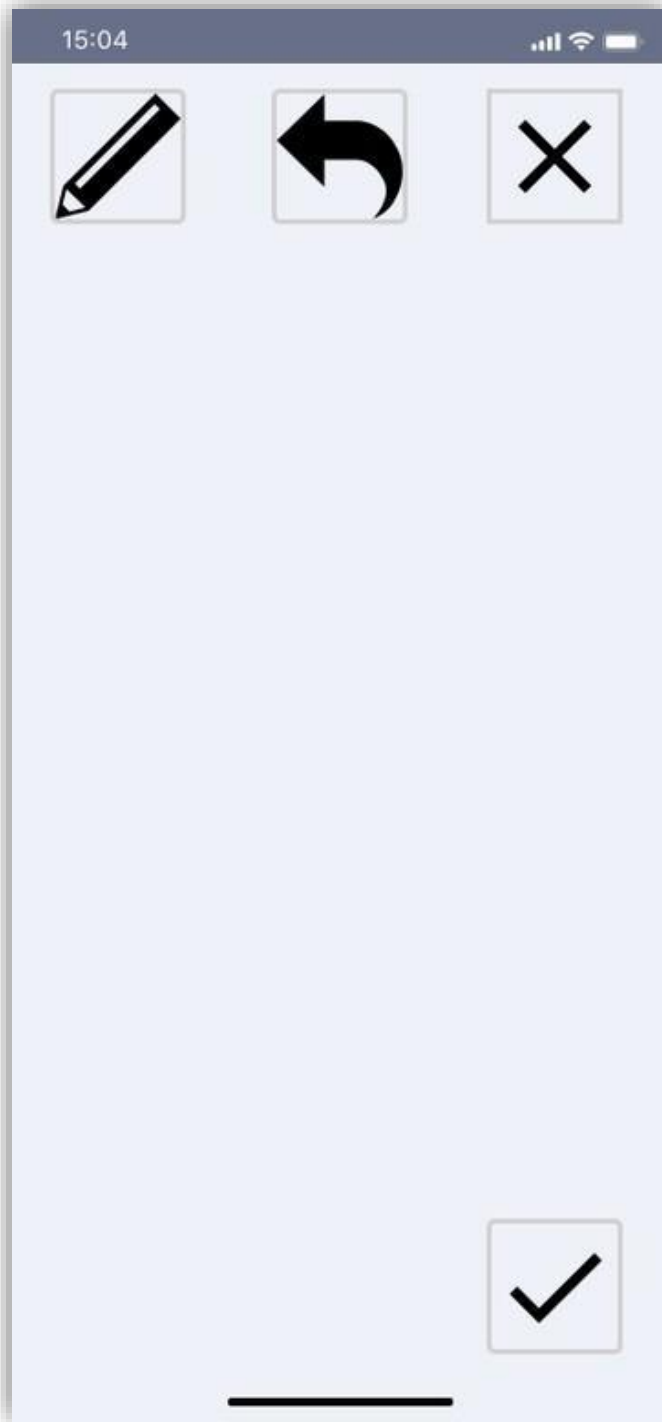


Figura 7: Maquetado de Pantalla de dibujo

## Keyboard – mobile screen

Cuenta con un teclado científico especializado para esta aplicación, un texto que se va modificando conforme se escriba con el teclado y un botón para volver

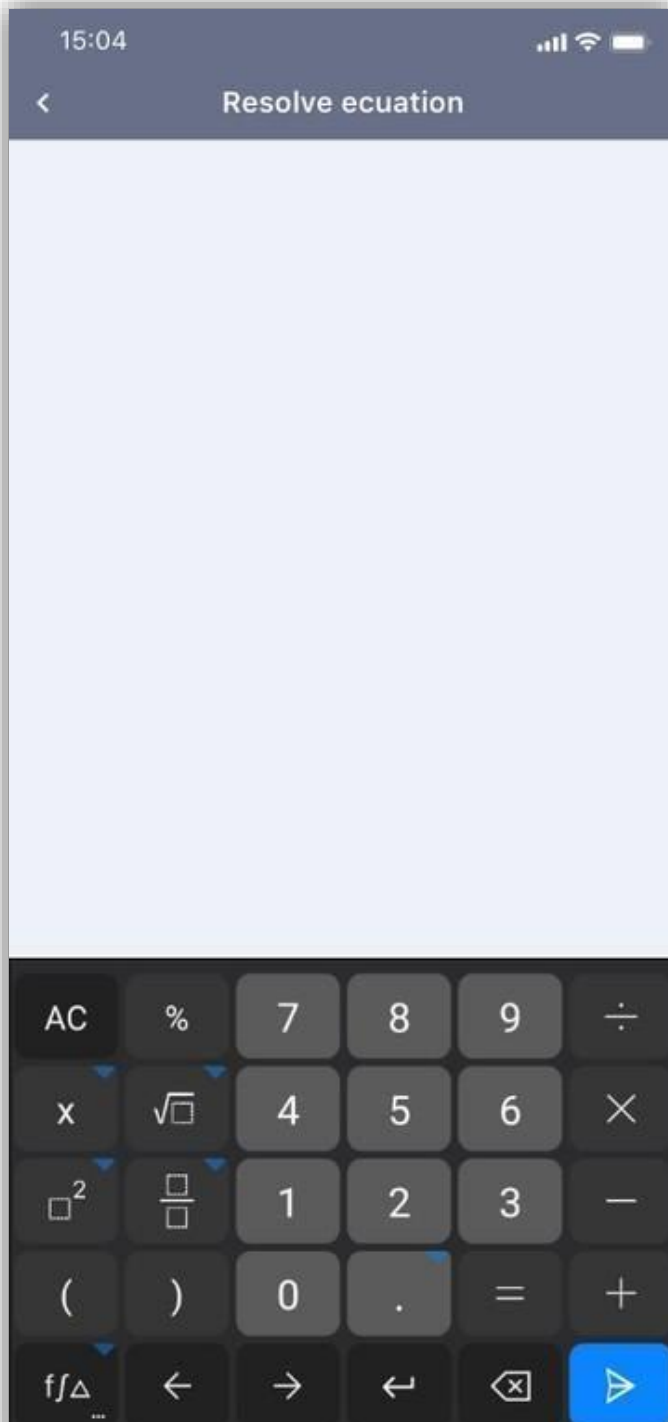
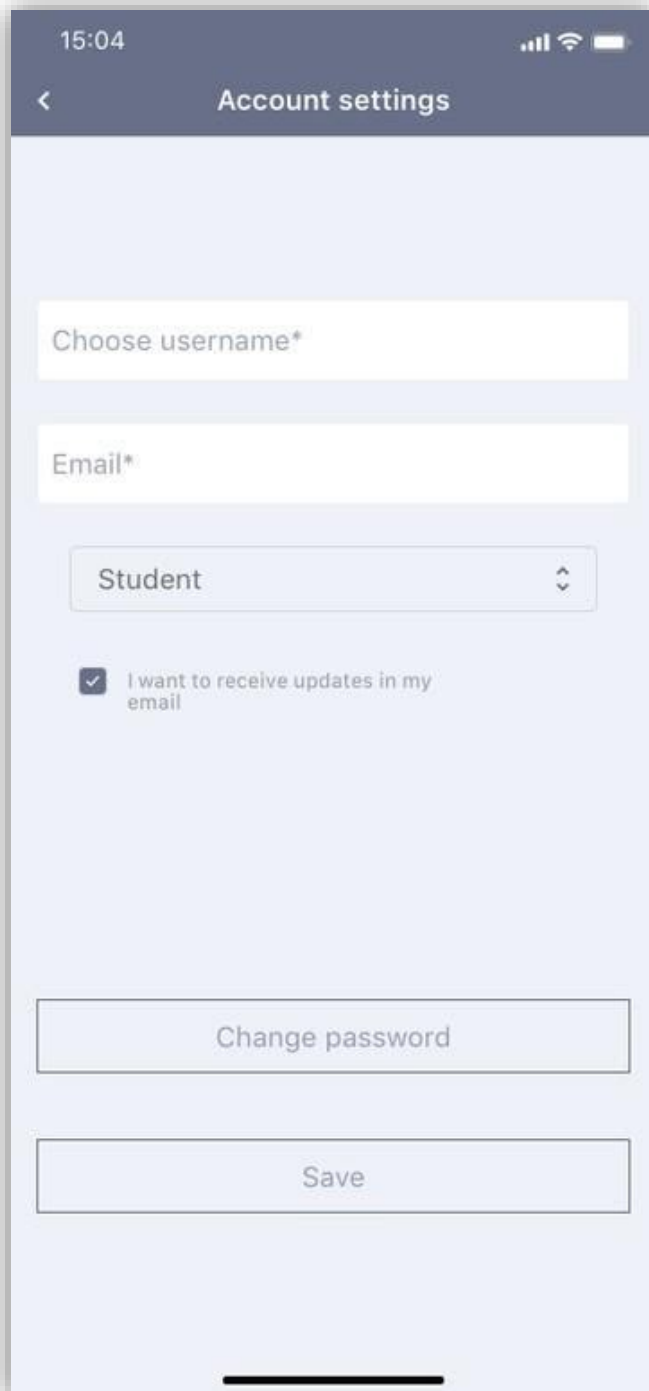


Figura 8: Maquetado de Pantalla de teclado



## Account settings – mobile screen

Cuenta con 2 campos de texto (usuario y email), un scroll (estudiante o investigador), una checkbox (se desea recibir correos), y 2 botones (cambiar contraseña y salvar) además de un botón para volver.

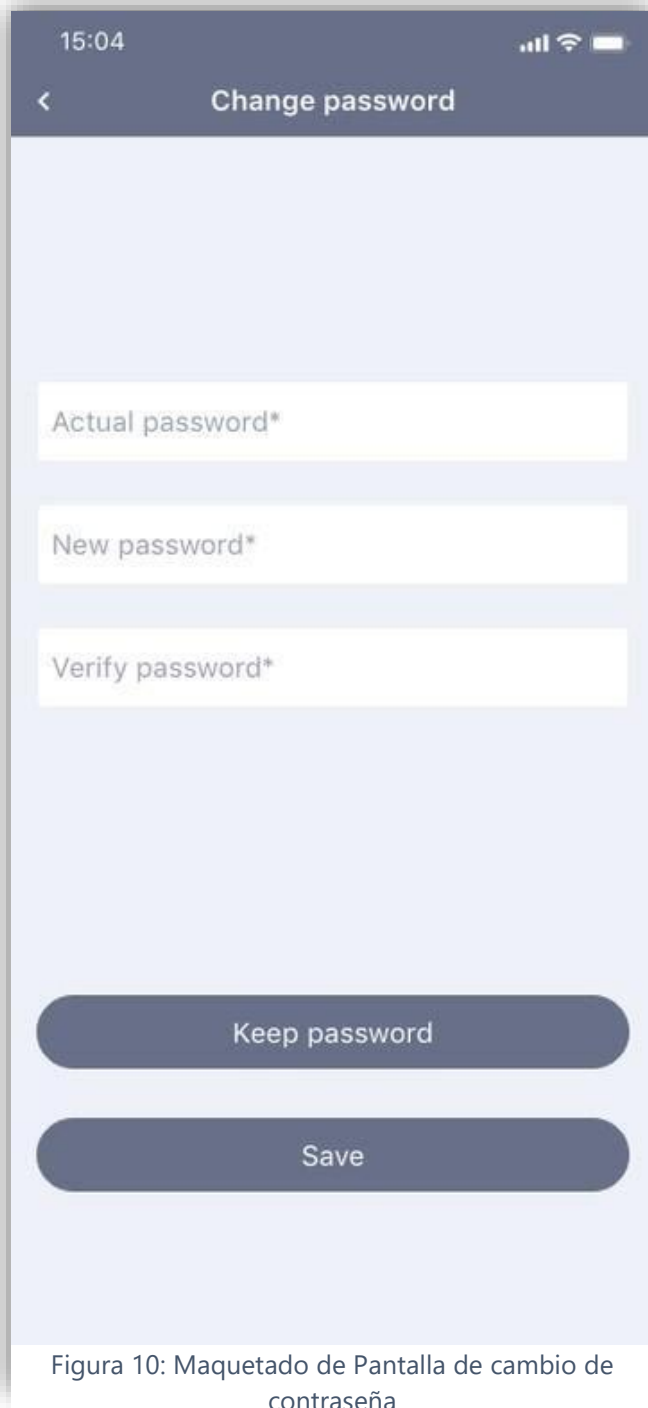


The image shows a mobile app mockup of the 'Account settings' screen. At the top, there is a status bar with the time '15:04' and signal icons. Below it is a dark blue header with a back arrow and the title 'Account settings'. The main content area is light blue and contains the following elements from top to bottom: a text input field with the placeholder 'Choose username\*', another text input field with the placeholder 'Email\*', a dropdown menu currently showing 'Student' with a double arrow icon, a checkbox that is checked with the label 'I want to receive updates in my email', a button labeled 'Change password', and a button labeled 'Save'. At the very bottom, there is a black horizontal bar representing the mobile home indicator.

Figura 9: Maquetado de Pantalla de configuración

## Password – mobile screen

Cuenta con 3 campos de texto (contraseña actual, nueva contraseña y confirmar contraseña) y 2 botones (quedarse con la anterior contraseña y guardar) además de un botón para volver.



## History – mobile screen

Cuenta con una lista con las ecuaciones que se hacen, cada elemento de la lista cuenta con 2 textos (la ecuación y la fecha en la que se ingresó), 2 botones (para fijar la ecuación y para eliminar la ecuación) y un botón para volver.

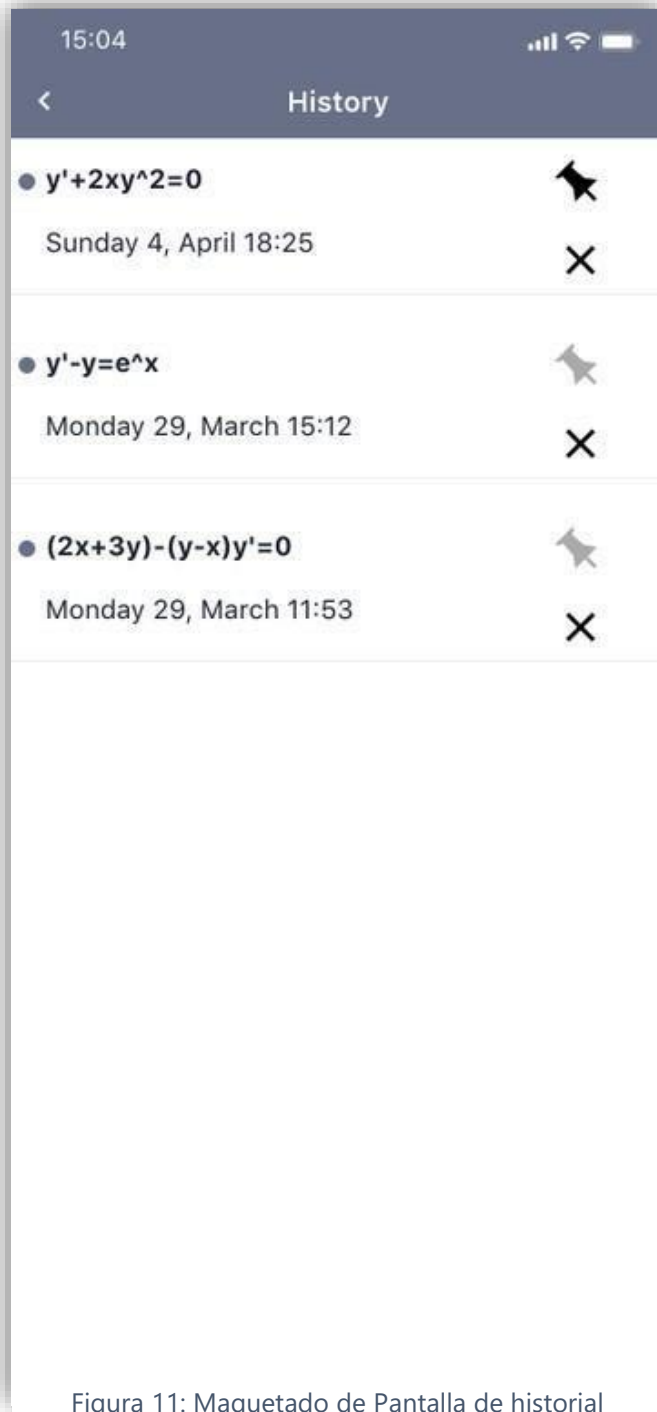


Figura 11: Maquetado de Pantalla de historial

## Welcome – desktop screen

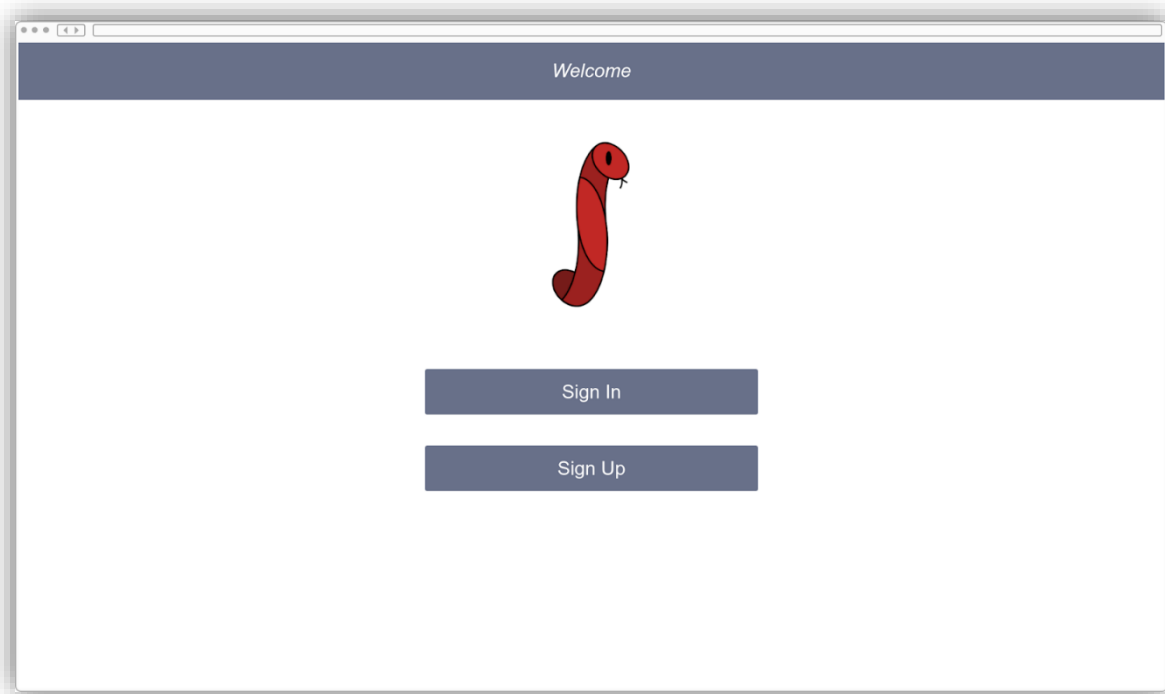


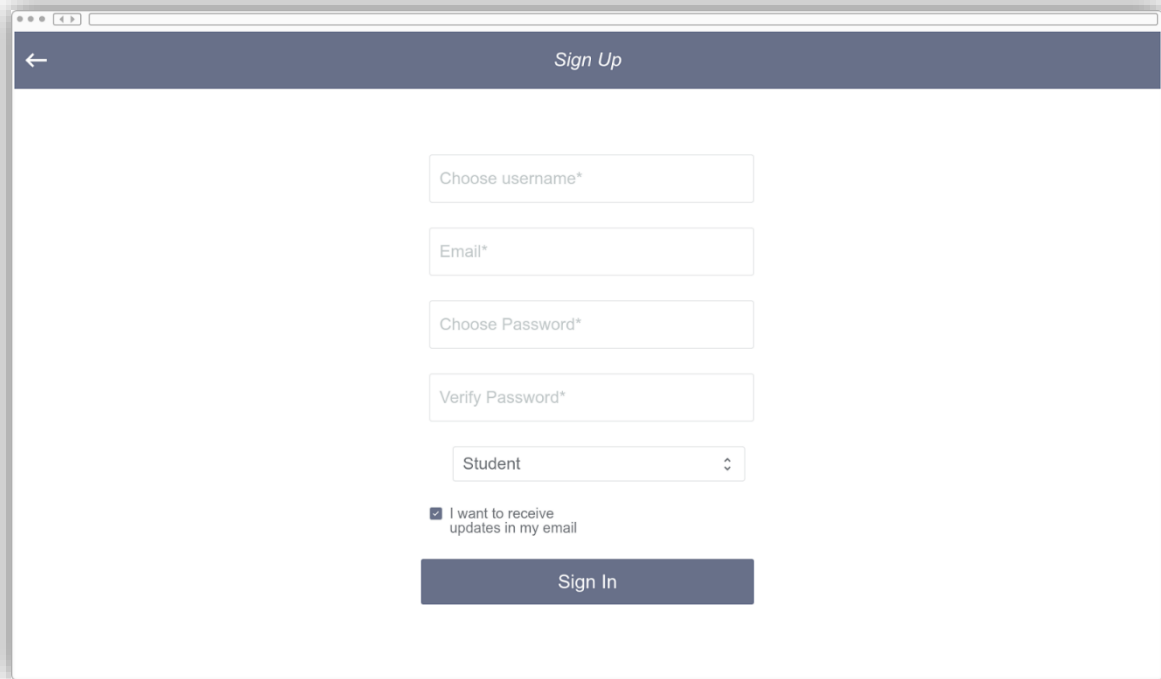
Figura 12: Maquetado de Pantalla de inicio (escritorio)

## Sign in – desktop screen



Figura 13: Maquetado de Pantalla de inicio de sesión (escritorio)

## Sign up – desktop screen



A desktop browser window mockup showing a 'Sign Up' form. The browser's address bar is empty. The page has a dark blue header with a back arrow on the left and the text 'Sign Up' in the center. The form is centered on the page and consists of five input fields: 'Choose username\*', 'Email\*', 'Choose Password\*', 'Verify Password\*', and a dropdown menu labeled 'Student'. Below the dropdown is a checkbox labeled 'I want to receive updates in my email'. At the bottom of the form is a dark blue button labeled 'Sign In'.

Figura 14: Maquetado de Pantalla de crear cuenta (escritorio)

## Menu – desktop screen

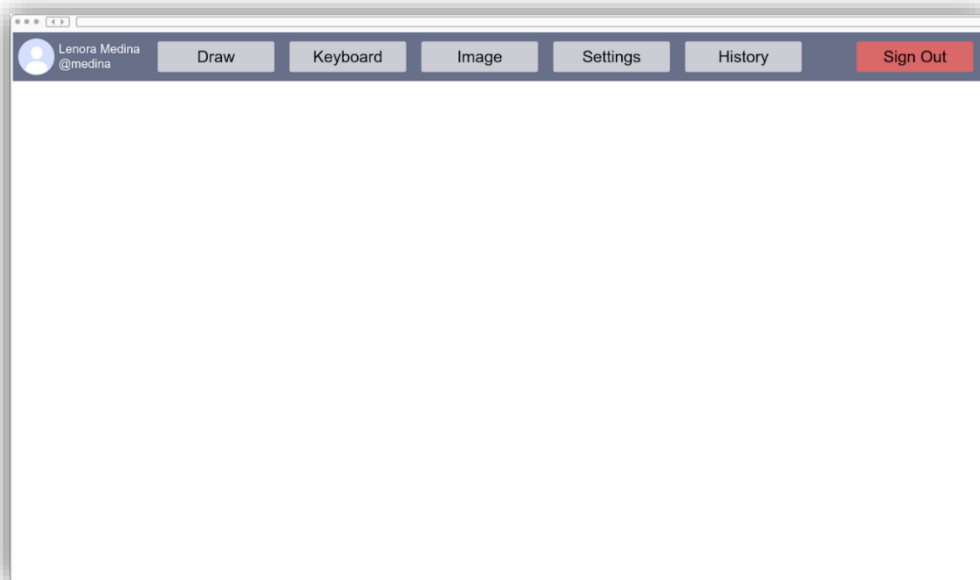


Figura 15: Maquetado de Pantalla de menú principal (escritorio)

## Draw – desktop screen

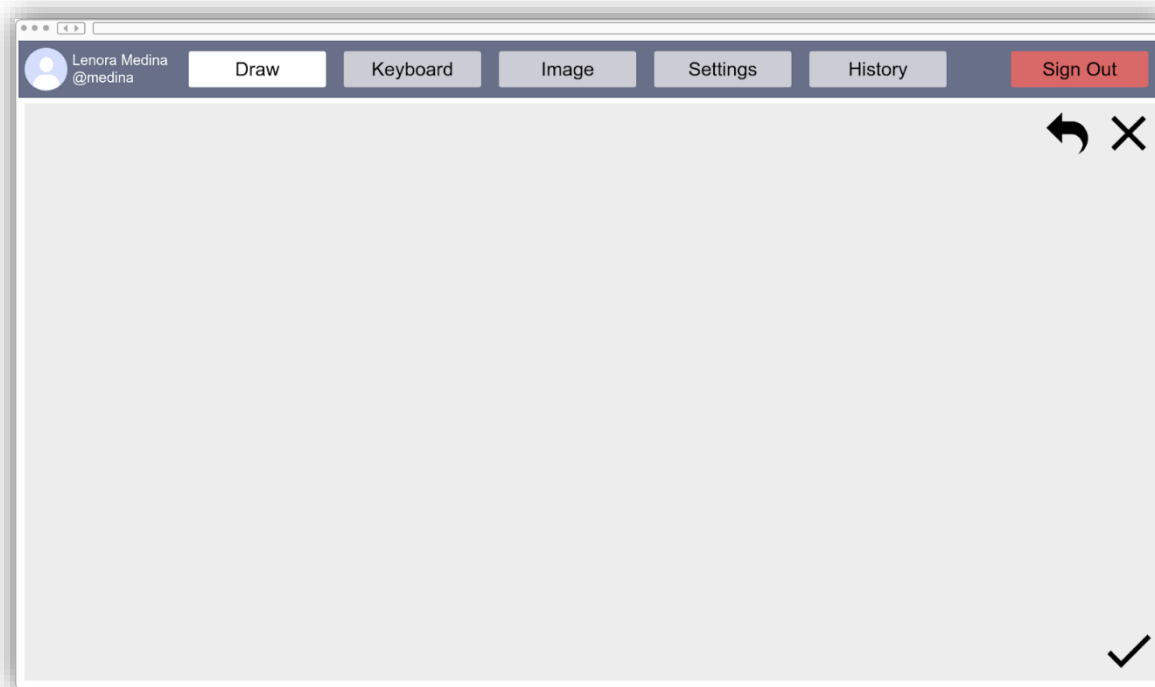


Figura 16: Maquetado de Pantalla de dibujo (escritorio)

## Keyboard – desktop screen

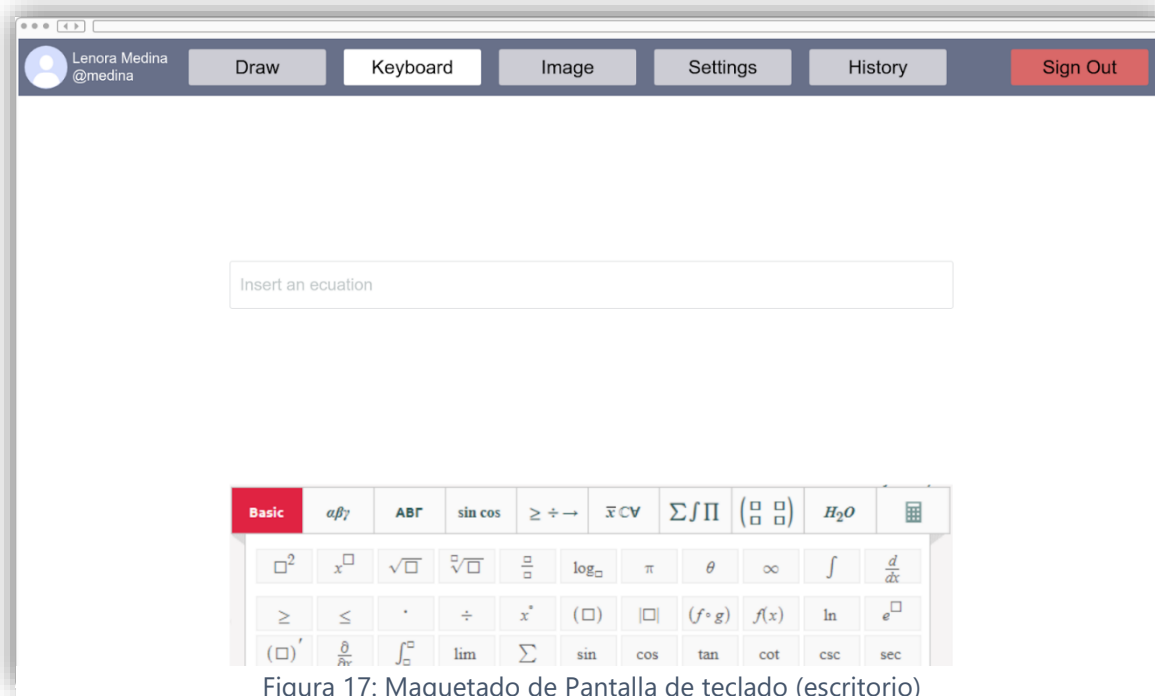


Figura 17: Maquetado de Pantalla de teclado (escritorio)

## Image – desktop screen

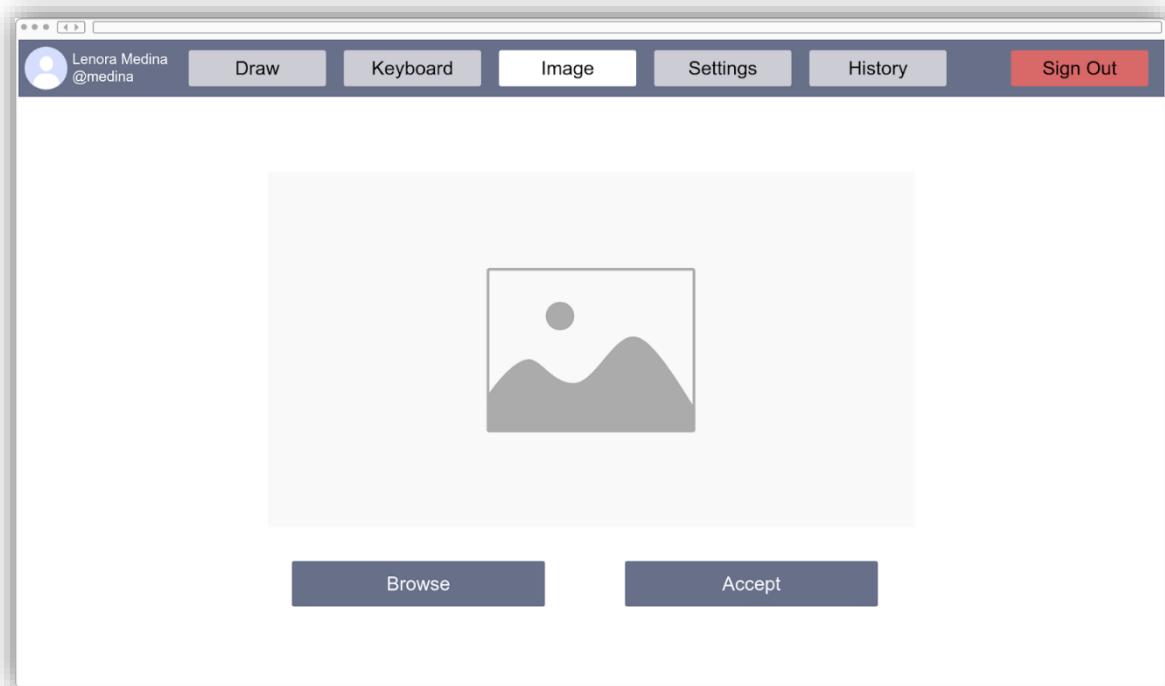


Figura 18: Maquetado de Pantalla de imagen (escritorio)

## Account settings – desktop screen

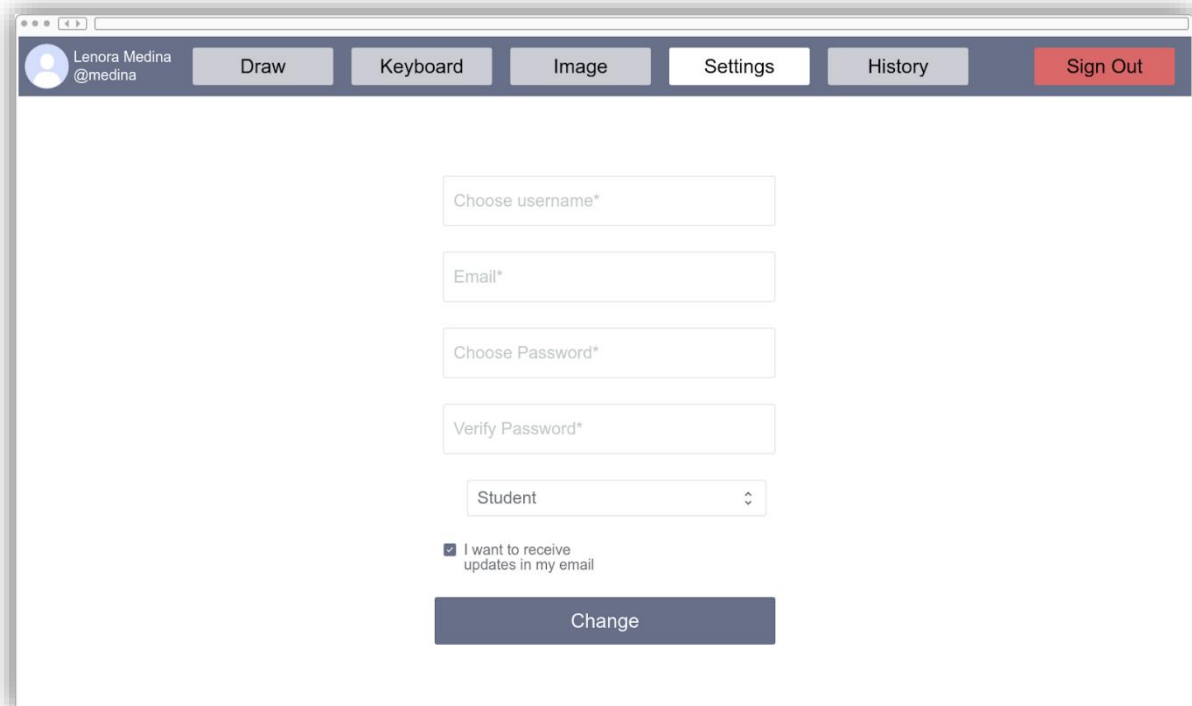


Figura 19: Maquetado de Pantalla de configuración (escritorio)

## History – desktop screen

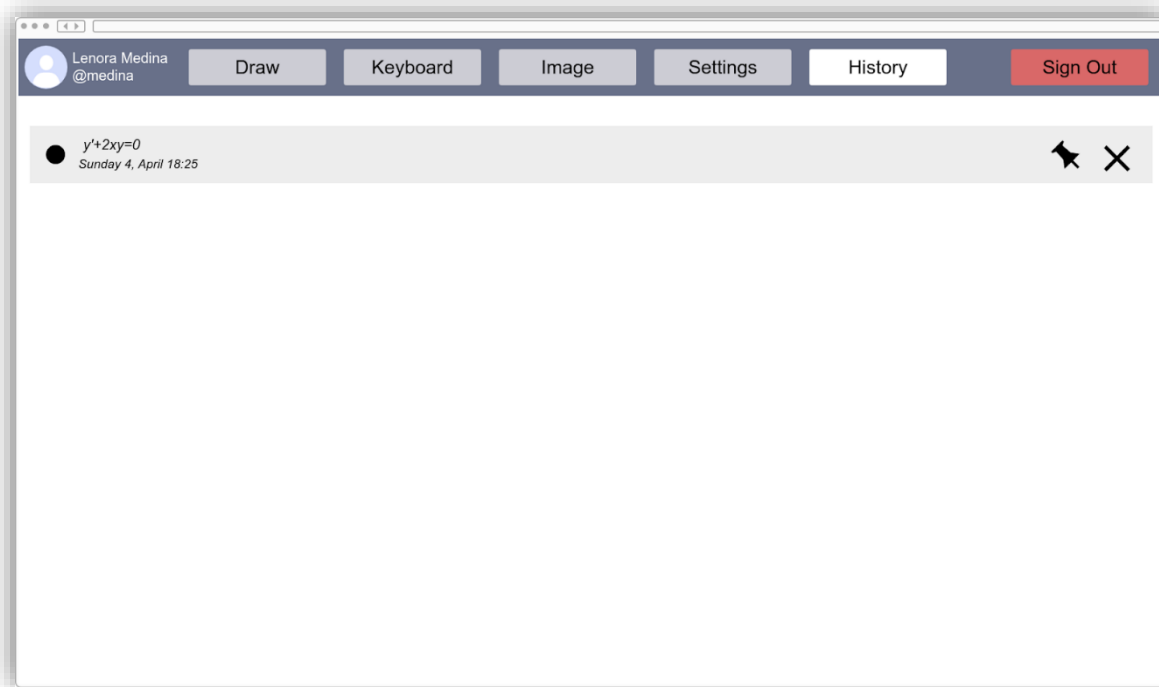


Figura 20: Maquetado de Pantalla de historial (escritorio)



# Capítulo III:

## Descripción del ODE Solver

# 1. Describiendo el proyecto

---

## 1.1. Panorama general

La parte del sistema que se pretende desarrollar puede describirse como la interacción entre un cliente que podrá ser cualquiera de las presentaciones de la aplicación y un servidor encargado de brindar el soporte matemático necesario para resolver la ecuación diferencial.

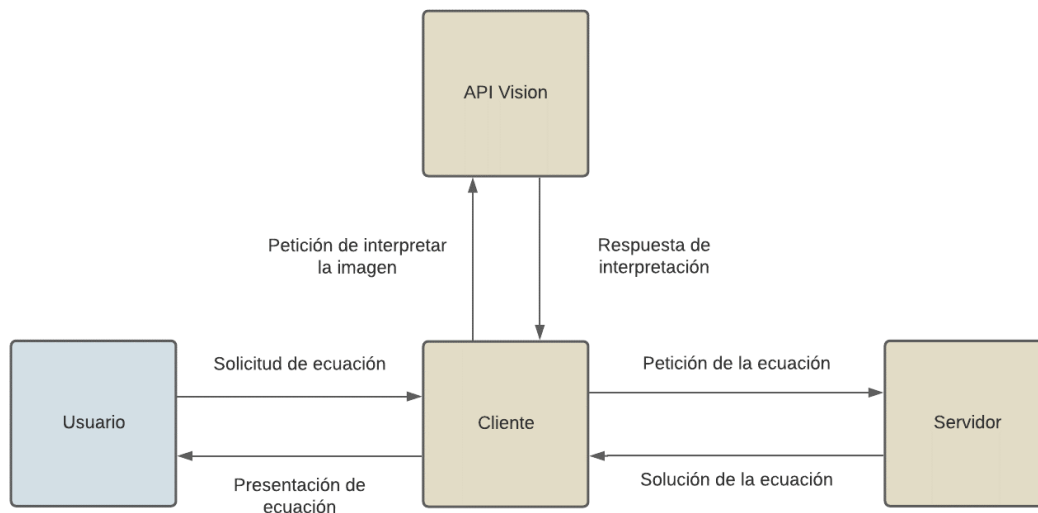


Figura 21: Diagrama de interacción Usuario - Cliente - Servidor

## 1.2. Interacción modular del cliente

La aplicación cliente puede estudiarse a través de su estructura modular:

- **Módulo de recepción:** Este módulo se encarga de recibir la información del usuario (por cualquiera de los medios de entrada permitidos) y guardar esta información para su posterior interpretación.
- **Módulo de interpretación:** Este módulo se encarga de traducir la información contenida en la entrada (que puede ser vía imagen, texto en

pantalla o texto en un formato matemático) a una entidad informática que puede ser clasificada como un tipo de ecuación diferencial según sea el caso. Se encarga de validar que la entrada pueda ser interpretada como una ecuación diferencial.

- **Módulo de formateo:** Este módulo se encarga de generar un formato a la entrada con la intención de que pueda ser interpretada por el servidor de manera adecuada. Junto con el módulo de despliegue representan los "traductores" entre una cadena de texto ordinaria y una expresión matemática. Se pretende que el formato utilizado sea Math LaTeX debido a su extensivo uso, las comodidades de operación que ofrece y la gran cantidad de documentación que existe apoyándose de expresiones en este formato.
- **Módulo de despacho:** Este módulo se encarga de enviar una solicitud al servidor seleccionado para resolver la ecuación diferencial detectada con la información pertinente para que pueda ser interpretado en el servidor. Se encarga también de codificar la ecuación diferencial en un formato que el servidor pueda entender para darle solución.
- **Módulo de retención:** Este módulo se encarga de retener la respuesta generada por el servidor de la petición, porque es en este módulo donde la respuesta a la ecuación diferencial entra al sistema. Su principal función es mantener la conexión con el servidor mientras genera y envía la solución de la ecuación diferencial hacia el sistema.
- **Módulo de despliegue:** Este módulo se encarga de expandir la respuesta obtenida desde el servidor en entidades informáticas que pueden ser manejadas en el sistema para generar una versión menos abstracta de la solución que la recibida por el servidor.
- **Módulo de presentación:** Este módulo se encarga de exponer al usuario la solución de la ecuación diferencial en una presentación entendible por él. Su función incluye el interpretar la solución obtenida en el módulo anterior en elementos visuales que formarán la solución mostrada al usuario.

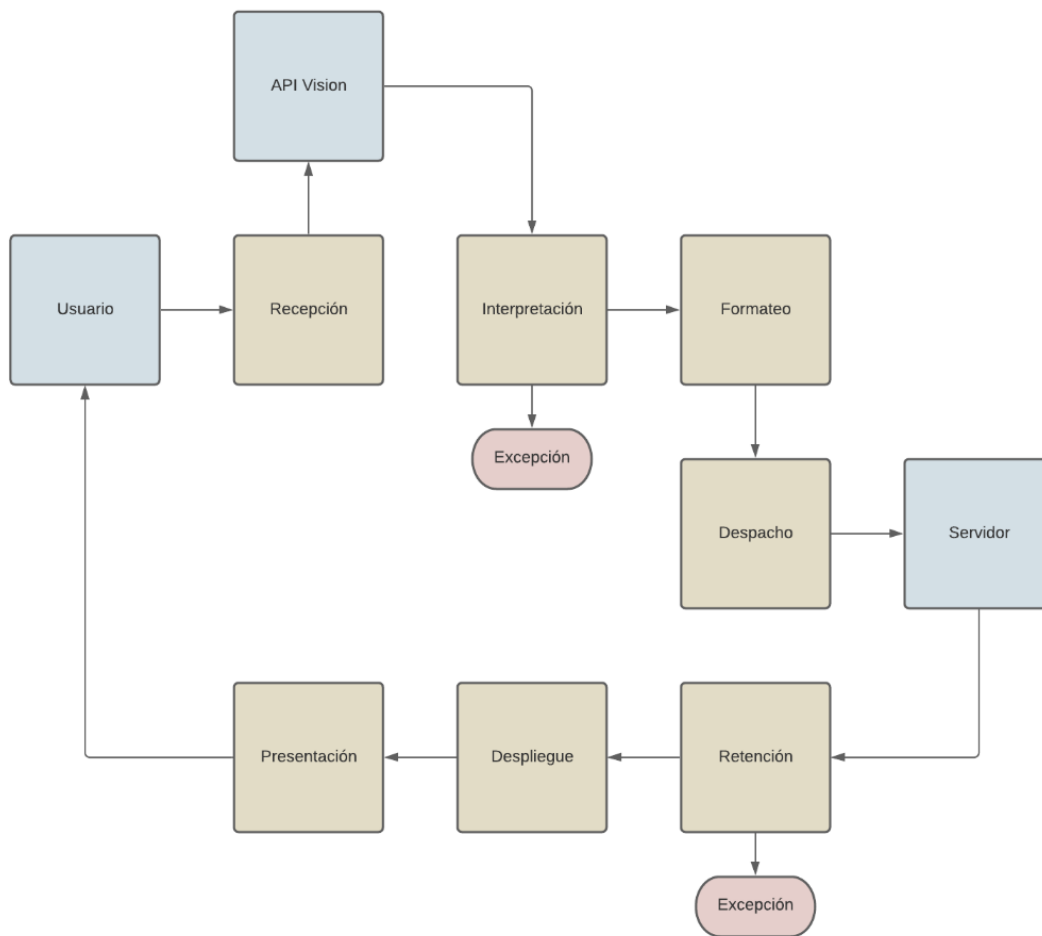


Figura 22: Diagrama de módulos del cliente

En el diagrama siguiente podemos resumir la interacción y orden de aparición de los módulos durante la solución de la ecuación (los bloques **Usuario**, **API Vision** y **Servidor** son los módulos externos y el resto son los módulos del cliente). Si hablamos del estado de la información en cada uno de los módulos, podemos decir que:

- De **Usuario** hacia **Recepción** la información se manda a través de una imagen en formato jpg (en los primeros dos tipos de ingreso [ver módulo de recepción]).
- De **Recepción** a **API Vision** la información se manda a través de una imagen mapeada o bitmap.

- De API Vision a Interpretación la información se manda a través de una cadena de caracteres.
- De Interpretación a Formateo la información se manda a través de una cadena de caracteres.
- De Formateo a Despacho la información se manda a través de una cadena de caracteres con formato.
- De Despacho a Servidor la información se manda a través de un objeto JSON.
- De Servidor a Retención la información se manda a través de un objeto JSON.
- De Retención a Despliegue la información se manda a través de un objeto JSON.
- De Despliegue a Presentación la información se manda a través de una lista de cadenas de caracteres con formato.
- De Presentación a Usuario la información se manda a través de imágenes vectorizadas (formato svg o equivalente) y cadenas de caracteres que se presentan en pantalla como texto plano.

En cuanto a las excepciones que se indican en el diagrama son aquellos eventos que frenan por completo el proceso de resolución y mandan una de las respuestas de error explicadas anteriormente según sea el caso. En el caso de la interpretación, su excepción corresponde a no encontrar una ecuación diferencial dentro de lo que fue interpretado en la imagen enviada a la API Vision; mientras que en el caso de la retención hace referencia a encontrar un error marcado dentro del objeto JSON que devuelve el servidor.

## 1.3. Módulo de recepción

Al usuario se le presenta un menú en donde puede seleccionar la manera en que va a interactuar con el sistema. Para todas las plataformas se pretende que las 3 selecciones sean posibles: Por una fotografía, por escritura en la pantalla y por texto.

### Ingreso por fotografía

Para que el usuario ingrese una fotografía se tienen dos opciones en el caso de un dispositivo móvil: tomar una fotografía o bien subir una fotografía desde la galería. En el caso de las versiones de escritorio solo se podrá realizar por selección desde la

galería. En principio la fotografía puede ser cualquiera, aunque el sistema se encargará de estandarizar el tamaño y la resolución de la imagen ingresada; esto con la intención de facilitar la interpretación del texto en la imagen.

Esta interacción se lleva a cabo por clic: Uno para seleccionar la opción de fotografía, otro para el tipo (galería o cámara), otro para tomar foto o seleccionar imagen, y un último de confirmación. Una vez seleccionada la imagen, se le mostrará una previsualización de la imagen al usuario con el fin de que pueda confirmar la calidad de su imagen y que el sistema haya capturado de la mejor manera la ecuación después de la estandarización comentada. Esta interacción es igual en la versión web.

## Ingreso por texto en pantalla

Para que el usuario ingrese la ecuación por texto en pantalla se tiene que, una vez seleccionada esta vía de entrada por clic, se presentará una región (a manera de lienzo) en donde el usuario podrá “escribir” su ecuación. En el caso de los dispositivos móviles (y en general con pantalla táctil) el usuario hará esto por medio de arrastrar su dedo para dibujar los diferentes símbolos que conforman a la ecuación diferencial. Para el caso de dispositivos de escritorio no táctiles el usuario deberá mantener presionado el botón de clic en el ratón y arrastrar para generar los símbolos de la ecuación.

Durante todo el ingreso de la ecuación, se le presenta un botón para confirmar que ha concluido de escribir su ecuación en la pantalla. Una vez presionado el botón, el sistema genera una imagen a partir de lo dibujado y procede igual que en el caso anterior.

## Ingreso por texto

Para que el usuario ingrese el texto de su ecuación, se le presentará una pantalla tipo calculadora donde por medio de botones el usuario puede ir ingresando los elementos que conforman a su ecuación diferencial. El menú de botones contiene todos los símbolos matemáticos que conforman a todas las ecuaciones diferenciales que puede resolver el sistema. A medida que el usuario ingresa con los botones su ecuación diferencial, se le muestra una vista en tiempo real de su ecuación para que pueda comparar con el ejercicio que se desea resolver. Durante todo el tiempo de ingreso hay un botón que le permite al usuario dar por terminado el ingreso y pasar

a la siguiente etapa de resolución. Este proceso de ingreso es análogo para todas las plataformas del sistema.

(**Nota:** Debido a que este proceso resulta mucho más controlado con los otros dos se pretende que sea capaz de omitir los siguientes dos módulos, ya que estos tienen sentido solo cuando hablamos del texto interpretado en una imagen y todas las libertades que existen en ello; con esto decimos que el ingreso por texto pasa directamente al módulo de despacho al generar una cadena con formato adecuado al tiempo que el usuario ingresa su ecuación por medio de los botones restringidos con los que interactúa en este apartado).

A excepción del último método de ingreso que fue descrito, se realiza una representación de la imagen por medio de un mapeo que obtiene la información más representativa de regiones de pixeles conocido como Bitmap. Este mapa es enviado hacia una API capaz de interpretar el texto contenido en ella por medio de un exhaustivo análisis con redes neuronales.

## 1.4. Módulo de interpretación

Una vez que la API regresa en una cadena de caracteres el texto que pudo ser reconocido en la imagen, el módulo de interpretación entra en acción. Mediante el uso de comparaciones de formato condicional (esto es, buscar que una determinada cadena posea ciertas reglas de formato) se comprueba si dicha cadena corresponde o no a una ecuación diferencial. Las comparaciones realizadas buscan que exista una consistencia matemática en lo que el usuario ingresó en el sistema.

- Entre los elementos que se buscan durante las comparaciones de formato condicional son:
- Exista exactamente un signo de igualdad
- Antes y después del signo de igualdad existen caracteres
- No se han incluido caracteres especiales
- A cada operador de jerarquía (paréntesis, por ejemplo) tenga exactamente una apertura y una clausura y específicamente en ese orden
- Cada operador se encuentra entre exactamente dos expresiones
- Cada letra en la expresión solo puede ser acompañada por exactamente un número posterior (exponente) y uno predecesor (coeficiente)
- Las derivadas se connotan por medio de la notación de primas (apóstrofes)

- No se expresan funciones de manera explícita ( $f(x)$ , por ejemplo).
- Deben de existir exactamente dos letras en la ecuación y deben ser exactamente 'x', 'y'. Pueden aparecer más de una vez
- Solo se permiten derivadas sobre la letra 'y', pues se asume que 'y' es una función de 'x'

Entre otros más específicos. En particular, podemos resumir en que "Solo se aceptan ecuaciones diferenciales cuya única incógnita es  $y(x)$ ".

## 1.5. Módulo de formateo

Con la expresión validada como una ecuación con expresiones matemáticas completas se espera que la ecuación pueda ser representada en un formato de escritura matemática como LaTeX. Este módulo se encarga de tomar la cadena interpretada y darle el conocido formato LaTeX con la intención de representar de una manera clara la jerarquía y disposición de los símbolos matemáticos en el texto interpretado. Para este paso se pretende realizar una traducción explícita de cada una de las expresiones por medio de comparaciones forzadas en cada uno de los elementos de la cadena. Esas comparaciones se realizan buscando palabras y caracteres clave que puedan aparecer en la expresión original; estos elementos, una vez encontrados, se reproducen con su equivalente connotación en LaTeX, formato que se basa la anidación de operadores matemáticos (como "funciones") y sus respectivos parámetros.

Se puede encontrar un catálogo completo de los símbolos que pueden ser representados en notación LaTeX y su interpretación matemática en el siguiente enlace:

[https://oeis.org/wiki/List\\_of\\_LaTeX\\_mathematical\\_symbol](https://oeis.org/wiki/List_of_LaTeX_mathematical_symbol)





## 1.6. Módulo de despacho

Este módulo posee dos etapas de operación. En la primera o fase activa se encarga de encapsular la expresión con el formato LaTeX en un objeto JSON con la intención de poder ser transmitido por medio de una solicitud hacia el servidor. El proceso de serialización JSON se lleva a cabo por medio de los diferentes soportes que ofrecen los entornos de desarrollo para hacerlo.

Una vez se ha mandado la solicitud al servidor se mostrará un mensaje indicando que se está procesando su ecuación (decimos que el módulo de despacho pasó a su fase pasiva). Mientras se encuentra en este periodo el servidor realiza toda la parte lógica de la interpretación y resolución de la ecuación. Este proceso puede tardar a lo más 30 segundos (dependiendo de la demanda del servidor y la conexión del usuario). Durante esta etapa el usuario no puede realizar ninguna acción con el sistema. La intención es que este procedimiento se haga en background, de modo que, aunque el usuario ponga en pausa al sistema la comunicación con el servidor permanece y se sigue buscando la resolución.

Durante este periodo de espera, el sistema puede regresar al usuario alguna de las siguientes advertencias indicando que hubo un error durante el proceso de resolución de la ecuación:

- No se detectó una ecuación diferencial
- Se perdió la conexión con el servidor
- No se tiene una solución para tu ecuación diferencial
- Se excedió el tiempo de espera

Independientemente de estas salidas abruptas, al usuario le parece una opción aceptar y regresar al menú inicial de captura de su ecuación diferencial. En caso de que el servidor mande alguna respuesta (favorable o desfavorable), el siguiente módulo comienza con su trabajo y finaliza su participación en el módulo de despacho.

## 1.7. Módulo de Retención

Una vez que se obtiene una respuesta del servidor, el módulo de retención del cliente comienza a operar. Se encarga de verificar que no exista una señal de error dentro del empaquetado JSON que acaba de llegar del servidor. La existencia de una señal

de error en el paquete indicaría que algo fue mal en el servidor durante la resolución de la ecuación diferencial y por lo tanto la solución no puede ser presentada. En el caso de encontrar un error se manda una excepción que interrumpe el proceso y finalmente se le muestra al usuario un mensaje describiendo el motivo por el cual hay no pudo resolverse su ecuación.

Es importante recalcar que es este módulo es el encargado de detectar la presencia de excepciones generadas por la respuesta o no respuesta por parte del servidor hacia la aplicación cliente. Independientemente de cuál sea el tipo de excepción generada se le mostrará al usuario una descripción detallada de que fue lo que produjo la excepción en términos que pueda comprender.

Hablando un poco sobre las **pantallas de excepción**, los elementos básicos las componen son los siguientes:

- Nombre de la excepción (el cual se asocia directamente con la fuente que produjo la excepción).
- Descripción detallada de la excepción, la cual a su vez se compone de diferentes elementos dependiendo del tipo de excepción que se haya detectado.
- Recomendaciones para que el usuario para evitar la excepción en peticiones posteriores que sean similares o iguales al servidor. Estas recomendaciones también están ligadas con el tipo de excepción con la que se esté tratando.

Estos tres elementos propios de cada una de las excepciones se presentarán en un formato análogo al que se utiliza para mostrar los pasos para la solución de una ecuación diferencial, de modo que podrían considerarse una especie de “pasos” sin incluir todos los elementos que eso implicaría (definido más adelante).

A continuación, se presentan los diferentes tipos de excepciones con la definición de cada uno de los puntos presentados arriba que deben incluir sus pantallas de excepción:

**Excepción de tiempo:** El servidor tardó más de lo esperado en dar una respuesta para la ecuación diferencial que fue ingresada. Para continuar, se aconseja que siga las siguientes recomendaciones:

- Verifique que su dispositivo tenga una conexión estable a internet. Las conexiones deficientes pueden afectar la comunicación con el servidor.
- Reinicie la aplicación e intente ingresar nuevamente la ecuación.

- Intente la vía de ingreso por teclas (III): es la más sencilla de procesar por parte del servidor.

**Excepción de interpretación:** El servidor no pudo identificar correctamente la entrada como una ecuación diferencial. Para continuar, se aconseja que siga las siguientes recomendaciones:

- Verifique que la ecuación diferencial haya sido escrita correctamente. En caso de utilizar alguna vía de entrada por imagen (I, II) asegúrese de que la ecuación esté completamente capturada y lo más enfocada posible; también verifique que la pantalla de previsualización coincida con la ecuación que desea resolver
- Verifique que el formato de la ecuación diferencial sea el aceptado por la aplicación.
- En caso de que lo anterior no resulte, utilice la entrada por teclas (III): es la más sencilla de identificar por parte del servidor.
- Reinicie la aplicación e intente ingresar nuevamente la ecuación por la vía III.

**Excepción de conexión:** No ha sido posible hacer contacto con el servidor para resolver la ecuación diferencial. Para continuar, se aconseja que siga las siguientes recomendaciones:

- Verifique que su dispositivo esté conectado a internet. Asegúrese, además, de que la conexión sea estable.
- Reinicie la aplicación e intente ingresar nuevamente la ecuación.
- Espere al menos 10 minutos e intente nuevamente. En caso de que aún se encuentre desconectado, comuníquese con el equipo para reportar el caso.

**Excepción de completitud:** El servidor no ha sido capaz de concluir la solución para la ecuación diferencial presentada. El servidor ha podido realizar el siguiente avance en la solución:

[Se inyecta el avance parcial de la solución].

Para continuar, se aconseja que siga las siguientes recomendaciones:

- Utilice la entrada de la ecuación por la vía III; es la entrada más simple de procesar para el servidor.
- Realice el desarrollo de los primeros pasos (algebraicos) que se presentan en la solución parcial encontrada por el servidor e intente nuevamente con la nueva expresión: es posible que en esta ocasión se agreguen los pasos necesarios para dar con la solución de la ecuación.
- Utilice un software externo para concluir con la solución de la ecuación diferencial (se recomienda el uso de Wolfram Alpha para ello).

Las excepciones dentro del servidor que pueden producir una excepción de completitud desde la perspectiva del cliente serán descritas más adelante en el apartado correspondiente al servidor.

## 1.8. Módulo de despliegue

Con el empaquetado JSON ya verificado se espera que el servidor pudo completar el procedimiento y este ha regresado con éxito al cliente. Este pequeño módulo se encarga de pasar este objeto JSON en una lista de pasos (que incluyen texto con formato) que serán mostradas posteriormente al usuario. Para este proceso (deserealización) se utilizarán los diferentes soportes que ofrecen los entornos de desarrollo.

## 1.9. Módulo de presentación

La presentación de resultados es una lista de los diferentes pasos para llegar a la solución del problema. En caso de que no se tenga una lista de pasos (para las ecuaciones más complejas), solo se muestra el resultado. La lista de pasos lleva dos partes: explicación y ecuación que representa al paso en cuestión. El usuario puede interactuar con estos resultados, de manera que puede colapsar la solución para que solo se muestre el desarrollo algebraico o bien que incluya la explicación de cada paso.

Al final de la solución, se le expone una serie de enlaces asociados al tipo de ecuación resuelto que le permiten investigar más a fondo sobre el desarrollo de esas ecuaciones diferenciales en concreto. Estos enlaces están registrados en una base de datos en la cual se almacenarán los detalles más usuales de la aplicación; se describe

más a detalle en la sección de “Base de Datos” de este mismo capítulo. La presentación de resultados es análoga en todas las plataformas. En el caso de utilizar el modo investigador, se le presentará información adicional asociada a la ecuación que el servidor pueda devolver (rango, dominio, gráfica). Esta situación variará según el tipo de ecuación diferencial que se haya ingresado.

Los detalles acerca de que incluye la solución de una ecuación diferencial se exponen en el apartado del servidor en este mismo documento.

## 1.10. Introducción al servidor

El Servidor será quién se encargue de dar solución a la ecuación diferencial introducida por el usuario por medio del cliente dado. Para comenzar a describir el funcionamiento del Servidor es necesario definir algunos conceptos que serán de ayuda posteriormente.

En primer lugar, una **ecuación diferencial** es una ecuación matemática que relaciona una función con sus derivadas. En las matemáticas aplicadas, las funciones usualmente representan cantidades físicas, las derivadas representan sus razones de cambio, y la ecuación define la relación entre ellas. Cómo estas relaciones son muy comunes, las ecuaciones diferenciales juegan un rol primordial en diversas disciplinas, incluyendo la ingeniería, la física, la química, la economía, y la biología.

En las matemáticas puras, las ecuaciones diferenciales se estudian desde perspectivas diferentes, la mayoría concernientes al conjunto de las soluciones de las funciones que satisfacen la ecuación. Solo las ecuaciones diferenciales más simples se pueden resolver mediante fórmulas explícitas; sin embargo, se pueden determinar algunas propiedades de las soluciones de una cierta ecuación diferencial sin hallar su forma exacta.

Debido a la complejidad alojada en el trabajo simbólico de las ecuaciones diferenciales en medios informáticos, se pretende utilizar una serie de herramientas capaces de brindar un soporte para el manejo de expresiones simbólicas. En este caso en concreto, se decidió implementar la tecnología de **SymPy**, que se trata de un módulo libre de código desarrollado en Python que posee una serie de métodos que simplifican el trabajo con expresiones simbólicas. Más adelante se enlistan los elementos utilizados de este módulo durante la construcción de la solución paso a paso de la ecuación.

## 1.11. Estableciendo la escala de dificultad

A lo largo de las siguientes definiciones estaremos hablando de un concepto de **dificultad**. La dificultad es una medida que utilizaremos para el control de cuán grande puede ser un proceso y con ello cuán tardado puede llegar a ser. Esta medida no posee otro sentido más que ofrecer un valor que describa los pasos con los que se llevará a cabo la solución. Como más adelante se definirá, los diferentes pasos utilizados poseerán medidas de dificultad distintas entre ellos. La dificultad en los pasos está determinada por dos partes principalmente:

- La cantidad de caracteres necesarios a comparar que se encuentra en una posición favorable para realizar un paso (la expresión coincide con lo buscado).
- La complejidad de la aplicación del paso, medida que es baja cuando no hay variantes en la aplicación y es alta cuando sí las hay.

Las medidas de dificultad actualmente propuestas son una aproximación en función a lo que podemos apreciar desde el diseño teórico del proyecto, por lo que estas medidas pueden variar durante la implementación práctica por medio de diferentes experimentos con el soporte simbólico que se pretende emplear para completar ciertos pasos.

La dificultad representa un parámetro solo de desarrollo; es decir, no posee ningún sentido para el usuario final y este no será expuesto de ninguna forma en la solución del proyecto. Con el fin de tener un punto de comparación, se presentan los siguientes límites de dificultad:

- Límite por paso individual: 500 (alrededor de 15 pasos atómicos o recursivos).
- Límite de dificultad global: 5000 (alrededor de 10 pasos compuestos en su máxima dificultad).

## 1.12. Pasos

La definición de una solución “paso a paso” puede resultar ambigua debido a que no existe una única definición para un paso. Para resolver esta situación, se propuso la siguiente definición para los pasos:

En general, diremos que un **paso** es una asociación binaria (entre dos objetos) de una expresión algebraica (normalmente una ecuación) y una leyenda que le describe. Definimos además una **solución** como una colección finita de pasos. Dada una ecuación diferencial, el programa retornará una solución en caso de que pueda encontrarse. Dicha solución es la misma para una misma ecuación diferencial.

Podemos entonces decir que para construir un paso se requiere de una manipulación algebraica de una expresión matemática previamente dada. La idea es que partiendo de la expresión dada por el usuario se puedan construir los pasos que compondrán la solución. Siguiendo esta idea, podemos decir que es posible construir todos los pasos por medio de manipulaciones algebraicas definidas. Dedicaremos el resto de este apartado a describir cada uno de los pasos que se pueden presentar y a asignarles un nivel de dificultad asociado al desarrollo de este a nivel informático.

Podemos separar los pasos más elementales en dos grupos: aquellos que se operan sobre una igualdad y aquellos que no lo hacen. Siendo rigurosos con esta idea, podemos definir al operador  $\gamma$  como una manipulación algebraica. Entonces tenemos que los dos pasos son como siguen:

- Paso sobre la igualdad:  $f = g \Rightarrow \gamma[f] = \gamma[g]$
- Paso sin la igualdad  $f \Rightarrow \gamma[f]$

Donde  $f, g$  son expresiones algebraicas cualesquiera.

A aquellos pasos que se operan sobre una igualdad les llamaremos operaciones algebraicas, mientras que a los pasos que no se aplican sobre una igualdad serán conocidos como transformaciones algebraicas. En general, tenemos que para construir una operación o transformación algebraica se utilizarán métodos de matemáticas simbólicas concretos, simplificados y sin intermediarios. Para trabajar con estos pasos se utilizarán los métodos en SymPy de manipulación algebraica. Las manipulaciones algebraicas que se utilizarán con su nivel de dificultad son:

**Suma**

- Operación:  $f + h = g + h$
- Transformación:  $f \leftarrow f + h$
- Dificultad: 1

**Resta**

- Operación:  $f - h = g - h$
- Transformación:  $f \leftarrow f - h$
- Dificultad: 1

**Multipliación**

- Operación:  $f * h = g * h$
- Transformación:  $f \leftarrow f * h$
- Dificultad: 1

**División**

- Operación:  $f / h = g / h$
- Transformación:  $f \leftarrow f / h$
- Dificultad: 1

**Composición**

- Operación:  $h(f) = h(g)$
- Transformación:  $f \leftarrow h(f)$
- Dificultad: 1

(Note que aplicar un paso sobre la igualdad es equivalente a realizar un paso sin igualdad para cada lado de la igualdad, por esto se podría considerar que el paso mínimo es la transformación; no obstante, hablamos de hablamos del paso de la igualdad como si fuera atómico por practicidad en la construcción de algoritmos con los que trabajaremos más adelante).

Por medio de combinaciones de estos pasos elementales se pueden construir pasos más complejos, los cuáles consideramos como los pasos más pequeños a mostrar en la solución con la intención de no realizar una exposición demasiado extensa de la solución y perder detalle en las ideas clave de la solución. Los **pasos compuestos** son aquellos que pueden ser descritos como una combinación finita de pasos elementales; poseen un propósito específico y representan la idea completa de un



paso. Estos pasos son los que aparecerán en la exposición final de la solución, y pueden separarse en dos grupos dependiendo del propósito con el que se lleven a cabo: algebraicos e integrales.

## 1.13. Paso Algebraico

Se define a un paso algebraico como una combinación de pasos elementales que realizan un trabajo meramente algebraico sobre la expresión. Dichos pasos tienen fines de manipulación y ordenamiento de la ecuación. Pese a que la cantidad de pasos elementales que componen a un paso algebraico es finita, no se puede conocer con certeza la cantidad exacta de ellos que se requiere para llevar a cabo el paso en cuestión, por lo que su dificultad no puede generalizarse a un valor estándar.

No obstante, con base a las ecuaciones más frecuentes con las que se pretende trabajar, se puede realizar una estimación de la cantidad de pasos elementales que le tomará; con la intención de no abandonar la idea de que es una aproximación, la dificultad se presenta en forma de rango. Los posibles pasos algebraicos que se podrán presentar son:

### **Sustitución**

- Reemplaza todas las ocurrencias de un término por una nueva expresión.
- Dificultad: 5-10.

### **Expansión**

- Busca eliminar todos los productos con la intención de dejar todo en términos de sumandos por ambos lados de la igualdad.
- Dificultad: 5-10.

### **Simplificación**

- Busca eliminar todos los factores diferentes a 0 y las sumas nulas por ambos lados de la igualdad.
- Dificultad: 7-10.

## Despeje

- Busca dejar por un lado de la igualdad una variable o expresión particular y del otro lado el resto de los elementos de la expresión.
- Dificultad: 10-15.

Con la intención de no obstruir el desarrollo del proyecto por el desarrollo de estas primeras etapas de manipulación algebraica, se pretenden utilizar los métodos de la librería SymPy dispuestos para llevar a cabo las operaciones presentadas anteriormente. *El resto de los métodos expuestos en esta sección serán desarrollados durante la construcción del proyecto a menos que se indique lo contrario.*

Los pasos algebraicos estarán guiados según sea el tipo de ecuación diferencial que sea detectado. Para esto, es necesario definir los tipos de ecuaciones diferenciales que se podrán resolver, así como las diferentes estrategias que se pretenden aplicar para detectarlas y darles solución. Con la intención de separar estos dos procesos, existen dos módulos que llevan a cabo esta función.

## 1.14. Módulo de clasificación

El módulo de clasificación es el encargado de clasificar el tipo de ecuación diferencial que es ingresado. Se trata del primer módulo que entra en acción en el servidor. El servidor posee soporte para presentar las soluciones paso a paso de los siguientes tipos de ecuaciones diferenciales:

- Primer Orden Separable
- Primer Orden Lineal
- Primer Orden Homogénea
- Primer Orden Exacta
- Primer Orden reducible a Lineal
- Orden Superior Lineal

La manera de operar del módulo puede resumirse en el siguiente par de diagramas en donde el primero representa la fase de examinación y el segundo la fase de comprobación.

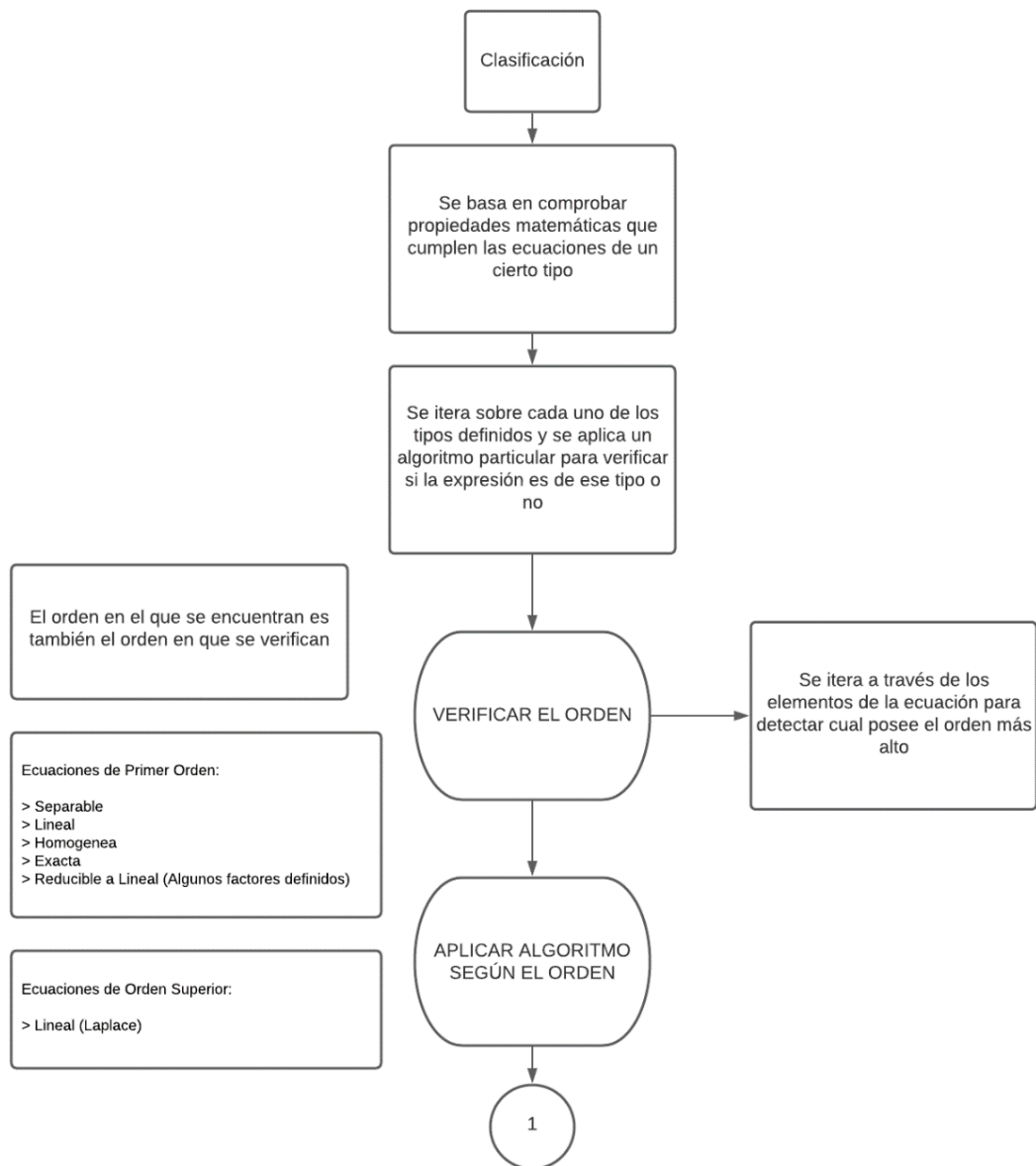


Figura 23: Diagrama de fase 1 de clasificación

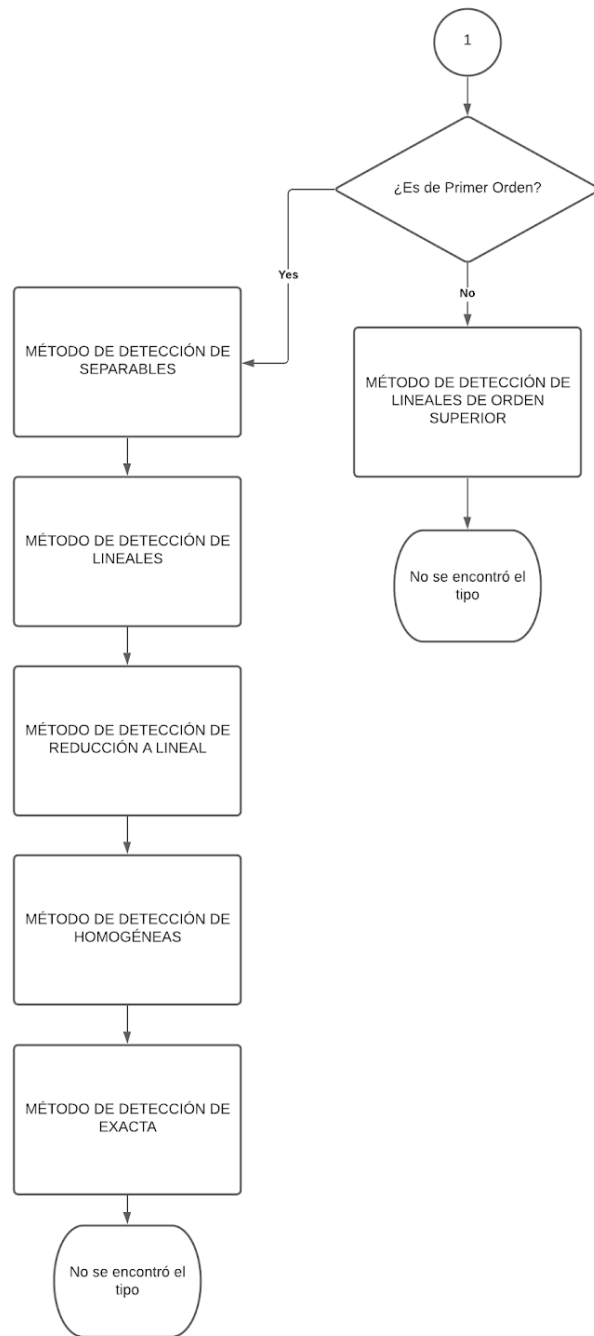


Figura 24: Diagrama de fase 2 de clasificación

Para poder detectar un tipo en concreto se realizan factorizaciones forzadas para llegar a las firmas generales que se describen en breve. Las factorizaciones forzadas buscan realizar los siguientes pasos en el orden en que se indica:

- Eliminar las fracciones multiplicando por el denominador ambos lados de la igualdad
- Igualar a 0 pasando a restar todo lo del lado derecho
- Factorizar los términos que poseen  $y'$
- Factorizar los términos que poseen ' $y$ ' (para las lineales de primer orden)
- Factorizar los términos que poseen de ' $y$ ' (para las reducibles a lineal)
- Factorizar los términos para cada uno de los grados de la derivada con respecto de ' $y$ ' (para las de orden superior)

Estudiaremos ahora cada uno de los métodos de detección, específicos para cada tipo de ecuación diferencial. La validación se hace verificando que los factores (polinomios) cumplan las propiedades definidas para cada caso.

### **Primer Orden Separable**

Si la ecuación es de la forma

$$p(x) + q(y)y' = 0$$

### **Primer Orden Lineal**

Si la ecuación es de la forma

$$q(x) + p(x)y + y' = 0$$

### **Primer Orden Reducible a Lineal (Bernoulli)**

Si la ecuación es de la forma

$$q(x)y^n + p(x)y + y' = 0$$

### Primer Orden Homogénea

Si la ecuación es de la forma

$$q(x, y) + p(x, y)y' = 0$$

Y además se cumple que  $q(x, y), p(x, y)$  son funciones homogéneas del mismo grado; esto es:

$$q(tx, ty) = t^n q(x, y) \quad \forall t$$

$$p(tx, ty) = t^n p(x, y) \quad \forall t$$

### Primer Orden Exacta

Si la ecuación es de la forma

$$q(x, y) + p(x, y)y' = 0$$

Y además se cumple que

$$\frac{\partial q}{\partial y} = \frac{\partial p}{\partial x}$$

### Orden superior con coeficientes constantes

Si la ecuación es de la forma

$$c_n y^{(n)} + c_{n-1} y^{(n-1)} + \dots + c_1 y' + c_0 y = 0$$

Decimos que es de orden  $n$  con coeficientes constantes.

Si la ecuación diferencial es clasificada con éxito este representará el primer paso de la solución: la forma algebraica y la justificación de que es del tipo en concreto. En el caso excepcional de que el sistema sea incapaz de determinar con precisión el tipo de ecuación diferencial del que se trata entonces se utilizará el módulo de SymPy para la clasificación de la ecuación diferencial.

Si el tipo no coincide con ninguno de los que se tiene soporte para mostrar el paso a paso, se procede directamente a resolver la ecuación diferencial con el módulo de SymPy (DSolve) y se regresa la solución sin pasos hacia el cliente. Si el tipo coincide, entonces se continúa con normalidad solo que el primer paso no mostraría una

prueba de porque la ecuación diferencial es de un determinado tipo (el paso 1 no tendría descripción por lo que estaría incompleto).

## 1.15. Módulo de control

El otro módulo que compone el manejo de la solución es el conocido como módulo de control, el cual opera de manera distinta para cada uno de los tipos de ecuaciones diferenciales para los que el servidor tiene soporte. Una vez que la ecuación diferencial fue clasificada con éxito en alguno de los tipos para los que se tiene soporte se procede a aplicar el algoritmo definido para la solución. Los algoritmos que se utilizan para resolver cada una de las ecuaciones diferenciales están descritos en la sección teórica del capítulo de este documento.

## 1.16. Paso integral

Se define a un paso integral como una combinación de pasos elementales que desempeñan la integración de una función. Un paso integral comienza con la expresión explícita de la integral y concluye con el resultado de dicha integral en caso de tener una solución. Si la integral no posee solución matemática conocida, entonces el paso integral regresa un mensaje indicando que no se tiene solución simbólica para dicha integral junto con la integral en su forma explícita. Debido a la gran variedad de integrales que pueden encontrarse, se plantea definir dos tipos de pasos integrales en función del papel que desempeñan dentro de la solución de la integral.

Un **paso integral recursivo** será aquel que no resuelve la integral, pero genera nuevas integrales que pueden resolverse posteriormente; mientras que un **paso integral atómico** será aquel que resuelve directamente la integral sin intermediarios. Se busca entonces que todo paso integral puede expresarse como una cantidad finita de pasos integrales atómicos por medio de una cantidad finita de pasos integrales recursivos.

## Paso integral atómico

En general diremos que los pasos integrales son los pasos integrales más pequeños que pueden darse. Como ya se explicó, estos pasos cuentan como compuestos y por lo tanto aparecen en la solución presentada al usuario al final. En el **Anexo 01**, se mostrará un catálogo con las diferentes integrales que se consideran atómicas en el sistema, las cuales se obtuvieron por medio de varios listados de las integrales más comunes que aparecen en ecuaciones diferenciales.

Es importante notar que el resultado de estas integrales muchas veces no se consideraría “directo” en una solución presentada en la escuela para un problema dado, por lo que se pretende mostrar estos pasos integrales junto con una pequeña demostración o enlace hacia una demostración de porque son verdaderas (especialmente en las que no pertenecen al primer apartado). De cualquier forma, esta información adicional se obtiene desde la Base de Datos que se describe más adelante.

## Paso integral recursivo

Los pasos integrales recursivos son mucho más complejos a los pasos vistos previamente. Si bien ya se había tocado el concepto de recursión en los primeros pasos algebraicos, no se abordó de manera especial debido a que esa parte se implementará por medio de SymPy y no representa mayor problema. El problema con los pasos recursivos (no solo los integrales) nace con la posibilidad de generar diferentes caminos para llegar a una solución o al menos para buscarla.

Si se revisa el catálogo de las integrales atómicas, se puede verificar que ninguna de ellas puede expresarse inmediatamente como la suma de dos funciones, esto es:

$$f(x)dx = [g(x) + h(x)]dx$$

De modo que el primer paso recursivo de todos será aquel que busca reducir a la expresión en sumas de expresiones que no pueden descomponerse de manera aditiva:

$$\int f(x)dx = \int [g(x) + h(x)]dx = \int g(x)dx + \int h(x)dx$$



Que posee una dificultad de 5 debido a que es relativamente sencillo encontrar el carácter aditivo que separa a las dos expresiones. De este modo, lo primero que se hace es realizar este paso recursivo tantas veces como sea necesario con la intención de dejar la integral original como la suma de integrales que no pueden descomponerse más con este paso:

$$\int f(x)dx = \int \sum_{i=0}^k p_i(x) = \sum_{i=0}^k \int p_i(x)$$

En caso de que sea posible llegar a la expresión anterior, se pretende entonces resolver cada una de las integrales del lado derecho partiendo del supuesto de que son atómicas.

En caso de que alguna de ellas no coincida con ninguna de las presentadas en el catálogo de integrales atómicas se requiere de una ejecución meticulosa de la **integración por partes**. La integración por partes nos genera una nueva integral a partir de una derivada y una integral:

$$\int u dv = uv - \int v du$$

Donde  $u$  y  $v$  son funciones de  $x$  y además  $du = u'dx$ ,  $dv = v'dx$  donde la prima indica la derivada. Las diferentes formas de tomar a las funciones  $u$ ,  $v$  dentro de la expresión generan los caminos de los que hablé previamente. Existen algunos resultados conocidos que se derivan de una buena elección de las funciones en la integración partes, de modo que estos resultados pueden permitir al servidor encontrar una ruta conocida para resolver la integral y no tener problemas con la generación de caminos múltiples.

En el **Anexo 02** se presenta el catálogo de los resultados más comunes que surgen mediante la ejecución de la integración por partes de manera adecuada sobre las expresiones. Note que las nuevas integrales pueden ser o no atómicas, por lo que es imperativo aplicar de manera sucesiva pasos recursivos (aunque sea más de una vez) hasta resolver por completo la integral en cuestión.

Existe la posibilidad de que la integral no pueda ser identificada dentro de ninguno de los dos catálogos de pasos integrales. En dado caso, se utilizará una examinación fragmentada de la integral para determinar diferentes caminos a partir de la integración por partes que podrían llevar eventualmente a la solución de la integral. Observemos que la integración por partes requiere de manera forzada que alguna de las funciones pueda ser integrada ( $v$ ), mientras que la otra debe ser posible de

derivar. En general, el proceso de derivación se realizará por medio de los métodos que SymPy posee para ello.

Consideraremos el caso en que la función está en su forma con factores, esto es:

$$f(x) = \prod_{i=1}^k p_i(x)$$

Ya que de otra forma diremos que  $f(x)$  no puede ser descompuesta para la integración por partes y con ello se agotan todas las herramientas del servidor para resolver la integral.

En caso favorable de que la función esté representada como el producto de factores, existe entonces una cantidad de factores finita que puedes representar a la función y donde cada uno de ellos ya no puede ser dividido a su vez en factores más pequeños de manera inmediata

$$\int f(x) = \int \prod_{i=1}^k p_i(x)$$

Para encontrar estos factores no factorizables se emplea una metodología similar a la empleada en la separación de los sumandos: se busca que dentro del nivel más bajo en la jerarquía de operaciones se tengan los signos correspondientes para indicar un producto. Por ejemplo:

$$f(x) = x^2 * 3 \ln(x) * \sin(x)$$

Posee los factores:

$$\{x^2, 3, \ln(x), \sin(x)\}$$

Suponiendo que es posible separar a la función en una cantidad finita de factores (digamos  $k$ ), entonces existe una cantidad igualmente finita de tomar a los factores  $u$  y  $v$  para la integración por partes.

Podemos obtener esta cantidad estudiando la cantidad de factores que se pretende que compongan a una de las funciones, ya que el resto de los factores serán quienes compongan a la otra función para preservar la definición de la función original. De este modo, la pregunta se vuelve de cuántas maneras es posible tomar  $1, 2, 3, \dots, k$  factores distintos del conjunto de los  $k$  factores. Haciendo las cuentas, tenemos que para tomar  $i$  elementos del conjunto se tienen:

$$k_i = \frac{k!}{i!(k-i)!}$$

Por lo que tenemos que la cantidad de formas de tomar a la pareja  $(u, v)$  son:

$$\sum_{i=0}^k k_i = k_0 + k_1 + k_2 + \dots + k_k = 2^k$$

De modo que la cantidad de caminos posibles que se pueden generar mediante integración por partes crece de manera exponencial a medida que la cantidad de factores crece de manera lineal. Aunque las formas de selección crecen rápidamente no todas ellas conducen a una integración por partes viables pues el producto de los factores seleccionados no genera una función de la cual se tenga registrada su integral ya sea en el catálogo de integrales atómicas o en el de recursivas definidas.

De este modo, una vez generado el mapa de todas las alternativas disponibles se descartan todas aquellas que conducen a una integral que no sea atómica o recursiva definida. Mediante este proceso se garantiza que el término  $uv$  del lado derecho de la integración por partes podrá ser definido y se espera que la integral del lado derecho ofrezca una nueva óptica que podría entrar en las integrales recursivas definidas, atómicas o en una nueva ejecución de integración por partes.

En resumen, decimos que el paso recursivo llamado integral por partes se define como sigue:

- Se obtiene la composición en factores de la función
- Se obtiene el mapa de todas las opciones posibles para el término  $du$
- Se descartan aquellas opciones de  $du$  para las cuales la integral no se tiene en el catálogo de integrales atómicas ni en el de recursivas definidas
- Las opciones restantes se prueban una por una por el orden en que fueron generadas. Las opciones quedan almacenadas en un espacio temporal y son descartadas por medio de las excepciones de dificultad explicadas en el apartado de dificultad. Cuando un camino es descartado entonces regresa a la lista de caminos disponibles y continua con el siguiente.

En caso de agotar todos los caminos posibles, se determina que la integral no puede ser resuelta por el sistema.

## 1.17. Árbol de caminos y Módulo de integración

Aunque el verificar que las opciones sean integrables reduce en gran medida los caminos, existe la posibilidad de que la cantidad de caminos crezca abruptamente en la medida de que varias integrales aplican integración por partes de manera sucesiva, generando entonces un **árbol de caminos**.

Definimos el **grado de un paso** como la cantidad de caminos que genera su aplicación. Hay que notar que todos los pasos integrales atómicos y recursivos definidos poseen grado 1, ya que existe una única manera de aplicarlos y por lo tanto generan una secuencia única para el desarrollo del paso. La integración por partes es un paso con grado variable, el cual depende de los caminos válidos que puedan generarse. Cuando un paso es de grado 0 significa que no existen caminos viables generados a partir de él por lo que decimos que ese camino es trunco y ya no se explora más en él.

Aunque el verificar que las opciones sean integrables reduce en gran medida los caminos, existe la posibilidad de que la cantidad de caminos crezca abruptamente en la medida de que varias integrales aplican integración por partes de manera sucesiva, generando entonces un árbol de caminos.

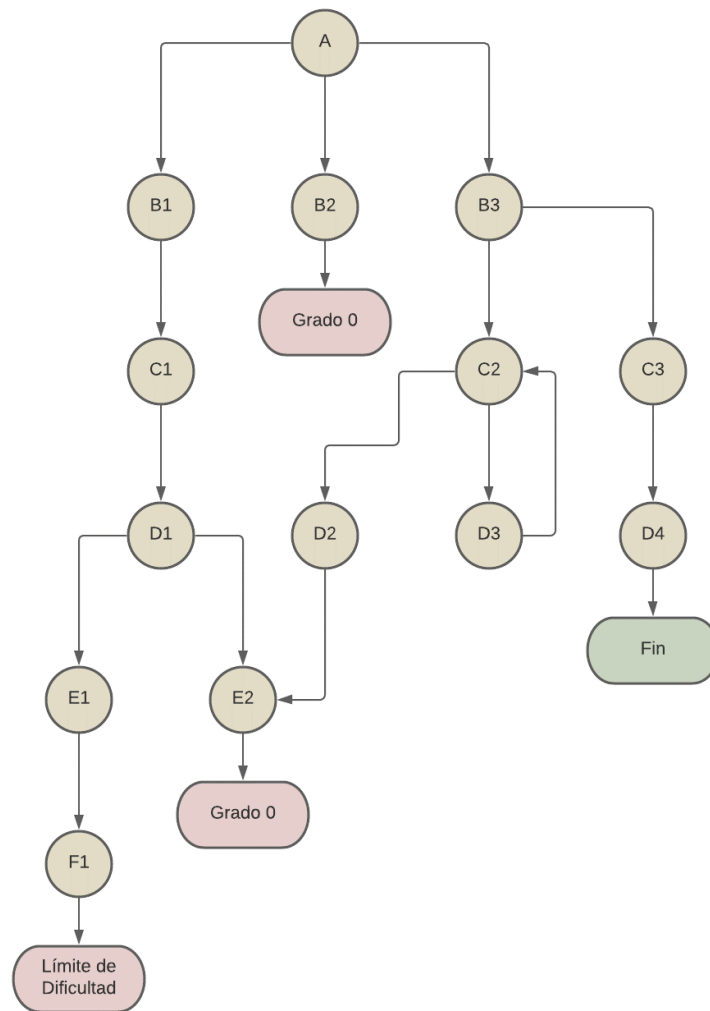


Figura 25: Diagrama de ejemplo de árbol de caminos

El diagrama muestra un ejemplo simplificado de cómo luce un árbol de caminos exponiendo algunas de las situaciones que pueden presentarse. Algunos de las cuestiones importantes que es necesario tomar en cuenta son:

- Cada una de las rutas de arriba hacia abajo del árbol se interpreta como un paso compuesto y cada uno de los movimientos puede ser un paso recursivo o atómico. Cada uno de estos pasos compuestos posee su propio control de dificultad con sus respectivas banderas. En caso de que exista un desborde de dificultad se corta el examen a través de ese camino y se procede a estudiar el siguiente.

- Todos los estados de la expresión son almacenados una sola vez en la memoria temporal con la intención de detectar relaciones entre los caminos que puedan conducir a atajos o a ciclos sin fin que de otra manera sería imposible distinguir (véase el ejemplo de  $D1 \rightarrow E2$  y  $D2 \rightarrow E2$  y el ciclo en  $C2 \rightarrow D3 \rightarrow C2$ ). Si no se hiciera una distinción con los ciclos la dificultad global del paso se extendería abruptamente y el estudio podría verse truncado rápidamente con pocos casos estudiados pero repetitivos entre sí. Cuando se detecta un ciclo, entonces se corta ese camino y no se generan más rutas a través de él.
- En caso de que se encuentre más de un camino para llegar a la solución entonces se toma el más corto, ya que solo existe una solución y al haber múltiples caminos significa que existen atajos dentro de la ruta de la solución. Para cumplir este punto, hay que también decir que siempre se estudian todos los caminos posibles, con la intención de dar con la solución más simple.
- La dificultad propia de cada uno de los caminos representa un límite en la longitud del camino, esto es, por cuantos estados distintos puede pasar antes de completar el paso. No obstante, no representa una barrera en la extensión horizontal del árbol de caminos por lo que un solo paso puede poseer muchos caminos con dificultades menores al límite. Para esto, definimos la dificultad de un árbol de caminos como la dificultad asociada a la suma de la cantidad de caminos que puede contener el árbol. Si consideramos que cada uno de estos caminos está sobre la máxima complejidad que se le permite, entonces la dificultad máxima del árbol estará relacionada con la cantidad de caminos a la máxima complejidad que puede mantener. Si el árbol está sobre el límite de dificultad permitido, entonces ya no puede generar más caminos a pesar de que el camino actual no se encuentre sobre el máximo posible.
- La dificultad del árbol "combina" las dificultades de los pasos compartidos por varios caminos, de modo que si existen varias rutas muy complejas pero que comparten gran parte de su recorrido la dificultad adicional de cada uno sobre el árbol será solo en lo que difieren de entre todos los demás caminos (esto es, si tomamos la dificultad del camino desde  $A \rightarrow F1$  y la de  $A \rightarrow E2$  solo contamos una vez la dificultad del recorrido para ir desde  $A \rightarrow D1$  en lugar de contarlo de manera independiente para cada uno de los caminos a la hora de calcular la dificultad del árbol de caminos).

**Nota:** Los valores para los límites de dificultad se calcularán de manera experimental una vez que se realicen pruebas con los algoritmos con la intención de que el tiempo de resolución de los pasos sea el más adecuado y además se abarque una gran cantidad de posibilidades sobre ellos.

Es importante no confundir los caminos posibles con la cantidad de integrales que se están atacando, la cual depende del paso integral recursivo de separar la expresión con sus componentes aditivas. Con esto, tenemos que, si un determinado paso integral se descompone en una cantidad finita de integrales y de las cuáles más de una requiere de integración por partes, entonces se sigue que existen más caminos sobre el paso general en función de los caminos de cada una de las integrales que se tomen en cuenta. Esto no afecta la percepción del árbol que hemos definido previamente, pues puede considerarse como un paso que genera muchos caminos en lugar de varios caminos simultáneos. La existencia de muchos caminos afecta la composición horizontal del árbol y con ello se procede con el control previamente establecido.

El siguiente diagrama (Figura 6) muestra cuáles serían los pasos agregados a la solución (que se mostrarán al usuario), que son aquellos que están iluminados. El rectángulo representa la intervención del **módulo de integración**. En la Figura 7 se muestran los principales componentes de este módulo y la manera en la que interactúan:

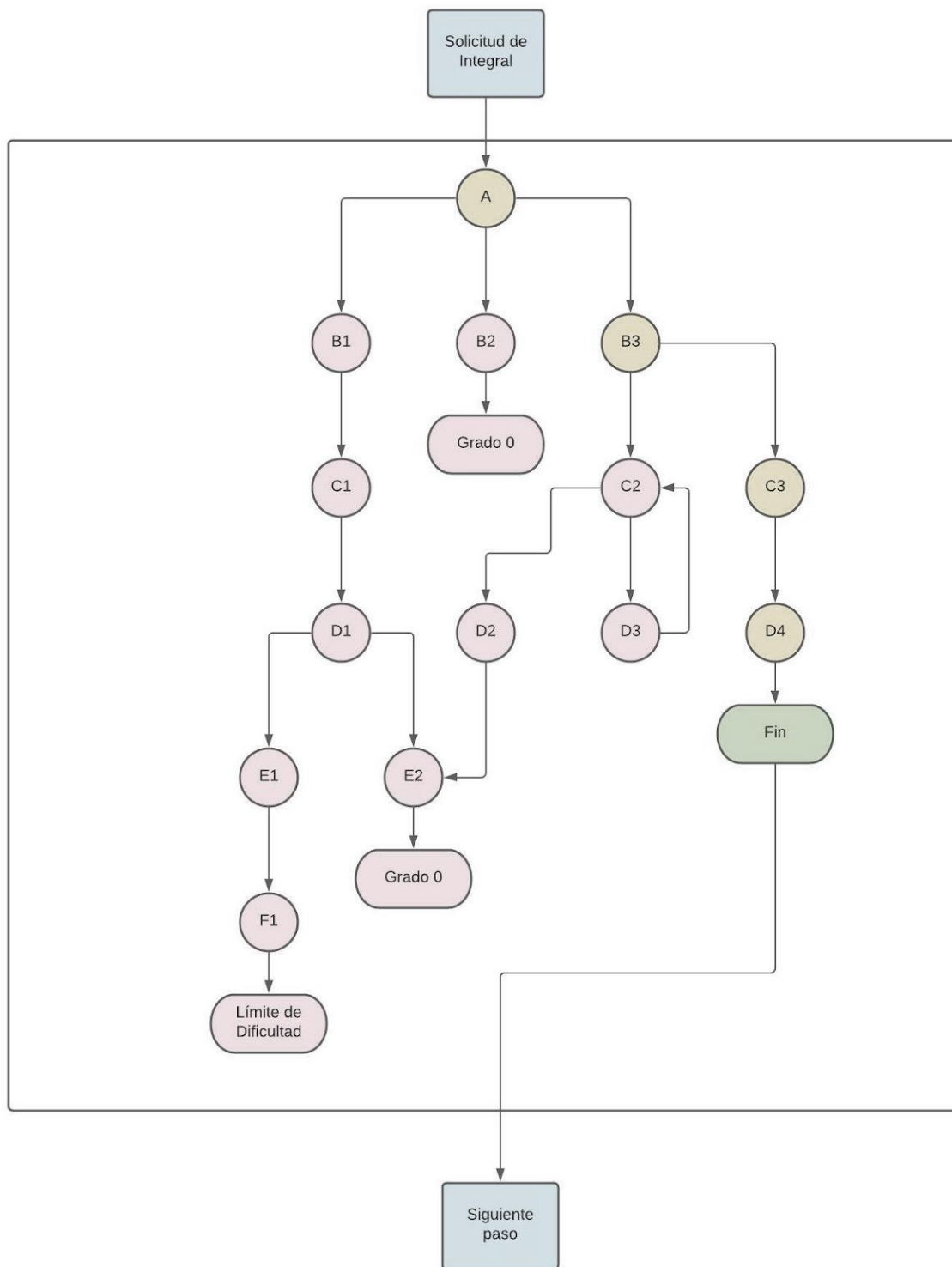


Figura 26: Diagrama ejemplo de árbol de caminos y pasos añadidos



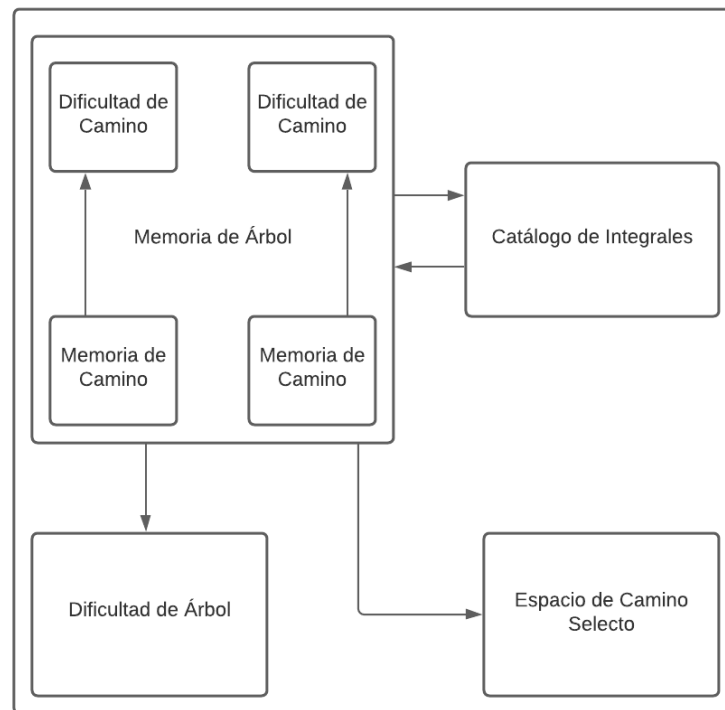


Figura 27: Módulo de integración

En caso de que no sea posible encontrar un camino en el árbol sin superar la dificultad permitida, entonces se procede a construir un único paso por medio de la herramienta de integración de SymPy. Este paso no posee descripción ni pasos intermedios pues SymPy solo ejecuta de manera directa su algoritmo propio. En caso de que SymPy tampoco sea capaz de encontrar una solución se dice que el paso quedó definitivamente trunco y se trunca la solución hasta este paso integral.

## 1.18. Módulo traductor y serializador

Recordemos que la expresión será analizada de manera simbólica mediante SymPy, mientras que la expresión ingresa bajo el formato LaTeX empacado en JSON desde la aplicación cliente y debe regresar bajo el mismo formato encapsulado en JSON. Es por este motivo, el servidor requiere otro par de módulos que llevan a cabo las funciones de serialización/deserialización de la información y formato LaTeX/Simbólico; los cuales llevan por nombre **Módulo Serializador** y **Módulo Traductor** respectivamente. Para desarrollarlos, se pretenden utilizar los diferentes

métodos proporcionados por módulos pre desarrollados en Python (que es el lenguaje motor de todo el Servidor). El módulo traductor se utiliza con frecuencia a medida que se van sumando pasos a la solución, mientras que el módulo Serializador solo se utiliza al inicio para recibir la información y al final para encapsular los pasos obtenidos en el proceso de solución.

## 1.19. Interacción modular del servidor

Los módulos del servidor descritos (clasificación, control e integración) interactúan entre ellos como se describe en el siguiente diagrama:

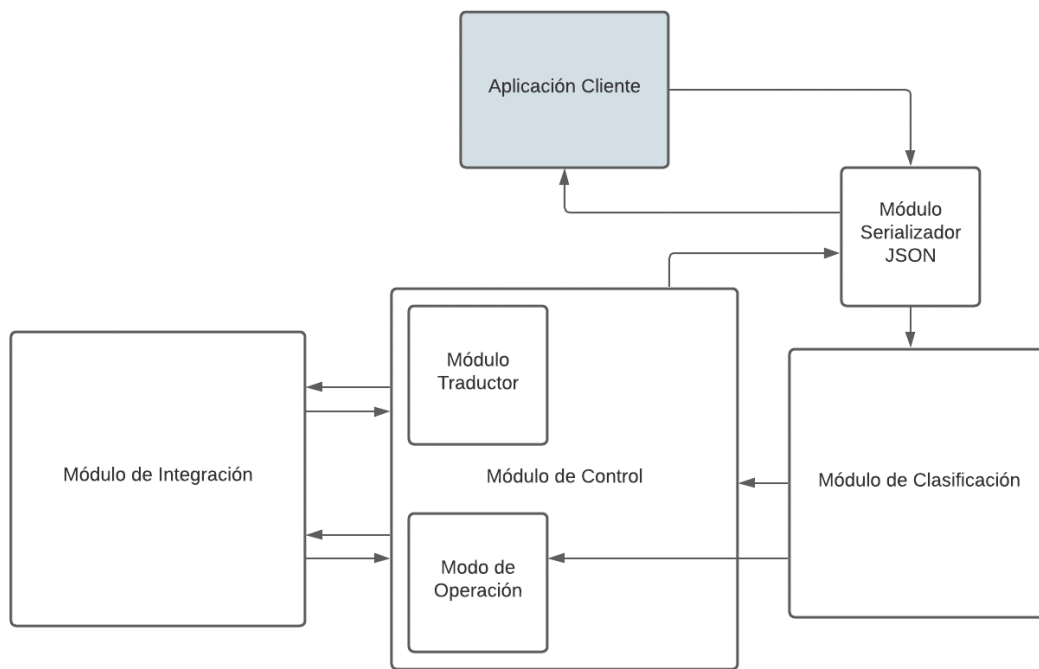


Figura 28: Módulos del servidor

Es importante notar que los diferentes procesos de solución que se llevan a cabo en el módulo de control requieren intervenciones esporádicas del módulo de integración para continuar con su desarrollo. El módulo de clasificación solo tiene una intervención temprana durante el proceso de solución.

## 1.20. Recuento de excepciones del servidor

En general, decimos que durante la operación de resolver una ecuación diferencial por el lado del servidor puede generarse una variedad de excepciones de las cuales distinguimos en dos clases: las generales y las modulares. Las excepciones generales son aquellas que detienen por completo la resolución de la ecuación diferencial y envían una señal de solución truncada hacia el cliente; mientras que las modulares son aquellas que se producen cuando uno de los módulos desarrollados es incapaz de completar correctamente uno de los pasos y se requiere la intervención de SymPy para buscar completarlo.

Por lo anterior, decimos que las excepciones modulares desembocan la intervención de SymPy y en caso de que la participación no sea favorable para continuar con el desarrollo paso a paso de la solución se desemboca una excepción general.

Las excepciones modulares del sistema son:

- No se pudo completar un paso integral debido a un nodo de grado 0
- No se pudo completar un paso algebraico debido a un nodo de grado 0
- Sobreflujo de dificultad en un paso integral
- Sobreflujo de dificultad en un paso algebraico
- No se pudo clasificar la ecuación diferencial
- Sobreflujo de dificultad global

A excepción de las últimas dos excepciones modulares, todas implican una intervención de SymPy para completar un paso, ya sea integral o algebraico. En estos casos de excepción SymPy intenta llevar a cabo dicho paso sin generar los pasos intermedios que se requieren para ello. La intervención de SymPy para estas excepciones puede tener dos opciones de resultado: puede o no llevarse a cabo correctamente. En caso de que se lleve correctamente, se agrega ese paso a la solución de la ecuación sin mostrar el desarrollo intermedio de esta; de otra forma el servidor desemboca la excepción general correspondiente con el tipo de paso que no se pudo concretar (algebraico o integral).

Cabe mencionar que la cuenta de dificultad de la solución no se reinicia con la intervención de SymPy, sino que permanece igual a la dificultad contada hasta antes de la excepción modular; esto con la intención de que una solución con muchos pasos inconclusos por dificultad genere un sobreflujo de dificultad global y con ello SymPy lleve a cabo la solución global de la ecuación.

Para los últimos dos tipos de excepciones, la intervención de SymPy puede resultar de maneras diferentes. Para el caso en donde no se puede clasificar la ecuación diferencial (véase módulo de clasificación), se utiliza SymPy para encontrar el tipo de la ecuación diferencial. De este modo, existen 3 variantes importantes:

- SymPy encuentra el tipo de ecuación y coincide con uno de los tipos del servidor
- SymPy encuentra el tipo de ecuación, pero no coincide con uno de los tipos del servidor
- SymPy no encuentra el tipo de ecuación

El primer caso sería resultado de alguna forma algebraica de la ecuación que no pudo ser atendida correctamente por el módulo de clasificación, por lo que en este caso se continúa con la solución por parte del servidor sin intervención de SymPy para los pasos que sigan. No obstante, es posible que se requiera de la intervención de SymPy para el desarrollo algebraico de la expresión en la primera etapa de solución para llegar a la forma que puede ser atendida por el módulo de control y comenzar con el desarrollo de la solución paso a paso de acuerdo con el algoritmo correspondiente.

Para los otros dos casos SymPy terminará por completo la solución, pues en ambos casos se tiene que el servidor no posee un soporte para desarrollar la solución paso a paso de dicha ecuación. Para el segundo y tercer caso, se intentará que SymPy desarrolle la solución sin pasos de la solución, mediante el módulo DSolve. En caso de que se dé con una solución, se envía al cliente bajo la nota con la excepción de que no pudo encontrarse una solución paso a paso para la ecuación.

## 1.21. API Vision

Un aspecto que hace falta definir es la parte del reconocimiento visual para los primeros dos métodos de ingreso de la ecuación al sistema. Esta parte del proceso se piensa resolver mediante el uso de una API desarrollada por Google y que es accesible de manera parcialmente gratuita por medio de un proyecto de Firebase. Esta API (de Google Vision) realiza la tarea de interpretar el texto contenido en una imagen por medio del análisis de densidad de píxeles y redes neuronales.

Es posible configurar esta API para obtener resultados en una forma determinada.

Para nuestro caso, la lectura de texto se llevará a cabo por medio de un formato estricto de caracteres con las siguientes normas:

- **D(y, x)** representa la derivada de la función **y** con respecto de la variable **x**. Este es el único caso en donde se permite el uso de la **coma (,)**.
- **+, -, \*, /, ^, =** son los únicos símbolos definidos para operaciones entre las variables y representan suma, resta, multiplicación, división, potencia e igualdad respectivamente.
- **(, )** son los únicos caracteres permitidos para agrupar operaciones y establecer jerarquía entre ellas.
- **sin(), cos(), tan(), sqrt(), ln()** son las funciones permitidas por el sistema. Reciben como parámetro exclusivamente una expresión simbólica y cada una de ellas puede considerarse como una de ellas.
- **E, Pi** son las únicas constantes que permite el sistema además de los dígitos del 0 al 9 incluyendo el **punto decimal (.)**

## 1.22. Base de datos

El último detalle que es necesario definir del sistema es la implementación de una pequeña base de datos encargada de almacenar una serie de datos que serán utilizados por el sistema. Entre los datos que se pretende almacenar se encuentran:

- Datos de los usuarios (credenciales, configuraciones personalizadas de la aplicación e historial de consultas del sistema)
- Catálogos de Integrales

Debido a que la información es variada y además puede cambiar su estructura con futuras actualizaciones de la aplicación, se utilizará una base de datos no relacional. El principal candidato para esta tarea es el gestor de bases de datos no relacionales que posee **Firebase**, aprovechando que ya fue implementado para la API de reconocimiento de imágenes.

La idea de almacenar la información requerida por el servidor en una base de datos externa al servidor surge debido a que las actualizaciones del sistema serían dirigidas a la cantidad de pasos definidos y ecuaciones y no a la manera de operar del servidor; esto es, separar la parte actualizable del proyecto de la fija.

Tanto el cliente como el servidor serán capaces de conectar con la base de datos con la intención de obtener información diferente (en el caso del cliente información relacionada al usuario y del servidor para obtener información acerca de sus procesos a realizar). Se posee una copia del catálogo dentro del servidor con la intención de no realizar tantas solicitudes a la base de datos (una por cada solicitud de ecuación diferencial). Esta copia se actualiza cada cierto tiempo (15 días, por ejemplo) en función de los cambios aplicados en la base de datos.

## 2. Requerimientos funcionales

---

### 2.1. Inicio de sesión

Código	Descripción
<b>RF001</b>	Al iniciar la aplicación cliente por primera vez se mostrará una pantalla en donde se da la bienvenida a la aplicación y se solicita la autenticación por parte del usuario. La pantalla mostrará dos botones: iniciar sesión o crear una cuenta.
<b>RF002</b>	Oprimiendo el botón de crear cuenta, se mostrará una pantalla con cuatro campos de texto y uno de selección. En el primer campo de texto se indicará el nombre del usuario, en el segundo campo un correo electrónico, en el tercero una contraseña y en el cuarto una verificación de la contraseña. En el campo de selección múltiple se debe seleccionar el tipo de usuario: Docente/Investigador o Estudiante. Existe además un botón para enviar los datos y una casilla para verificar si se desean mandar noticias de la aplicación al correo seleccionado.
<b>RF003</b>	Una vez enviados los datos de creación de cuenta, la aplicación ejecutará una consulta a la base de datos para verificar la existencia del nombre de usuario y el correo. En caso de que alguno ya esté dado de alta, regresará a la pantalla indicando el error en el campo correspondiente.
<b>RF004</b>	Una vez enviados los datos de creación de cuenta, la aplicación revisará que las contraseñas coinciden. Si no coinciden, se regresará a la pantalla indicando el error sobre las contraseñas.

<i>Código</i>	<i>Descripción</i>
<b>RF005</b>	<p>Oprimiendo el botón de iniciar sesión, se mostrará una pantalla con dos campos de texto. En el primer campo de texto se indicará el nombre de usuario o correo. En el segundo campo se indicará la contraseña.</p> <p>Existe además un botón para enviar los datos.</p>
<b>RF006</b>	Una vez enviados los datos de inicio de sesión, la aplicación ejecutará una consulta a la base de datos para verificar la existencia del nombre de usuario o correo. En caso de que no exista dicho usuario, regresará a la pantalla indicando que no existe el usuario o correo.
<b>RF007</b>	En caso de haber creado una cuenta exitosamente, se iniciará sesión con ella.
<b>RF008</b>	En caso de haber iniciado la aplicación tras haberla cerrado con autenticación completa sin cerrar sesión, se omite el inicio de sesión (persistencia de sesión). Esta información se guarda en un archivo en el dispositivo del usuario.
<b>RF009</b>	El archivo local de persistencia de sesión guardará el nombre de usuario, el correo, la contraseña del usuario y el tipo de usuario (investigador o estudiante).



## 2.2. Configuración de cuenta

Código	Descripción
--------	-------------

**RF010** Una vez iniciada la sesión, la aplicación mostrará al usuario el menú principal, el cual será de la forma de navegación horizontal (menú desplegable del lado izquierdo). Las opciones de este menú serán: Resolver una ecuación, opciones de cuenta, historial y cerrar sesión. Por defecto se estará seleccionada la opción de resolver ecuación. El menú podrá verse y esconderse oprimiendo el botón de menú (tipo sándwich).

**RF011** Al seleccionar la opción de "cerrar sesión", se borrará el contenido del archivo para iniciar sesión y se regresará al menú inicial de inicio de sesión.

**RF012** Al seleccionar la opción de "opciones de cuenta", se mostrará una pantalla con 2 campos de texto editables, una casilla de verificación, un campo de selección múltiple y dos botones. Los campos de texto tendrán el nombre de usuario y correo respectivamente. La casilla de verificación indicará si el usuario desea que le lleguen correos de actualizaciones de la aplicación. El campo de selección indicará el tipo de usuario. El primer botón indicará si se desea cambiar la contraseña y el segundo indicará si se desean guardar los datos actuales.

**RF013** Al oprimir el botón de cambiar contraseña se desplegarán tres nuevos campos de texto y un nuevo botón. Los campos de texto serán: actual contraseña, nueva contraseña, verificar contraseña anterior. El botón será "conservar contraseña"

**RF014** Al oprimir el botón de conservar contraseña, se colapsan los campos de texto de contraseña y se regresará el botón de "cambiar contraseña".

Código	Descripción
--------	-------------

<b>RF015</b>	Al oprimir el botón de guardar cambios, la aplicación verificará (en caso de que se tenga extendido el cambio de contraseña) que la contraseña actual coincida con la ingresada en el campo correspondiente y que las nuevas contraseñas coinciden. Para comparar, realiza una consulta a la base de datos. En caso de que no coincida alguna de ellas, la aplicación mostrará un error en estos campos para que el usuario cambie sus valores.
--------------	---

<b>RF016</b>	Una vez verificadas las contraseñas (en caso de ser necesario), se verificará que el nuevo nombre de usuario y/o correo no estén registrados actualmente en el sistema. En caso de que alguno de ellos ya se encuentre, la aplicación mostrará un error en estos campos para que el usuario cambie sus valores.
--------------	---

<b>RF017</b>	En caso de que la verificación del cambio de información de usuario sea válida, se realiza una actualización sobre la base de datos y sobre el archivo de sesión actual del usuario, sustituyendo el nombre de usuario y la contraseña por los nuevos valores.
--------------	--

## 2.3. Historial

Código	Descripción
--------	-------------

<b>RF018</b>	Al seleccionar la opción de "historial" se desplegará una lista con las ecuaciones más recientes que haya realizado el usuario. Dichos valores serán consultados cada vez que el usuario acceda a este apartado mediante una consulta hacia la base de datos. Cada elemento de la lista debe contener la fecha y hora en la que se consultó la ecuación y una vista previa en LaTeX de la misma, así como un botón de cruz y uno de pin.
--------------	--

Código	Descripción
<b>RF019</b>	La consulta de los elementos del historial traerá consigo 3 elementos: la fecha de la solicitud, el formato LaTeX para que se muestre la preview y el formato JSON de la solicitud (este último no se muestra al usuario).
<b>RF020</b>	Al hacer clic sobre una de las ecuaciones mostradas en el historial, la aplicación automáticamente pasa a mandar la solicitud al servidor (la solicitud en JSON que fue recogida de la base de datos). Durante este proceso, la aplicación pasará al estado de espera de solicitud descrito en el módulo de despacho.
<b>RF021</b>	En caso de que no se tenga registro en el historial del usuario, la pantalla de esta opción mostrará un mensaje que indique: "Genera una solicitud para que aparezca en tu historial".
<b>RF022</b>	Al hacer click sobre el botón de "cruz" de uno de los elementos de la lista, se mandará un mensaje al usuario para confirmar que se desea eliminar permanentemente el elemento de su historial. En caso de que el usuario confirme dicha acción, el registro desaparecerá de la pantalla y será eliminado de la base de datos. En caso contrario, solo se cierra el mensaje de alerta.
<b>RF023</b>	Al hacer click sobre el botón de "pin" de unos de los elementos de la lista, se oscurecerá el botón de pin de manera que aparente estar seleccionado. El elemento pasará a estar encima de toda la lista y en la base de datos se indicará que dicho valor se encuentra fijado por el usuario. Al dar clic nuevamente sobre este botón se removerá el pin visualmente y en la base de datos.
<b>RF024</b>	Los elementos del historial se despliegan siguiendo el orden: <ul style="list-style-type: none"> <li>• Pines en orden cronológico descendente</li> <li>• Ecuaciones no pinadas en orden cronológico descendente</li> </ul>

## 2.4. Base de datos

Código	Descripción
<b>RF025</b>	El gestor de la base de datos (Firebase) estará a cargo de eliminar aquellas solicitudes con más de 15 días de antigüedad del historial que no hayan sido fijadas por el usuario.
<b>RF026</b>	En caso de realizar una actualización al sistema que requiera ser notificada a los usuarios, el gestor de la base de datos enviará un correo a aquellos usuarios que tengan en su configuración que desean recibir notificaciones de la aplicación.
<b>RF027</b>	La base de datos guardará datos para la aplicación (cliente) y para el servidor. Los datos que se deben almacenar de la aplicación cliente están agrupados por usuarios, mientras que los del servidor están agrupados por pasos y secuencias de pasos.
<b>RF028</b>	Los usuarios almacenados en la base de datos deben de contener la siguiente información: Nombre de usuario, correo, contraseña, verificación de correos, historial (compuesto de la entrada en LaTeX, la solicitud en JSON y la fecha   hora de solicitud) y tipo de cuenta.

## 2.5. Módulo de recepción

Código	Descripción
<b>RF029</b>	Al seleccionar la opción de "Resolver Ecuación" la aplicación mostrará 3 botones indicando los 3 métodos de solicitud que soporta la aplicación: "Por imagen", "Por dibujo", "Por teclado".
<b>RF030</b>	Al seleccionar la opción de " Por imagen", este botón aparecerá como seleccionado y los otros dos serán reemplazados por los botones "Cámara", "Galería" respectivamente.
<b>RF031</b>	Al seleccionar la opción "Cámara" existen dos escenarios: se lanza un intento para utilizar la cámara del dispositivo (el cual puede terminar exitosamente con el retorno de una imagen al programa principal o terminar mal y no retornar nada) o bien no se cuentan con los permisos actualmente y se solicita el permiso nuevamente del uso de la cámara del dispositivo en cuestión. En este segundo caso, se requiere volver a presionar el botón para lanzar el intento una vez concedidos los permisos.
<b>RF032</b>	En caso de que el intento del uso de la cámara se haya lanzado y haya resultado exitoso, se guardará la imagen capturada en un archivo de naturaleza temporal en la carpeta principal del programa para su posterior interpretación. En caso de que haya existido algún error con el intento, se mandará una notificación indicando los detalles del error que puedan ser interpretados y se regresará al paso anterior (selección de cámara o galería).
<b>RF033</b>	Al seleccionar la opción "Galería" existen dos escenarios: se lanza un intento para acceder a los archivos (galería) del dispositivo (el cual puede terminar exitosamente con el retorno de una imagen al programa principal o terminal mal y no retornar nada) o bien no se cuentan con los permisos actualmente y se solicita el permiso nuevamente del acceso a la galería del dispositivo en cuestión. En este segundo caso, se requiere volver a presionar el botón para lanzar el intento una vez concedidos los permisos.

<i>Código</i>	<i>Descripción</i>
<b>RF034</b>	En caso de que el intento de acceso a la galería se haya lanzado y haya resultado exitoso, se guardará una copia de la imagen seleccionada en un archivo de naturaleza temporal en la carpeta principal del programa para su posterior interpretación. En caso de que haya existido algún error con el intento, se mandará una notificación indicando los detalles del error que puedan ser interpretados y se regresará al paso anterior (selección de cámara o galería).
<b>RF035</b>	Al seleccionar la opción "Por dibujo", se mostrará al usuario un lienzo de dimensiones relativas al tamaño del display con el que se trabaje. Dicho lienzo estará completamente en blanco y tendrá tres botones en la parte superior: uno con un lápiz, otro con una flecha hacia atrás y otro para cerrar. En la parte inferior del lienzo se tendrá un botón que diga aceptar.
<b>RF036</b>	Dentro del lienzo, al seleccionar la opción de lápiz el usuario podrá escribir con libertad en el lienzo por medio del dispositivo puntero que se utilice (mouse, dedo, pluma táctil, tableta gráfica). Los trazos se irán dibujando a medida que el usuario los haga (modalidad de Paint).
<b>RF037</b>	Dentro del lienzo, al seleccionar la opción de la flecha hacia atrás se eliminará el último trazo realizado por el usuario (un trazo será definido por cada vez que el usuario presiona sobre el lienzo hasta que lo deja de hacer). En caso de no haber trazos, la flecha no genera cambios
<b>RF038</b>	Dentro del lienzo, al seleccionar la opción de la cruz, se mandará una advertencia al usuario indicando si en realidad desea descartar el lienzo trabajado hasta ese punto. En caso de que el usuario acepte el descarte, el lienzo se cerrará y regresará a la pantalla de selección de método de ingreso.
<b>RF039</b>	Dentro del lienzo, al seleccionar la opción de aceptar, se mandará guardará el lienzo actual como una imagen de naturaleza temporal en la carpeta principal de la aplicación para su posterior interpretación.

<i>Código</i>	<i>Descripción</i>
<b>RF040</b>	Una vez completado alguno de los procesos de entrada de imagen (por lienzo o foto), la aplicación mostrará una pantalla en donde se verá la previsualización de la imagen que se ha generado seguida de un par de botones de confirmación. En caso de aceptar dicha imagen, se avanza al siguiente paso; de otra forma, se regresa a la pantalla de selección de método de ingreso y la imagen temporal es eliminada de la memoria del dispositivo.
<b>RF041</b>	Una vez verificada la imagen temporal en la carpeta principal se realizará un filtrado sobre esta imagen para generar un formato estándar que pueda ser interpretado por la API Vision de Google Cloud. El filtrado incluye: poner en escala de grises, normalizar el tamaño de la imagen y normalizar la resolución.
<b>RF042</b>	La imagen modificada será enviada por medio de un formato en Base64 a un proceso corriendo en Google Cloud con la API Vision. Dicha API será configurada para recibir las solicitudes de la aplicación e interpretar el texto contenido en la imagen enviada.
<b>RF043</b>	Una vez que se tenga respuesta por parte de la API Vision, la aplicación puede proceder en dos formas diferentes: la API no encontró ningún texto en la imagen, o bien la API encontró algún texto en la imagen. Si no se encontró texto, la aplicación manda una notificación al usuario indicando que no se encontró ninguna ecuación diferencial en el texto que se introdujo y se regresará a la pantalla de métodos de ingresos de ecuación. En caso de que se encuentre algún texto en la imagen se pasa dicho texto al módulo de interpretación.

Código	Descripción
--------	-------------

**RF044**

Al seleccionar la opción “Por teclado”, la aplicación mostrará una pantalla conformada por una línea a manera de cuadro de texto editable y un panel con teclas para introducir símbolos matemáticos. Los símbolos que componen al teclado serán:

Primer Grupo:

- > Dígitos del 0 al 9.
- > Las letras ‘x’, ‘y’, ‘e’.
- > Las palabras reservadas ‘pi’, ‘cos’, ‘sin’, ‘tan’, ‘ln’, ‘sec’, ‘csc’, ‘cot’, ‘acos’, ‘asin’, ‘atan’.
- > Los operadores aritméticos: ‘+’, ‘-’, ‘\*’, ‘/’, ‘^’.
- > Los operadores de agrupación: ‘(’, ‘)’, ‘[’, ‘]’, ‘{’, ‘}’.
- > El operador de igualdad: ‘=’.
- > El operador diferencial ‘’ ‘’.
- > El punto decimal ‘.’.

Segundo Grupo:

- > Las flechas de movimiento del cursor (izquierda, derecha).
- > Borrar toda la entrada.
- > Borrar el carácter anterior al cursor.
- > Terminar la ecuación.

**RF045**

A medida que el usuario presione las teclas de ingreso se construirá la cadena de acuerdo con lo especificado en los requerimientos no funcionales RNF 043 - RNF 046. Se mostrará la vista previa en el cuadro texto en formato LaTeX en tiempo real de edición.

**RF046**

Una vez presionada la tecla de enviar, la cadena pasará al módulo de interpretación sin solicitar la intervención de la API de Google Vision.



## 2.6. Módulo de interpretación

Código	Descripción
--------	-------------

<b>RF047</b>	A partir de la string que regrese ya sea Google Vision o el Módulo de Recepción pasará por una serie de verificaciones para establecer si el texto contenido posee la forma de una ecuación diferencial válida para el sistema.
--------------	---

<b>RF048</b>	Las validaciones del módulo deben realizarse en el mismo orden en el que se exponen y basta con una sola de ellas que no se cumpla para considerar que el formato de la entrada no es válido. En caso de que sea así, la aplicación termina el flujo de la petición y manda la pantalla de error correspondiente a "Anomalía de interpretación".
--------------	--

<b>RF049</b>	En caso de que la cadena cumpla con todas las validaciones del módulo, se enviará al siguiente módulo en el mismo formato con el que fue verificada.
--------------	--

---

## 2.7. Módulo de formateo

Código	Descripción
--------	-------------

<b>RF050</b>	Por medio de la cadena validada se construirá una nueva cadena en formato LaTeX. Con base al formato de entrada propuesto, se utilizará una serie de funciones para realizar los cambios pertinentes en la expresión para que pase a formato LaTeX serán los establecidos en los RNF 68 - RNF 71
--------------	--

<b>RF051</b>	La cadena LaTeX generada se mostrará al usuario en el centro de la pantalla marcando que esa es la ecuación que se enviará al servidor. En dicha vista el usuario puede decidir si continuar o declinar en el proceso de solución de la ecuación. En caso de declinar, se cortará el flujo de solución de la ecuación hasta entonces y se regresará al menú de selección de ingreso de ecuación diferencial.
--------------	--

<b>RF052</b>	Al aceptar la ecuación mostrada, se mandará la string con el nuevo formato al módulo de despacho que se encontrará en su primera fase (fase activa)
--------------	---

## 2.8. Módulo de despacho

Código	Descripción
<b>RF053</b>	<p>Al ingresar a la fase activa, se creará una cadena JSON que contenga los siguientes elementos:</p> <ul style="list-style-type: none"> <li>&gt; La cadena LaTeX generada</li> <li>&gt; Un identificador propio de la sesión que está enviando la solicitud.</li> </ul>
<b>RF054</b>	<p>Al generar la cadena JSON, se guardará dicha cadena en la base de datos, añadiendo además la siguiente información:</p> <ul style="list-style-type: none"> <li>&gt; Fecha de solicitud</li> <li>&gt; Hora de la solicitud</li> <li>&gt; Tipo de usuario que realiza la solicitud</li> </ul> <p>La información será enviada por medio de un conector proporcionado por el gestor de bases de datos a utilizar (Firebase). Se guardará como un nuevo registro en el historial del usuario que realiza dicha solicitud como no pineado por defecto.</p>
<b>RF055</b>	<p>El empaquetado JSON será enviado mediante una HTTP Request hacia el servidor, en particular al módulo de Serializador JSON. La solicitud estará bajo el método POST.</p>
<b>RF056</b>	<p>Una vez que se haya mandado la solicitud y guardado la consulta en el historial, el módulo pasará a la "fase pasiva", en donde la aplicación cliente quedará a la espera de la respuesta (o no respuesta) por parte del servidor. En esta modalidad el usuario no puede interactuar de ninguna forma con la aplicación.</p>
<b>RF057</b>	<p>Mientras el flujo de resolución de la ecuación permanezca en este módulo (ya sea de forma activa o pasiva) se le mostrará al usuario la pantalla de "espera", en la cual habrá un mensaje indicando que su petición se está procesando e indicando que esa ecuación ya puede ser visible desde su historial.</p>
<b>RF058</b>	<p>Durante la fase pasiva, la aplicación quedará a la espera de una de las posibles respuestas del servidor. Es posible que se produzca alguna anomalía en el flujo de la solución, en cuyo caso la aplicación mostrará la pantalla de anomalía correspondiente, saliendo así de la fase pasiva y terminando el flujo de solución</p>

Código	Descripción
--------	-------------

<b>RF059</b>	Cuando la aplicación requiera mostrar una pantalla de anomalía, contendrá los siguientes datos:
--------------	---

- > Nombre de la anomalía (el cual se asocia directamente con la fuente que produjo la excepción)
- > Descripción detallada de la anomalía, la cual a su vez se compone de diferentes elementos dependiendo del tipo de anomalía que se haya detectado.
- > Recomendaciones para que el usuario evite la anomalía en peticiones posteriores que sean similares o iguales al servidor. Estas recomendaciones también están ligadas con el tipo de anomalía con la que se esté tratando.

Además de un botón de aceptar dicho mensaje. Al aceptar, se regresa a la prevista de la ecuación por sí el usuario desea volver a mandar su solicitud al servidor.

<b>RF060</b>	Al lanzar la solicitud hacia el servidor, la aplicación establecerá un timeout para la espera de la respuesta por parte del servidor. En caso de que el tiempo se exceda al establecido en (...), se lanzará una anomalía de tiempo, pasando a mostrar la pantalla de anomalía correspondiente
--------------	--

## 2.9. Módulo de despliegue

<i>Código</i>	<i>Descripción</i>
<b>RF061</b>	A partir de la respuesta empaquetada por el servidor, la aplicación deserializará el contenido del JSON para extraer los pasos contenidos en formato LaTeX en cadenas de caracteres.
<b>RF062</b>	El módulo deberá extraer las cadenas string de los pasos en la situación de anomalía de incompletitud, teniendo entonces una interacción continua con el módulo de despacho.
<b>RF063</b>	Cada una de las cadenas en formato LaTeX que se extraigan serán enviadas al módulo de presentación, incluyendo aquellas que vienen provengan de una anomalía de completitud.
<b>RF064</b>	El módulo también se encargará de recibir los mensajes propios de cada tipo de anomalía que puedan provenir desde el servidor, así como su correspondiente interpretación para mostrar una determinada pantalla de anomalía según sea el caso.

## 2.10. Módulo de presentación

Código	Descripción
--------	-------------

<b>RF065</b>	La aplicación mostrará al usuario una pantalla que contenga todos los pasos en orden que son necesarios para resolver su ecuación diferencial. Para mostrar las expresiones algebraicas, se utilizarán módulos de compilación LaTeX previamente desarrollados por otros desarrollados (los mismos utilizados para la muestra de la vista previa). Al final de la presentación, se incluirán los siguientes botones para concluir con la interacción del usuario con esa solicitud: > Aceptar y cerrar > Exportar a pdf > Exportar a LaTeX > Compartir
<b>RF066</b>	Al seleccionar la opción de “Aceptar y Cerrar”, la aplicación terminará el flujo de solución de la ecuación y se mostrará el menú inicial al usuario.
<b>RF067</b>	Al seleccionar la opción de exportar a pdf, se lanzará una nueva pantalla emergente para que el usuario seleccione el destino y el nombre por defecto que deberá llevar el archivo (esto aplica para todos los dispositivos de la PWA).
<b>RF068</b>	Al seleccionar el nombre y ubicación de exportación del documento, se utilizará una tecnología externa desarrollada para transformar un documento en formato LaTeX a pdf (compilación de LaTeX). Al concluir la exportación correctamente, se regresará al menú previo donde se muestra la solución de la ecuación.
<b>RF069</b>	Al seleccionar la opción de exportar a LaTeX, se le mostrará al usuario una pantalla emergente con el código LaTeX correspondiente a la solución de la ecuación diferencial. En esta pantalla estarán las opciones de cerrar o copiar en el portapapeles. El texto no podrá editarse, pero si seleccionarse para copiar fragmentos específicos. Al cerrar, se regresa al menú anterior donde se muestra la solución.
<b>RF070</b>	Al seleccionar la opción de compartir, se le mostrará un pequeño formulario en donde el usuario debe indicar el correo a quién desea compartir la solución de la ecuación diferencial y una lista en donde se indica la lista de elementos que desea enviar: solución en pdf y solución en LaTeX, así como un botón para enviar.

Código	Descripción
<b>RF071</b>	Al marcar alguna de las casillas para el envío por correo de la solución, se creará el correspondiente elemento marcado (la solución en pdf y/o LaTeX). El correo se enviará por medio de una instancia gestora de correos provista por Firebase, a nombre de la aplicación e incluirá información de la cuenta remitente (correo y nombre de usuario), así como la ecuación que fue enviada (en LaTeX) y los correspondientes elementos que el remitente haya decidido adjuntar. En caso de haber sido generados previamente, los tomará de la dirección de memoria designada por el usuario previamente.
<b>RF072</b>	En el caso de haya generado alguna anomalía, el módulo se limitará a mostrar la información correspondiente a la situación que generó la anomalía. La manera en la que despliega la información será análoga a la de los pasos.
<b>RF073</b>	En el caso de que la cuenta que mandó la solicitud esté en modalidad de investigador, la aplicación mostrará además de la solución la gráfica asociada con la familia de curvas que son solución de su ecuación diferencial, así como sus raíces y puntos críticos. Para la gráfica, se utilizará un conversor de base 64 para reconstruir la imagen.

## 2.11. Módulo de serialización

*Código Descripción*

- |              |  |
|--------------|--|
| <b>RF074</b> | El servidor será capaz de recibir solicitudes en formato JSON lanzadas desde una instancia de la aplicación cliente. Dicha solicitud debe contener los elementos mínimos establecidos en los módulos de la aplicación cliente.   |
| <b>RF075</b> | Una vez recibida la solicitud, el servidor extraerá la ecuación planteada en formato LaTeX y el tipo de cuenta que mandó la solicitud. Guardará estos valores en cadenas de caracteres para su posterior análisis.   |
| <b>RF076</b> | Al ser el punto de unión, el módulo creará una cola de solicitudes tanto para la entrada como para la salida del servidor. El modelo de esta fila será FIFO, y atenderá las solicitudes a medida que vayan llegando al servidor y se encuentre a su vez listo para construir la siguiente solución.                            |
| <b>RF077</b> | El servidor almacenará temporalmente las direcciones de red de quiénes realizan las solicitudes para así desarrollar las filas de solicitudes y ser capaz de regresar la solicitud a la instancia cliente que corresponda.   |
| <b>RF078</b> | Una vez construida la solución de la ecuación diferencial, el módulo se encargará de poner en formato JSON los pasos de la ecuación una vez hayan sido pasados a formato LaTeX. Se enviará el paquete JSON de vuelta a la instancia de la aplicación correspondiente a manera de respuesta a la HTTP Request.                  |
| <b>RF079</b> | En el caso de que la solicitud haya sido por parte de una cuenta investigador, se adjuntará además la gráfica de la solución y sus puntos característicos en el paquete JSON. En el caso de la gráfica, se mandará a través de una cadena en base 64 para que pueda ser enviada a través de la respuesta de la solicitud POST. |
| <b>RF080</b> | El módulo tendrá una memoria local que se irá llenando a medida que se construyan los diferentes pasos de la solución; cada uno de ellos en su correspondiente formato LaTeX y descripción.  |
| <b>RF081</b> | En el caso de que se desate una anomalía en la solución de la ecuación, el servidor debe serializar los pasos que se hayan realizado hasta antes del desborde de la anomalía y serán enviados a la aplicación cliente correspondiente como anomalía de completitud.  |



## 2.12. Módulo de clasificación

### Código Descripción

- RF082** A partir de la ecuación recibida en formato LaTeX por parte del módulo de serialización, el servidor tendrá una intervención del módulo traductor previa a la fase de clasificación para pasar del formato de string LaTeX al formato de string aceptado por SymPy. Para ello, se utilizarán los grupos de intercambio definidos en los requerimientos RNF 091 con la ayuda del módulo traductor.
- RF083** Con la ecuación guardada en la clase correspondiente de SymPy, el servidor realizará una serie de manipulaciones algebraicas para determinar el tipo de ecuación diferencial del que se trata, basándose en las ecuaciones diferenciales que el sistema puede resolver (RNF 092). La validación se hará bajo el mismo orden siempre y al encontrar un patrón de coincidencia con una de las formas.
- RF084** Para detectar el patrón de una ecuación, el servidor realizará los siguientes pasos a manera general para llegar a una de las formas esperadas:
- Eliminar las fracciones multiplicando por el denominador ambos lados de la igualdad
  - l. Igualar a 0 pasando a restar todo lo del lado derecho
  - l. Factorizar los términos que poseen  $y'$
  - l. Factorizar los términos que poseen  $y'$  (para las lineales de primer orden)
  - /. Factorizar los términos que poseen de  $y'$  (para las reducibles a lineal)
  - Factorizar los términos para cada uno de los grados de la derivada con respecto de  $y'$  (para las de orden superior)
- RF085** El servidor indicará que la ecuación diferencial es de primer orden separable si se llega a la forma:
- $$p(x) + q(y)y' = 0$$
- Donde  $p(x)$  y  $q(y)$  son funciones en esencia diferentes que contienen sólo términos en base a las variables expresadas  $x$ ,  $y$ . Una vez detectada la ecuación de este tipo, el servidor enviará la ecuación al módulo de control junto con una etiqueta que indica que es una ecuación diferencial de primer orden separable. La comparación se realizará mediante la verificación de caracteres consecutivos que componen a la ecuación diferencial.

Código	Descripción
--------	-------------

<b>RF086</b>	El servidor indicará que la ecuación diferencial es de primer orden lineal si se llega a la forma:
--------------	--

$$y' + p(x)y + q(x) = 0$$

Donde  $p(x)$  y  $q(x)$  son funciones en esencia diferentes que se expresan en términos de 'x'. Una vez detectada la ecuación de este tipo, el servidor enviará la ecuación al módulo de control junto con una etiqueta que indica que es una ecuación diferencial de primer orden lineal. La comparación se realizará mediante la verificación de caracteres consecutivos que componen a la ecuación diferencial.

<b>RF087</b>	El servidor indicará que la ecuación diferencial es de primer orden homogénea si se llega a la forma:
--------------	---

$$p(x, y)y' + q(x, y) = 0$$

Donde  $p(x, y)$  y  $q(x, y)$  son funciones en esencia diferentes que se expresan en términos de 'x' y 'y'. Una vez detectada la ecuación de esta forma, aplicará una validación de homogeneidad utilizando la validación que indica la ecuación:

$$p(tx, ty) = t^n p(x, y); q(tx, ty) = t^n q(x, y)$$

Para ello se utilizará una sustitución en las variables de la ecuación y se buscará la factorización de los términos como indica la ecuación (esto con el apoyo de SymPy). En caso de que ambas expresiones sean ciertas, el servidor identificará a la ecuación diferencial como de primer orden homogénea, así que se enviará al módulo de control junto con la etiqueta correspondiente. La comparación se realizará mediante la verificación de caracteres consecutivos que componen a la ecuación diferencial.

Código	Descripción
--------	-------------

<b>RF088</b>	El servidor indicará que la ecuación diferencial es de primer orden exacta si se llega a la forma:
--------------	--

$$p(x, y)y' + q(x, y) = 0$$

Donde  $p(x, y)$  y  $q(x, y)$  son funciones en esencia diferentes que se expresan en términos de 'x' y 'y'. Una vez detectada una ecuación de esta forma, aplicará una validación de exactitud utilizando la validación que indica la siguiente ecuación:

$$\frac{\partial p}{\partial x} = \frac{\partial q}{\partial y}$$

Para ello se utilizará una derivación parcial sobre cada una de las funciones 'p' y 'q' con respecto a 'x' y 'y', respectivamente (esto con el apoyo del módulo diferenciador de SymPy). En caso de que ambas expresiones sean ciertas, el servidor identificará a la ecuación diferencial como de primer orden exacta, así que se enviará al módulo de control junto con la etiqueta correspondiente. La comparación se realizará mediante la verificación de caracteres consecutivos que componen a la ecuación diferencial.

<b>RF089</b>	El servidor indicará que la ecuación diferencial es de orden superior lineal con coeficientes constantes si se llega a la forma:
--------------	--

$$c_n y^{(n)} + c_{n-1} y^{(n-1)} + c_{n-2} y^{(n-2)} + \dots + c_1 y' + c_0 y = 0$$

Donde  $y^{(n)}$  indica la n-ésima derivada de 'y' con respecto a 'x'; y  $c(n)$  indica una constante numérica. Una vez detectada la ecuación de esta forma, se enviará al módulo de control con la etiqueta correspondiente. La comparación se realizará mediante la verificación de caracteres consecutivos que componen a la ecuación diferencial.

<b>RF090</b>	En el caso de que una misma ecuación pueda ser identificada como dos tipos de los anteriores, el servidor seleccionará aquel tipo que se haya identificado primero.
--------------	---

<i>Código</i>	<i>Descripción</i>
<b>RF091</b>	Si la ecuación puede ser identificada correctamente como alguna de las anteriores, el servidor agregará un primer paso a la solución el cual lleva por descripción que se identificó al tipo de ecuación diferencial de un determinado tipo, mientras que el cuerpo del paso se compone de la validación correspondiente que fue cierta para determinar el tipo de la ecuación diferencial. Todo esto se manda al módulo traductor, para pasar a formato LaTeX y luego ser recolectada por el módulo de serialización.
<b>RF092</b>	En caso de que no se encuentre ninguna coincidencia con los tipos anteriores, el servidor solicitará una intervención de SymPy, en particular del módulo DSolve para buscar clasificar a la ecuación diferencial. De los tipos que puede identificar SymPy se buscará que sean equivalentes a los que se plantean en el sistema.
<b>RF093</b>	En caso de que SymPy identifique a la ecuación diferencial como uno de los tipos soportados por el sistema, el servidor continuará con la solución mandando al módulo de control la ecuación y la etiqueta correspondiente a su tipo. En este caso, el servidor agregará el paso de identificación a la solución sin incluir el cuerpo (es decir, no se coloca la validación correspondiente para identificar a la ecuación de un determinado tipo).
<b>RF094</b>	En caso de que SymPy identifique el tipo de ecuación pero no corresponda con ninguno de los tipos soportados por el sistema, el servidor solicitará la intervención de SymPy para concluir con la resolución de la ecuación diferencial. En este caso, añadirá un primer paso a la solución que lleve por descripción: "El servidor no fue capaz de construir los pasos, pero pudo encontrar una solución", seguido de la solución que encuentre el módulo de SymPy a la ecuación diferencial. Bajo este escenario, la solución consta de únicamente ese paso.
<b>RF095</b>	En caso de que Sympy no sea capaz de identificar el tipo de ecuación diferencial, se desembocará una anomalía de clasificación (completitud). El servidor construirá un único paso en la solución indicando que la ecuación diferencial no pudo ser resuelta por el sistema. Se enviará la solución a la aplicación cliente con la etiqueta correspondiente al tipo de anomalía que fue detectado.

## 2.13. Módulo de control

<i>Código</i>	<i>Descripción</i>
<b>RF096</b>	El servidor tendrá en la memoria local una lista de los pasos que son necesarios para resolver un determinado tipo de ecuación diferencial. Mediante la etiqueta provista por el módulo de clasificación el servidor seguirá una de las secuencias de pasos y construirá la solución a partir de ella.
<b>RF097</b>	El módulo de control llevará el registro de cuáles pasos ya se realizaron y en qué estado fueron concluidos, así como un conteo de la dificultad acumulada de la solución. La dificultad global se calculará sumando la dificultad de cada uno de los pasos completados.
<b>RF098</b>	El módulo de control solo será capaz de llevar el flujo de solución de una ecuación a la vez, por lo que el resto de las solicitudes se colocarán en cola esperando a completar todos los pasos de una solicitud dada antes de comenzar a resolverse.
<b>RF099</b>	El módulo de control tendrá la intervención constante del módulo traductor para pasar a formato LaTeX los pasos contruidos y se enviarán al módulo de serialización para que sean agregados a la solución construida.
<b>RF100</b>	El módulo de control utilizará la intervención de SymPy para manipular algebraicamente las expresiones que resulten de los pasos anteriores. Las manipulaciones realizadas serán acorde a lo establecido en la solución de la ecuación diferencial en cuestión.
<b>RF101</b>	El módulo de control tendrá la intervención constante del módulo de integración en función a lo dictado por los pasos establecidos para el tipo de ecuación diferencial.

Código	Descripción
--------	-------------

**RF102** En el caso de que la ecuación diferencial sea de primer orden separable, la lista de pasos que seguirá el módulo de control será:

- > Expresar la derivada de la función 'y' en función de términos diferenciales ( $dy/dx$ ). Esto se realizará por medio de una sustitución de SymPy.
- > Restar a ambos lados de la ecuación la expresión  $q(y)dy/dx$ . Esto se realizará por medio de una resta de SymPy.
- > Multiplicar ambos lados de la ecuación por  $dx$  y simplificar el lado derecho. Esto se realizará por medio de una multiplicación de SymPy.
- > Integrar ambos lados de la ecuación. Esto se realizará por medio de la intervención del módulo de integración. Primero se integra el lado izquierdo y después el derecho.
- > Despejar (en caso de ser posible) la variable 'y' resultante de la integración.

**RF103** En el caso de que la ecuación diferencial sea de primer orden lineal, la lista de pasos que seguirá el módulo de control será por dos partes:

Obtener el factor de integración

- > Expresar que la derivada del producto del factor de integración ( $v$ ) con 'y' con respecto a  $x$  debe ser igual al propio factor multiplicado por la derivada de 'y' con respecto de 'x' más el producto de  $p(x)$ , ' $v$ ' y 'y'. Esto se realizará por medio de la creación de una nueva ecuación con SymPy que tenga estos elementos ( $p(x)$  será representado de acuerdo con su definición en la ecuación original).
- > Desarrollar el lado izquierdo por medio de la regla de la derivada del producto de un par de funciones, de manera que queda  $v*dy/dx + y*dv/dx$ . Esto se realizará mediante una sustitución del lado izquierdo con SymPy.
- > Cancelar los términos comunes por ambos lados de la ecuación (el término  $v*dy/dx$ ). Esto se realizará mediante una resta simbólica por ambos lados con SymPy.
- > Multiplicar ambos lados de la ecuación por  $dx$  y simplificar el lado derecho. Esto se realizará por medio de una multiplicación simbólica con SymPy.
- > Dividir por ambos de la ecuación  $v$ , de modo que queda  $dv/v$  del lado izquierdo y  $p(x)dx$  del otro. Esto se realizará por medio de la división simbólica con SymPy.

- > Integrar ambos lados de la ecuación, de modo que el lado derecho quede como  $\ln(v)$  y el izquierdo se realiza mediante la intervención del módulo de integración.
- > Despejar  $v$  mediante la exponenciación con base natural por ambos lados de la ecuación, de modo que el lado izquierdo queda ' $v$ ' y del lado derecho ' $e$ ' elevado al resultado de la integral del paso anterior. Para esto se utilizará la exponenciación de SymPy.

**RF104** La segunda parte de los pasos de la solución de una ecuación diferencial de primer orden lineal tendrá los siguientes pasos:

- > Se multiplica la ecuación original por  $v(x)$  (factor de integración) en ambos lados. Se utilizará la multiplicación simbólica de SymPy para ello.
- > Se reemplazan los primeros dos términos del lado izquierdo por  $d(vy)/dx$ , al ser la condición bajo la que se creó el factor. Esto se realizará mediante una sustitución con SymPy.
- > Restar a ambos lados de la ecuación el término  $q(x)v(x)$ . Esto se realizará por medio de la resta simbólica de SymPy.
- > Multiplicar ambos lados de la ecuación por  $dx$ . Esto se realizará mediante la multiplicación simbólica de SymPy.
- > Integrar ambos lados de la ecuación, de modo que el lado izquierdo queda como  $vy$  mientras que el derecho queda el negativo de la integral de  $q(x)v(x)dx$ . Esta parte se resolverá por medio del módulo de integración.
- > Dividir ambos lados de la ecuación por  $v(x)$ , para obtener finalmente el valor de  $y(x)$  del lado izquierdo. Este paso se realizará por medio de la división simbólica de SymPy.

Código	Descripción
--------	-------------

<b>RF105</b>	<p>En el caso de que la ecuación diferencial sea de primer orden homogénea, la lista de pasos que seguirá el módulo de control será la siguiente:</p> <ul style="list-style-type: none"> <li>&gt; Se plantea el cambio de variable mediante la ecuación <math>y = ux</math>. Esto se hará por medio de una nueva ecuación de SymPy.</li> <li>&gt; Se derivada la igualdad anterior de modo que queda <math>dy/dx = x*du/dx + u</math>. Esto se realiza mediante la diferenciación de ambos lados con SymPy.</li> <li>&gt; Se realiza la sustitución de la derivada de 'y', y de 'y' con base a las ecuaciones anteriores. Esto se realiza por medio de la sustitución simbólica de SymPy.</li> <li>&gt; Se factoriza 'x' de los términos contenidos en <math>p(x, vx)</math>, <math>q(x, vx)</math>. Esto se realiza por medio de la herramienta de factorización de SymPy.</li> <li>&gt; Se eliminan los términos factorizados previamente mediante la división simbólica de SymPy, de modo que queda todo en términos de <math>v</math> y un producto separable de la forma <math>x/dx</math></li> <li>&gt; Expandir el producto <math>p(1, v) * (x*dv/dx + v)</math> del lado izquierdo. Esto se realizará por medio de la multiplicación simbólica de SymPy.</li> <li>&gt; Dejar de un solo lado de la igualdad <math>p(1, v) * x * dv/dx</math> pasando a restar el resto del lado izquierdo al derecho. Para ello se usará la resta simbólica de SymPy.</li> </ul>
--------------	--



Código	Descripción
--------	-------------

**RF106** En el caso de que la ecuación diferencial sea exacta, la lista de pasos que seguirá el módulo de control será la siguiente:

- > Se factorizan todos los términos que contengan el diferencial  $dy/dx$ . Esto se realiza por medio de la herramienta de factorización de SymPy.
- > Se manipula la expresión hasta que quede de la forma  $M+N(dy/dx) = 0$ , esto es dejando todos los términos de un lado de la ecuación ya sea restando o sumando. Para ello se usará la resta y suma simbólica de SymPy.
- > Se integra  $M$  respecto a  $x$ , manteniendo  $y$  como constante. Dejando la constante de integración indicada como función de  $y$ . Esta parte se resolverá por medio del módulo de integración.
- > Se deriva la expresión anterior respecto a  $y$ . Esto se realiza mediante la diferenciación con SymPy.
- > Se iguala la derivada de la constante obtenida a los términos de  $N$  que no contienen  $x$ . Esto se hará por medio de una nueva ecuación de SymPy.
- > Se integra la constante respecto a  $y$ . Esta parte se resolverá por medio del módulo de integración.
- > Se sustituye el valor de la constante en el resultado de la integral de  $M$  respecto a  $x$ . Esto se realiza por medio de la sustitución simbólica de SymPy.

**RF107** En caso de que se use Laplace para el caso en el que la ecuación diferencial sea lineal, la lista de pasos que seguirá el módulo de control será:

- > Aplicar la transformada de Laplace sobre cada uno de los términos de la ecuación diferencial. Para esto se utilizará lo expresado en el apartado de "La Transformada de Laplace" del módulo de integración RF150 - RF153
- > A partir de la ecuación generada, despejar " $L\{s\}$ ", de modo que quede expresada explícitamente como función de  $s$ . Esto se realizará por medio de factorizaciones, sumas, restas y multiplicaciones por parte de SymPy.
- > Reescribir la expresión de  $L\{s\}$  de forma que quede en términos de funciones que tengan una definición directa en la transformada de Laplace, para aplicar la inversa. Para esto se utilizará lo expresado en el apartado de "La Transformada Inversa de Laplace" del módulo de integración

> Aplicar la transformada inversa de Laplace sobre ambos términos de la ecuación. Para esto se utilizará lo expresado en el apartado de “La Transformada Inversa de Laplace” del módulo de integración.

> En caso de ser posible, despejar la función ‘y’ para dejarla explícitamente en términos de ‘x’. Esto se realizará por medio de factorizaciones, sumas, restas y multiplicaciones por parte de SymPy.

**RF108** A medida que se realicen los pasos en la solución correspondiente se comenzará a acumular de manera aditiva la dificultad, de modo que cada paso completado aporta una dificultad global acorde a la dificultad de cada uno de los subpasos que fueron necesarios para llevarlo a cabo (las dificultades se expresan en (RNF 093). Esta medida se almacenará durante todo el tiempo que la solicitud esté siendo atendida por el servidor.

**RF109** En caso de que la dificultad supere la barrera de dificultad máxima (RNF-094), el servidor terminará el flujo de solución de la ecuación bajo el concepto de anomalía de completitud. El servidor solicitará entonces la intervención del módulo DSolve de SymPy para llegar a la solución de la ecuación sin obtener los pasos realizados para llegar a la misma.

**RF110** En caso de que el módulo de SymPy sea capaz de resolver la ecuación original, se añadirá un paso extra en la solución incompleta de la ecuación que contenga la respuesta a la que llegó DSolve acompañada de la descripción correspondiente de que el servidor encontró que dicha función es solución de la ecuación diferencial más no pudo completar la solución hasta ella. La solución enviada al cliente tendrá entonces los pasos contenidos hasta antes del sobreflujo de dificultad y la solución encontrada por DSolve. En caso contrario, solo se manda la solución incompleta al cliente (no se agrega el paso).

**RF111** En el caso de que durante el proceso de solución de la ecuación diferencial el módulo de control no sea capaz de determinar cuál es el siguiente paso algebraico debido a un error interno o bien por desborde de dificultad generado por las búsquedas recursivas de completar alguno de estos, el servidor buscará la intervención de SymPy para llevar a cabo la tarea del modo más directo posible utilizando las funciones correspondientes para la tarea.

<i>Código</i>	<i>Descripción</i>
<b>RF112</b>	En caso de que el SymPy sea capaz de completar el paso para el cual fue invocado, se continuará con el flujo normal de solución del problema considerando que la dificultad global permanecerá igual a cómo quedó tras el sobreflujo de dificultad del paso (dicho de otra forma: la intervención de SymPy tendrá un coste de dificultad igual al máximo de dificultad permitido para un paso de este tipo).
<b>RF113</b>	En el caso de que SymPy no sea capaz de completar el paso algebraico para el cual fue invocado, se terminará el flujo de solución de la ecuación, lanzando una anomalía interna de incompletitud de paso. Esta anomalía será tomada por el mismo servidor y realizará los mismos pasos realizados para el caso de anomalía de incompletitud generada por el sobreflujo de dificultad (se utiliza DSolve para resolver la ecuación, y sus dos casos posibles).
<b>RF114</b>	Cuando el módulo de control requiera de una intervención del módulo de integración permanecerá en modo de espera en lo que el módulo de integración consigue devolver la integral solicitada con sus respectivos subpasos. Dentro del modo de espera el módulo de control no se podrán recibir nuevas solicitudes de solución ni tampoco se realizarán tareas en paralelo a este hilo de solución.
<b>RF115</b>	En caso de que el módulo de integración retorne una "no solución" (no pudo resolverse) de la integral solicitada, el módulo de control mandará una anomalía de incompletitud bajo el concepto de la no solución de un paso, por lo que se seguirá con los pasos estipulados en RF110 - RF112. (Solicitar intervención de SymPy, en caso favorable se sigue con la solución, de otra forma se lanza la anomalía de manera general y se busca la solución general de la ecuación diferencial).
<b>RF116</b>	En caso de que el módulo de integración retorne exitosamente el paso para el que fue invocado, el módulo de control mandará el paso con sus correspondientes subpasos al módulo traductor para su formateo y posterior almacenamiento.
<b>RF117</b>	La memoria de dificultad del módulo deberá guardar el respectivo puntaje de cada uno de los pasos y subpasos, guardando la relación entre ellos por medio de un identificador único, de modo que sea posible distinguir cuáles subpasos corresponden a cada paso y la dificultad de cada uno de ellos.

<i>Código</i>	<i>Descripción</i>
<b>RF118</b>	La memoria del módulo de control deberá almacenar los patrones de cadenas esperados para cada uno de los subpasos posibles, de manera que solo se requerirá de algunas sustituciones sobre la misma para llegar a la descripción del subpaso específico (i.e. "Se integra el lado izquierdo {1} de la ecuación", donde {1} representa el espacio a sustituir con el valor del lado izquierdo de la ecuación).
<b>RF119</b>	En el caso de que el módulo consiga dar con la solución de la ecuación diferencial y el tipo de usuario que realizó la solicitud sea estudiante, se terminará el flujo de solución y el módulo de control pasará a modo de espera de la siguiente solicitud de ecuación (en caso de haber fila de espera, tomará la siguiente solicitud de la fila).
<b>RF120</b>	<p>En el caso de que el módulo consiga dar con la solución de la ecuación diferencial y el tipo de usuario que realizó la solicitud sea docente o investigador, el módulo de control tendrá 4 tareas adicionales antes de concluir con el flujo del programa:</p> <ul style="list-style-type: none"> <li>&gt; Encontrar la gráfica de la familia de curvas de solución.</li> <li>&gt; Encontrar las raíces de la familia de curvas</li> <li>&gt; Encontrar los puntos críticos de la familia de curvas.</li> <li>&gt; Encontrar los puntos de inflexión de la familia de curvas.</li> </ul> <p>Las indicaciones para cada una de estas tareas deberán quedar almacenadas nuevamente en el módulo de control.</p>
<b>RF121</b>	Para encontrar la gráfica de la familia de curvas de la solución, el módulo de control utilizará las librerías de NumPy y Matplotlib. Se realizará la gráfica de la solución por medio de una variación del parámetro arbitrario obtenido por la integración (lo mismo aplica para varios parámetros) sustituyendo diferentes valores constantes para generar una curva solución. Se realizará un barrido de 10 variaciones del parámetro para construir el gráfico de las curvas solución.
<b>RF122</b>	El gráfico de la familia de curvas que se obtenga mediante Matplotlib será almacenado en una carpeta de archivos temporales dentro del mismo módulo. Dicha imagen será construida con los métodos de configuración de Matplotlib correspondientes.
<b>RF123</b>	Una vez construida la imagen, el módulo de control mandará una señal al módulo traductor indicando la dirección donde fue alojada la imagen para su posterior formateo.

Código	Descripción
--------	-------------

- |              |   |
|--------------|---|
| <b>RF124</b> | Para encontrar las raíces de la familia de curvas, el módulo de control utilizará SymPy para resolver la ecuación $y(x) = 0$ , donde 'y' es la solución de la ecuación diferencial. Quedará en términos del parámetro arbitrario constante. En caso de que no sea posible dar con la solución, se utilizará el módulo de solución numérica forzada para encontrar valores aproximados por medio de métodos numéricos para las raíces considerando los mismos 10 barridos utilizados para construir el gráfico.  |
| <b>RF125</b> | En caso de que fuera posible encontrar la expresión analítica de las raíces de la ecuación, se mandará al módulo traductor bajo la leyenda "Raíces analíticas de la ecuación". En caso contrario, se mandarán los 10 resultados obtenidos por medio de los métodos numéricos con sus respectivos valores de constantes tomadas bajo la leyenda "Raíces obtenidas bajo los valores: [lista de valores]".   |
| <b>RF126</b> | Para encontrar los puntos críticos de la familia de curvas, el módulo de control utilizará SymPy para resolver la ecuación $y'(x) = 0$ , donde 'y'' es la derivada de la solución de la ecuación diferencial. Quedará en términos del parámetro arbitrario constante. En caso de que no sea posible dar con la solución, se utilizará el módulo de solución numérica forzada para encontrar valores aproximados por medio de métodos numéricos para los puntos críticos considerando los mismos 10 barridos utilizados para construir el gráfico.           |
| <b>RF127</b> | En caso de que fuera posible encontrar la expresión analítica de los puntos críticos de la ecuación, se mandará al módulo traductor bajo la leyenda "Puntos críticos analíticos". En caso contrario, se mandarán los 10 resultados obtenidos por medio de los métodos numéricos con sus respectivos valores de constantes tomadas bajo la leyenda "Puntos críticos obtenidos bajo los valores: [lista de valores]".   |
| <b>RF128</b> | Para encontrar los puntos de inflexión de la familia de curvas, el módulo de control utilizará SymPy para resolver la ecuación $y''(x) = 0$ , donde 'y''' es la derivada de la solución de la ecuación diferencial. Quedará en términos del parámetro arbitrario constante. En caso de que no sea posible dar con la solución, se utilizará el módulo de solución numérica forzada para encontrar valores aproximados por medio de métodos numéricos para los puntos de inflexión considerando los mismos 10 barridos utilizados para construir el gráfico. |

<i>Código</i>	<i>Descripción</i>
<b>RF129</b>	En caso de que fuera posible encontrar la expresión analítica de los puntos de inflexión de la ecuación, se mandará al módulo traductor bajo la leyenda "Puntos de inflexión analíticos". En caso contrario, se mandarán los 10 resultados obtenidos por medio de los métodos numéricos con sus respectivos valores de constantes tomadas bajo la leyenda "Puntos de inflexión obtenidos bajo los valores: [lista de valores]".

## 2.14. Módulo de Integración

<i>Código</i>	<i>Descripción</i>
<b>RF130</b>	El módulo de integración deberá recibir exclusivamente una expresión a integrar y retornará una no solución (mensaje de incompletitud) o una solución que se compondrá exclusivamente de un paso integral.
<b>RF131</b>	El módulo de integración deberá guardar de manera dinámica los diferentes subpasos que se generen durante la búsqueda de la solución. Dichos subpasos deberán contener la expresión correspondiente a una cadena de ecuación de SymPy y una descripción asociada a lo realizado en este subpaso.
<b>RF132</b>	El módulo de integración deberá almacenar una serie de pasos integrales atómicos definidos en la memoria local del programa a manera de catálogo de integrales. Este catálogo será una réplica de un catálogo definido en una base de datos ajena al servidor, la cual tendrá un período diario de actualización.
<b>RF133</b>	El módulo de integración deberá almacenar de manera dinámica la dificultad asociada a cada subpaso y el valor de dificultad global. A medida que se construyan nuevos subpasos para completar la integral deberán actualizarse estos indicadores.
<b>RF134</b>	En caso de que el paso integral supere el límite de dificultad para un paso integral RNF-095, el módulo de integración retornará una no solución bajo el concepto de sobreflujo de dificultad en el paso local.
<b>RF135</b>	El flujo de trabajo del módulo de integración estará basado en realizar comparaciones de expresiones con el catálogo de integrales atómicas definidas e ir las acumulando como parte de una solución, por lo que se utilizarán pasos recursivos para explorar nuevas expresiones que puedan llegar a una forma atómica. Los pasos recursivos también son contados para la dificultad acumulada de acuerdo con lo establecido en (RNF-096). El catálogo de integrales almacenará también las definiciones de estos pasos recursivos y también serán replicados de lo establecido en la base de datos.
<b>RF136</b>	El módulo de integración utilizará la comparación de caracteres estricta para verificar la forma de la integral que se solicita; esto es, revisará que cada uno de los caracteres se encuentre según la forma esperada de la integral. El catálogo de las cadenas esperadas para cada una de las integrales es el <b>ANEXO 1</b> (integrales atómicas).

Código	Descripción
<b>RF137</b>	El módulo de integración comparará la cadena contra todos los tipos de integrales en el catálogo de integrales atómicas en el orden en el que se expresan. Si se encuentra una coincidencia, se aplica la integral correspondiente generando un subpaso que lleva por expresión algebraica el lado izquierdo el integrando bajo el signo integral y del derecho la solución correspondiente según el catálogo. Se agregará además la descripción correspondiente como una entrada extra al objeto que almacena a la cadena con la ecuación (Tupla con descripción y ecuación correspondiente). La dificultad del sub paso se añadirá a la cuenta global.
<b>RF138</b>	Si la petición actual ya no tiene ninguna integral pendiente, se regresan los subpasos acumulados hacia el módulo de control para constituir el paso integral que inicialmente fue solicitado. Esto marca el fin de la intervención del módulo de integración. Si se encuentra sobre una ramificación, se conservará la ramificación y pasará a quedarse guardada en el stock de ramificaciones completadas que son solución de la petición; y se pasará a la siguiente ramificación.
<b>RF139</b>	En caso de que la cadena no coincida con el patrón buscado, se buscará que sea de la forma $f(x) + g(x)$ , para verificar si puede ser descompuesta en la suma de dos integrales. En caso de ser posible, se resuelve cada una de ellas sobre la misma línea de flujo de solución (esto es, una misma solicitud se convierte en dos solicitudes ordenadas). Para cada una de las integrales generadas se repetirán los mismos pasos de solución como si fueran dos solicitudes diferentes, aunque la dificultad global sigue siendo la misma. Al realizar este cambio, se añadirá la dificultad global de acuerdo con lo establecido en RNF-097.
<b>RF140</b>	En caso de que la cadena no coincida con la separación en sumas de la integral, se realizará una búsqueda de la composición de factores funcionales de la expresión; esto es, se buscará descomponer a la función como el producto de una cantidad finita de funciones más simples. Esto se realizará mediante comparación forzada de caracteres y operaciones simbólicas de SymPy como la división y factorización. El resultado debe ser una expresión de la forma $f(x) = g_1(x)g_2(x)g_3(x)g_4(x)...g_k(x)$ .



<i>Código</i>	<i>Descripción</i>
<b>RF141</b>	Si no es posible encontrar al menos dos factores funcionales, el módulo solicitará indicará al módulo de control que ocurrió una anomalía de incompletitud, por lo que se regresará una "no solución" por parte de su intervención. En caso de que se encuentre se llegue a esta incompletitud sobre una rama y queden más ramas disponibles, solo se terminará la revisión de esta rama y se pasará a la siguiente. En caso de encontrarse sobre la última rama disponible, se lanzará la anomalía de incompletitud.
<b>RF142</b>	En caso de encontrar al menos un par de factores funcionales, el módulo deberá generar tantas ramas de integrales como combinaciones se puedan tomar para las parejas de factores funcionales (desde 0 factores hasta k). Cada una de estas ramas representa una posible transformación mediante la integración por partes que lleve a la solución de la ecuación (el límite de ramas permitidas será el indicado en (RNF-098).
<b>RF143</b>	En caso de que una determinada expresión pueda generar más ramas del límite establecido, se tomarán solamente las primeras combinaciones que se mantengan dentro del rango máximo, de modo que todos los nodos tendrán una cantidad de ramificaciones acotada de 0 hasta el máximo.
<b>RF144</b>	Cada una de las ramificaciones será guardada en la memoria del módulo de integración, de modo que pueden existir copias de los mismos nodos entre las diferentes ramificaciones. Al crear una serie de ramificaciones sobre un nodo, es necesario hacer tantas copias como ramificaciones se pretendan crear de todo el camino anteriormente recorrido para la solución.
<b>RF145</b>	Si sobre una ramificación se encuentra una incompletitud, se borrará toda la rama para liberar la memoria utilizada para nuevas ramas. El límite de ramas es para ramas activas (las eliminadas disminuyen la cuenta). No obstante, la dificultad acumulada al trabajar con las ramas no se eliminará de la cuenta global.
<b>RF146</b>	El efecto de construir una ramificación seleccionado una combinación de factores funcionales de una expresión tiene un coste de dificultad que será añadido a la cuenta global RNF-096
<b>RF147</b>	Si al trabajar sobre una ramificación se llega a un nodo (expresión) por la que ya se había pasado antes en la misma ramificación, decimos que entró en un ciclo y el módulo eliminará dicha ramificación.

<i>Código</i>	<i>Descripción</i>
<b>RF148</b>	Si se llega a una anomalía de incompletitud general (ya no quedan ramas activas posibles de ser completadas), el módulo verificará en el stock de ramificaciones completadas cual posee una menor dificultad total, y esa será la ramificación que se enviará como respuesta a la solicitud.
<b>RF149</b>	Si al trabajar sobre una ramificación se llega a un nodo (expresión) que ya existe en otra ramificación, decimos que una de las ramificaciones puede ser simplificada por medio de la otra, por lo que el módulo eliminará la ramificación que tenga mayor dificultad para llegar desde el nodo origen hasta el nodo común que fue encontrado.
<b>RF150</b>	Cada ramificación posee un límite de dificultad como está expresado en RNF-099, de modo que antes de agregar un nuevo nodo el módulo de integración deberá verificar que la dificultad actual se encuentre por debajo del máximo. En caso de que se encuentre por encima, el módulo dejará de examinar dicha rama y la eliminará de la memoria de ramas activas.
<b>RF151</b>	Los límites de dificultades serán exactamente los mismos independientemente del número de ramificación que se esté abordando o si se está generando una nueva solicitud de integral, como en el caso de $f(x) + g(x)$ . De este modo, el único indicador que posee memoria de todo lo que se ha realizado es el contador de dificultad global. Antes de realizar cualquier operación se verificará que se encuentra bajo la cota de dificultad máxima para todo el árbol de caminos (RNF-100). En caso de que se supere el límite, el módulo concluirá la búsqueda de la solución y mandará directamente la anomalía de incompletitud al módulo de control.
<b>RF152</b>	El módulo de integración se utilizará también para generar la transformada de Laplace de una expresión. Para ello, el módulo utilizará las definiciones atómicas de la transformada definidas en RNF-102. Se utilizará la comparación estricta de caracteres de las expresiones del catálogo con la forma que tiene la expresión solicitada en el orden en que aparecen. En caso de coincidir, aplicará la transformada expresada, de lo contrario pasará a la siguiente comprobación.
<b>RF153</b>	En caso de que la expresión no coincida con ninguno de los elementos definidos para la transformada de Laplace, el módulo retornará al módulo de control una señal de anomalía de incompletitud, para la cual el módulo de control solicitará la intervención de SymPy.

<i>Código</i>	<i>Descripción</i>
<b>RF154</b>	El módulo de integración se utilizará para obtener la transformada inversa de Laplace de una expresión. Para ello, el módulo utilizará las mismas definiciones que se utilizan para generar la transformada en primer lugar, solo que la verificación será con el otro elemento (el del otro dominio). Se utilizará la comparación estricta de caracteres de las expresiones del catálogo con la forma que tiene la expresión solicitada en el orden en que aparecen. En caso de coincidir, aplicará la transformada expresada, de lo contrario pasará a la siguiente comprobación.
<b>RF155</b>	En caso de que la expresión no coincida con ninguno de los elementos definidos para la transformada inversa de Laplace, el módulo retornará al módulo de control una señal de anomalía de incompletitud, para la cual el módulo de control solicitará la intervención de SymPy.

## 2.15. Módulo de Traducción

<i>Código</i>	<i>Descripción</i>
<b>RF156</b>	El módulo traductor recibirá una cadena en el formato con el que la trabaja SymPy. Solo podrá recibir una cadena a la vez.
<b>RF157</b>	El módulo traductor utilizará el módulo de LaTeX contenido en SymPy para traducir la cadena recibida a formato LaTeX.
<b>RF158</b>	El módulo traductor añadirá un encabezado LaTeX a la cadena generada por medio del módulo de SymPy, el cual incluirá el número de paso que se está expresando. Estos letreros serán obtenidos desde la solicitud de traducción por parte del módulo de control y solo se realizará la solicitud de los elementos correspondientes.
<b>RF159</b>	El módulo traductor añadirá una descripción LaTeX a la cadena generada por medio del módulo de SymPy, la cual incluirá el número de subpaso correspondiente. Estos letreros serán obtenidos desde la solicitud de traducción por parte del módulo de control y solo se realizará la sustitución de los elementos correspondientes.
<b>RF160</b>	El código LaTeX generado para una intervención del módulo corresponde exactamente a un paso, el cual será enviado al módulo de serialización para su almacenamiento y serialización con el resto de los pasos. El módulo no contendrá entonces memoria propia.
<b>RF161</b>	En caso de que el módulo de control solicite su intervención para mandar la gráfica de curvas, se utilizará un módulo externo de Python para pasar la imagen actual a un formato en Base64. La cadena resultante será etiquetada con el tag correspondiente en LaTeX para imágenes en Base64. Una vez en LaTeX, se añadirá una descripción de la imagen (i.e. Gráfica de la solución) y se enviará al módulo serializador como los otros pasos.

### 3. Requerimientos no funcionales

---

Código	Descripción
<b>RNF001</b>	Al iniciar la aplicación cliente por primera vez se mostrará una pantalla en donde se da la bienvenida a la aplicación y se solicita la autenticación por parte del usuario. La pantalla mostrará dos botones: iniciar sesión o crear una cuenta.
<b>RNF002</b>	Oprimiendo el botón de crear cuenta, se mostrará una pantalla con cuatro campos de texto y uno de selección. En el primer campo de texto se indicará el nombre del usuario, en el segundo campo un correo electrónico, en el tercero una contraseña y en el cuarto una verificación de la contraseña. En el campo de selección múltiple se debe seleccionar el tipo de usuario: Docente/Investigador o Estudiante. Existe además un botón para enviar los datos y una casilla para verificar si se desean mandar noticias de la aplicación al correo seleccionado.
<b>RNF003</b>	El nombre de usuario deberá tener de 6 a 30 caracteres de longitud. El nombre de usuario puede ser cualquier combinación de letras, números o símbolos.
<b>RNF004</b>	Solo se admitirán extensiones de correo @gmail, @hotmail, @yahoo, @live, @outlook para evitar la creación de cuentas temporales.
<b>RNF005</b>	La contraseña deberá ser tener de 6 a 30 caracteres de longitud.
<b>RNF006</b>	La aplicación podrá correr en los siguientes navegadores: Google Chrome, Mozilla Firefox, Microsoft Edge.
<b>RNF007</b>	La aplicación requerirá de un espacio en memoria máxima de 30Mb.
<b>RNF008</b>	Recién iniciada la sesión la aplicación pedirá permiso al usuario para tener acceso a la cámara, en caso de que el usuario rechace el permiso la aplicación no podrá tomar foto. Si el usuario intenta tomar una foto tras haber negado el permiso la aplicación volverá a pedir el permiso de acceso correspondiente de nuevo.

<i>Código</i>	<i>Descripción</i>
<b>RNF009</b>	Recién iniciada la sesión la aplicación pedirá permiso al usuario para tener acceso a los archivos del ordenador. En caso de que el usuario rechace el permiso la aplicación no podrá subir imágenes. Si el usuario intenta subir una imagen tras haber negado el permiso, la aplicación pedirá el permiso de acceso correspondiente de nuevo.
<b>RNF010</b>	Recién iniciada la sesión la aplicación pedirá permiso al usuario para aceptar las cookies requeridas por la aplicación, si el usuario no las acepta y continúa en el sitio se tomarán por aceptadas.
<b>RNF011</b>	El sistema debe asegurar que los datos estén protegidos del acceso no autorizado.
<b>RNF012</b>	La consulta a la base de datos para comprobar la existencia previa de un nombre de usuario y correo electrónico tendrá una duración máxima de 1 minutos en cuyo caso se reportará un error de acceso.
<b>RNF013</b>	El sistema debe ser capaz de operar adecuadamente con hasta 10,000 usuarios con sesiones concurrentes.
<b>RNF014</b>	Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menos de 2 segundos.
<b>RNF015</b>	Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.
<b>RNF016</b>	La compañía que provee el servicio de hosting de la base de datos será Heroku por tanto nos atenemos a las políticas que esta compañía imponga sobre la base de datos.
<b>RNF017</b>	El intermediario entre la aplicación web y la base de datos será Firebase por tanto nos atenemos a las políticas que el servicio de Firebase impone.
<b>RNF018</b>	Si el usuario muestra inactividad por más de 6 meses su cuenta será eliminada para evitar la saturación del sistema por cuentas falsas.
<b>RNF019</b>	La aplicación debe ser compatible con todas las versiones de Windows, desde Windows 7.
<b>RNF020</b>	El sistema se acogerá a las reglas de las licencias generales públicas (GNU), es decir será gratuito, código abierto en el que cualquiera podrá cambiar el software, sin patentes y sin garantías.

<i>Código</i>	<i>Descripción</i>
<b>RNF021</b>	El correo electrónico será utilizado únicamente para comunicar anuncios de la aplicación y operaciones de configuración de la cuenta.
<b>RNF022</b>	La base de datos seguirá un modelo no relacional, basado en almacenamiento JSON, debido a la estructura dinámica de la información que se busca almacenar.
<b>RNF023</b>	La base de datos tendrá un máximo de registros de 1000 ecuaciones en el historial, Una vez superado el límite, se eliminarán las más antiguas que no hayan sido fijadas. Si las 1000 están fijadas, ya no se guardarán las nuevas entradas en el historial.
<b>RNF024</b>	El trazado sobre el lienzo será de un solo grosor (punto medio) y de color negro absoluto (#000000).
<b>RNF025</b>	Los archivos que sean adjuntados deberán ser de extensión jpg, png y jpeg únicamente y no se admitirá otro tipo de archivo.
<b>RNF026</b>	Las imágenes adjuntas podrán tener una resolución máxima de 1920x1080 píxeles.
<b>RNF027</b>	Será posible subir un solo archivo adjunto a la vez y su tamaño no deberá exceder los 20Mb.
<b>RNF028</b>	El tiempo de subida de los archivos que se adjunten no deberá exceder de 1 minuto.
<b>RNF029</b>	Solo se manejan 1800 peticiones por minuto, de lo contrario se retrasarán dichas peticiones.
<b>RNF030</b>	Se podrán manejar hasta 16 imágenes por petición
<b>RNF031</b>	El objeto JSON utilizado para enviar la petición a la API vision no deberá exceder de los 10Mb.
<b>RNF032</b>	La imagen no será utilizada para ningún propósito distinto al de proveer el servicio de detección de texto dentro de la misma.
<b>RNF033</b>	El sistema solo acepta una única ecuación diferencial en la barra de entrada.

<i>Código</i>	<i>Descripción</i>
<b>RNF034</b>	El sistema solo resolverá los tipos de ecuaciones diferenciales siguientes: EDO, Lineales, Homogéneas y de orden n.
<b>RNF035</b>	El sistema sólo reconocerá la palabra reservada D(). Como derivada de una función al interpretar la imagen.
<b>RNF036</b>	El usuario deberá ingresar la ecuación sin errores matemáticos de lo contrario se producirá un error. (ej: "++", "+-", etc..)
<b>RNF037</b>	Las ecuaciones deberán ser de solo 2 variables y se identificarán como 'y' y 'x', cualquier otra letra será tomada como constante.
<b>RNF038</b>	La variable "y", será tomada como la función a encontrar
<b>RNF039</b>	Todas las ecuaciones ingresadas deberán contener una y solo una vez el signo "=".
<b>RNF040</b>	Las ecuaciones deberán contener un máximo de 100 caracteres.
<b>RNF041</b>	Cada paso dentro de la resolución de la ecuación tendrá un tiempo máximo de 30 seg, por tanto el tiempo máximo de espera para la resolución será la suma de los tiempos de cada paso.
<b>RNF042</b>	La ecuación escrita deberá ser traducida a formato latex y deberá cumplir con los estándares latex.
<b>RNF043</b>	Al presionar alguna de las teclas del primer grupo, se concatenan los caracteres en cuestión a la cadena que se tenga en ese momento en el cuadro de texto editable, en la parte donde se encuentra el cursor. El cursor termina en la última posición de la secuencia ingresada, de modo que se distinga la cadena "izquierda", la cadena "ingresada" y la cadena "derecha".
<b>RNF044</b>	Al presionar las teclas de movimiento del cursor, la actual posición del cursor se moverá exactamente un carácter hacia la dirección indicada por la tecla en cuestión (derecha o izquierda)
<b>RNF045</b>	Al hacer click en alguna parte del campo de texto editable el cursor viajará a la parte de la cadena en la cual se realizó la acción. En caso de que se haga click al final de la cadena, el cursor se irá hacia el final del mismo campo de texto



Código	Descripción
--------	-------------

<b>RNF046</b>	<p>Para el caso del uso de una computadora, no será válida la escritura por medio del teclado de las expresiones definidas por el teclado mostrado en la pantalla. Sin embargo, existirá un mecanismo de "hotkeys" para el uso del teclado al escribir la ecuación:</p> <p>Tecla: Botón correspondiente del teclado</p> <p>0, 1, ..., 9: '0', '1', '2', ..., '9'</p> <p>x, y, e: 'x', 'y', 'e'</p> <p>Alt + p: 'pi'</p> <p>Alt + c: 'cos'</p> <p>Alt + s: 'sin'</p> <p>Alt + t: 'tan'</p> <p>Alt + l: 'ln'</p> <p>Alt + Shift + s: 'sec'</p> <p>Alt + Shift + c: 'csc'</p> <p>Alt + Shift + t: 'cot'</p> <p>Alt + Shift + a: 'asin'</p> <p>Alt + Shift + o: 'acos'</p> <p>Alt + Shift + 'n': 'atan'</p> <p>+, -, *, /, ^: '+', '-', '*', '/', '^'</p> <p>(, ), {, }, [, ]: '(', ')', '{', '}', '[', ']'</p> <p>=: '='</p> <p>., ', ', ..: '.', "'", '..: ''</p> <p>Left arrow: 'Mover cursor a la izquierda'</p> <p>Right arrow: 'Mover el cursor a la derecha'</p> <p>Remove: 'Eliminar el último carácter'</p> <p>Alt + Rem: 'Eliminar toda la entrada'</p> <p>Alt + Enter: 'Enviar ecuación'</p>
---------------	---

Código	Descripción
<b>RNF047</b>	<p>Referente a las pantallas de anomalías, la anomalía de tiempo incluirá los siguientes elementos:</p> <p>Anomalía de tiempo: El servidor tardó más de lo esperado en dar una respuesta para la ecuación diferencial que fue ingresada. Para continuar, se aconseja que siga las siguientes recomendaciones:</p> <p>Verifique que su dispositivo tenga una conexión estable a internet. Las conexiones deficientes pueden afectar la comunicación con el servidor  Reinicie la aplicación e intente ingresar nuevamente la ecuación.  Intente la vía de ingreso por teclas (III): es la más sencilla de procesar por parte del servidor</p>
<b>RNF048</b>	<p>Referente a las pantallas de anomalías, la anomalía de interpretación incluirá los siguientes elementos:</p> <p>Anomalía de interpretación: El servidor no pudo identificar correctamente la entrada como una ecuación diferencial. Para continuar, se aconseja que siga las siguientes recomendaciones</p> <p>Verifique que la ecuación diferencial haya sido escrita correctamente. En caso de utilizar alguna vía de entrada por imagen (I, II) asegúrese de que la ecuación esté completamente capturada y lo más enfocada posible; también verifique que la pantalla de previsualización coincida con la ecuación que desea resolver  Verifique que el formato de la ecuación diferencial sea el aceptado por la aplicación  En caso de que lo anterior no resulte, utilice la entrada por teclas (III): es la más sencilla de identificar por parte del servidor.  Reinicie la aplicación e intente ingresar nuevamente la ecuación por la vía III</p>

Código	Descripción
--------	-------------

**RNF049** Referente a las pantallas de anomalías, la anomalía de conexión incluirá los siguientes elementos:

Anomalía de conexión: No ha sido posible hacer contacto con el servidor para resolver la ecuación diferencial. Para continuar, se aconseja que siga las siguientes recomendaciones:

Verifique que su dispositivo esté conectado a internet. Asegúrese, además, de que la conexión sea estable.

Reinicie la aplicación e intente ingresar nuevamente la ecuación.

Espere al menos 10 minutos e intente nuevamente. En caso de que aún se encuentre desconectado, comuníquese con el equipo para reportar el caso.

**RNF050** Referente a las pantallas de anomalías, la anomalía de completitud incluirá los siguientes elementos:

Excepción de completitud: El servidor no ha sido capaz de concluir la solución para la ecuación diferencial presentada. El servidor ha podido realizar el siguiente avance en la solución:

[Aquí iría el avance parcial de la solución que pudo encontrar el servidor]

Para continuar, se aconseja que siga las siguientes recomendaciones:

Utilice la entrada de la ecuación por la vía III; es la entrada más simple de procesar para el servidor.

Realice el desarrollo de los primeros pasos (algebraicos) que se presentan en la solución parcial encontrada por el servidor e intente nuevamente con la nueva expresión: es posible que en esta ocasión se agreguen los pasos necesarios para dar con la solución de la ecuación

Utilice un software externo para concluir con la solución de la ecuación diferencial (se recomienda el uso de Wolfram Alpha para ello).

<i>Código</i>	<i>Descripción</i>
<b>RNF051</b>	El cuadro de texto de la vista previa de la ecuación se ajustará al tamaño del dispositivo.
<b>RNF052</b>	El tiempo de intervención de la API Google Vision no deberá ser mayor a 10 segundos.
<b>RNF053</b>	En el contexto del primer grupo de validaciones(formato básico), la cadena deberá tener exactamente un solo signo de igualdad.
<b>RNF054</b>	En el contexto del primer grupo de validaciones(formato básico), deberán existir caracteres antes y después del signo de igualdad.
<b>RNF055</b>	En el contexto del primer grupo de validaciones(formato básico), la cadena tendrá sólo los caracteres anteriormente permitidos.
<b>RNF056</b>	En el contexto del segundo grupo de validaciones(formato lógico), cada operador de jerarquía (paréntesis, por ejemplo) debe tener exactamente una apertura y una clausura, específicamente en ese orden.
<b>RNF057</b>	En el contexto del segundo grupo de validaciones(formato lógico), cada operador deberá encontrarse entre dos expresiones.
<b>RNF058</b>	En el contexto del segundo grupo de validaciones(formato lógico), cada letra en la expresión solo puede ser acompañada por exactamente un número posterior (exponente) y uno predecesor (coeficiente).
<b>RNF059</b>	En el contexto del tercer grupo de validaciones(formato técnico), el operador "derivada" será sólo aplicable a la variable "y".
<b>RNF060</b>	En el contexto del tercer grupo de validaciones(formato técnico), la variable "y" representa siempre la función buscada.
<b>RNF061</b>	En el contexto del tercer grupo de validaciones(formato técnico), la variable "x" representa siempre la variable de la que depende la función buscada (i.e. "y").
<b>RNF062</b>	En el contexto del tercer grupo de validaciones(formato técnico), las derivadas se connotan por medio de la notación de primas(apóstrofes).
<b>RNF063</b>	La cadena que devuelve el módulo de interpretación se enviará en formato JSON.

<i>Código</i>	<i>Descripción</i>
<b>RNF064</b>	El archivo JSON enviado deberá pesar un máximo de 5 Kbyte.
<b>RNF065</b>	El identificador de la sesión deberá ser un número entero positivo
<b>RNF066</b>	La fecha y hora de las solicitudes realizadas se obtendrán de acuerdo con la zona horaria del cliente.
<b>RNF067</b>	El sistema durará un máximo de 2 min en la fase de pasiva.
<b>RNF068</b>	En el contexto de la cadena en formato LaTeX del módulo de formateo, la cadena debe encontrarse entre llaves de peso (\$).
<b>RNF069</b>	En el contexto de la cadena en formato LaTeX del módulo de formateo, a cada operador matemático le corresponde un par de llaves ({}), de tal forma que el operador quedará dentro.
<b>RNF070</b>	En el contexto de la cadena en formato LaTeX del módulo de formateo, en el caso de los operadores de división, exponenciación, trigonométricos, anti-trigonométricos y logarítmicos serán reemplazados por los correspondientes en LaTeX.
<b>RNF071</b>	En el contexto de la cadena en formato LaTeX del módulo de formateo, en caso de que la ecuación contenga llaves ({}), deberán ser reemplazadas con su correspondiente escape (\{, \}).
<b>RNF072</b>	La cadena LaTeX generada por el módulo de formateo se ajustará a la pantalla del usuario.
<b>RNF073</b>	En caso de aceptar la cadena LaTeX generada por el módulo de formateo, el servidor tardará un máximo de 10 segundos en devolver una respuesta.
<b>RNF074</b>	En caso de declinar la cadena LaTeX generada por el módulo de formateo, el sistema tardará un máximo de 30 segundos en regresar al menú de selección.
<b>RNF075</b>	La string que se mandará del módulo de formateo al módulo de despacho no puede exceder los 500 caracteres.
<b>RNF076</b>	El tiempo máximo de espera de la solicitud HTTP request (timeout) será de 1 min

<i>Código</i>	<i>Descripción</i>
<b>RNF077</b>	El tamaño máximo de la solicitud HTTP request será de 5 kb.
<b>RNF078</b>	La pantalla de espera deberá tener un tamaño relativo a la pantalla del usuario. (Esta pantalla será la misma para el reporte de anomalía).
<b>RNF079</b>	Las anomalías serán reportadas en un tiempo máximo de 1 min.
<b>RNF080</b>	La respuesta exportada a pdf no se descargará si el usuario no ha dado permiso a acceder a los archivos al inicio, en cuyo caso se volverá a pedir permiso.
<b>RNF081</b>	Los límites y criterios de la exportación de la respuesta a pdf se atiene a las características de la API utilizada para realizar dicha conversión.
<b>RNF082</b>	Al compartir una solución, la opción de compartir se bloqueará por 5 min si se realizan más de 5 intentos en los que el correo electrónico buscado no existe.
<b>RNF083</b>	En caso de mandar la información por correo electrónico, ésta no podrá ser mayor a 500MB.
<b>RNF084</b>	El sistema solo guardará el correo electrónico en caso de que el usuario así lo decida.
<b>RNF085</b>	En modalidad de investigador, solo se mostrará la solución gráfica en caso de que exista alguna.
<b>RNF086</b>	El conversor base 64 que se utilizará para reconstruir la imagen tiene una capacidad máxima de operación de 192MB.
<b>RNF087</b>	El módulo de serialización tendrá una memoria con una capacidad máxima de 5k bytes.
<b>RNF088</b>	La validez de las operaciones aritméticas estará determinada por los algoritmos de resolución de cada tipo de ecuación diferencial, así como el método utilizado en el módulo de algebra simbólica de SymPy.
<b>RNF089</b>	Una ecuación diferencial puede ser reconocida por el servidor como de un y solo un tipo.

Código	Descripción
<b>RNF090</b>	DSolve de SymPy dejara de buscar soluciones o coincidencias tras un periodo de 1 min (timeout). Esto está definido por el código fuente de DSolve.
<b>RNF091</b>	La notación de SymPy para las ecuaciones diferenciales desde LaTeX requiere: Eliminar los caracteres de escape \ Eliminar los delimitadores \$ Reemplazar las fracciones por diagonal Reemplazar $\exp^{\wedge}$ por $**$ Eliminar agrupadores $\{ \}$
<b>RNF092</b>	El sistema podrá resolver este tipo de ecuaciones diferenciales: Primer orden separable Primer orden lineal Primer orden homogénea Primer orden exacta Primer orden reducible a lineal Orden superior lineal
<b>RNF093</b>	Se usarán dos tipos de pasos para expresar la dificultad de la ecuación: Pasos elementales: Suma (1), resta (1), composición (1). Pasos algebraicos: Sustitución (7), expansión (7), simplificación (10), despeje (15).
<b>RNF094</b>	La ecuación tendrá una dificultad máxima de 10,000 puntos.
<b>RNF095</b>	Un paso integral tendrá una dificultad máxima de hasta 1,000 puntos.
<b>RNF096</b>	Se usarán 20 puntos de dificultad para generar un paso recursivo del tipo integración por partes.
<b>RNF097</b>	La dificultad asociada a una descomposición por suma de funciones en un subpaso integral $f(x) = g(x) + h(x)$ tendrá una dificultad de 5 puntos.
<b>RNF098</b>	La cantidad de ramas posibles del módulo de integración será 16 en caso de escala local y 4096 en caso de escala local.

<i>Código</i>	<i>Descripción</i>
<b>RNF099</b>	La dificultad máxima que puede alcanzar una ramificación del módulo de integración es igual a la máxima de un paso integral (en este caso, 1000 puntos).
<b>RNF100</b>	La cantidad máxima de dificultad de un árbol de caminos 1024 veces la dificultad máxima de una ramificación.



## 4. Costo y materiales

### 4.1. Proyección de gastos con modelo COCOMO

Para la estimación de los costos del proyecto se ha considerado la evaluación en el modelo COCOMO ya que este es medianamente conocido, estandarizado y sencillo de aplicar.

Debido a la experiencia de los programadores en el equipo, así como el tamaño que tiene este con relación al proyecto, nuestro grupo de trabajo cae en la categoría de orgánico. Para realizar una estimación rápida utilizaremos el modelo básico, en el que los costos se aproximan en base al esfuerzo realizado.

Los parámetros para la categoría orgánica adoptan los siguientes valores:

Proyecto	a	b	C	d
Orgánico	2.4	1.05	2.5	0.38

Los cálculos y formulas se muestran como:

Parámetro	Formula	Resultado
Esfuerzo	$E = a(KLOC)^b$	$E = 2.4(10)^{1.05} = 27$
Tiempo	$t = c(E)^d$	$t = 2.25(27)^{0.38} = 7.87$
Personal requerido	$P = E/t$	$P = 27/7.87 = 3.43, t_2 = 9$

Donde KLOC son las kilo-líneas de código, y el tiempo esta medido en meses, vemos que el tiempo estimado es distinto tomando en cuenta que fuéramos más personas en el equipo, como en esta ocasión particular el equipo está conformado por 3 integrantes se utilizara la segunda tasa de tiempo aproximada.

Tomando en cuenta que el salario por hora de un programador promedio es de aproximadamente \$46.38 *mxn*, y tomando en cuenta 8 horas de trabajo diario obtenemos que el desarrollo del proyecto tiene un costo aproximado de:

$$Costo = 46.38 * (9) * (30.5) * 8 = \$101,850.48 \text{ mxn}$$

## 4.2. Materiales para el desarrollo

Podemos separar los materiales en hardware/recursos físicos y en recursos/tecnologías de software estos son:

<b>HARDWARE/FÍSICOS</b>	<b>SOFTWARE/TECNOLOGÍAS</b>
COMPUTADORA	Python
ELECTRICIDAD	Javascript
SERVICIO DE INTERNET	CSS
	HTML5
	React
	Firebase
	Google Vision
	Sympy
	Flask
	Postman
	GitHub
	Heroku

## 5. Actividades

---

- |     |  |
|-----|--|
| 001 | Desarrollar la pantalla de inicio (logueo)   |
| 002 | Desarrollar la pantalla de crear cuenta  |
| 003 | Desarrollar la validación de cuenta inexistente al crear una nueva                         |
| 004 | Desarrollar la validación de contraseñas válidas al crear la cuenta                        |
| 005 | Desarrollar la pantalla de inicio de sesión  |
| 006 | Desarrollar verificación de usuario existente al iniciar sesión                            |
| 007 | Desarrollar inicio automático con creación de cuenta exitosa                               |
| 008 | Desarrollar la validación de persistencia con archivos locales                             |
| 009 | Desarrollar la estructura del archivo de persistencia                                      |
| 010 | Desarrollar la pantalla del menú principal   |
| 011 | Desarrollar la funcionalidad de cerrar sesión  |
| 012 | Desarrollar la pantalla de configuración de la cuenta                                      |
| 013 | Desarrollar la sub-pantalla de cambiar contraseña  |
| 014 | Desarrollar interactividad de la sub-pantalla de cambio de contraseña                      |
| 015 | Desarrollar la verificación de nuevas contraseñas  |
| 016 | Desarrollar la verificación de nuevo correo y/o username                                   |
| 017 | Generar el cambio de credenciales sobre la base de datos                                   |
| 018 | Desarrollar la pantalla de historial   |
| 019 | Desarrollar la consulta y recepción de cada elemento del historial                         |
| 020 | Desarrollar la funcionalidad de re-consulta al hacer click sobre un elemento del historial |
| 021 | Desarrollar pantalla por defecto para el historial   |
| 022 | Desarrollar interacción de eliminar elemento del historial                                 |
| 023 | Desarrollar interacción de dar pin a un elemento del historial                             |
| 024 | Generar una consulta que reproduzca los elementos del historial en el orden correcto       |
| 025 | Generar evento para la eliminación del historial por tiempo                                |
| 026 | Generar evento de envío de correos en caso de una actualización en la base                 |
| 027 | Diseñar una base de datos de acuerdo con lo que se pretende almacenar                      |

<b>028</b>	Generar una base de datos de acuerdo con el diseño
<b>029</b>	Desarrollar pantalla de menú de opciones de resolución
<b>030</b>	Desarrollar pantalla de menú para ingreso por imagen
<b>031</b>	Desarrollar intervención de cámara para capturar imagen
<b>032</b>	Generar el almacenamiento temporal de la imagen de cámara
<b>033</b>	Desarrollar intervención de la galería para seleccionar imagen
<b>034</b>	Generar el almacenamiento temporal de la imagen de galería
<b>035</b>	Desarrollar pantalla de ingreso por dibujo
<b>036</b>	Desarrollar la interacción de dibujar sobre el lienzo
<b>037</b>	Desarrollar la interacción de eliminar último trazo
<b>038</b>	Desarrollar la interacción de eliminar el lienzo y regresar al menú de opciones
<b>039</b>	Generar el almacenamiento temporal de la imagen generada en el lienzo
<b>040</b>	Desarrollar pantalla de previsualización de imagen a manejar
<b>041</b>	Desarrollar algoritmo de normalización de la imagen ingresada
<b>042</b>	Generar una instancia de Google API Vision para el uso de la aplicación
<b>043</b>	Solicitar la intervención de la instancia de Vision para la imagen
<b>044</b>	Recibir respuesta de API Vision y mandar al módulo de interpretación
<b>045</b>	Desarrollar pantalla de ingreso por teclado
<b>046</b>	Desarrollar interacción de escritura de ecuación por teclas
<b>047</b>	Generar guardado de cadena por teclado y mandar al módulo de interpretación
<b>048</b>	Desarrollar las validaciones de la cadena interpretada por la recepción
<b>049</b>	Desarrollar la anomalía de interpretación
<b>050</b>	Desarrollar el lanzamiento de una anomalía de interpretación por interpretación errónea
<b>051</b>	Desarrollar pantalla de anomalía de interpretación
<b>052</b>	Desarrollar la comunicación entre los módulos de interpretación y formateo
<b>053</b>	Desarrollar un conversor de la string validada a LaTeX
<b>054</b>	Desarrollar pantalla de previsualización del LaTeX detectado
<b>055</b>	Integrar un módulo externo para compilar y mostrar LaTeX
<b>056</b>	Desarrollar interacción de aceptar y mandar LaTeX al módulo de despacho

<b>057</b>	Generar la serialización de los elementos en JSON
<b>058</b>	Guardar el JSON generado en el historial (sobre la base de datos)
<b>059</b>	Enviar la solicitud al servidor
<b>060</b>	Desarrollar la fase pasiva del módulo de despacho (timeout y wait)
<b>061</b>	Desarrollar la pantalla de fase pasiva del módulo de despacho
<b>062</b>	Desarrollar interrupción de fase pasiva por respuesta o anomalía
<b>063</b>	Desarrollar pantalla de anomalía genérica
<b>064</b>	Desarrollar anomalía de tiempo
<b>065</b>	Desarrollar el lanzamiento de anomalía de tiempo por timeout
<b>066</b>	Generar los modelos para guardar la solución
<b>067</b>	Deserializar la solución en los modelos
<b>068</b>	Desarrollar anomalía de completitud
<b>069</b>	Desarrollar deserialización parcial por anomalía de completitud
<b>070</b>	Desarrollar comunicación entre módulo de despliegue y presentación
<b>071</b>	Desarrollar la clasificación de anomalías por parte del servidor
<b>072</b>	Desarrollar pantalla de presentación
<b>073</b>	Desarrollar la funcionalidad de aceptar y cerrar en presentación
<b>074</b>	Desarrollar la funcionalidad de exportar a pdf en presentación
<b>075</b>	Integrar un módulo externo para compilar LaTeX a pdf
<b>076</b>	Desarrollar la interacción de guardar el pdf exportado
<b>077</b>	Desarrollar la funcionalidad de exportar a LaTeX en presentación
<b>078</b>	Desarrollar pantalla de exportación a LaTeX
<b>079</b>	Desarrollar la funcionalidad de compartir en presentación
<b>080</b>	Desarrollar pantalla de compartir
<b>081</b>	Gestionar correos de Firebase para mandar los documentos a compartir
<b>082</b>	Desarrollar verificación de documentos a compartir existentes
<b>083</b>	Desarrollar el volcado de información en anomalías sobre sus pantallas
<b>084</b>	Desarrollar la presentación de información de investigador
<b>085</b>	Desarrollar la recepción de cadenas JSON por parte del servidor
<b>086</b>	Desarrollar la deserialización del JSON recibido
<b>087</b>	Desarrollar estructura de fila de solicitudes al server
<b>088</b>	Implementar seguridad en la recepción de solicitudes del server
<b>089</b>	Desarrollar el directorio de solicitudes y direcciones del server

<b>090</b>	Desarrollar la serialización de la solución
<b>091</b>	Desarrollar el empaquetado adicional del modo investigador
<b>092</b>	Desarrollar la estructura de datos necesaria para almacenar los pasos acumulados
<b>093</b>	Desarrollar la anomalía de completitud desde el server
<b>094</b>	Desarrollar serialización de la anomalía de completitud desde el server
<b>095</b>	Desarrollar la interacción entre el clasificador y el traductor
<b>096</b>	Integrar SymPy a los módulos del servidor
<b>097</b>	Desarrollar la clasificación de la ecuación (general)
<b>098</b>	Desarrollar la clasificación de la ecuación (separable)
<b>099</b>	Desarrollar la clasificación de la ecuación (lineal orden 1 y reducible a lineal)
<b>100</b>	Desarrollar la clasificación de la ecuación (homogénea)
<b>101</b>	Desarrollar la clasificación de la ecuación (exacta)
<b>102</b>	Desarrollar la clasificación de la ecuación (lineal orden superior)
<b>103</b>	Desarrollar discriminante de orden (primer encuentro)
<b>104</b>	Desarrollar el paso 0: clasificación, y mandar al traductor
<b>105</b>	Gestionar la intervención de DSolve para clasificación forzada
<b>106</b>	Desarrollar el paso 0 incompleto: clasificación, y mandar al traductor
<b>107</b>	Gestionar la intervención de DSolve para resolución forzada si no hay soporte para el tipo
<b>108</b>	Desarrollar la anomalía de clasificación desde el server
<b>109</b>	Desarrollar la solución incompleta por anomalía de clasificación
<b>110</b>	Enviar solución incompleta por anomalía de clasificación al cliente
<b>111</b>	Desarrollar la estructura de datos necesaria para almacenar la lista de pasos por tipo
<b>112</b>	Desarrollar la estructura de datos necesaria para almacenar cuales pasos han sido realizados
<b>113</b>	Desarrollar la estructura de datos necesaria para generar la fila de solicitudes al control
<b>114</b>	Desarrollar la comunicación entre el módulo de control y traductor
<b>115</b>	Desarrollar las instrucciones algebraicas simples en forma de métodos de SymPy
<b>116</b>	Desarrollar la conexión entre el módulo de control e integración

<b>117</b>	Desarrollar las instrucciones en forma de métodos de SymPy para separables
<b>118</b>	Desarrollar las instrucciones en forma de métodos de SymPy para lineales de orden 1 (I)
<b>119</b>	Desarrollar las instrucciones en forma de métodos de SymPy para lineales de orden 1 red
<b>120</b>	Desarrollar las instrucciones en forma de métodos de SymPy para homogéneas
<b>121</b>	Desarrollar las instrucciones en forma de métodos de SymPy para exactas
<b>122</b>	Desarrollar las instrucciones en forma de métodos de SymPy para lineal de orden superior
<b>123</b>	Desarrollar la estructura de datos necesaria para almacenar la dificultad global de la solución
<b>124</b>	Desarrollar anomalía de completitud desde el server
<b>125</b>	Generar una anomalía de completitud si la solve alcanza el límite máximo de dificultad global
<b>126</b>	Gestionar la intervención de SymPy para la resolución de la ecuación por anomalía. completitud
<b>127</b>	Desarrollar la solución incompleta por anomalía de completitud con solución de SymPy
<b>128</b>	Desarrollar la solución incompleta por anomalía de completitud sin solución de SymPy
<b>129</b>	Solicitar la intervención de SymPy para completar un paso algebraico
<b>130</b>	Evaluar la intervención de SymPy para completar un paso algebraico
<b>131</b>	Retomar flujo de solución con instrucciones del módulo de control
<b>132</b>	Generar anomalía de completitud por error en intervención algebraica de SymPy
<b>133</b>	Desarrollar la espera por parte del módulo de control a la respuesta del módulo integrador
<b>134</b>	Gestionar la intervención de SymPy para la resolución de la integral por anomalía de completitud
<b>135</b>	Generar anomalía de completitud por error en intervención integral de SymPy
<b>136</b>	Gestionar la intervención del módulo traductor para guardar un paso integral

<b>137</b>	Implementar en la estructura de datos de la dificultad un guardado individual de dificultad
<b>138</b>	Desarrollar la estructura de control necesaria para almacenar los patrones de muestra de cada subpaso
<b>139</b>	Desarrollar la señal de fin de solución para el caso de solución completa (estudiante)
<b>140</b>	Desarrollar los pasos adicionales para el módulo de control para investigador
<b>141</b>	Integrar los módulos NumPy y Matplotlib al server
<b>142</b>	Generar los gráficos de la solución con NumPy y Matplotlib
<b>143</b>	Almacenar los gráficos generados por Matplotlib en memoria temporal del server
<b>144</b>	Desarrollar la señal de gráficos completos (modo investigador)
<b>145</b>	Desarrollar el algoritmo para encontrar las raíces de la solución (analítico)
<b>146</b>	Desarrollar el algoritmo para encontrar las raíces de la solución (numérico)
<b>147</b>	Desarrollar las señales de raíces completas (modo investigador)
<b>148</b>	Desarrollar el algoritmo para encontrar los puntos críticos de la solución (analítico)
<b>149</b>	Desarrollar el algoritmo para encontrar los puntos críticos de la solución (numérico)
<b>150</b>	Desarrollar las señales de puntos críticos completos (modo investigador)
<b>151</b>	Desarrollar el algoritmo para encontrar los puntos de inflexión de la solución (analítico)
<b>152</b>	Desarrollar el algoritmo para encontrar los puntos de inflexión de la solución (numérico)
<b>153</b>	Desarrollar las señales de puntos de inflexión completos (modo investigador)
<b>154</b>	Desarrollar los modelos necesarios para el manejo de nodos y secuencias de nodos
<b>155</b>	Desarrollar la estructura de datos necesaria para generar una fila en las solicitudes internas
<b>156</b>	Desarrollar la estructura de datos necesaria para almacenar cada nodo y secuencia de nodos
<b>157</b>	Gestionar una replicación del catálogo de los pasos integrales desde la base de datos hacia el server



<b>158</b>	Desarrollar la estructura de datos necesaria para almacenar los pasos integrales atómicos replicados
<b>159</b>	Generar el evento de actualización de réplica del catálogo diariamente
<b>160</b>	Desarrollar la estructura de datos necesaria para almacenar la dificultad global de la integral
<b>161</b>	Implementar en la estructura de datos del módulo integrador el almacenamiento de dificultad por subpaso
<b>162</b>	Desarrollar la anomalía de sobreflujo de dificultad en paso integral
<b>163</b>	Generar una anomalía de sobreflujo de dificultad en paso integral si se excede el límite de dificultad
<b>164</b>	Desarrollar la estructura de datos necesaria para almacenar los pasos integrales recursivos replicados
<b>165</b>	Desarrollar las comparaciones estrictas de caracteres para detectar la integral atómica desde el catálogo
<b>166</b>	Desarrollar la secuencia de comparación estricta sobre el catálogo y su salida
<b>167</b>	Desarrollar la validación de ausencia de solicitudes pendientes
<b>168</b>	Desarrollar el envío final de la solución en caso de ausencia de solicitudes pendientes
<b>169</b>	Desarrollar la comparación de partición por suma de integrales de la integral
<b>170</b>	Desarrollar la búsqueda de factores funcionales de la expresión a integrar (I)
<b>171</b>	Desarrollar la búsqueda de factores funcionales de la expresión a integrar (II)
<b>172</b>	Desarrollar la validación de cantidad de factores funcionales (min)
<b>173</b>	Generar anomalía de completitud si el conteo de factores funcionales es menor al mínimo
<b>174</b>	Desarrollar la estructura de datos necesaria para la gestión de las ramificaciones
<b>175</b>	Desarrollar un algoritmo para la generación de ramificaciones mediante integral por partes
<b>176</b>	Desarrollar la validación de cantidad de factores funcionales (máx.)
<b>177</b>	Desarrollar un algoritmo de corte para el caso de ramificaciones por encima del límite

<b>178</b>	Desarrollar las copias necesarias para cada uno de los nodos presentes en ramificaciones
<b>179</b>	Desarrollar un algoritmo para la eliminación de ramificaciones que alcancen la incompletitud
<b>180</b>	Desarrollar el algoritmo del reajuste de dificultad tras eliminación de ramificaciones
<b>181</b>	Desarrollar un algoritmo para la eliminación de ramificaciones que queden cicladas
<b>182</b>	Desarrollar la verificación de la cantidad ramas activas tras eliminar una rama
<b>183</b>	Desarrollar la selección de la rama completada de menor dificultad como solución al llegar a estudiar todas las ramas
<b>184</b>	Desarrollar un algoritmo para la eliminación de ramificaciones que puedan simplificarse con otra
<b>185</b>	Desarrollar la verificación de dificultad máxima que puede alcanzar una ramificación
<b>186</b>	Desarrollar un algoritmo para la eliminación de ramificaciones que alcancen su dificultad máxima
<b>187</b>	Desarrollar la verificación de dificultad máxima que puede alcanzar el conjunto de todas las ramificaciones
<b>188</b>	Generar una anomalía de completitud si se alcanza la máxima dificultad posible del árbol
<b>189</b>	Desarrollar la estructura de control necesaria para almacenar las transformadas de Laplace atómicas
<b>190</b>	Desarrollar las comparaciones estrictas de caracteres para detectar Laplace atómico desde el catálogo
<b>191</b>	Generar una anomalía de completitud si no se encuentra una transformada de Laplace conocida en la expresión
<b>192</b>	Desarrollar la estructura de control necesaria para almacenar las transformadas de Laplace inversas atómicas
<b>193</b>	Desarrollar las comparaciones estrictas de caracteres para detectar Laplace atómico desde el catálogo
<b>194</b>	Generar una anomalía de completitud si no se encuentra una transformada inversa de Laplace conocida en la expresión
<b>195</b>	Desarrollar la estructura de datos necesaria para que el módulo traductor forme filas de solicitudes

- |            |   |
|------------|---|
| <b>196</b> | Implementar el módulo LaTeX contenido en SymPy para traducir generar la ecuación LaTeX                  |
| <b>197</b> | Desarrollar un algoritmo para dar formato de encabezados LaTeX a las descripciones de los pasos         |
| <b>198</b> | Desarrollar un algoritmo para dar formato de sub encabezados LaTeX a las descripciones de los sub pasos |
| <b>199</b> | Desarrollar la comunicación entre el módulo traductor y el serializador                                 |
| <b>200</b> | Integrar un módulo para el formateo de imágenes en Base64   |
| <b>201</b> | Implementar el módulo de formateo en Base64 para representar en LaTeX la imagen codificada              |

# Capítulo IV:

## Codificación del servidor

# 1. Configuración de la API

---

## 1.1. Introducción

En esta sección se abordará la estructura básica de rutas y funciones que fueron desarrollados en el proyecto para montar un servidor de Flask en formato API. Hablaremos del desarrollo del archivo principal que corre en el servidor: el punto de entrada `api.py`; y también se da una perspectiva de la estructura de archivos del servidor.

## 1.2. Imports

El archivo comienza importando las funciones que van a ser utilizadas para cada una de las llamadas del archivo. Analizaremos cada bloque de imports por separado. En el primer bloque:

```
from flask import Flask, request, jsonify, after_this_request
from flask_cors import CORS
```

se importan las clases y métodos que provee Flask para la comunicación en formato API utilizando HTTP Requests y JSON como formato básico de la información manejada.<sup>0</sup>

Después tenemos los imports:

```
from anomalies.classification_anomaly import ClassificationAnomaly
from anomalies.completeness_anomaly import CompletenessAnomaly
```

que están asociados a las excepciones personalizadas que fueron desarrolladas para el proyecto. Estas excepciones son dos para el lado del servidor:

- La de **clasificación**, la cual es lanzada cuando el sistema no reconoce la ecuación diferencial como uno de los tipos permitidos.

- La de **completitud**, la cual es lanzada cuando el sistema no es capaz de resolver la ecuación diferencial. Existen diferentes formas en las que el sistema puede lanzar este tipo de excepción, las cuáles son atendidas en el apartado de excepciones del servidor.

En el siguiente bloque de imports:

```
from sympy import parse_expr, latex
from compiler.textopdf import text_to_pdf
```

se incluye el control de expresiones de SymPy para realizar conversiones entre los formatos LaTeX, SymPy string y Python String. Además, el segundo método importado fue desarrollado para obtener el pdf (compilado de LaTeX) de la solución que le es presentada al cliente de su ecuación diferencial. Se hablará más a detalle de su funcionamiento al abordar el módulo de compilación.

En el último bloque de imports:

```
from solvers.controller import solve
from parsers.parse_sympy import parseSympy, parseLatex
from vision.text_detection import detectText
from integrals.updater import get_client, write_atm_integrals,
write_rec_integrals
```

se incluyen los métodos destinados a las funciones representativas de los siguientes módulos:

- Controlador (destinado a resolver la ecuación diferencial).
- Traductor (destinado a traducir la ecuación de entrada en formato LaTeX y SymPy).
- Detección de Texto en Imagen (intervención de la API de Google Cloud para Vision Services).
- Integrador (parte del módulo destinada para atender el evento diario de actualización del catálogo).

## 1.3. Estructura del archivo api.py

Después de las llamadas para importar a los módulos externos, la estructura del archivo se distribuye en 3 secciones:

- Configurar el servidor de Flask
- Cabeceras de rutas y métodos asociados
- Inicializar el servidor de Flask

## 1.4. Configuración de CORS

El primero de estos bloques (el de configuración) luce como sigue:

```
app = Flask(__name__)  
CORS(app)
```

en el cual se aplican las configuraciones por defecto de CORS (Cross-Origin Resource Sharing) para la comunicación de agentes externos con la instancia del servidor Flask que generamos.

## 1.5. Rutas y métodos

La segunda sección de este archivo posee los métodos y rutas que podrá atender el servidor de Flask. La forma general con la que están escritos estos métodos es la siguiente:

```
@app.route("/route", methods = ["POST", "GET"])  
def function_name():  
    # function body
```

Para la API desarrollada en este proyecto, las rutas que son atendidas por nuestro servidor son:

- `/solve`: Acepta llamadas en GET y POST. Recibe la ecuación diferencial a resolver en LaTeX y regresa su solución paso a paso escrita en LaTeX.
- `/parse/latex`: Acepta llamadas en GET y POST. Recibe la ecuación a resolver y regresa el código LaTeX correspondiente a la misma.
- `/image/text`: Acepta llamadas en POST. Recibe una imagen codificada en Base64 y devuelve el texto interpretado en ella.
- `/pdf`: Acepta llamadas POST. Recibe el código LaTeX de la solución construida y devuelve el pdf de correspondiente codificado en Base64.
- `/update/atm`: Acepta llamadas en POST. Actualiza el catálogo de integrales atómicas.
- `/update/rec`: Acepta llamadas en POST. Actualiza el catálogo de integrales recursivas.



## 1.6. Ruta para obtener solución de ecuación

### /solve

Este método manda a llamar a la función `solve()` del módulo controlador. Obtiene la ecuación diferencial codificada en LaTeX a partir del body de la request realizada por parte del cliente. El método posee además protección para solicitudes malformadas ya sea por un JSON mal escrito o por la ausencia del campo buscado. Una vez llamado el proceso de solución, el módulo queda a la espera de alguna excepción, teniendo soporte especial para las propias del servidor (de clasificación y completitud) y un soporte genérico para otras excepciones no clasificadas. En cualquiera de estos casos, se envía una respuesta codificada en JSON al cliente en el cual se encapsula el resultado de la intervención del servidor para resolver la ecuación diferencial.

```
@app.route("/solve", methods = ["POST", "GET"])
def getSolve():
    # Check inputs
    jsonInput = request.get_json()
    if (jsonInput == None) :
        return jsonify({ "status": "error on json" })

    inputString = jsonInput["equation"]
    user_type = jsonInput["type"]

    if (inputString == None or user_type == None) :
        return jsonify({ "status": "error on string" })

    try:
        solution = solve(inputString, user_type)
        # No anomaly in solve call
        return jsonify({
            "status": "ok",
            "solution": str(solution)
        })

    # Classification error in solve call
    except ClassificationAnomaly as clsa:
```

```
        return jsonify({
            "exception": "classification",
            "status": clsa.args[0],
            "solution": str(clsa.final_solve)
        })

# Completeness error in solve call
except CompletenessAnomaly as ca:
    return jsonify({ "exception": "completeness",
        "status": ca.args[0],
        "solution": str(ca.partial_solution)
    })

# Unexpected error during execution of solve method
except Exception as e:
    return jsonify({ "exception": "generic",
        "status": e.args[0]
    })
```

## 1.7. Ruta para obtener LaTeX de ecuación

### `/parse/latex`

Este método manda a llamar a la función `parseSympy()` del módulo traductor, el cual recibe una string en el formato de texto solicitado al usuario desde el cliente para expresar su ecuación diferencial (el cual se obtiene en el body de la request en JSON) y entrega la expresión simbólica equivalente de la misma en un objeto manipulable por SymPy. Esta expresión es utilizada como entrada en el método `latex()` de SymPy, el cual devuelve la string correspondiente al LaTeX de la misma. El método tiene protección para malformaciones del JSON en la request y para alguna excepción que pueda generarse con la intervención de alguno de los dos métodos previamente mencionados.

```
@app.route("/parse/latex", methods = ["POST", "GET"])
def parseToLatex():
    # Check inputs
    jsonInput = request.get_json()
    if (jsonInput == None) :
        return jsonify({ "status": "error on json" })
    inputString = jsonInput["equation"]
    if (inputString == None) :
        return jsonify({ "status": "error on string" })

    try:
        equation = parseSympy(inputString)
        equationLatex = latex(equation)
    except Exception as e:
        return jsonify({ "status": e.args[0] })

    response = jsonify({ "equation": equationLatex + " = 0",
        "status": "ok"
    })

    return response
```

## 1.8. Ruta para obtener texto de imagen

### `/image/text`

Este método manda a llamar a la función `detectText()` del módulo de vision (detección de texto por imagen), el cual recibe una string que es la representación en Base64 de la imagen de la cual el cliente desea obtener el texto, la cual se obtiene en el body de la request en JSON. El método tiene protección para malformaciones del JSON en la request y para alguna excepción que pueda generarse con la intervención de la API de Google Vision en la llamada del método mencionado.

```
@app.route("/parse/latex", methods = ["POST", "GET"])
def getTextFromImage():

    @after_this_request
    def add_header(response):
        response.headers.add('Access-Control-Allow-Origin', '*')
        return response

    # Check inputs
    jsonInput = request.get_json()
    if (jsonInput == None) :    if (jsonInput == None):
        return jsonify({ "status": "error on json" })
    inputString = jsonInput["image"]
    if (inputString == None):
    if (inputString == None) :
        return jsonify({ "status": "error on string" })

    # Call function

    try:
        text = detectText(inputString)
        equation = parseSympy(inputString)
        equationLatex = latex(equation)
    except Exception as e:
        print(e.args[0])
        return jsonify({ "status": e.args[0] })
```

## 1.9. Ruta para obtener PDF de solución

### /pdf

Este método manda a llamar a la función `text_to_pdf()` del módulo de compilación, el cual recibe una string que es el texto en formato LaTeX de su solución, la cual se obtiene a su vez desde el body de la request en formato JSON. El método tiene protección para malformaciones del JSON en la request y para alguna excepción que pueda generarse con la intervención del módulo compilador de LaTeX.

```
@app.route("/pdf", methods = ["POST"])
def getPDFFromLatex():
    # Check inputs
    jsonInput = request.get_json()
    if (jsonInput == None):
        return jsonify({ "status": "error on json" })
    inputString = jsonInput["latex"]
    if (inputString == None):
        return jsonify({ "status": "error on string" })

    # Call function
    try:
        text = text_to_pdf(inputString)
    except Exception as e:
        print(e)
        return jsonify({ "status": e.args[0] })

    response = jsonify({ "text": text , "status": "ok" })
    return response
```

## 1.10. Ruta para actualizar catálogo atómico

### **/update/atm**

Este método manda a llamar a la función `write_atm_integrals()` del módulo de integración, el cual reescribe el catálogo de integrales atómicas del servidor. Para esta operación, se obtiene una instancia de un conector con Firebase que permita realizar la consulta; este cliente se obtiene con el método `get_client()` del mismo módulo. Es importante notar que el conector está definido como global para que se utilice la misma instancia a través de los métodos del archivo.

```
@app.route("/update/atm", methods = ["POST"])
def updateAtmIntegrals():
    print("Updating atomic integrals")
    global db
    if db is None:
        db = get_client()

    write_atm_integrals(db)
    return "200"
```

## 1.11. Ruta para actualizar catálogo recursivo

### **/update/rec**

Este método manda a llamar a la función `write_rec_integrals()` del módulo de integración, el cual reescribe el catálogo de integrales recursivas del servidor. Para esta operación, se obtiene una instancia de un conector con Firebase que permita realizar la consulta; este cliente se obtiene con el método `get_client()` del mismo módulo. Es importante notar que el conector está definido como global para que se utilice la misma instancia a través de los métodos del archivo.

```
@app.route("/update/rec", methods = ["POST"])
def updateRecIntegrals():
    print("Updating recursive integrals")
    global db
    if db is None:
        db = get_client()

    write_rec_integrals(db)
    return "200"
```

## 1.12. Iniciar el servidor

Finalmente, la última sección del archivo es la que echa a andar el servidor de Flask que ha sido configurado con las políticas de CORS y sus rutas:

```
if __name__ == "__main__":  
    global db  
    db = None  
    app.run(debug = True, port = 4000, host='26.142.66.43')
```

Vemos que además de configurar el host de nuestro servidor Flask también se inicializa nuestro conector de base de datos que estará gestionando los catálogos de integrales con Firebase. Es importante mencionar que este objeto debe ser configurado como global dentro de este archivo para que maneje la misma instancia dentro de todos los métodos que requieren de su intervención.



## 2. Controlador

---

### 2.1. Introducción

El módulo controlador es el encargado de llevar el flujo de las soluciones de cada uno de los tipos de ecuaciones diferenciales. Es por ello por lo que la intervención de este módulo se lleva en dos partes: Una etapa de clasificación y una de resolución. En este apartado hablaremos de la estructura general del archivo `controller.py` y en los siguientes apartados se dará mayor profundidad al estudio de cada parte del clasificador y de los solvers.

### 2.2. Imports

Se importan las excepciones personalizadas del sistema: La de completitud y la de clasificación.

```
from anomalies.classification_anomaly import ClassificationAnomaly
from anomalies.completeness_anomaly import CompletenessAnomaly
```

Luego se importan las librerías básicas para el manejo simbólico de SymPy

```
from sympy import *
from sympy.abc import x
```

En el siguiente bloque se importan 3 cosas:

- El método `parseLatex()` del módulo traductor, el cual genera una expresión simbólica a partir de una string en LaTeX
- Cada uno de los métodos de solución para los 6 tipos de ecuaciones diferenciales del sistema, los cuales son: `separable`, `linear`, `homogénea`, `exacta`, `reducible` y de `orden_n`

- El método `classify()` del módulo clasificador, el cuál identifica una ecuación diferencial como una de las 6 categorías antes mencionadas

```
from parsers.parse_sympy import parseLatex
from solvers.sys_solvers.solve_separable import solveSeparable
from solvers.sys_solvers.solve_linear import solveLinear
from solvers.sys_solvers.solve_homogeneous import solveHomogeneous
from solvers.sys_solvers.solve_exact import solveExact
from solvers.sys_solvers.solve_reducible_linear import
solveReducibleToLinear
from classifiers.classifier import classify
```

## 2.3. Estructura del archivo controller.py

El archivo `controller.py` es básicamente la definición de la función `solve()`, la cual espera dos parámetros:

- **inputString:** Se espera que sea una string que contenga la ecuación diferencial a resolver en formato LaTeX
- **user\_type:** Se espera que sea una string con alguno de los siguientes dos valores: "teacher", "student". El valor por defecto es "student". El método retorna un arreglo de pasos para resolver la ecuación diferencial. La estructura de este arreglo es, en formato JSON:

```

// Solución
[
  // Paso 1
  [
    // Título del Paso 1
    "Paso 1",
    // Subpasos del Paso 1
    [
      "Texto 1",
      "Ecuación 1",
      "Texto 2",
      "Ecuación 2",
      // ...,
      "Texto k",
      "Ecuación k"
    ]
  ],
  // Paso 2
  [
    // Título del Paso 2
    "Paso 2",
    // Subpasos del Paso 2
    [
      // ...,
    ]
  ],
  // ...,
  // Paso n
  [
    // Título del Paso N
    "Paso N",
    // Subpasos del Paso N
    [
      // ...,
    ]
  ]
]

```

Es posible analizar la definición de la función en cuatro secciones, en base a la operación que realizan:

- Interpretar la expresión desde el LaTeX cómo una expresión simbólica de SymPy.
- Llamada al clasificador.
- Llamada al solver correspondiente al tipo.
- Manejo de excepciones.

## 2.4. Interpretación

```
def solve(inputString, user_type):
    # Parse Latex expression
    try:
        equation = parseLatex(inputString)
    except Exception as e:
        raise e
```

Se realiza la interpretación por medio de la función `parseLatex()`, la cual está definida en el módulo traductor del proyecto. Mediante una base de comparaciones con expresiones regulares y los métodos proporcionados por SymPy para la manipulación de LaTeX se obtiene la expresión simbólica que le corresponde a la string. En caso de que se produzca una excepción durante la ejecución de la función, ésta es enviada hacia el archivo `api.py` y es atendida para ser enviada al cliente.

## 2.5. Clasificación

```
# Trying to catch a completeness anomaly
try:
    # Classify ODE. Could raise a Classification Anomaly
    odeType = classify(str(equation) + "= 0")
    print("ODE TYPE: " + odeType)
    print()
```

Se realiza la clasificación por medio de la función `classify()`, la cual está definida en el módulo clasificador del proyecto. Mediante una base de comparaciones de los árboles de clases de las expresiones de SymPy se determina si una expresión simbólica tiene la forma de una determinada ecuación diferencial. Se explorará más a detalle de la forma buscada para cada ecuación diferencial en los siguientes apartados.

## 2.6. Dificultad

```
# Global difficulty
global global_difficulty
global_difficulty = 0
```

Antes de comenzar con la construcción de la ecuación diferencial, se inicia el valor del limitador de búsqueda de solución conocido llamado `global_difficulty`. Esta variable es un entero que sirve para medir de manera indirecta el tiempo y recursos que ha utilizado el servidor para explorar la ecuación e ir construyendo la solución. A medida que el proceso avanza, su valor se incrementa en diferentes valores dependiendo de la intervención o manipulación que se realiza. Si el sistema excede un valor límite de dificultad global suspende la construcción de la solución de la ecuación diferencial y arroja una excepción de completitud. Las diferentes fuentes de incremento de dificultad serán exploradas a mayor profundidad en los apartados relacionados a los solvers.

## 2.7. Llamada a los solvers

Una vez resuelta la intervención del clasificador, se utiliza a la variable `odeType` para verificar cual es la ecuación diferencial encontrada. Los valores esperados de esta variable al concluir la clasificación son:

- "separable": Ecuación diferencial en variables separable de primer orden.
- "linear": Ecuación diferencial linear de primer orden.
- "reducible": Ecuación diferencial en forma de Bernoulli.
- "homogeneous": Ecuación diferencial homogénea de primer orden.
- "exact": Ecuación diferencial exacta.
- "superior": Ecuación diferencial linear de orden superior.

La intervención del clasificador puede, además, arrojar una excepción de clasificación para el caso en que la ecuación a estudiar no sea reconocida por el sistema como ninguno de los tipos que se están buscando.

```
if odeType == "separable":
    solveArray = solveSeparable(str(equation) + "= 0", 'y', user_type)
    print("Global Difficulty: " + str(global_difficulty))
    return solveArray[1]

elif odeType == "linear":
    solveArray = solveLinear(str(equation) + "= 0", 'y')
    print("Global Difficulty: " + str(global_difficulty))
    return solveArray[1]

elif odeType == "reducible":
    solveArray = solveReducibleToLinear(str(equation) + "= 0")
    print("Global Difficulty: " + str(global_difficulty))
    return solveArray[1]

elif odeType == "homogeneous":
    solveArray = solveHomogeneous(str(equation) + "= 0", user_type)
    print("Global Difficulty: " + str(global_difficulty))
    return solveArray[1]

elif odeType == "exact":
    solveArray = solveExact(str(equation) + "= 0")
    print("Global Difficulty: " + str(global_difficulty))
    return solveArray[1]
```

Cada una de las llamadas a los solvers puede arrojar una excepción de completitud que detiene la búsqueda de la ecuación diferencial. Cada llamada al solver retorna un arreglo con diferentes elementos de interés de la solución de la ecuación diferencial. Debido a que las estrategias de resolución de las ecuaciones diferenciales en forma de Bernoulli y Homogéneas se basan en la reducción mediante un cambio de variable a forma Linear y Separable respectivamente, estos últimos dos métodos también son utilizados durante la construcción de la solución, pero sobre otra variable (por eso en estas ecuaciones es pasado el atributo 'y', el cual es el esperado por el sistema según lo indicado en la propuesta. Cuando se llaman con el cambio de variable, se utiliza el símbolo 'u' para que la solución sea comprensible para el usuario).

## 2.8. Manejo de excepciones

Con base a las intervenciones del clasificador y los solvers, el controlador trabaja para construir una solución parcial para el usuario dependiendo del tipo de excepción que haya concluido con el flujo normal de la solución.

```
# Classification error on classify call
except ClassificationAnomaly as clsa:
    print("undefined classification")
    try:
        # Launch DSolve intervention for solving
        # an undefined ODE type on server
        solveSingle = dsolve(Eq(equation, 0), Function('y')(x))

        # Create single step in case of found a solution
        solveArray = []
        step = []
        step.append("- Solve with DSolve (backup system): " +
            "\\\ \\\")
        subSteps = []
        h0 = "The server was not able to build the steps for " + \
            "the solution." + "\\\ \\\") + \
            "However, the solution found was the following:" + \
            "\\\ \\\")
        eq0 = "$" + latex(solveSingle) + "$" + "\\\ \\\")
        subSteps.append(h0)
        subSteps.append(eq0)
        step.append(subSteps)
        solveArray.append(step)
        clsa.set_final_solve(solveArray)
```

```
except Exception as e:
    print(e.args[0])
finally:
    raise clsa
```

Con base a las intervenciones del clasificador y los **solvers**, el controlador trabaja para construir una solución parcial para el usuario dependiendo del tipo de excepción que haya concluido con el flujo normal de la solución. **CompletenessAnomaly** es llamada en caso de que la ecuación se haya podido clasificar, pero no concluir hasta su resolución, de forma que el sistema devuelve la solución parcial encontrada y la solución definitiva obtenida mediante **sympy**.

```
# Classification error on classify call
except CompletenessAnomaly as ca:
    print("unsolvable by the system")
    print("Global Difficulty: " + str(global_difficulty))
    try:
        # Launch DSolve intervention for solving
        # an uncompleted ODE on server
        solveSingle = dsolve(Eq(equation, 0), Function('y')(x))
        step = []
        step.append("- Solve with DSolve (backup system): " +
            "\\\n \\\n")
        subSteps = []
        h0 = "The server was not able to complete the steps for " +
            "the solution." + "\\\n \\\n" + \
            "However, the solution found was the following:" + \
            "\\\n \\\n"
        eq0 = "$" + latex(solveSingle) + "$" + "\\\n \\\n"
        subSteps.append(h0)
        subSteps.append(eq0)
        step.append(subSteps)
        ca.append_final_solve(step)
    except Exception as e:
        print(e.args[0])
    finally:
        raise ca
```



## 3. Solucionadores

### 3.1. Solver de ODEs Separables de Primer Orden

El archivo del `solve_separable.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución.

```
from timers.custom_threads import PropagatingThread
from anomalies.completeness_anomaly import CompletenessAnomaly
from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr

from algebraics.operations import *
from integrals.integrator import *
from analytics.investigator import *
```

```
def solveSeparable(odeString, functionName, user_type):
    """
    -----
    # Init solve
    -----
    """
    solveArray = []

    try:
        """
        -----
        # Initial algebraic analysis
        -----
        """
        odeLeftString = odeString.split("=")[0]
        odeRightString = odeString.split("=")[1]
        odeLeftSym = parse_expr(odeLeftString)
        odeRightSym = parse_expr(odeRightString)
        y = Function(functionName)

        equation = Eq(alg_subs(odeLeftSym, odeRightSym), 0)
        left = equation.args[0]
        express = alg_solve(Eq(left, Integer(0)), Derivative(y(x), x))
        aux = alg_simplify(express[0])
        aux = alg_expand(aux)
        aux = alg_factor(aux)
```

Una vez inicializado el arreglo y expandidos ambos lados de la ecuación, se intenta llevarla a la forma general buscada para detectar cuales serán cada una de las funciones de la ecuación diferencial.

```

...
-----
# Step 01: Detect separable structure
-----
...
solveArray.append([])
step = solveArray[0]
step.append("- Identify the separable equation and its parts" + "\\\n
\\\\")
step.append([])
subSteps = step[1]

functionF = Integer(1)
functionG = Integer(1)

if type(aux) is Add:
    if functionName in str(aux):
        functionG = aux
    else:
        functionF = aux
else:
    if not (functionName in str(aux)):
        functionF = aux
    else:
        for term in aux.args:
            if functionName in str(term):
                functionG = alg_mul(functionG, term)
            else:
                functionF = alg_mul(functionF, term)

functionG = alg_mul_inv(functionG)
expr = alg_div(functionF, functionG)

h0 = "With algebra, transform the expression: " + "\\\n
\\\\\n
"
subSteps.append(h0)

eq0 = "$" + latex(Eq(odeLeftSym, odeRightSym)) + "$" + "\\\n
\\\\\n
"
subSteps.append(eq0)

h1 = "into the equation: " + "\\\n
\\\\\n
"
subSteps.append(h1)

eq1 = "$" + latex(Derivative(y(x), x)) + " = " + latex(expr) + "$" + "\\\n
\\\\\n
"

```

A partir de la forma general, se separan las funciones para que de ambos lados de la ecuación tengamos dos funciones de variables diferentes con su respectivo diferencial. Se agregan los respectivos textos y ecuaciones en LaTeX al arreglo de pasos de la solución a medida que se construyen.

```
h2 = "which has the form: " + "\\\\ \\\\"
subSteps.append(h2)

eq2 = "$" + latex(Derivative(y(x), x)) + " = " +
latex(alg_div(Function('f')(x), \
Function('g')(Symbol(functionName)))) + "$" + "\\\\ \\\\"
subSteps.append(eq2)

h3 = "where: " + "\\\\ \\\\"
subSteps.append(h3)

eq3 = "$g{\\left( "+functionName+"\\right)} = " + latex(functionG) + "$ \\\\"
subSteps.append(eq3)

eq4 = "$f{\\left(x \\right)} = " + latex(functionF) + "$ \\\\"
subSteps.append(eq4)

h4 = "So, it is 1st order separable" + "\\\\ \\\\"
subSteps.append(h4)
```

```
...
-----
# Step 02: Separate functions
-----
...
solveArray.append([])
step = solveArray[1]
step.append("- Separate functions" + "\\\\ \\\\"
step.append([])
subSteps = step[1]

functionG = alg_substitution(functionG, y(x), Symbol(functionName))
left = alg_mul(functionG, Symbol('(d'+functionName+')'))
right = alg_mul(functionF, Symbol('(dx)'))

h0 = "Multiply by the differential of x and multiply by " + \
f"g({functionName})" + ", so the result is " + \
f"g({functionName})" + " and " + "f(x)" + \
" with their respective differentials" + "\\\\ \\\\"

subSteps.append(h0)
```

El siguiente paso constituye integrar ambos lados de la igualdad. Para ello, se hace la llamada al módulo integrador, el cual se explica con más detalle en secciones posteriores. Esta llamada devuelve a su vez una serie de subpasos que son anexados a la solución final concatenando ambos arreglos.

```

'''
-----
# Step 03: Integrate Left Side
-----
'''
solveArray.append([])
step = solveArray[2]
step.append("- Solve left" + "\\\\ \\")
step.append([])
subSteps = step[1]

left = alg_div(left, Symbol('(d'+functionName+')'))

h0 = "Integrate left side with respect to " + functionName + "\\\\ \\\"
subSteps.append(h0)

eq0 = "$\int{" + latex(left) + "} d"+functionName+"$"
subSteps.append(eq0 + "\\\\ \\")

left = alg_expand(left)
left_int_solve = int_solve(left, Symbol(functionName))
left = left_int_solve["solution"]

subSteps.append("-----" + "\\\\ \\")
for int_substep in left_int_solve["steps"]:
    subSteps.append(int_substep["text"] + "\\\\ \\")
    subSteps.append(int_substep["symbol"] + "\\\\ \\")
    subSteps.append("-----" + "\\\\ \\")
eq0 = "$" + latex(left) + " = " + latex(right) + "$" + "\\\\ \\\"
subSteps.append(eq0)

```

Se integra de la misma forma el otro lado de la igualdad. Los pasos se van añadiendo a la solución general exactamente del mismo modo que antes.

```

...
-----
# Step 04: Integrate Right Side
-----
...
solveArray.append([])
step = solveArray[3]
step.append("- Solve right" + "\\\\ \\")
step.append([])
subSteps = step[1]

h0 = "Integrate right side with respect to x" + "\\\\ \\\"
subSteps.append(h0)

right = alg_div(right, Symbol('(dx)'))
eq0 = "$\int{" + latex(right) + "} dx$"
subSteps.append(eq0 + "\\\\ \\")

right = alg_expand(right)
right_int_solve = int_solve(right, x)
right = right_int_solve["solution"]

subSteps.append("-----" + "\\\\ \\")
for int_substep in right_int_solve["steps"]:
    subSteps.append(int_substep["text"] + "\\\\ \\")
    subSteps.append(int_substep["symbol"] + "\\\\ \\")
    subSteps.append("-----" + "\\\\ \\")

```

Se igualan ambos lados de la ecuación (los resultados de ambas integrales). Se llega a la solución implícita de la ecuación diferencial con esto, ya que el resultado presenta a la función 'y' sin derivadas. Esto es indicado al usuario como parte del paso número 5 de la solución.

```

...
-----
# Step 05: Equate Both Sides
-----
...
solveArray.append([])
step = solveArray[4]
step.append("- Get implicit solution" + "\\\\ \\")
step.append([])
subSteps = step[1]

h1s5 = "Equate both sides" + "\\\\ \\\"
subSteps.append(h1s5)

exp1s5 = "$" + latex(left) + " = " + latex(right) + "$" + "\\\\ \\\"
subSteps.append(exp1s5)

h2s5 = "Substract right side from both sides and add the arbitrary constant
C. " + \
    "The implicit answer is: " + "\\\\ \\\"
subSteps.append(h2s5)

express = Add(left, Mul(right, Integer(-1)), Symbol('C'))
eq1s5 = "$" + latex(express) + "$ = 0" + "\\\\ \\\"
subSteps.append(eq1s5)

```

Dependiendo de la forma que adquiriera la solución implícita se busca obtener la solución explícita despejando a la variable 'y'. En caso de que existan varias soluciones para este despeje, se le presentan al usuario todas las posibles. En caso de que el usuario este registrado como tipo "teacher", se hace la llamada a los métodos analíticos para entregar un análisis de los puntos notables de cada una de las soluciones de las ecuaciones diferenciales. Todas estas intervenciones son reguladas por el timer "Propagating Thread", del cual será descrito más adelante.

```

...
-----
# Step 06: Get Explicit Solve
-----
...
solveArray.append([])
step = solveArray[5]
step.append("- Get the explicit solution solving for " + functionName +
"\\\\ \\")
step.append([])
subSteps = step[1]

global finalSolve
finalSolve = []

def final_solve_timeout(expression, symbol):
    global finalSolve
    finalSolve = solve(expression, symbol)

try:
    process = PropagatingThread(target = final_solve_timeout, args=(express,
Symbol(functionName)))
    process.start()
    process.join(timeout=5)

    for singleSolve in finalSolve:
        eq1s6 = Eq(y(x), singleSolve)
        subSteps.append("$" + latex(eq1s6) + "$" + "\\\\ \\")

        # Analytic intervention for all the single solves if is teacher
        if (user_type == 'teacher'):
            print("Teacher")
            try:
                roots = []
                roots_process = PropagatingThread(target = get_roots, args =
[singleSolve, roots])
                roots_process.start()
                roots_process.join(timeout = 3)

                h0 = "Whose roots are: " + "\\\\ \\")
                subSteps.append(h0)
                subIndex = 1
                for root in roots:
                    eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(root) + "$" +
"\\\\ \\")
                    subIndex = subIndex + 1
                    subSteps.append(eq0)

            except Exception as e:
                print("Error with roots")
                print(e)

```

```

try:
    critics = []
    critics_process = PropagatingThread(target = max_min, args =
[singleSolve, critics])
    critics_process.start()
    critics_process.join(timeout = 3)

    h0 = "Whose critics are: " + "\\\\ \\\\"
    subSteps.append(h0)
    subIndex = 1
    for critic in critics:
        eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(critic) + "$"
+ "\\\\ \\\\"
        subIndex = subIndex + 1
        subSteps.append(eq0)

except Exception as e:
    print("Error with critics")
    print(e)

try:
    inflexions = []
    inflexions_process = PropagatingThread(target = inflexion_points,
args = [singleSolve, inflexions])
    inflexions_process.start()
    inflexions_process.join(timeout = 3)

    h0 = "Whose inflexions are: " + "\\\\ \\\\"
    subSteps.append(h0)
    subIndex = 1
    for inflexion in inflexions:
        eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(inflexion) +
"$" + "\\\\ \\\\"
        subIndex = subIndex + 1
        subSteps.append(eq0)

except Exception as e:
    print("Error with inflexions")
    print(e)

```

En caso de que se trate de un usuario de tipo "teacher", se agrega un último paso para indicar las gráficas de las diferentes soluciones. Esto solo se lleva a cabo para todas las soluciones explícitas de la ecuación diferencial; de no encontrar ninguna no se agrega este paso.



```

    if (user_type == "teacher" and functionName == 'y'):
        ...
        -----
        # Step 07: Generate Plot
        -----
        ...
        solveArray.append([])
        step = solveArray[6]
        step.append("- Graphs" + "\\\\\\ \\\\")
        step.append([])
        subSteps = step[1]

        for singleSolve in finalSolve:
            # Add plot step to solution
            print("Creating plot")

            try:
                plot_string = create_plot(singleSolve)[1:]
                plot_string = plot_string.replace("\\n", "")
            except Exception as e:
                print(e)

            subSteps.append(plot_string)
            print("Plot appended")
        except:
            subSteps.append("Can not get the explicit solution solving for " +
functionName + "\\\\\\ \\\\")

```

Por último, tanto la expresión simbólica de la solución como todos los pasos se retornan de la llamada del solucionador como respuesta para el controlador para una solución completa o bien una llamada intermedia para las ecuaciones de tipo homogéneas. En caso de que durante alguna de las intervenciones se haya provocado una excepción de completitud, se encapsulan los pasos que se hayan obtenido y son enviados al controlador para que este a su vez los regrese al usuario (esto también incluye las soluciones parciales del módulo integrador).

```

def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve():
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [display_solve(), solveArray, finalSolve]

except CompletenessAnomaly as ca:

    if ca.partial_solution[0][0] == "partial integral":
        step = solveArray[len(solveArray) - 1]
        subSteps = step[1]
        subSteps.append("-----" + "\\\\ \\")

        for int_substep in ca.partial_solution[0][1]:
            subSteps.append(int_substep["text"] + "\\\\ \\")
            subSteps.append(int_substep["symbol"] + "\\\\ \\")
            subSteps.append("-----" + "\\\\ \\")

    ca.set_partial_solution(solveArray)

raise ca

```

## 3.2. Solver de ODEs Homogéneas de Primer Orden

El archivo del `solve_homogeneous.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución.

```

from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr
from solvers.sys_solvers.solve_separable import *

```

```

def solveHomogeneous(odeString, functionName, user_type):
    ...
    -----
    # Init solve
    -----
    ...
    # Init solve array
    solveArray = []

    try:
        ...
        -----
        # Initial algebraic analysis
        -----
        ...

        odeLeftString = odeString.split("=")[0]
        odeRightString = odeString.split("=")[1]

        odeLeftSym = parse_expr(odeLeftString)
        odeRightSym = parse_expr(odeRightString)

        y = Function(functionName)
        equation = Eq(odeLeftSym - odeRightSym, 0)

        # Step 1
        left = equation.args[0]
        exp = solve(left, Derivative(y(x), x))
        aux = expand(exp[0])

        left = Derivative(y(x), x)
        # Define the change of variable
        functionF = aux

```

En el primer paso de la solución se encuentra el cambio de variable adecuado para la ecuación diferencial, el cual busca llevar la ecuación diferencial a una presentación en variables separables. El cambio de variable es descrito para el usuario en este paso.

```

'''
-----
# Step 01: Propose the appropriate variable change to reduce to separable
-----
'''
solveArray.append([])
step = solveArray[0]
step.append("- Propose the appropriate variable change to reduce to
separable" + "\\\\ \\")
step.append([])
subSteps = step[1]

#Define function u(x)
u = Function('u')

h0 = "Since it's homogeneous, the derivative can be expressed as a
function that is also homogeneous, that is:" + "\\\\ \\\"
subSteps.append(h0)

eq0 = "$" + latex(Eq(Derivative(y(x), x), functionF)) + "$" + "\\\\ \\\"
subSteps.append(eq0)

#Perform the substitution
functionF = functionF.subs(y(x), Mul(u(x), x))
left = Add(Mul(Derivative(u(x), x), x), u(x))

h1 = "Using the change of variable: " + "\\\\ \\\"
subSteps.append(h1)

eq1 = "$" + latex(Eq(y(x), Mul(u(x), x))) + "$" + "\\\\ \\\"
subSteps.append(eq1)

h2 = "Whose derivative is: " + "\\\\ \\\"
subSteps.append(h2)

eq2 = "$" + latex(Eq(Derivative(y(x), x), Add(u(x), Mul(Derivative(u(x),
x)))))) + "$" + "\\\\ \\\"
subSteps.append(eq2)

h3 = "Carrying out the changes for the function and its derivative in the
original equation: " + "\\\\ \\\"
subSteps.append(h3)

eq3 = "$" + latex(Eq(left, functionF)) + "$" + "\\\\ \\\"
subSteps.append(eq3)

left = Add(left, Mul(functionF, Integer(-1)))
left = expand(left)
separableODE = Eq(left, Integer(0))

```

La ecuación en variables separables resultante del paso anterior se resuelve llamando al método antes descrito. Los pasos que entrega la intervención del método para resolver separables se añaden al arreglo general de pasos de la ecuación diferencial. A partir de las soluciones encontradas, se regresa el cambio de variable utilizado al inicio para llegar a la solución explícita. Después de esto, procede el mismo análisis de la intervención analítica para puntos críticos y gráficas del modo investigador.

```
h4 = "Simplifying: " + "\\\ \\\ \"
subSteps.append(h4)

eq4 = "$" + latex(separableODE) + "$" + "\\\ \\\ \"
subSteps.append(eq4)

h5 = "Wich is first order separable" + "\\\ \\\ \"
subSteps.append(h5)

solutionSeparable = solveSeparable(str(separableODE.args[0]) + "= 0", 'u',
user_type)
solveArray += solutionSeparable[1]

#Final Solution
solveForU = solutionSeparable[2]
```

```
...
-----
# Step 02: Get Explicit Solve
-----
...
solveArray.append([])
step = solveArray[1]
step.append("- Undo the variable change" + "\\\ \\\ \")
step.append([])
subSteps = step[1]

global finalSolve
finalSolve = []

def final_solve_timeout(expression, symbol):
    global finalSolve
    finalSolve = solve(expression, symbol)
```

```

    if (len(solveForU)) > 0:
        try:
            process = PropagatingThread(target = final_solve_timeout,
            args=(solveForU, Symbol(functionName)))
            process.start()
            process.join(timeout=5)

            for singleSolve in finalSolve:
                eq1s6 = Eq(y(x), singleSolve)
                subSteps.append("$" + latex(eq1s6) + "$" + "\\\\ \\")

            # Analytic intervention for all the single solves if is teacher
            if (user_type == 'teacher'):
                print("Teacher")
                try:
                    roots = []
                    roots_process = PropagatingThread(target = get_roots, args =
[singleSolve, roots])
                    roots_process.start()
                    roots_process.join(timeout = 3)

                    h0 = "Whose roots are: " + "\\\\ \\\"
                    subSteps.append(h0)
                    subIndex = 1
                    for root in roots:
                        eq0 = "$" + "x_{ " + str(subIndex) + " } = " + latex(root) +
"$" + "\\\\ \\\"
                        subIndex = subIndex + 1
                        subSteps.append(eq0)

                    except Exception as e:
                        print("Error with roots")
                        print(e)

                try:
                    critics = []
                    critics_process = PropagatingThread(target = max_min, args =
[singleSolve, critics])
                    critics_process.start()
                    critics_process.join(timeout = 3)

                    h0 = "Whose critics are: " + "\\\\ \\\"
                    subSteps.append(h0)
                    subIndex = 1
                    for critic in critics:
                        eq0 = "$" + "x_{ " + str(subIndex) + " } = " + latex(critic) +
"$" + "\\\\ \\\"
                        subIndex = subIndex + 1
                        subSteps.append(eq0)

                    except Exception as e:
                        print("Error with critics")
                        print(e)

```

```

        try:
            inflexions = []
            inflexions_process = PropagatingThread(target =
inflexion_points, args = [singleSolve, inflexions])
            inflexions_process.start()
            inflexions_process.join(timeout = 3)

            h0 = "Whose inflexions are: " + "\\\ \\\ \"
            subSteps.append(h0)
            subIndex = 1
            for inflexion in inflexions:
                eq0 = "$" + "x_" + str(subIndex) + " = " + latex(inflexion)
+ "$" + "\\\ \\\ \"
                subIndex = subIndex + 1
                subSteps.append(eq0)

        except Exception as e:
            print("Error with inflexions")
            print(e)

```

Al igual que en la solución de las ecuaciones diferenciales separables, es posible que exista una anomalía de completitud en alguna de las intervenciones realizadas (ya sea del integrador o del solver para ODEs separables). Esta excepción es registrada y lanzada al controlador para mostrar la solución parcial al usuario.

```

    if (user_type == "teacher"):
        ...
        -----
        # Step 07: Generate Plot
        -----
        ...
        solveArray.append([])
        step = solveArray[6]
        step.append("- Graphs" + "\\\\\\ \\\\")
        step.append([])
        subSteps = step[1]

        for singleSolve in finalSolve:
            # Add plot step to solution
            print("Creating plot")

            try:
                plot_string = create_plot(singleSolve)[1:]
                plot_string = plot_string.replace("\\n", "")
            except Exception as e:
                print(e)

            subSteps.append(plot_string)
            print("Plot appended")
        except:
            subSteps.append("Can not get the explicit solution solving for " +
functionName + "\\\\\\ \\\\")

def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve(solveArray):
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [ display_solve(solveArray), solveArray ]

except CompletenessAnomaly as ca:

    if ca.partial_solution[0][0] == "partial integral":
        step = solveArray[len(solveArray) - 1]
        subSteps = step[1]
        subSteps.append("-----" + "\\\\\\ \\\\")

        for int_substep in ca.partial_solution[0][1]:
            subSteps.append(int_substep["text"] + "\\\\\\ \\\\")
            subSteps.append(int_substep["symbol"] + "\\\\\\ \\\\")
            subSteps.append("-----" + "\\\\\\ \\\\")
        ca.set_partial_solution(solveArray)
    raise ca

```



### 3.3. Solver de ODEs Lineares de Primer Orden

```

from timers.custom_threads import PropagatingThread
from anomalies.completeness_anomaly import CompletenessAnomaly
from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr

from algebraics.operations import *
from integrals.integrator import *
from analytics.investigator import *

```

```

def solveLinear(odeString, functionName, user_type):
    """
    -----
    # Init solve
    -----
    """
    solveArray = []

    try:
        odeLeftString = odeString.split("=")[0]
        odeRightString = odeString.split("=")[1]
        odeLeftSym = parse_expr(odeLeftString)
        odeRightSym = parse_expr(odeRightString)
        y = Function(functionName)
        equation = Eq(odeLeftSym - odeRightSym, 0)
        left = equation.args[0]
        exp = alg_solve(left, Derivative(y(x), x))
        aux = alg_expand(exp[0])

        left = Derivative(y(x), x)

        functionF = parse_expr("0")
        functionG = parse_expr("0")

        for term in aux.args:
            if functionName in str(term):
                functionF = Add(functionF, Mul(term, Pow(y(x), Integer(-1))))
            else:
                functionG = Add(functionG, term)

        functionF = Mul(functionF, Integer(-1))
        functionF = simplify(functionF)
        functionG = simplify(functionG)

        right = alg_add(functionG, Mul(Integer(-1), functionF, y(x)))
    
```

El archivo del `solve_linear.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución. Una vez que se tiene todo inicializado, se explora la ecuación para identificar cuáles son las funciones que caracterizan a la ODE linear. Esta exploración se hace comparando los elementos que contiene cada término de la ecuación. Este proceso constituye el paso 1 de la solución.

```

'''
-----
# Step 01: Identify the linear equation
-----
'''
solveArray.append([])
step = solveArray[0]
step.append("- Identify the linear equation and its parts" + "\\\\ \\")
step.append([])
subSteps = step[1]

h0 = "With algebra, transform the expression: " + "\\\\ \\\"
subSteps.append(h0)

eq0 = "$" + latex(Eq(odeLeftSym, odeRightSym)) + "$" + "\\\\ \\\"
subSteps.append(eq0)

h1 = "into the equation: " + "\\\\ \\\"
subSteps.append(h1)

eq1 = "$" + latex(Derivative(y(x), x)) + " = " + latex(exp) + "$" + "\\\\ \\\"
subSteps.append(eq1)

h2 = "which has the form: " + "\\\\ \\\"
subSteps.append(h2)

eq2 = "$" + latex(Derivative(y(x), x)) + " = " +
latex(Add(Function('g')(x), Mul(Function('f')(x), Integer(-1), y(x)))) + "$" +
"\\\\ \\\"
subSteps.append(eq2)

h3 = "where: " + "\\\\ \\\"
subSteps.append(h3)

eq3 = "$g{\\left(x \\right)} = " + latex(functionG) + "$ \\\\ \\\"
subSteps.append(eq3)

eq4 = "$f{\\left(x \\right)} = " + latex(functionF) + "$ \\\\ \\\"
subSteps.append(eq4)

h4 = "So, it is 1st order linear" + "\\\\ \\\"
subSteps.append(h4)

```

El siguiente paso de la solución es calcular el factor integral que necesita la ecuación diferencial para quedar en variables separables. Este proceso requiere de la intervención del módulo de integración.

```

'''
-----
# Step 02: Calculate integral factor
-----
'''
solveArray.append([])
step = solveArray[1]
step.append("- Calculate integral factor" + "\\\\ \\")
step.append([])
subSteps = step[1]

h1s2 = "Lets propose a function M(x) such that: " + "\\\\ \\\"
subSteps.append(h1s2)

eqAux = Eq(Mul(Function('M')(x), Function('f')(x)),
Derivative(Function('M')(x), x))
eq1s2 = "$" + latex(eqAux) + "$" + "\\\\ \\\"
subSteps.append(eq1s2)

h2s2 = "Substituting: " + "\\\\ \\\"
subSteps.append(h2s2)

eq2s2 = "$" + latex(Eq(Mul(Function('M')(x), functionF),
Derivative(Function('M')(x), x))) + "$" + "\\\\ \\\"
subSteps.append(eq2s2)

h3s2 = "Which is a 1st order separable differential equation. Hence solving
for M(x)" + "\\\\ \\\"
subSteps.append(h3s2)

functionM = Pow(E, Integral(functionF, x))
eq3s2 = "$" + latex(Eq(Mul(Pow(Function('M')(x), Integer(-1)),
Symbol('dM(x)'), Mul(functionF, Symbol('(dx)')))) + "$" + "\\\\ \\\"
subSteps.append(eq3s2)

eq4s2 = "$" + latex(Eq(log(Function('M')(x)), Integral(functionF, x))) +
"$" + "\\\\ \\\"
subSteps.append(eq4s2)

functionF = expand(functionF)
exponentM = integrate(expand(functionF), x)
eq5s2 = "$" + latex(Eq(log(Function('M')(x)), exponentM)) + "$" + "\\\\
\\\\\"
subSteps.append(eq5s2)

eq6s2 = "$" + latex(Function('M')(x) + " = " + latex(Pow(E, exponentM))+
"$" + "\\\\ \\\"
subSteps.append(eq6s2)

```

El factor recién calculado es utilizado para cambiar la forma de la ecuación a separable por medio del cambio de variable conveniente.

```

...
-----
# Step 03: Reduce to 1st order separable ODE
-----
...
solveArray.append([])
step = solveArray[2]
step.append("- Reduce to 1st Order Separable ODE" + "\\\\ \\")
step.append([])
subSteps = step[1]

functionM = functionM.replace(Integral(functionF, x), exponentM)
functionM = Pow(E, exponentM)
functionM = simplify(functionM)

h1s3 = "Multiplying the original equation by M(x):" + "\\\\ \\\"
subSteps.append(h1s3)

equation = Eq(left, right)
left = Mul(left, functionM)
right = Add(Mul(Integer(-1), functionF, y(x), functionM), Mul(functionG,
functionM))
equation = Eq(left, right)

left = Add(left, Mul(functionF, y(x), functionM))
right = Add(right, Mul(functionF, y(x), functionM))
equation = Eq(left, right)
equationaux = Eq(expand(Mul( Function('M')(x), left, pow(functionM,
Integer(-1)))), Mul( Function('M')(x), right, pow(functionM, Integer(-1))))

eq1s3 = "$" + latex(equationaux) + "$" + "\\\\ \\\"
subSteps.append(eq1s3)

h2s3 = "By the definition of M(x) this is equivalent to: " + "\\\\ \\\"
subSteps.append(h2s3)

eq2s3 = "$" + latex(Add(Mul(Derivative(y(x),x),
Function('M')(x)),Mul(y(x), Derivative(Function('M')(x), x)))) + " = " +
latex(Mul(functionG, Function('M')(x))) + "$" + "\\\\ \\\"
subSteps.append(eq2s3)

h3s3 = "Notice that the left hand side can be reduce by the chain rule to:
" + "\\\\ \\\"
subSteps.append(h3s3)

eq3s3 = "$" + latex(Add(Mul(Derivative(y(x),x),
Function('M')(x)),Mul(y(x), Derivative(Function('M')(x), x)))) + " = " +
latex(Derivative(Mul(y(x), Function('M')(x)),x)) + "$" + "\\\\ \\\"
subSteps.append(eq3s3)

```

```

h4s3 = "Therefore: " + "\\\\ \\\\"
subSteps.append(h4s3)7

eq4s3 = "$" + latex(Derivative(Mul(y(x), Function('M')(x)),x)) + " = " +
latex(Mul(functionG, Function('M')(x))) + "$" + "\\\\ \\\\"
subSteps.append(eq4s3)

h5s3 = "Which again is a 1st order separable differential equation" + "\\\\
\\\\"
subSteps.append(h5s3)

equationaux = Eq(Symbol('dM(x)'+functionName+'(x)'),
factor(Mul(Symbol('(dx)'),functionG, Function('M')(x))))
eq5s3 = "$" + latex(equationaux)+ "$" + "\\\\ \\\\"
subSteps.append(eq5s3)

```

Con la forma actual de la ecuación, es posible llegar a la solución de la forma separable mediante una simple integración del lado derecho; esto se lleva a cabo en el paso 4. Es importante recalcar que esta integral puede ser particularmente compleja debido a que involucra el resultado previo del factor integrando que, en principio, es una función exponencial compuesta con alguna otra integral. Con esto, la complejidad de este tipo de ecuaciones diferenciales reside en este paso.

Al llegar a la solución de la integral, solo resta deshacer el cambio de variable utilizado para llegar a la solución explícita de la ecuación diferencial. Al llegar a este punto, se realizan las intervenciones analíticas para el gráfico y puntos característicos de la solución para los usuarios de tipo investigador.

```

...
-----
# Step 04: Get implicit solve
-----
...
solveArray.append([])
step = solveArray[3]
step.append("- Get implicit solution" + "\\\\ \\")
step.append([])
subSteps = step[1]

left = Derivative(Mul(functionM, y(x)), x)

equation = Eq(left, right)
left = Mul(left, Pow(Derivative(Mul(functionM, y(x)), x), Integer(-1)),
Symbol('d'), Mul(y(x), functionM))
right = Mul(right, Symbol('dx'))
equation = Eq(left, right)
h6s3 = "Integrating the left hand side, and indicating the integral at
right hand side: " + "\\\\ \\\"
subSteps.append(h6s3)

left = Mul(y(x), functionM)
right = Mul(right, Pow(Symbol('dx'), Integer(-1)))
eq6s3 = "$" + latex(Symbol('M(x)'+functionName+'(x)'))+ " = " +
latex(Integral(Mul(Mul(right, Pow(functionM, Integer(-
1))),Function('M')(x)),x)) + "$" + "\\\\ \\\"
subSteps.append(eq6s3)

h7s3 = "Substituting M(x) = " + "\\\\ \\\"
subSteps.append(h7s3)

eqAux1 = "$" + latex(functionM) + "$" + "\\\\ \\\"
subSteps.append(eqAux1)

equationaux = Eq(left, Integral(right,x))
eq7s3 = "$" + latex(equationaux)+ "$" + "\\\\ \\\"
subSteps.append(eq7s3)

right = expand(right)
right = integrate(right, x)
right = Add(right, Symbol('C'))
equation = Eq(left, right)

h8s3 = "Integrating the right hand side: " + "\\\\ \\\"
subSteps.append(h8s3)

eq8s3 = "$" + latex(equation)+ "$" + "\\\\ \\\"
subSteps.append(eq8s3)

```

```

...
-----
# Step 05: Obtain solution
-----
...
solveArray.append([])
step = solveArray[4]
step.append("- Solve for " + functionName + "\\\ \\\")
step.append([])
subSteps = step[1]

left = y(x)
right = Mul(right, Pow(functionM, Integer(-1)))
right = simplify(right)
equation = Eq(left, right)
h9s3 = "Solve for " + "\\\ \\\")
subSteps.append(h9s3)

eqAux = latex(Symbol(functionName+'(x)')) + + "\\\ \\\")
eq9s3 = "$" + latex(equation) + "$" + "\\\ \\\")
subSteps.append(eq9s3)

global finalSolve
finalSolve = []

def final_solve_timeout(expression, symbol):
    global finalSolve
    finalSolve = solve(expression, symbol)

try:
    process = PropagatingThread(target = final_solve_timeout,
args=(equation, Symbol(functionName)))
    process.start()
    process.join(timeout=5)

    for singleSolve in finalSolve:
        eq1s6 = Eq(y(x), singleSolve)
        subSteps.append("$" + latex(eq1s6) + "$" + "\\\ \\\")
# Analytic intervention for all the single solves if is teacher
if (user_type == 'teacher'):
    print("Teacher")
    try:
        roots = []
        roots_process = PropagatingThread(target = get_roots, args =
[singleSolve, roots])
        roots_process.start()
        roots_process.join(timeout = 3)

```

```

        h0 = "Whose roots are: " + "\\\\ \\\\"
        subSteps.append(h0)
        subIndex = 1
        for root in roots:
            eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(root) + "$" +
"\\\\ \\\"
            subIndex = subIndex + 1
            subSteps.append(eq0)

    except Exception as e:
        print("Error with roots")
        print(e)

    try:
        critics = []
        critics_process = PropagatingThread(target = max_min, args =
[singleSolve, critics])
        critics_process.start()
        critics_process.join(timeout = 3)

        h0 = "Whose critics are: " + "\\\\ \\\"
        subSteps.append(h0)
        subIndex = 1
        for critic in critics:
            eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(critic) + "$"
+ "\\\\ \\\"
            subIndex = subIndex + 1
            subSteps.append(eq0)

    except Exception as e:
        print("Error with critics")
        print(e)

    try:
        inflexions = []
        inflexions_process = PropagatingThread(target = inflexion_points,
args = [singleSolve, inflexions])
        inflexions_process.start()
        inflexions_process.join(timeout = 3)

        h0 = "Whose inflexions are: " + "\\\\ \\\"
        subSteps.append(h0)
        subIndex = 1
        for inflexion in inflexions:
            eq0 = "$" + "x_" + str(subIndex) + "} = " + latex(inflexion) +
"$" + "\\\\ \\\"
            subIndex = subIndex + 1
            subSteps.append(eq0)

    except Exception as e:
        print("Error with inflexions")
        print(e)

```



```

if (user_type == "teacher"):
    ...
    -----
    # Step 07: Generate Plot
    -----
    ...
    solveArray.append([])
    step = solveArray[6]
    step.append("- Graphs" + "\\\n \\\n")
    step.append([])
    subSteps = step[1]

    for singleSolve in finalSolve:
        # Add plot step to solution
        print("Creating plot")

        try:
            plot_string = create_plot(singleSolve)[1:]
            plot_string = plot_string.replace("\\n", "")
        except Exception as e:
            print(e)

        subSteps.append(plot_string)
        print("Plot appended")
    except:
        subSteps.append("Can not get the explicit solution solving for " +
functionName + "\\\n \\\n")

```

Una vez contruidos todos los pasos, se envía la respuesta al controlador o a la llamada del método que aparece en las ecuaciones en la forma Bernoulli según sea el caso. En caso de que alguna de las intervenciones algebraicas o integrales desaten una excepción de completitud, se registra la solución parcial con los pasos completados hasta entonces y es enviada al controlador para que a su vez la envíe al usuario final.

```

def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve(solveArray):
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [display_solve(solveArray), solveArray]

except CompletenessAnomaly as ca:
    if ca.partial_solution[0][0] == "partial integral":
        step = solveArray[len(solveArray) - 1]
        subSteps = step[1]
        subSteps.append("-----" + "\\\ \\\ \\\ \\\")

        for int_substep in ca.partial_solution[0][1]:
            subSteps.append(int_substep["text"] + "\\\ \\\ \\\ \\\")
            subSteps.append(int_substep["symbol"] + "\\\ \\\ \\\ \\\")
            subSteps.append("-----" + "\\\ \\\ \\\ \\\")

    ca.set_partial_solution(solveArray)
    raise ca

```

### 3.4. Solver de ODEs Reducibles a Lineares de Primer Orden (Forma de Bernoulli)

```
from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr
from sympy.parsing.latex import parse_latex

from solvers.sys_solvers.solve_linear import *
```

```
def solveReducibleToLinear(odeString):
    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)

    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)

    solveArray = []

    left = equation.args[0]
    exp = solve(left, Derivative(y(x), x))
    aux = expand(exp[0])

    left = Derivative(y(x), x)

    functionF = parse_expr("0")
    functionG = parse_expr("0")

    n = Integer(0)

    aux = Mul(aux, Pow(y(x), Integer(-1)))
    aux = simplify(aux)

    for term in aux.args:
        if 'y' in str(term):
            for subTerm in term.args:
                if 'y' in str(subTerm):
                    n = Add(subTerm.args[1], Integer(1))
                    subG = Mul(term, Pow(subTerm, Integer(-1)))
                    functionG = Add(functionG, subG)
            else:
                functionF = Add(functionF, term)
```

El archivo del `solve_reducible.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución. Una vez que se tiene todo inicializado, se explora la ecuación para identificar cuáles son las funciones que caracterizan a la ODE en forma de Bernoulli, así como el exponente de reducción. Esta exploración se hace comparando los elementos que contiene cada término de la ecuación. Este proceso constituye el paso 1 de la solución.

```

step = []
step.append("- Identify the reducible to linear equation, its parts and
degree" + "\\\\ \\\\")
subSteps = []
h0 = "From the equation its degree is given by: " + n + "\\\\ \\\\"
subSteps.append(h0)

functionF2 = Mul(functionF, Integer(Add(Integer(1), Mul(Integer(-1), n))))
functionG2 = Mul(functionG, Integer(Add(Integer(1), Mul(Integer(-1), n))))

u = Function('u')
h1 = "Finding the right substitution in the parameter u(x)" + "\\\\ \\\\"
h2 = "Substituting into the equation, yields " + "\\\\ \\\\"
equation = Eq(Add(Derivative(u(x), x), Mul(functionF2, u(x))), functionG2)
h3 = "$" + latex(equation) + "$" + "\\\\ \\\\"
h4 = "Which is linear" + "\\\\ \\\\"

subSteps.append(h1)
subSteps.append(h2)
subSteps.append(h3)
subSteps.append(h4)
step.append(subSteps)
solveArray.append(step)

odeStringEqLeft = equation.args[0]
odeStringEqRigth = equation.args[1]
odeStringLinear = str(odeStringEqLeft) + "=" + str(odeStringEqRigth)

solveFromLinear = solveLinear(odeStringLinear, 'u')

solveArray += solveFromLinear[1]

```

Una vez realizado el cambio de variable, se resuelve la ecuación diferencial linear resultante y se agregan los pasos a la solución general. Con base a la solución obtenida por parte del solve\_linear.py, se deshace el cambio de variable realizado y se agregan los pasos restantes para el modo investigador en caso de que el usuario tenga una cuenta de este tipo.

```
def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve(solveArray):
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [ display_solve(solveArray), solveArray ]
```

### 3.5. Solver de ODEs Exactas de Primer Orden

```

from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr
from sympy.parsing.latex import parse_latex
from anomalies.completeness_anomaly import CompletenessAnomaly

from algebraics.operations import *
from integrals.integrator import *
from analytics.investigator import *

```

```

def solveExact(odeString, functionName, user_type):
    '''
    -----
    # Init solve
    -----
    '''
    try:
        '''
        -----
        # Initial algebraic analysis
        -----
        '''
        # init solve array
        solveArray = []

        odeLeftString = odeString.split("=")[0]
        odeRightString = odeString.split("=")[1]

        odeLeftSym = parse_expr(odeLeftString)
        odeRightSym = parse_expr(odeRightString)

        y = Function('y')
        equation = Eq(odeLeftSym - odeRightSym, 0)
        equation = equation.subs(y(x), Symbol('y'))

        functionP = Integer(0)
        functionQ = Integer(0)
        functionF = Integer(0)
        leftPartial = Integer(0)
        functionG = Function('g')

```

El archivo del `solve_exact.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución. Una vez que se tiene todo inicializado, se explora la ecuación para identificar cuáles son las funciones que caracterizan a la ODE exacta. Esta exploración se hace comparando los elementos que contiene cada término de la ecuación. Este proceso constituye el paso 1 de la solución.

```

...
-----
# Step 01: Detect exact structure
-----
...

solveArray.append([])
step = solveArray[0]
step.append("- Identify the exact equation and its parts" + "\\\\ \\")
step.append([])
subSteps = step[1]

for term in equation.args[0].args:
    if 'Derivative' in str(term):
        functionQ = Add(functionQ, Mul(term, Pow(Derivative(Symbol('y'), x),
Integer(-1))))
    else:
        functionP = Add(functionP, term)

h0 = "With algebra, transform the expression: " + "\\\\ \\\"
subSteps.append(h0)
eq0 = "$" + latex(Eq(odeLeftSym, odeRightSym)) + "$" + "\\\\ \\\"
subSteps.append(eq0)
h1 = "into the equation: " + "\\\\ \\\"
subSteps.append(h1)
eq1 = "$" + latex(equation) + "$" + "\\\\ \\\"
subSteps.append(eq1)
h2 = "which has the form: " + "\\\\ \\\"
subSteps.append(h2)
eq2 = "$" + latex(Function('P')(Symbol('x,y')) + " + " + latex(
Mul(Function('Q')(Symbol('x,y')), Derivative(y(x),x))) + " = 0" + "$" + "\\\\ \\\"
subSteps.append(eq2)
h3 = "where: " + "\\\\ \\\"
subSteps.append(h3)
eq3 = "$" + latex(Function('Q')(Symbol('x,y')) + " = " + latex(functionQ)
+ "$ \\\\ \\\"
subSteps.append(eq3)
eq4 = "$" + latex(Function('P')(Symbol('x,y')) + " = " + latex(functionP)
+ "$ \\\\ \\\"
subSteps.append(eq4)
h4 = "So, it is an exact differential equation" + "\\\\ \\\"
subSteps.append(h4)

```

Como lo indica la teoría, el siguiente paso es buscar expresar una de las funciones como resultado de la derivada parcial de una función constante que caracteriza a la ODE exacta (en este caso,  $F(x,y)$ ). Para ello, se realiza el álgebra necesaria y se realiza la integral correspondiente. Los pasos de la integral son añadidos a la solución general como parte del paso 2.

```
#Step 2
'''
-----
# Step 02: Obtain F(x,y) as a result of P(x,y)
-----
'''

solveArray.append([])
step = solveArray[1]
step.append("- Obtain F(x,y) as a result of P(x,y) " + "\\\\ \\")
step.append([])
subSteps = step[1]

functionF = integrate(functionP, x)
functionF = Add(functionF, functionG(Symbol('y')))

h1s2 = "Lets integrate the function " + "\\\\ \\\"
subSteps.append(h1s2)

eqAux1 = "$" + latex(Function('P')(Symbol('x'), Symbol('y'))) + "$" +
"\\\\ \\\"
subSteps.append(eqAux1)

eqAux = Eq(Function('F')(Symbol('x'), Symbol('y')),
Integral(Function('P')(Symbol('x'), Symbol('y')),x))
eq1s2 = "$" + latex(eqAux) + "$" + "\\\\ \\\"
subSteps.append(eq1s2)

h2s2 = "Substituting " + "\\\\ \\\"
subSteps.append(h2s2)

eqAux2 = "$" + latex(Function('P')(Symbol('x'), Symbol('y'))) + " = " +
latex(functionP) + "$" + "\\\\ \\\"
subSteps.append(eqAux2)

eqAux = Eq(Function('F')(Symbol('x'), Symbol('y')), Integral(functionP,x))
eq2s2 = "$" + latex(eqAux) + "$" + "\\\\ \\\"
subSteps.append(eq2s2)

h3s2 = latex("Integrating: ") + "\\\\ \\\"
subSteps.append(h3s2)

eqAux = Eq(Function('F')(Symbol('x'), Symbol('y')), functionF)
eq3s2 = "$" + latex(eqAux) + "$" + "\\\\ \\\"
subSteps.append(eq3s2)
```



```

#Step 3
'''
-----
# Step 03: Use the properties of the exact equation to find g(y)
-----
'''
solveArray.append([])
step = solveArray[2]
step.append(latex("- Use the properties of the exact equation to find
(y)") + "\\\ \\\ \\\ \\\")
step.append([])
subSteps = step[1]

partialF = diff(functionF, Symbol('y'))
leftPartial = Add(partialF, Mul(functionQ, Integer(-1)))
rightGSolveSide = solve(leftPartial, Derivative(functionG(Symbol('y')),
Symbol('y')))

h1s3 = "To find g(y) differentiate respect y the function: " + "\\\ \\\ \\\ \\\
\\\\"
subSteps.append(h1s3)
eq1s3 = "$" + latex(Derivative(Function('F')(Symbol('x')), Symbol('y')),
Symbol('y'))) + " = " + latex(partialF) + "$" + "\\\ \\\ \\\ \\\\"
subSteps.append(eq1s3)

eqAux = "$" + latex(Function('F')(Symbol('y'), Symbol('x'))) + "$" +
"\\ \\\ \\\ \\\\"
subSteps.append(eqAux)

h2s3 = "This by definition of a exact diferential equation must be equal
to " + "\\\ \\\ \\\ \\\\"
subSteps.append(h2s3)

eqAux1 = "$" + latex(Function('Q')(Symbol('x'), Symbol('y'))) + "$" +
"\\ \\\ \\\ \\\\"
subSteps.append(eqAux1)

h3s3 = "Hence equating: " + "\\\ \\\ \\\ \\\\"
subSteps.append(h3s3)

eq2s3 = "$" + latex(partialF) + " = " + latex(functionQ) + "$" + "\\\ \\\ \\\ \\\
\\\\"
subSteps.append(eq2s3)

h4s3 = "Solving for: " + "\\\ \\\ \\\ \\\\"
subSteps.append(h4s3)

eqAux2 = "$" + latex(Derivative(Function('g')(Symbol('y')), Symbol('y')))
+ "$" + "\\\ \\\ \\\ \\\\"
subSteps.append(eqAux2)

eq3s3 = "$" + latex(Derivative(Function('g')(Symbol('y')), Symbol('y')))
+ " = " + latex(rightGSolveSide[0]) + "$" + "\\\ \\\ \\\ \\\\"
subSteps.append(eq3s3)

```

Para construir el paso 3, se utiliza la definición de la ecuación diferencial exacta para llegar a completar la constante de integración (función de 'y') a la que se llega en el paso anterior. Con esto se llega a la solución implícita de la ecuación diferencial exacta. Se hace lo propio para el paso 4.

```
#Step 4
'''
-----
# Step 04: Get g(y) and particular F(x,y)
-----
'''
solveArray.append([])
step = solveArray[3]
step.append("- Get g(y) and particular F(x,y) " + "\\\\ \\")
step.append([])
subSteps = step[1]

gIntValue = integrate(rightGSolveSide[0], Symbol('y'))
functionF = Add(functionF, Mul(functionG(Symbol('y')), Integer(-1)),
gIntValue)

h1s4 = "Integrating both sides to get " + "$" +
latex(Function('g')(Symbol('y'))) + "$" + "\\\\ \\\"
subSteps.append(h1s4)

eq1s4 = "$" + latex(Function('g')(Symbol('y'))) + " = " +
latex(Integral(rightGSolveSide[0], Symbol('y'))) + "$" + "\\\\ \\\"
subSteps.append(eq1s4)

eq2s4 = "$" + latex(Function('g')(Symbol('y'))) + " = " +
latex(gIntValue) + "$" + "\\\\ \\\"
subSteps.append(eq2s4)

h2s4 = "Substituting this result into F(x,y)" + "\\\\ \\\"
subSteps.append(h2s4)

eq3s4 = "$" + latex(Function('F')(Symbol('x,y'))) + " = " +
latex(functionF) + "$" + "\\\\ \\\"
subSteps.append(eq3s4)

functionF = Add(functionF, Symbol('C'))
h3s4 = "Simplifying and using F(x,y) as a constant C, we get: " + "\\\\
\\\\\"
subSteps.append(h3s4)

functionF = simplify(functionF)

eq4s4 = "$" + latex(Integer(0)) + " = " + latex(functionF) + "$" + "\\\\
\\\\\"
subSteps.append(eq4s4)
```

```

'''
-----
# Step 05: Get the explicit solution solving for y
-----
'''

solveArray.append([])
step = solveArray[4]
step.append("- Get the explicit solution solving for y" + "\\\\ \\")
step.append([])
subSteps = step[1]

global finalSolve
finalSolve = []

def final_solve_timeout(expression, symbol):
    global finalSolve
    finalSolve = solve(expression, symbol)

try:
    process = PropagatingThread(target = final_solve_timeout,
args=(functionF, Symbol(functionName)))
    process.start()
    process.join(timeout=5)

    for singleSolve in finalSolve:
        eq1s5 = Eq(y(x), singleSolve)
        subSteps.append("$" + latex(eq1s5) + "$" + "\\\\ \\")

        # Analytic intervention for all the single solves if is teacher
        if (user_type == 'teacher'):
            print("Teacher")
            try:
                roots = []
                roots_process = PropagatingThread(target = get_roots, args =
[singleSolve, roots])
                roots_process.start()
                roots_process.join(timeout = 3)

                h0 = "Whose roots are: " + "\\\\ \\\"
                subSteps.append(h0)
                subIndex = 1
                for root in roots:
                    eq0 = "$" + "x_{" + str(subIndex) + "} = " + latex(root) + "$" +
"\\\\ \\\"
                    subIndex = subIndex + 1
                    subSteps.append(eq0)

            except Exception as e:
                print("Error with roots")
                print(e)

```

En caso de ser posible, el paso 5 busca despejar la función para obtener la solución explícita de la ecuación diferencial. Después en los pasos 6 y 7 se agregan los elementos propios para la solución de tipo investigador. Finalmente, se agregan los subpasos a la solución final y es regresada al controlado.

```

if (user_type == "teacher"):
    ...
    -----
    # Step 07: Generate Plot
    -----
    ...
    solveArray.append([])
    step = solveArray[6]
    step.append("- Graphs" + "\\\\ \\")
    step.append([])
    subSteps = step[1]

    for singleSolve in finalSolve:
        # Add plot step to solution
        print("Creating plot")

        try:
            plot_string = create_plot(singleSolve)[1:]
            plot_string = plot_string.replace("\\n", "")
        except Exception as e:
            print(e)

        subSteps.append(plot_string)
        print("Plot appended")

    except:
        subSteps.append("Can not get the explicit solution solving for " +
functionName + "\\\\ \\")

```

```

def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve(solveArray):
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [ display_solve(solveArray), solveArray ]

except CompletenessAnomaly as ca:

    if ca.partial_solution[0][0] == "partial integral":
        step = solveArray[len(solveArray) - 1]
        subSteps = step[1]
        subSteps.append("-----" + "\\\\ \\")

        for int_substep in ca.partial_solution[0][1]:
            subSteps.append(int_substep["text"] + "\\\\ \\")
            subSteps.append(int_substep["symbol"] + "\\\\ \\")
            subSteps.append("-----" + "\\\\ \\")

    ca.set_partial_solution(solveArray)

```

## 3.6. Solver de ODEs de Orden Superior con Coeficientes Constantes

El archivo del `solve_n_order.py` comienza con los imports necesarios para el control algebraico de los pasos de la solución. Después se define el gran cuerpo de la función, en el cual se inicializa el arreglo de pasos de la solución. Una vez que se tiene todo inicializado, se explora la ecuación para identificar cuáles son las funciones que caracterizan a la ODE de orden superior. Esta exploración se hace comparando los elementos que contiene cada término de la ecuación. Este proceso constituye el paso 1 de la solución.

```
from sympy import *
from sympy.abc import x
from sympy.parsing import parse_expr
from anomalies.completeness_anomaly import CompletenessAnomaly
```

```
def solveNLinear(odeString, functionName, user_type):
```

```
    try:
        ...
        -----
        # Init solve
        -----
        ...
        # init solve array
        solveArray = []
        ...
        -----
        # Initial algebraic analysis
        -----
        ...
        odeLeftString = odeString.split("=")[0]
        odeRightString = odeString.split("=")[1]

        odeLeftSym = parse_expr(odeLeftString)
        odeRightSym = parse_expr(odeRightString)

        y = Function(functionName)
        equation = Eq(odeLeftSym - odeRightSym, 0)
        equation = equation.subs(y(x), Symbol('y'))
        equationsolve = parse_expr("E**(r*x)")
```

El primer paso de la solución se construye mediante la sustitución de la solución propuesta de la ecuación diferencial considerando solo la parte homogénea de la misma. Este paso se describe mediante los textos y ecuaciones mostrando los pasos para llevarlo a cabo.

En un segundo paso, se presenta la ecuación polinómica que se espera obtener al aplicar las derivadas con la solución propuesta en el paso anterior y simplificando. El resultado de este procedimiento nos lleva naturalmente al paso 3, en el cual se resuelve dicha ecuación y se obtienen los valores característicos de la solución propuesta de la ecuación diferencial. En caso de que sea homogénea, con esto se ha llegado a la solución final de la ecuación diferencial al sustituir en la ecuación original (paso 4); de lo contrario, comienza el método de variación de parámetros.

```
...
-----
# Step 01: Substitute the potential solution
-----
...
solveArray.append([])
step = solveArray[0]
step.append(" -Arrange the equation and subsitute the potential solution"+
"\\\\ \\")
step.append([])
subSteps = step[1]

h0 = "Initial equation is given by: " + "\\\\ \\\"
subSteps.append(h0)

eq0 = "$" + latex(Eq(odeLeftSym, odeRightSym)) + "$" + "\\\\ \\\"
subSteps.append(eq0)

eq1 = "$" + latex(equation) + "$" + "\\\\ \\\"
subSteps.append(eq1)

h1 = "Lets propose the solution: " + "\\\\ \\\"
subSteps.append(h1)

eq2 = "$" + latex(Eq(Symbol('y(x)'), equationsolve)) + "$" + "\\\\ \\\"
subSteps.append(eq2)

equation = equation.subs(Symbol('y'), equationsolve)
h2 = "Substituting" + "\\\\ \\\"
subSteps.append(h2)

eq3 = "$" + latex(equation) + "$" + "\\\\ \\\"
subSteps.append(eq3)
```

```

...
-----
# Step 02: Evaluate the solution on the SDE
-----
...

solveArray.append([])
step = solveArray[1]
step.append(" -Apply derivatives and simplify" + "\\\\ \\")
step.append([])
subSteps = step[1]

#Declare temp functions
functionP = Integer(0)
functionQ = Integer(0)
functionF = Integer(0)
leftPartial = Integer(0)
functionG = Integer(0)
functionS = Integer(0)
functionT = Integer(0)
#Mul(pow(x, -2), equationsolve.subs(Symbol('r'), 1))
for term in equation.args[0].args:
    if 'Derivative' in str(term):
        if type(term) is Mul:
            for subterm in term.args:
                if 'Derivative' in str(subterm):
                    try:
                        expression = term.subs(Derivative(equationsolve, x,
subterm.args[1].args[1]), diff( equationsolve, x, subterm.args[1].args[1]))
                        functionF = Add( functionF , expression)
                    except:
                        expression = term.subs(Derivative(equationsolve, x), diff(
equationsolve, x))
                        functionF = Add( functionF , expression)
                else:
                    expression = term.subs(Derivative(equationsolve, x,
term.args[1].args[1]), diff( equationsolve, x , term.args[1].args[1]))
                    functionF = Add( functionF , expression)
            else:
                if 'r' in str(term):
                    functionF = Add( functionF , term)
                else:
                    functionT = Mul(term, -1)
#Derivatives are subsituted
equation = Eq(functionF, 0)
h1 = "Applying derivatives: " + "\\\\ \\\"
subSteps.append(h1)
eq1 = "$" + latex(equation) + "$" + "\\\\ \\\"
subSteps.append(eq1)
#Factorize
equation = factor(equation)
h2 = "Simplifying: " + "\\\\ \\\"
subSteps.append(h2)

eq2 = "$" + latex(equation) + "$" + "\\\\ \\\"

```



```

subSteps.append(eq2)

...

-----
# Step 03: Obtaining the roots
-----
...

solveArray.append([])
step = solveArray[2]
step.append(" -Obtain roots" + "\\\\ \\")
step.append([])
subSteps = step[1]

#Since ex cannot be 0, then function Q must be zero
functionQ = expand(Mul(functionF, pow(equationsolve, -1)))
equationaux = Eq(functionQ, 0)

#Simplify if possible

h0 = "can't be zero then: " + "\\\\ \\\"
subSteps.append(h0)

equationaux = factor(equationaux)
eq1 = "$" + latex(equationaux) + "$" + "\\\\ \\\"
subSteps.append(eq1)

#Solve equation for r in Q
solutions = []
solutionlist = roots(functionQ)
for solution in solutionlist:
    for i in range(0, solutionlist[solution]):
        solutions.append(solution)
solutionlist = solutions

if not solutionlist:
    solutionlist = Poly(functionQ.as_numer_denom()[0]).nroots(10, 80)
    solutionlist = [ round(number, 3) for number in solutionlist ]

functionF = Integer(0)
i = 0
real = 0
imag = 0
lastsolution = None
coef = Integer(1)
FuncArray = []
Func = []
Rows = []

```

```

for solution in solutionlist:
    if solution.compare(lastsolution):
        coef = Integer(1)
    else:
        coef = Mul(coef, x)

    # if isinstance(solution, float):
    #     solution = round(solution, 3)
    real = re(solution)
    imag = im(solution)

    subSteps.append("$" + latex(Eq(Indexed(Symbol('r'), i), solution )) +
"$" + "\\\ \\\ \\\")

    equationsolvere = equationsolve.subs(Symbol('r'), real)
    equationsolveim = equationsolve.subs(Symbol('r'), (imag
*I)).rewrite(cos)

    functionP = Mul(equationsolve.subs(Symbol('r'), solution), coef)
    Rows.append(functionP)

    functionP = Mul(functionP, Indexed(Symbol("C", real = True), i))
    functionQ = Mul(equationsolvere, equationsolveim , Indexed(Symbol("C",
real = True), i), coef)
    functionG = Add(functionG, functionQ)
    functionF = Add(functionF, functionP)

    imagpart = Mul(im(equationsolveim.subs(x, re(x))).subs(re(x),
x),Indexed(Symbol("K"), i), equationsolvere, coef)
    realpart = Mul(re(equationsolveim.subs(x, re(x))).subs(re(x),
x),Indexed(Symbol("C", real = True), i), equationsolvere, coef)

    if imag == 0:
        equationsolveim = realpart
    else:
        equationsolveim = Add(realpart, imagpart)

    functionQ = equationsolveim
    functionS = Add(functionS, functionQ)
    i+=1
    lastsolution = solution

```

La naturaleza del método puede llevar a dar con soluciones complejas para la ecuación polinómica. Se utiliza la identidad de Euler para obtener una representación real de las partes complejas de la solución (mediante la expansión en funciones trigonométricas).

```

...
-----
# Step 04: Substitute the roots and Final homogeneous solution
-----
...
solveArray.append([])
step = solveArray[3]
step.append("-Substitute the roots and Final homogeneous solution"+ "\\\n"
\\\n")
step.append([])
subSteps = step[1]

#Get Matrices
expression = 0
Rowsaux2 = Rows.copy()
for i in range(0, len(Rowsaux2)+1):
    Func = []
    Rows = Rowsaux2.copy()
    if i == len(Rowsaux2):
        Func.append(Rows)
        for j in range(0, len(Rowsaux2) - 1):
            Rowsaux = []
            for funcs in Rows:
                Rowsaux.append(diff(funcs, x))
            Rows = Rowsaux
            Func.append(Rowsaux)
    else:
        Rows[i] = 0
        Func.append(Rows)
        for j in range(0, len(Rowsaux2) - 1):
            Rowsaux = []
            for funcs in Rows:
                expression = Add(expression, diff(funcs, x))
                Rowsaux.append(diff(funcs, x))
            Rows = Rowsaux.copy()
            if j == len(Rowsaux2) - 2:
                Rowsaux[i] = 1
            else:
                Rowsaux[i] = 0
            Func.append(Rowsaux)
FuncArray.append(Func)

h0 = "Substituting the obtained solutions in the proposed solution and
adding up: " + "\\\n \\\n"
subSteps.append(h0)

```

```

equation = Eq(Symbol('y(x)'), functionF)
subSteps.append("$" + latex(equation) + "$" + "\\\\ \\\\")

equationHomAux = Eq(Symbol('y(x)'), factor(expand(functionG)))
if (str(equation) != str(equationHomAux)):
    subSteps.append("$" + latex(equationHomAux) + "$" + "\\\\ \\\\")
equation = equationHomAux

equationHomAux = Eq(Symbol('y(x)'), expand(functionS))
if (str(equation) != str(equationHomAux)):
    subSteps.append("$" + latex(equationHomAux) + "$" + "\\\\ \\\\")
equation = equationHomAux

```

```

...
-----
# Step 05: Equate Both Sides
-----
...

solveArray.append([])
step = solveArray[4]
step.append(" -Obtain non constant coeficcients of complementary
function"+ "\\\\ \\\\")
step.append([])
subSteps = step[1]

h0 = "Searching up for the system of matrices: " + "\\\\ \\\\"
subSteps.append(h0)

if functionT != 0:
#Create and DIsplay Matrices
    Matrices = []
    for matrix in FuncArray:
        Matrices.append(Matrix(matrix))
        subSteps.append("$" + latex(Matrix(matrix)) + "$" + "\\\\ \\\\")

#Calculate the determinant of each Matrix
    solveArray.append([])
    step = solveArray[5]
    step.append(" -Obtaining determinants of each matrix:"+ "\\\\ \\\\")
    step.append([])
    subSteps = step[1]

    h0 = "Obtaining determinants of each function: " + "\\\\ \\\\"
    subSteps.append(h0)
    Dets = []
    j=0
    for matrix in Matrices:
        deti = matrix.det()
        subSteps.append("$" + latex(Eq(Indexed(Symbol('P'), j), deti)) + "$" +
"\\\\ \\\\")
        Dets.append(deti)
        j+=1

```

```

...
-----
# Step 06: Get Explicit Solve
-----
...
solveArray.append([])
step = solveArray[5]
step.append("-Obtaining Integral factors:" + " \\ \\ \\ \\ ")
step.append([])
subSteps = step[1]

#Calculate Integral Factors
Factors = []
subSteps.append("Taking into account that " + "$" +
latex(Eq(Function('T')(x), functionT)) + "$" + " \\ \\ \\ \\ ")
for i in range(0, len(Dets)-1):

    subSteps.append("$" + latex(Eq(Indexed(Symbol('U'), i),
Integral(nsimpify(Mul(Function('T')(x), Indexed(Symbol('P'), i),
pow(Indexed(Symbol('P'), len(Dets)-1), -1))), x))) + "$" + " \\ \\ \\ \\ ")
    subSteps.append("$" + latex(Eq(Indexed(Symbol('U'), i),
Integral(nsimpify(Mul(Function('T')(x), Dets[i], pow(Dets[len(Dets)-1], -
1))), x))) + "$" + " \\ \\ \\ \\ ")
    subSteps.append("$" + latex(Eq(Indexed(Symbol('U'), i),
Integral(nsimpify(Mul(functionT, Dets[i], pow(Dets[len(Dets)-1], -1))), x)))
+ "$" + " \\ \\ \\ \\ ")

    auxExpr = expand(Mul(functionT.simplify(), Dets[i],
pow(Dets[len(Dets)-1], -1)))
    auxExpr = auxExpr.rewrite(cos)
    auxExpr = auxExpr.subs(x, re(x))
    auxExpr = im(auxExpr)
    auxExpr = simplify(Mul(auxExpr, I))

    # if functionT.is_real:
    expandAux = expand(Mul(functionT.simplify(), Dets[i],
pow(Dets[len(Dets)-1], -1)))
    u = integrate(expandAux, x)
    # else:
    # u = integrate(expand(Mul(simplify(functionT.rewrite(cos)),
Dets[i], pow(Dets[len(Dets)-1], -1)), x))

    Factors.append(u)
    subSteps.append("$" + latex(Eq(Indexed(Symbol('U'), i), u)) + "$" +
" \\ \\ \\ \\ ")

```

```

...
-----
# Step 07: Get Explicit Solve
-----
...
solveArray.append([])
step = solveArray[6]
step.append(" -Obtain final complement by substituting the integral
factors" + "\\\ \\\ \\\ \\\ " + "in the constants of the homogeneous solution:" +
"\\ \\\ \\\ \\\ ")
step.append([])
subSteps = step[1]

functionT = functionF
for i in range(0, len(Dets)-1):
    functionT = functionT.subs(Indexed(Symbol('C', real = True), i),
Factors[i])

    h0 = latex("Substitution: ") + "\\\ \\\ \\\ \\\ "
    subSteps.append(h0)
    subSteps.append("$" + latex(expand(Eq(Symbol('Y(x)'), functionT))) + "$"
+ "\\\ \\\ \\\ \\\ ")
    subSteps.append("$" + latex(expand(Eq(Symbol('Y(x)'),
simplify(functionT)))) + "$" + "\\\ \\\ \\\ \\\ ")

    if not (functionT.is_real):
        h1 = "Rewriting complex functions in terms of cos and sin: " + "\\\ \\\
\\ \\\ "
        subSteps.append(h1)
        subSteps.append("$" + latex(Eq(Symbol('Y(x)'),
simplify(functionT.rewrite(cos)).rewrite(cos))) + "$" + "\\\ \\\ \\\ \\\ ")

```

La variación de parámetros se lleva a cabo utilizando las propiedades teóricas del Wroksiano en la parte no homogénea que fue encontrada en la ODE solicitada. En caso de que esta parte no sea nula, se obtiene el valor y es utilizado para la variación de la solución original (para el caso homogéneo) de la ODE. Es por esta razón que existen dos versiones para el paso 7.

```

...
-----
# Step 07: Substitute integral factors
-----
...
solveArray.append([])
step = solveArray[6]
step.append(" -Obtain final complement by substituting the integral
factors" + " \\\ \\\ \\\ \\\ " + "in the constants of the homogeneous solution:" +
" \\\ \\\ \\\ \\\ ")
step.append([])
subSteps = step[1]

subSteps.append(" -Finally the general solution is the addition of the
homogeneous solution and the particular solution:" + " \\\ \\\ \\\ \\\ ")

functionF = Add(simplify(functionF), functionT)
h0 = "Adding both the final solution is: " + " \\\ \\\ \\\ \\\ "
subSteps.append(h0)

eq1 = "$" + latex(simplify(expand(Eq(Symbol('y(x)'), functionF)))) + "$"
+ " \\\ \\\ \\\ \\\ "
subSteps.append(eq1)

if not functionF.is_real:
    functionF = trigsimp(logcombine(simplify(functionF.rewrite(cos)),
force=True))

h1 = "Expressing complex terms as sin and cos: " + " \\\ \\\ \\\ \\\ "
eq2 = "$" + latex(expand(Eq(Symbol('y(x)'), functionF))) + "$" + " \\\ \\\
\\ \\\ "

functionFAux = expand(functionF)

realpart = re(functionFAux.subs(x, re(x))).subs(re(x), x)
imagpart = im(functionFAux.subs(x, re(x))).subs(re(x), x)
for i in range(0, len(Dets)-1):
    imagpart = imagpart.subs(Indexed(Symbol("C", real=True),
i), Indexed(Symbol("K", real=True), i))

functionFAux = imagpart + realpart
eq3 = "$" + latex(Eq(Symbol('y(x)'),
simplify(simplify(functionFAux)))) + "$" + " \\\ \\\ \\\ \\\ "

imagpart = abs(imagpart)
if ((len(imagpart.args) > 1) or not (type(imagpart.args[0]) is arg)):
    subSteps.append(h1)
    subSteps.append(eq2)
    subSteps.append(eq3)

```

```

def display_step(step):
    stepStr = ""
    for subStep in step:
        stepStr += str(subStep)
    return stepStr

def display_solve(solveArray):
    solveStr = ""
    for stepAux in solveArray:
        solveStr += stepAux[0]
        solveStr += display_step(stepAux[1])
    return solveStr
return [ display_solve(solveArray), solveArray ]

except CompletenessAnomaly as ca:

    if ca.partial_solution[0][0] == "partial integral":
        step = solveArray[len(solveArray) - 1]
        subSteps = step[1]
        subSteps.append("-----" + "\\\\\\ \\\\")

        for int_substep in ca.partial_solution[0][1]:
            subSteps.append(int_substep["text"] + "\\\\\\ \\\\")
            subSteps.append(int_substep["symbol"] + "\\\\\\ \\\\")
            subSteps.append("-----" + "\\\\\\ \\\\")

    ca.set_partial_solution(solveArray)

raise ca

```

La solución final es enviada al controlador una vez que se agregan los detalles adicionales propios de una solución de tipo investigador si hace se requiere. Con esto se concluye la exposición de los solvers de las ecuaciones diferenciales que el sistema puede resolver.



## 4. Pasos algebraicos

---

El archivo `algebraics.py` contiene la información correspondiente a los pasos algebraicos que son utilizados durante la solución de la ecuación diferencial. Grosso modo, estas operaciones constituyen todas las intervenciones algebraicas que manipulan a las expresiones simbólicas en el sistema. Para esta tarea, se utilizan los métodos propios que ya provee SymPy. La diferencia radica en que estos métodos modifican la dificultad global de la solución a medida que son invocados; además verifican que la dificultad se encuentre del rango tolerado, de lo contrario arrojan la excepción de completitud que trunca la solución de la ODE.

```
# Classification error on classify call
from anomalies.completeness_anomaly import CompletenessAnomaly
from sympy import *
from utils.constant import \

EXPAND_STEP_DIFFICULTY, \
FACTOR_STEP_DIFFICULTY, \
MAX_GLOBAL_DIFFICULTY, \
SIMPLIFY_STEP_DIFFICULTY, \
SOLVE_STEP_DIFFICULTY, \
SUBSTITUTION_STEP_DIFFICULTY

import solvers.controller as controller
```

```

def alg_add(param1, param2):
    """
    ## Params
    * param1 (symbolic expression)
    * param2 (symbolic expression)

    ## Return (symbolic expression)
    * param1 + param2 and adds the amount of difficulty to the global count
    """

    controller.global_difficulty = controller.global_difficulty +
    ALGEBRAIC_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return Add(param1, param2)

def alg_subs(param1, param2):
    """
    ## Params
    * param1 (symbolic expression)
    * param2 (symbolic expression)

    ## Return (symbolic expression)
    * param1 - param2 and adds the amount of difficulty to the global count
    """

    controller.global_difficulty = controller.global_difficulty +
    ALGEBRAIC_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return Add(param1, Mul(param2, -1))

def alg_mul(param1, param2):
    """
    ## Params
    * param1 (symbolic expression)
    * param2 (symbolic expression)

    ## Return (symbolic expression)
    * param1 * param2 and adds the amount of difficulty to the global count
    """

    controller.global_difficulty = controller.global_difficulty +
    ALGEBRAIC_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return Mul(param1, param2)

```

```

def alg_div(param1, param2):
    """
    ## Params
    * param1 (symbolic expression)
    * param2 (symbolic expression)

    ## Return (symbolic expression)
    * param1 / param2 and adds the amount of difficulty to the global count
    """

    controller.global_difficulty = controller.global_difficulty +
    ALGEBRAIC_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return Mul(param1, Pow(param2, -1))

def alg_solve(equation, variable):
    """
    ## Params
    * equation (symbolic expression)
    * variable (symbolic expression)

    ## Return (symbolic expression)
    * solve variable as a function of the given equation
    """

    controller.global_difficulty = controller.global_difficulty +
    SOLVE_STEP_DIFFICULTY
    try:
        if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
            raise CompletenessAnomaly(["", []])
        return solve(equation, variable)
    except:
        print("Couldn't make the solve for the equation")
        raise CompletenessAnomaly(["", []])

def alg_simplify(param1):
    """
    ## Params
    * param1 (symbolic expression)

    ## Return (symbolic expression)
    * simplified expression of param1
    """

    controller.global_difficulty = controller.global_difficulty +
    SIMPLIFY_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return simplify(param1)

```

```

def alg_expand(param1):
    """
    ## Params
    * param1 (symbolic expression)

    ## Return (symbolic expression)
    * expanded expression of param1
    """

    controller.global_difficulty = controller.global_difficulty +
EXPAND_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return param1.expand(force = True)

def alg_factor(param1):
    """
    ## Params
    * param1 (symbolic expression)

    ## Return (symbolic expression)
    * factored expression of param1
    """

    controller.global_difficulty = controller.global_difficulty +
FACTOR_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return factor(param1)

def alg_mul_inv(param1):
    """
    ## Params
    * param1 (symbolic expression)

    ## Return (symbolic expression)
    * mul inverse of param1
    """

    controller.global_difficulty = controller.global_difficulty +
ALGEBRAIC_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return Pow(param1, Integer(-1))

```

```

def alg_substitution(expression, old_variable, new_variable):
    """
    ## Params
    * expression (symbolic expression)
    * old_variable (symbolic expression)
    * new_variable (symbolic expression)

    ## Return (symbolic expression)
    * substitutes in the expression the old_variable with the new_variable and
    returns the
    new expression
    """

    controller.global_difficulty = controller.global_difficulty +
    SUBSTITUTION_STEP_DIFFICULTY
    if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
        raise CompletenessAnomaly(["", []])
    return expression.subs(old_variable, new_variable)

```

## 5. Pasos analíticos

El archivo `analytics.py` contiene los métodos utilizados a la hora de construir una solución para el modo investigador. Estos métodos son: obtener raíces, máximos y mínimos locales, y puntos de inflexión de una función en particular (en este caso, la solución encontrada de la ecuación diferencial). Además, se define el método para generar el gráfico de la función mediante la variación de las constantes que aparezcan en la solución de la ODE; para esta tarea se genera un archivo temporal y se correspondiente Encoding en una string en Base64.

```
from sympy import *
import base64
import uuid
import os
x = Symbol('x')
# Researcher Properties and parameters

def get_roots(expression, solution_list):
    print(expression)
    solutions = []
    # expression = expression.subs(Symbol('C'), 0)
    try:
        solutions = solve(Eq(expression, 0), x)
        if len(solutions) == 0:
            raise Exception()
        for solution in solutions:
            solution_list.append(solution)
    except:
        try:
            solutions = Poly(expression.as_numer_denom()[0]).nroots(10, 80)
            print(solutions)
            solutions = [ round(number, 4) for number in solutions ]
        except:
            try:
                solutions = nsolve(expression, 0, verify = false)
                for solution in solutions:
                    solution_list.append(solution)
            # solutionlist = [ round(number, 3) for number in solutionlist ]
        except Exception as e:
            raise e
    # if not solutionlist:
    #     solutionlist = Poly(expression.as_numer_denom()[0]).nroots(10, 80)
    #     solutionlist = [ round(number, 3) for number in solutionlist ]
```

```

def validate(expression, solution_list):
    tolerance = 0.5
    if solution_list is None:
        return False
    for solution in solution_list:
        eval = expression.evalf(subs = {x:solution})
        if re(eval) > tolerance or re(eval) < -1 * tolerance:
            return False
    return True

def max_min(expression, solution_list):
    # expression = parse_expr(expression)
    try:
        dexpression = diff(expression, x)
        solutions = []
        get_roots(dexpression, solutions)
        if validate(dexpression, solutions):
            for solution in solutions:
                solution_list.append(solution)
    except Exception as e:
        raise e

def inflexion_points(expression, solution_list):
    # expression = parse_expr(expression)
    try:
        dexpression = diff(expression, x, 2)
        solutions = []
        get_roots(dexpression, solutions)
        if validate(dexpression, solutions):
            for solution in solutions:
                solution_list.append(solution)
    except Exception as e:
        raise e

def create_plot(expression):
    p0 = plot(expression.subs(Symbol('C'), -10), show = False)
    min = -9
    max = 10
    for i in range(min, max, 1):
        pi = plot(expression.subs(Symbol('C'), i), show = False, ylim = (-300,
300))
        p0.append(pi[0])
    try:
        id = str(uuid.uuid1())
        file = f'analytics/images/{id}.png'
        p0.save(file)
        base64_string = None
        with open(file, 'rb') as image_file:
            base64_string = str(base64.encodebytes(image_file.read()))
        os.remove(file)
        return base64_string
    except Exception as e:
        print(e)

```

## 6. Anomalías del servidor

---

### 6.1. Anomalía de clasificación

El archivo `Classification_anomaly.py` contiene la definición de la clase que representa a la excepción que será lanzada en el servidor a la hora de buscar clasificar una ODE. Esta clase tiene por clase base a **Exception** y tiene una estructura muy similar: un mensaje y un cuerpo. Se agrega un método para transformar a formato JSON (en este caso, un diccionario de Python) para cuando se agregue como parte de la respuesta al usuario en caso de ser necesario. Además, tiene un campo para almacenar la solución final de la ODE, ya que al ser detectada por el controlador busca resolver la ODE mediante una intervención completa de SymPy. La solución final consiste solo en una expresión simbólica contenida en un solo paso.

```
class ClassificationAnomaly(Exception):
    def __init__(self):
        self.base_message = "It was not possible to complete the solution of the
differential equation because it could not be detected by our server or the
backup system as one of the supported types."
        self.final_solve = []
        super().__init__(self.base_message)

    def set_final_solve(self, solve):
        self.final_solve = solve

    def to_json(self):
        return {"message": self.base_message, "solution": self.final_solve}
```



## 6.2. Anomalía de completitud

El archivo `Classification_anomaly.py` contiene la definición de la clase que representa a la excepción que será lanzada en el servidor a la hora de resolver una ODE. Esta clase tiene por clase base a **Exception** y tiene una estructura muy similar: un mensaje y un cuerpo. Se agrega un método para transformar a formato JSON (en este caso, un diccionario de Python) para cuando se agregue como parte de la respuesta al usuario en caso de ser necesario. Además, tiene un campo para almacenar la solución parcial de la ODE, ya que al ser detectada por el controlador busca resolver la ODE mediante una intervención completa de SymPy. La solución final consiste en todos los pasos que se hayan podido completar y la intervención adicional de SymPy.

En general, las anomalías de completitud pueden ser lanzadas por un sobreflujo de dificultad global o local, así como por alguna obstrucción para la generación del siguiente paso; esta situación suele ser mucho más común en las intervenciones del integrador que en las del módulo de pasos algebraicos.

```
class CompletenessAnomaly(Exception):
    def __init__(self, partial_solution):
        self.base_message = "It was not possible to complete the solution of
the differential equation. The server managed to perform these steps: "
        super().__init__(self.base_message)
        self.partial_solution = partial_solution

    def set_partial_solution(self, partial_solution):
        self.partial_solution = partial_solution

    def append_final_solve(self, solve):
        self.partial_solution.append(solve)

    def to_json(self):
        return {"partial": self.partial_solution, "message": self.base_message,
"kind": self.kind}
```

## 7. Clasificador

---

### 7.1. Introducción

El `Classifier.py` tiene la tarea de clasificar una ODE en alguna de las 6 categorías que el sistema puede resolver. El archivo se compone de un método a exportar y una serie de métodos propios del módulo; el método a exportar recibe a la ODE y retorna una string que identifica el tipo de ODE que fue interpretado durante la etapa de clasificación. Dado que una misma ODE puede entrar en varias clasificaciones de manera simultánea, se busca clasificarla en orden de dificultad de tipo para garantizar que la solución presentada al usuario sea la más simple posible.

Cada uno de los métodos particulares de clasificación retorna un booleano que indica si la ecuación es del tipo buscado. Para llevar a cabo la etapa de clasificación, se utiliza mayoritariamente el primer paso de cada una de las soluciones y se busca que las funciones características de cada tipo de ODE coincidan en composición con la forma buscada (por lo general, las variables que lo componen). La exploración sobre cada uno de los términos se lleva a cabo por medio de la búsqueda con expresiones regulares y comparación de tipos.

El archivo comienza con la importación de las dependencias de SymPy para el parseo de expresiones y las clases simbólicas de Python. Además se incluye a la anomalía de clasificación, la cual será arrojada hacia el controlador en caso de que no sea posible clasificar a la ODE en uno de los tipos soportados por el sistema.

```
from anomalies.classification_anomaly import ClassificationAnomaly
from sympy import *
from sympy.abc import x,z
from sympy.parsing import parse_expr
```

## 7.2. Clasificador de ODEs Separables de Primer Orden

```
def checkSeparable(odeString, functionName):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]
    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)
    y = Function(functionName)
    equation = Eq(odeLeftSym - odeRightSym, 0)
    left = equation.args[0]
    try:
        express = solve(Eq(left, Integer(0)), Derivative(y(x), x))
    except:
        return False
    if (len(express) == 0):
        return False
    aux = simplify(express[0])
    aux = aux.expand(force=True)
    aux = factor(aux)

    functionF = Integer(1)
    functionG = Integer(1)
    if type(aux) is Add:
        if functionName in str(aux):
            functionG = aux
        else:
            functionF = aux
    else:
        if not (functionName in str(aux)):
            functionF = aux
        else:
            for term in aux.args:
                if functionName in str(term):
                    functionG = Mul(functionG, term)
                else:
                    functionF = Mul(functionF, term)

    if functionName in str(functionF):
        return False
    functionG = functionG.subs(y(x), Symbol(functionName))
    # functionG = functionG.subs(Symbol('x'), Symbol(functionName))
```

```

if 'x' in str(functionG):
    return False
# functionG = functionG.subs(Symbol(functionName), Symbol('x'))
functionG = functionG.subs(Symbol(functionName), y(x))

if (Mul(functionF, functionG)) == aux:
    return True
return False

```

## 7.3. Clasificador de ODEs Homogéneas de Primer Orden

```

def checkHomogeneous(odeString):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)

    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)

    left = equation.args[0]
    exp = solve(left, Derivative(y(x), x))
    aux = expand(exp[0])

    left = Derivative(y(x), x)
    functionF = aux

    u = Function('u')

    functionF = functionF.subs(y(x), Mul(u(x), x))
    left = Add(Mul(Derivative(u(x), x), x), u(x))

    left = Add(left, Mul(functionF, Integer(-1)))
    left = expand(left)
    separableODE = Eq(left, Integer(0))
    try:
        print(str(separableODE.args[0]) + "= 0")
        return checkSeparable(str(separableODE.args[0]) + "= 0", "u")
    except:
        return False

```

## 7.4. Clasificador de ODEs Lineales de Primer Orden

```
def checkLinear(odeString):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)

    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)
    left = equation.args[0]
    try:
        express = solve(Eq(left, Integer(0)), Derivative(y(x), x))
    except:
        return False
    if (len(express) == 0):
        return False
    aux = expand(express[0])

    left = Derivative(y(x), x)

    functionF = parse_expr("0")
    functionG = parse_expr("0")

    for term in aux.args:
        if "y" in str(term):
            functionF = Add(functionF, Mul(term, Pow(y(x), Integer(-1))))
        else:
            functionG = Add(functionG, term)

    functionF = Mul(functionF, Integer(-1))
    functionF = simplify(functionF)
    functionG = simplify(functionG)

    if 'y' in str(functionF):
        return False
    if 'y' in str(functionG):
        return False
    right = Add(functionG, Mul(Integer(-1), functionF, y(x)))
    right = right.expand(force=True)

    if Add(aux, Mul(Integer(-1), right)).simplify() == Integer(0):
        return True
    return False
```

## 7.5. Clasificador de ODEs Reducibles a Lineares de Primer Orden (Forma de Bernoulli)

```
def checkReducibleLinear(odeString):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)

    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)

    left = equation.args[0]
    exp = solve(left, Derivative(y(x), x))
    aux = expand(exp[0])

    functionF = parse_expr("0")
    functionG = parse_expr("0")

    aux = Mul(aux, Pow(y(x), Integer(-1)))
    aux = simplify(aux)

    for term in aux.args:
        if 'y' in str(term):
            for subTerm in term.args:
                if 'y' in str(subTerm):
                    try:
                        n = Add(subTerm.args[1], Integer(1))
                        subG = Mul(term, Pow(subTerm, Integer(-1)))
                        functionG = Add(functionG, subG)
                    except:
                        n = None
                else:
                    functionF = Add(functionF, term)

    print(n)
    print(functionF)
    print(functionG)

    return False

    if functionG != 0 and functionF != 0:
        return True
    else:
        return False
```

## 7.6. Clasificador de ODEs Exactas de Primer Orden

```
def checkExact(odeString):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)

    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)
    equation = equation.subs(y(x), Symbol('y'))

    functionP = Integer(0)
    functionQ = Integer(0)

    for term in equation.args[0].args:
        if 'Derivative' in str(term):
            functionQ = Add(functionQ, Mul(term, Pow(Derivative(Symbol('y')),
x), Integer(-1))))
        else:
            functionP = Add(functionP, term)

    if (functionP == 0 or functionQ == 0):
        return False

    partialP = diff(functionP, Symbol('y'))
    partialQ = diff(functionQ, Symbol('x'))

    if Add(partialP, Mul(Integer(-1), partialQ)).simplify() == Integer(0):
        return True

    return False
```

## 7.7. Clasificador de ODEs de Orden Superior con Coeficientes Constantes

```
def checkSuperiorOrder(odeString):
    # Testing completeness anomaly
    # return False

    odeLeftString = odeString.split("=")[0]
    odeRightString = odeString.split("=")[1]

    odeLeftSym = parse_expr(odeLeftString)
    odeRightSym = parse_expr(odeRightString)
    y = Function('y')
    equation = Eq(odeLeftSym - odeRightSym, 0)
    equation = equation.subs(y(x), Symbol('y'))
    equationsolve = parse_expr("E**(r*z)")

    for term in equation.args[0].args:
        if 'Derivative' in str(term):
            if type(term) is Mul:
                for subterm in term.args:
                    if 'Derivative' in str(subterm):
                        try:
                            expression = term.subs(Derivative(equationsolve,
z, subterm.args[1].args[1]), diff( equationsolve, x, subterm.args[1].args[1]))
                            functionF = Add( functionF , expression)
                            if 'x' in str(functionF):
                                return False
                        except:
                            expression = term.subs(Derivative(equationsolve,
z), diff( equationsolve, x))
                            functionF = Add( functionF , expression)
                            if 'x' in str(functionF):
                                return False
                    else:
                        expression = term.subs(Derivative(equationsolve, z,
term.args[1].args[1]), diff( equationsolve, z , term.args[1].args[1]))
                        functionF = Add( functionF , expression)
                        if 'x' in str(functionF):
                            return False
            else:
                if 'r' in str(term):
                    functionF = Add( functionF , term)
                    if 'x' in str(functionF):
                        return False
                else:
                    functionT = Mul(term, -1)

    return True
```



## 7.8. Clasificación de respaldo con SymPy

En caso de que la ODE no sea identificada con éxito por ninguno de los clasificadores que se han desarrollado, se utiliza la intervención de Dsolve (en concreto, del método `classify_ode`) para verificar si SymPy es capaz de clasificar la ODE como uno de los tipos soportados por el sistema. En caso de que sea la Dsolve logre clasificarla con éxito, se continua con el flujo normal de solución como si uno de los clasificadores desarrollados hubiera hecho la tarea; de otra forma se arroja una anomalía de clasificación hacia el controlador.

```
# Check if the equation is linear of superior order with constant
coefficients
try:
    if checkSuperiorOrder(odeString):
        return "superior"
except:
    print("Non Superior Order")

# Use dsolve to classify the equation
try:
    odeSym = parse_expr(odeString.split('=')[0])
    odeEq = Eq(odeSym, 0)

    # Gets all the possible ways to see the equation
    odeClass = classify_ode(odeSym, Function('y')(x))

    # Iterate in odeClass list and check if match
    for odeType in odeClass:
        print(odeType)
        if odeType is 'separable':
            return 'separable'
        elif odeType is '1st_exact':
            return 'exact'
        elif odeType is '1st_linear':
            return 'linear'

except Exception as e:
    print(e.args[0])

raise ClassificationAnomaly()
```

## 8. Compilación a LaTeX

---

El módulo de LaTeX a PDF se encarga de compilar la solución que le fue entregada al usuario en formato PDF. Para esta tarea, se utiliza el LaTeX que fue entregado como solución al usuario y el módulo externo de **pdflatex**, así como la gestión de archivos locales del servidor para guardar la solución a compilar en un .tex, obtener el .pdf, codificarlo en Base64, regresarlo al cliente y borrar los archivos auxiliares para la compilación.

```
import uuid
import pdflatex as ptex
import base64
import os
```

```
def text_to_pdf(latex):
    # Create .tex file
    id = str(uuid.uuid1())
    filepath = f'compiler/files/{id}.tex'

    with open(filepath, "w") as file_tex:
        file_tex.write("\\documentclass{article}\\n")
        file_tex.write("\\usepackage{amsmath}\\n")
        file_tex.write("\\begin{document}\\n")
        file_tex.write("\\noindent\\n")
        file_tex.write(latex + "\\n")
        file_tex.write("\\end{document}\\n")

    # Create .pdf file
    os.system(f"pdflatex -output-directory=compiler/files {filepath}")

    # Read .pdf file
    encoded_string = None
    with open(f'compiler/files/{id}.pdf', 'rb') as file_pdf:
        encoded_string = str(base64.b64encode(file_pdf.read()))

    # Remove .tex, .pdf, .aux, .log files
    os.remove(filepath)
    os.remove(f'compiler/files/{id}.pdf')
    os.remove(f'compiler/files/{id}.aux')
    os.remove(f'compiler/files/{id}.log')
    encoded_string = encoded_string[2:]
    encoded_string = encoded_string[:-1]

    return encoded_string
```

# 9. Integrador

---

## 9.1. Introducción

El módulo integrador (`integrator.py`) tiene la función de llevar a cabo las integrales que son solicitadas por cada uno de los solvers para la solución de las ecuaciones diferenciales. La solución de estas integrales se da de manera simbólica y además genera la secuencia de subpasos necesarios para llegar a la solución. Como se expuso en la sección teórica, la idea es representar a la integral en cuestión en término de las integrales que se consideran atómicas en el sistema.

Al ser un proceso recursivo, el código del integrador es, en sí, la llamada a una función que construye un árbol de solución para la integral. Esta función se manda a llamar a sí misma en diferentes escenarios dependiendo de la situación en la que se encuentre, que van desde llamarse múltiples veces para la suma de varias integrales o bien para la resolución de una nueva integral generada a partir de una recursiva.

La sección de imports de este archivo incluye las variables definidas como límites en la búsqueda de la solución de integrales, el timer personalizado, las operaciones algebraicas, los catálogos tanto atómicos como recursivos y el manejo simbólico de SymPy.

```
from timers.custom_threads import PropagatingThread
from sympy import *
from algebraics.operations import *
from utils.constant import \
    ATOMIC_INTEGRAL_DIFFICULTY, \
    RECURSIVE_INTEGRAL_DIFFICULTY, \
    MAX_GLOBAL_DIFFICULTY, \
    MAX_INTEGRAL_LEVEL, \
    MAX_NODE_DIFFICULTY, \
    SYMPY_INTEGRAL

import solvers.controller as controller
import integrals.atomic_integrals as atm
import integrals.recursive_integrals as rec
```

El método para resolver una integral atómica itera sobre el catálogo de integrales (el cual se construye en base a las variables con respecto a las cuales se está llevando a cabo la integral). Para comparar si la integral en cuestión coincide con alguna del catálogo utiliza el método `match_integral` (definido después). Si coincide, entonces se adjunta el paso en un diccionario con un texto descriptivo y el símbolo correspondiente. En caso de que no coincida ninguna integral, retorna "None".

```
def int_atm_solve(expression, differential):
    # Variables of global scope in integrator module
    global variables
    variables = [
        {"symbol": Symbol('n'), "value": None},
        {"symbol": Symbol('a'), "value": None},
        {"symbol": Symbol('b'), "value": None},
        {"symbol": Symbol('c'), "value": None}
    ]

    # Create list of atm integrals using the appropriate differential
    atm.build_integrals(differential)

    # Factor expression for atm integral checking
    expression = alg_factor(expression)
    differential_expression = Mul(expression, Symbol('d'+str(differential)))

    # Iterate all along the list of atomic integrals and use the match
    function to
    # verify for a possible match
    index = 0
    for integral in atm.BASIC:
        if match_integral(differential_expression, integral, "atm", index):
            int_solution = atm.SOLVE[index]

            # Replace variables in integral solution
            for variable in variables:
                if srepr(variable['symbol']) in srepr(atm.SOLVE[index]):
                    int_solution = int_solution.subs(variable['symbol'],
variable['value'])

            # Return symbolic expression and difficulty
            return {"symbol": int_solution, "difficulty":
ATOMIC_INTEGRAL_DIFFICULTY}

            index = index + 1

    # If there's no match, then return None
    return None
```

De un modo similar, para resolver una integral recursiva se construye el catálogo y se itera sobre él. En este caso se regresa la nueva integral y el resultado parcial del paso recursivo en la solución. De igual forma, el valor para la no coincidencia de es "None". Ambos métodos son llamados después por el método que genera un árbol de solución.

```
def int_rec_solve(expression, differential):

    # Variables of global scope in integrator module
    global variables
    variables = [
        {"symbol": Symbol('n'), "value": None},
        {"symbol": Symbol('a'), "value": None},
        {"symbol": Symbol('b'), "value": None},
        {"symbol": Symbol('c'), "value": None}
    ]

    # Create list of atm integrals using the appropriate differential
    rec.build_integrals(differential)

    # Factor expression for atm integral checking
    expression = alg_factor(expression)
    differential_expression = Mul(expression, Symbol('d'+str(differential)))

    # Iterate all along the list of recursive integrals and use the match
    function to
    # verify for a possible match
    index = 0
    for integral in rec.RECURSIVE:
        if match_integral(differential_expression, integral, "rec", index):
            int_partial_solution = rec.PARTIAL_SOLVE[index]
            int_new_int = rec.NEW_INTEGRAL[index]

            # Replace variables in integral solution
            for variable in variables:
                if srepr(variable['symbol']) in
srepr(rec.PARTIAL_SOLVE[index]):
                    int_partial_solution =
int_partial_solution.subs(variable['symbol'], variable['value'])
                    int_new_int = int_new_int.subs(variable['symbol'],
variable['value'])

            # Return symbolic expression and difficulty
            return {"partial_symbol": int_partial_solution, "new_int_symbol":
int_new_int, "difficulty": RECURSIVE_INTEGRAL_DIFFICULTY}
            index = index + 1

    # If there's no match, then return None
    return None
```

Antes de pasar al creador de árboles, se tiene definido el método que se utilizará para suscribir en el timer el proceso de integral por defecto de SymPy.

```
def integrate_timeout(exp, dif):  
    global aux_int_sympy  
    aux_int_sympy = integrate(exp, dif)
```

En general, el método general de árboles de solución para integrales tiene el siguiente flujo de trabajo:

1. Revisar si el nivel de profundidad del árbol (longitud de rama) supera el máximo
2. Revisar si el integrando es un número (constante)
3. Revisar si el integrando está siendo multiplicado por una constante; esto es, de la forma  $kf(x)$  con  $k$  una constante
4. Revisar si el integrando está en el catálogo de integrales atómicas
5. Revisar si el integrando está en el catálogo de integrales recursivas
6. Revisar si el integrando puede representarse como la suma de funciones
7. Forzar integración por partes y verificar cada una de las posibles alternativas hasta encontrar alguna que funcione
8. Dejar la integral en manos de SymPy

## 9.2. Nivel de profundidad

El proceso recursivo de mandar a llamar a la función generadora de árboles está controlado por la variable de nivel. Existe un máximo valor que puede tomar esta profundidad, que indica a su vez la máxima cantidad de llamadas que puede hacer la función a sí misma antes de detener el proceso de búsqueda y solicitar la intervención de SymPy. Cuando esto ocurre, el proceso parcialmente resuelto es devuelto al usuario incluyendo el paso final de la intervención de SymPy para resolver la integral. Esto se manda hasta la primera llamada de la función generando una anomalía de completitud que va escalando entre la pila de llamadas hasta llegar. Este valor finalmente se regresa como anomalía de completitud al solver pero no detiene la solución de la ODE, sino solo de la integral en sí.

```

def tree_solve(expression, differential, level):
    global integral_solve_array
    global aux_int_sympy

    # Check level limit. If is too deep, then use integrate from sympy
    if level >= MAX_INTEGRAL_LEVEL:
        try:
            process = PropagatingThread(target=integrate_timeout,
args=(expression, differential))
            process.start()
            process.join(timeout=10)

            integral_solve_array.append({"left": expression,
"right": aux_int_sympy,
"level": level,
"difficulty": SYMPY_INTEGRAL,
"type": "SymPy Integral",
"symbol": "$\int{" + latex(expression) + "} d" + str(differential)
+ " = "+ latex(aux_int_sympy) + "$",
"text": "Using DSolve (backup system): "})

            return {"symbol": aux_int_sympy, "difficulty": SYMPY_INTEGRAL}
        except:
            raise CompletenessAnomaly(["", []])

```

### 9.3. Caso de Integrand Constante

En este caso se saca la constante de la integral y se resuelve la integral del diferencial (que está en el catálogo atómico).

```

# Check if the expression is a number
if expression.is_number:
    atomic_int = int_atm_solve(Integer(1), differential)
    right_side = alg_mul(expression, atomic_int["symbol"])

    integral_solve_array.append({"left": expression,
"right": right_side,
"level": level,
"difficulty": atomic_int["difficulty"],
"type": "Constant integral k",
"symbol": "$\int{" + latex(expression) + "} d" + str(differential) +
" = " +
    latex(right_side) + "$",
"text": "It is known that the solution is: "})
    return {"symbol": right_side, "difficulty": atomic_int["difficulty"]}

```

## 9.4. Caso de Integrando Constante por Función

En este caso se saca la constante de la integral, se divide la expresión original entre dicha constante y se manda a llamar a la función nuevamente aumentando en 1 el nivel de profundidad. Una vez que se tiene el resultado de la recursión, se multiplica por la constante que se retiró. Este será el resultado de esta integral.

```
# Check if the expression has the form k*f(x) with
# k a constant number. Try to solve the new integral
if type(expression) is Mul and expression.args[0].is_number:
    try:
        new_int = alg_div(expression, expression.args[0])
        prev_expression = "$\int{" + latex(expression) + "} d" +
str(differential) + "=" + latex(expression.args[0]) + "\int{" + latex(new_int)
+ "} d" + str(differential) + "$"

        integral_solve_array.append({"left": Integer(0),
"right": Integer(0),
"level": level,
"difficulty": 0,
"type": "Symbol",
"symbol": prev_expression,
"text": "Taking the constant out of the integral: "})

        aux_int = tree_solve(new_int, differential, level+1)
        integral_solve_array.append({"left": expression,
"right": aux_int["symbol"],
"level": level,
"difficulty": aux_int["difficulty"],
"type": "Constant integral kf(x)",
"symbol": "$\int{" + latex(expression) + "} d" + str(differential)
+ "=" +
        latex(Mul(aux_int["symbol"], expression.args[0]))+"$",
"text": "Multiplying back the constant we have: "})

        return { "symbol": alg_mul(expression.args[0], aux_int["symbol"]),
"difficulty": aux_int["difficulty"] }

    except CompletenessAnomaly as ca:
        raise ca
```



## 9.5. Caso de Integrando Atómico

Se manda a llamar a la función de resolver atómica. En caso de que el retorno sea None, significa que la integral no estaba en el catálogo y se continúa examinando. De otra forma, se agrega la solución de la integral de acuerdo con el diccionario que regresa el método.

```
# Check if the expression is atomic
atomic_int = int_atm_solve(expression, differential)
if atomic_int is not None:
    integral_solve_array.append({"left": expression,
                                "right": atomic_int["symbol"],
                                "level": level,
                                "difficulty": atomic_int["difficulty"],
                                "type": "Atomic",
                                "symbol": "$\int{" + latex(expression) + "} d" + str(differential) +
"=" +
                                latex(atomic_int["symbol"]) + "$",
                                "text": "It is known that the solution is: "})
    return atomic_int
```

## 9.6. Caso de Integrando Recursivo

Se manda a llamar a la función de resolver recursiva. En caso de que el retorno sea None, significa que la integral no estaba en el catálogo y se continúa examinando. De otra forma, se manda a llamar de nuevo a la función para la nueva integral generada; luego este resultado se añade con la otra parte de la integral recursiva y este será el valor de la integral en cuestión.

```

node_difficulty = 0

# Check if the expression is recursive one to one
# (from the list)
recursive_int = int_rec_solve(expression, differential)
if recursive_int is not None:
    node_difficulty = recursive_int["difficulty"]
    try:
        prev_expression = "$\int{" + latex(expression) + "} d" +
            str(differential) + "=" + latex(recursive_int["partial_symbol"]) + "+" +
            "\int{" + latex(recursive_int["new_int_symbol"].subs(Symbol('d' +
                str(differential))), 1)) + "} d" + str(differential) + "$"

        integral_solve_array.append({"left": Integer(0),
            "right": Integer(0),
            "level": level,
            "difficulty": 0,
            "type": "Symbol",
            "symbol": prev_expression,
            "text": "Using integration by parts, we rewrite the integral: "})

        solution_new_int =
tree_solve(recursive_int["new_int_symbol"].subs(Symbol('d' +
            str(differential))), 1), differential, level+1)
        node_difficulty = node_difficulty + solution_new_int["difficulty"]
        if node_difficulty >= MAX_NODE_DIFFICULTY:
            raise CompletenessAnomaly(["", []])

        right_side = alg_add(recursive_int["partial_symbol"],
solution_new_int["symbol"])
        integral_solve_array.append({"left": expression,
            "right": right_side,
            "level": level,
            "difficulty": node_difficulty,
            "type": "Recursive (list)",
            "symbol": "$\int{" + latex(expression) + "} d" + str(differential)
+ "=" +
            latex(right_side) + "$",
            "text": "Adding with the additional part of integration by parts:
"})

        return { "symbol": right_side, "difficulty": node_difficulty }
    except CompletenessAnomaly as ca:
        raise ca

```

## 9.7. Caso de Integrando en Suma de Funciones

```

# Expand expression to do the addition distribution and try to solve each
# sub integral
expression = alg_expand(expression)
int_solution = Integer(0)
if type(expression) is Add:
    prev_expression = "$\int{" + latex(expression) + "} d" +
str(differential) + "="
    for item in expression.args:
        partial_symbol = "\int{" + latex(item) + "} d" + str(differential)
+ "+"
        prev_expression = prev_expression + partial_symbol
    prev_expression = prev_expression[:-1]
    prev_expression = prev_expression + "$"
    integral_solve_array.append({"left": Integer(0),
"right": Integer(0),
"level": level,
"difficulty": 0,
"type": "Symbol",
"symbol": prev_expression,
"text": "We separate the integral into sums of integrals and solve
each of them: "})

    for item in expression.args:
        try:
            aux_int = tree_solve(item, differential, level+1)
            int_solution = alg_add(int_solution, aux_int["symbol"])
            node_difficulty = node_difficulty + aux_int["difficulty"]
            if node_difficulty >= MAX_NODE_DIFFICULTY:
                raise CompletenessAnomaly(["", []])

        except CompletenessAnomaly as ca:
            raise ca

# Check if the expression could be expressed as the addition of functions
if int_solution != Integer(0):
    integral_solve_array.append({"left": expression,
"right": int_solution,
"level": level,
"difficulty": node_difficulty,
"type": "Addition of integrals",
"symbol": "$\int{" + latex(expression) + "} d" + str(differential) +
"=" +
        latex(int_solution) + "$",
"text": "Adding the results of the integrals: "
    })

return {"symbol": int_solution, "difficulty": node_difficulty}

```

Se expande la función para ver si puede ser representada en forma de suma. En caso de que el resultado sea de tipo "Add", se procede con la solución de cada integral de cada uno de los sumandos. Las soluciones de cada una de ellas se registran en un arreglo que finalmente termina sumándose. El resultado de la suma será también el resultado de la integral.

## 9.8. Caso de Integración por Partes Forzada

Se expande la función para ver si puede ser representada en forma de suma. En caso de que el resultado sea de tipo "Add", se procede con la solución de cada integral de cada uno de los sumandos. Las soluciones de cada una de ellas se registran en un arreglo que finalmente termina sumándose. El resultado de la suma será también el resultado de la integral.

```
# Final test using forced integration by parts
expression = alg_factor(expression)
if len(expression.args) == 1 or type(expression) is not Mul:
    # Request a quick sympy intervention
    try:
        process = PropagatingThread(target=integrate_timeout,
args=(expression, differential))
        process.start()
        process.join(timeout=10)

        integral_solve_array.append({"left": expression,
"right": aux_int,
"level": level,
"difficulty": SYMPY_INTEGRAL,
"type": "SymPy Integral",
"symbol": "$\int{" + latex(expression) + "}" + str(differential)
+ " = " + latex(aux_int_sympy) + "$",
"text": "Using DSolve (backup system): "})
        return {"symbol": aux_int, "difficulty": SYMPY_INTEGRAL}
    except:
        # The requested integral could not be solved for any method
        raise CompletenessAnomaly([["", []]])
else:
    for integral_factor in expression.args:
        try:
            # Request a quick SymPy intervention to check if it's a
possible integral
```

```

process = PropagatingThread(target=integrate_timeout, args=(integral_factor,
differential))
    process.start()
    process.join(timeout=5)

    # The factor was quick-integrable. Proceed with integration by
parts
    factor_u = alg_div(expression, integral_factor)
    factor_du = diff(factor_u, differential)
    factor_v = aux_int

    factor_vdu = alg_mul(factor_v, factor_du)
    factor_uv = alg_mul(factor_u, factor_v)

    prev_expression = "$\int{" + latex(expression) + "} d" +
str(differential) + "=" + latex(factor_uv) + "-" + "\int{" + latex(factor_vdu)
+ "} d" + str(differential) + "$"
    integral_solve_array.append({"left": Integer(0),
    "right": Integer(0),
    "level": level,
    "difficulty": 0,
    "type": "Integral by parts",
    "symbol": prev_expression,
    "text": "Using integration by parts, we rewrite the integral:
"}})

    solution_new_int = tree_solve(factor_vdu, differential,
level+1)
    node_difficulty = node_difficulty +
solution_new_int["difficulty"]
    if node_difficulty >= MAX_NODE_DIFFICULTY:
        raise CompletenessAnomaly([["", []]])
    right_side = alg_subs(factor_uv, solution_new_int["symbol"])
    integral_solve_array.append({"left": expression,
    "right": right_side,
    "level": level,
    "difficulty": node_difficulty,
    "type": "Integral by parts",
    "symbol": "$\int{" + latex(expression) + "} d" +
str(differential) + "=" +
        latex(right_side) + "$",
    "text": "Adding with the additional part of integration by
parts: "})

    return { "symbol": right_side, "difficulty": node_difficulty }
    # Completeness Anomaly from the solve_tree iteration
except CompletenessAnomaly as ca:
    raise ca
    # Timeout exception or another exception on sympy integration
except:
    continue
    # Having exhausted all possible factors, throw an exception
    raise CompletenessAnomaly([["", []]])

```

## 9.9. Exportación del módulo

El módulo de integración es accesible por los solvers por medio del método **int\_solve**, el cuál llama a su vez al método generador de árboles del módulo integrador (es esta la llamada 0 de la pila recursiva). En caso se produzca una excepción en el proceso, se registra la solución parcial de la integral en la anomalía de completitud que luego es enviada al solver que hizo la llamada.

```
def int_solve(expression, differential):
    global integral_solve_array
    global aux_int_sympy

    integral_solve_array = []

    try:
        solution = tree_solve(expression, differential, 0)
        controller.global_difficulty = controller.global_difficulty +
solution["difficulty"]
        if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
            raise CompletenessAnomaly(["", []])
        print_solution()
        return {"solution": solution["symbol"], "steps": integral_solve_array}

    except CompletenessAnomaly:
        controller.global_difficulty = controller.global_difficulty +
MAX_NODE_DIFFICULTY
        if controller.global_difficulty >= MAX_GLOBAL_DIFFICULTY:
            raise CompletenessAnomaly(["", []])
        try:
            process = PropagatingThread(target=integrate_timeout,
args=(expression, differential))
            process.start()
            process.join(timeout = 10)

            print("Integral solved with sympy")
            print()
            print_solution()

            integral_solve_array.append({"left": expression,
"right": aux_int_sympy,
"level": 0,
"difficulty": SYMPY_INTEGRAL,
"type": "SymPy Integral",
"symbol": "$\int{" + latex(expression) + "} d" + str(differential)
+ " = " + latex(aux_int_sympy) + "$",
"text": "Using DSolve (backup system): "})
            return {"solution": aux_int_sympy, "steps": integral_solve_array}
        except Exception as e:
            raise CompletenessAnomaly(["partial integral",
integral_solve_array])
```

## 9.10. Comparación de integrales

La comparación de integrales se da mediante la verificación de tipos de cada uno de los elementos que se espera que compongan a una determinada integral del catálogo. Se itera sobre cada uno de los argumentos y se asignan las variables que en los catálogos aparecen como símbolos constantes. En la primera discrepancia se retorna falso.

```
def match_integral(expression, integral, tag, int_index):
    if integral == expression:
        return True
    global variables
    for variable in variables:
        if integral == variable['symbol']:
            if expression.is_number:
                if variable['value'] is None:
                    # Check if is a right value
                    EXCEPTIONS = []
                    if tag == "atm":
                        EXCEPTIONS = atm.EXCEPTIONS
                    elif tag == "rec":
                        EXCEPTIONS = rec.EXCEPTIONS
                    else:
                        return False

                    if verify_constant(EXCEPTIONS[int_index], integral,
expression):
                        variable['value'] = expression
                        return True
                    else:
                        return False
                else:
                    return variable['value'] == expression
            else:
                return False

    if type(expression) is type(integral):
        item_index = 0
        for item in integral.args:
            try:
                if match_integral(expression.args[item_index], item, tag,
int_index):
                    item_index = item_index + 1
                else:
                    return False
            except:
                return False
        return len(integral.args) == len(expression.args)
    else:
        return False
```

```

def verify_constant(exceptions, symbol, value):
    if len(exceptions) == 0:
        return True

    for exception in exceptions:
        if symbol == exception["symbol"]:
            if exception["type"] == "neq":
                # The injected and expected values are numbers
                if exception["value"].is_number:
                    return value != exception["value"]

                # The expected value is a symbol
                expected_variable = None
                for variable in variables:
                    if variable['symbol'] == exception['value']:
                        expected_variable = variable
                        break
                if expected_variable is None:
                    return False
                if expected_variable['value'] is None:
                    return True
                return value != expected_variable["value"]

            elif exception["type"] == "g":
                # The injected and expected values are numbers
                if exception["value"].is_number:
                    return value > exception["value"]

                # The expected value is a symbol
                expected_variable = None
                for variable in variables:
                    if variable['symbol'] == exception['value']:
                        expected_variable = variable
                        break
                if expected_variable is None:
                    return False
                if expected_variable['value'] is None:
                    return True
                return value > expected_variable["value"]

            elif symbol == Pow(exceptions["symbol"], 2):
                if exception["type"] == "neq":
                    # The expected value is a symbol
                    expected_variable = None
                    for variable in variables:
                        if Pow(variable['symbol'], 2) == exception['value']:
                            expected_variable = variable
                            break
                    if expected_variable is None:
                        return False
                    if expected_variable['value'] is None:
                        return True

```



```

if expected_variable['value'] is None:
    return True

    return Pow(value, 2) > Pow(expected_variable["value"], 2)

else:
    return False

```

```

def print_solution():
    global integral_solve_array
    for step in integral_solve_array:
        print("Left: " + str(step["left"]))
        print("Right: " + str(step["right"]))
        print("Level: " + str(step["level"]))
        print("Difficulty: " + str(step["difficulty"]))
        print("Type: " + str(step["type"]))
        print()

```

Durante el proceso de inyección de los valores en las variables simbólicas se verifica que estos valores sean consistentes entre sí y además sean valores permitidos en la forma simbólica de la integral. Un ejemplo claro de esto es en la integral de  $x^n$ , la cual tiene una forma polinómica como resultado siempre y cuando  $n \neq -1$ , en este caso se tiene una solución logarítmica. Los valores permitidos para cada integral están definidos dentro de los catálogos indicados en los arreglos de excepciones.

Existe una función de impresión de la integral que fue utilizada con propósitos de análisis del crecimiento de la dificultad y la efectividad de los pasos recursivos.

## 9.11. Actualización de los catálogos

En el archivo `updater.py` se tienen los métodos utilizados para la actualización de los catálogos de integrales (vistos en el archivo `api.py`). Estos eventos son invocados por una aplicación cliente dedicada únicamente a lanzar las peticiones de manera periódica a la aplicación. El método descrito a detalle junto con los catálogos fue agregado como anexo a este documento debido a su extensión y facilitar su lectura.

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore
from sympy import *
from shutil import move
from datetime import datetime
```

```
def get_client():
    cred = credentials.Certificate('./integrals/firebase-key.json')
    firebase_admin.initialize_app(cred)
    return firestore.client()

def read_hints(db):
    cat_ref = db.collection('recursive')
    int_docs = cat_ref.stream()
    for int_doc in int_docs:
        int_dict = int_doc.to_dict()

        hint_ref = cat_ref.document(int_doc.id).collection('hints')
        hint_docs = hint_ref.stream()

        for hint_doc in hint_docs:
            print("Here")
            hint_dict = hint_doc.to_dict()
            print(hint_dict)

            changes_ref = hint_ref.document(hint_doc.id).collection('changes')
            changes_doc = changes_ref.stream()

            change_text = ""
            for change_doc in changes_doc:
                change_dict = change_doc.to_dict()
                print(change_dict)

def push_integrals(db):
    #Defined at the end of the document ...
```

## 10. Intérpretes

El módulo de interpretación del servidor comienza con las dependencias de SymPy para el manejo simbólico, así como un parser de LaTeX a SymPy por defecto. También son esenciales las expresiones regulares.

```
from sympy import *
from sympy.abc import x
from sympy.parsing.latex import parse_latex

import re
```

```
def parseSympy(inputString):
    inputString = inputString.replace("D", "Derivative")
    inputString = inputString.replace("^", "**")
    inputString = inputString.replace("y", "y(x)")
    inputString = inputString.replace(
        "Derivative(y(x),x)", "(Derivative(y(x),x))")

    print(inputString)

    therms = inputString.split("=")
    if (len(therms) != 2):
        raise Exception("Invalid equation. Must be exactly one equal sign (=)")

    leftString = inputString.split("=")[0]
    rightString = inputString.split("=")[1]
    leftSymbol = parse_expr(leftString)
    rightSymbol = parse_expr(rightString)

    leftSide = Add(leftSymbol, Mul(rightSymbol, Integer(-1)))

    return leftSide
```

La función anterior obtiene la expresión en SymPy equivalente con base a una string recibida desde el cliente. En principio se utiliza una sustitución de caracteres y los intérpretes que posee SymPy para generar las clases simbólicas a partir de una string. Hay una pequeña etapa de verificación de formato de la ODE al verificar que contenga exactamente dos signos de igualdad.

```

def regexCorrect(variable):
    regexps = [r'(\sin[^\]]*\)\)\}', r'(\cos[^\]]*\)\)\}', r'(\tan[^\]]*\)\)\}',
               r'(\csc[^\]]*\)\)\}', r'(\sec[^\]]*\)\)\}', r'(\cot[^\]]*\)\)\}']

    for regexp in regexps:
        foo = re.compile(regexp)
        searchAll = foo.findall(variable)
        for oldValue in searchAll:
            variable = variable.replace(
                oldValue, "\\left(" + oldValue + "\\right)")
    return variable

def newRegexCorrect(variable):
    regexp = r'y\^\{[^\}]*\}\{\left(x \right)\}'
    foo = re.compile(regexp)
    searchAll2 = foo.findall(variable)
    for oldValue in searchAll2:
        startPowIndex = oldValue.index("{")
        endPowIndex = oldValue.index("}")
        powString = oldValue[(startPowIndex+1): endPowIndex]
        variable = variable.replace(
            oldValue, "M^{ " + powString + "}"
        )
    return variable

def fixMFunction(expression):
    for item in expression.args:
        if type(item) is Function('M'):
            arguments = item.args[0]
            expression = expression.subs(Function('M')(
                arguments), Mul(Symbol('M'), arguments))
        elif "Function('M')" in str(srepr(item)):
            newItem = fixMFunction(item)
            expression = expression.subs(item, newItem)
    return expression

```

Con base a los resultados particulares de cada expresión en LaTeX, se desarrollaron funciones para ajustar la traducción de vuelta a expresiones simbólicas. Por ejemplo, para el caso de las funciones trigonométricas fue necesario desarrollar una expresión regular para obtener el argumento sin que fuera afectado (esto es, conservar la dependencia y jerarquía de las operaciones inicial). Este mismo trabajo fue desarrollado para la función logaritmo, y en general las funciones que pueden anidarse. La función **parseLatex** es la que implementa estas correcciones sobre la ecuación en LaTeX para generar la forma simbólica original.

```

def fixLogarithm(expression):
    for item in expression.args:
        if type(item) is log:
            arguments = item.args[0]
            expression = expression.subs(item, ln(arguments))
        elif "log" in str(srepr(item)):
            newItem = fixLogarithm(item)
            expression = expression.subs(item, newItem)
    return expression

def fixXFunction(expression):
    for item in expression.args:
        if type(item) is Function('x'):
            arguments = item.args[0]
            expression = expression.subs(Function('x')(
                arguments), Mul(Symbol('x'), arguments))
        elif "Function('x')" in str(srepr(item)):
            newItem = fixXFunction(item)
            expression = expression.subs(item, newItem)
    return expression

```

```

def parseLatex(inputString):
    inputString = inputString.replace("\\\\", "\\")
    inputString = inputString.replace(
        "y{\\left(x \\right)}", "M"
    )
    inputString = inputString.replace(
        "log", "ln"
    )
    inputString = regexCorrect(inputString)
    inputString = newRegexCorrect(inputString)

    therms = inputString.split("=")
    if (len(therms) != 2):
        raise Exception("Invalid equation. Must be exactly one equal sign (=)")

    leftString = inputString.split("=")[0]
    leftSymbol = parse_latex(leftString)

    leftSymbol = fixMFunction(leftSymbol)
    leftSymbol = fixLogarithm(leftSymbol)
    leftSymbol = fixXFunction(leftSymbol)

    leftSymbolStr = str(leftSymbol)
    leftSymbolStr = leftSymbolStr.replace("M", "y(x)")
    leftSymbol = parse_expr(leftSymbolStr)

    leftSymbol = leftSymbol.subs(Symbol('e'), E)

    return leftSymbol

```

# 11. Temporizadores

La intervención de procesos simbólicos de SymPy utilizados a manera de respaldo de los métodos desarrollados no puede ser limitada en su complejidad y no existe una forma de predecir la efectividad de la intervención. Con esta idea, fue desarrollada la clase Propagating Thread (Hilo propagador) usando a la clase Thread como base. En general, se busca unir al hilo principal de ejecución la llamada de una función, pero asignándole un tiempo máximo de ejecución previo a una excepción (Timeout exception), lo cual permite interrumpir la llamada y prevenir tiempos de espera excesivamente altos.

```
from threading import Thread
import time

class PropagatingThread(Thread):
    def run(self):
        self.exc = None
        try:
            if hasattr(self, '_Thread__target'):
                # Thread uses name mangling prior to Python 3.
                self.ret = self._Thread__target(*self._Thread__args,
**self._Thread__kwargs)
            else:
                self.ret = self._target(*self._args, **self._kwargs)
        except BaseException as e:
            self.exc = e

    def join(self, timeout=None):
        super(PropagatingThread, self).join(timeout)
        if self.exc:
            raise self.exc
        try:
            return self.ret
        except:
            raise Exception("Timeout exception")
```

# 12. Utilidades

---

## 12.1. Constantes

```
'''
# Limits
'''
MAX_GLOBAL_DIFFICULTY = 100000
MAX_NODE_DIFFICULTY = 5000
MAX_INTEGRAL_LEVEL = 20

'''
# Algebraics
'''
ALGEBRAIC_STEP_DIFFICULTY = 10
SOLVE_STEP_DIFFICULTY = 20
SIMPLIFY_STEP_DIFFICULTY = 20
EXPAND_STEP_DIFFICULTY = 20
FACTOR_STEP_DIFFICULTY = 20
SUBSTITUTION_STEP_DIFFICULTY = 20

'''
# Integrals
'''
INTEGRATION_MAX_DIFFICULTY = 50
ATOMIC_INTEGRAL_DIFFICULTY = 60
RECURSIVE_INTEGRAL_DIFFICULTY = 20
SYMPY_INTEGRAL = 70
```

## 12.2. Permisos de Firebase

```
{
  "type": "service_account",
  "project_id": "pwa-for-ode",
  "private_key_id": "6ab545387d77a4efd44376d499dbf615f2c8f9c2",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQBZdt552sGpLLs\ne8ym
66E6jHS6rObAmdVoiK8caQoohcgS8WNRoQLdty3/W2giJl5PS+zBmmrVPGUo\nyaNTUrMISk
dXV6ULKbvfJsb9CPntoBlNwB3s9JRPIff/6Cd7oFCR3AUM+QzUGoEK\nxjdOPkwVShaydXlZ
MCplVmcAAw4dEO5IIteEHFteJYUNUB+jR8dp7R/OImfAGlbo\nYaLvX3Bhkf6/y2o6Nmgm2+
ZSLrL6lKJ+Qio8nnpnGSr/qGGjHaUlmXi/49b98nHQ5\nZlVFPEPxS1kkUUbx5Sn/R2TE3xoO
/iZzPtJAvqGG+lo/gT91v5r2xGDbeKMDqHMM\nByXNa7Q1AgMBAAECggEAKb6yQklXM9ihE7
A87k197pMXh6g7kgvwwHIIqgl0HzNl\n27K0Dn6sk95joYfqX8S89r6ncjJVRbYbIhSAwi9j
kesR/93IlqUej8wV+6YF06nz\nnGVzC9WnfUkZCWZSDgK0Q/cITMm6o1T0aQ5xbiFaB4+oQBo
OpKt9lnd2tFRViGHyo\nnprXSBu4UmplbDdWlSdaQflz4tftQfwlWgT6vSMZ0W/QNEms8edw+
Gwk8v4uRGKqG\ng9IMej0Axy7PCsYnUx9stHy870QOdn173prDM/i6Fkkiw/REkq0sMfS6gz
haW+NJ\nnjLKIHVYULGd2BAcen0RhlcAkX/f28+peKQ1AWdvW9wKBgQDkhXS75600znsjMtfe
\nSGU7xwCiyRo8/qJREwmovsfiArMXXNTOINHB4WBn7Ng86KhITkfs0DVgwMDXhPrU\ncA1t
Cp9NzA/FHf3XMs8CSdD1oxPqJ1tonX/G6EHL6feHtltOCMNkY2dWyzqX3hDm\nnld3kPR3+oq
XV0plrVzn6/AlDFwKBgQDZG7TdORbKiTEvtZ5QI5FABk/pPW5B3vjV\nn+7IBZE+vejOveOH5
aPudAi/dzphzgn9/QfwIlBqbqZyea+5BQGntbMHSn+12qj4h\nnRbFTk4kCdy3UipoK2gSV9v
z02EcIJ5Z/IJJ4d4BgS8Xtq4Dkech1QbhoS4zwYt96\nnSqNK14wCkwKBgGcqxOesDNAE8dyT
MIvhrYpMjkHvj1PpFTiK4rp23uBvHZE3jK8+\na9QevppMd9cPpiF2jcBj1SVIAAeMdnz/1V
/cAslpidkQ0C+eNYyv7JFovS3h9MeH\nnC0+uPT4k82YmcpQJKsUgh1/Mwjgbvf+K/2ISIX96
yGl++VqFr4V60WejAoGAD60M\ndhy1JJKbVpKKh8bJqhHKm1CwMVlkJi8yPyb9afIs31IHHv
cRk9iglCdiopR9mDav\nd/8ZnLu2b0njvhpki5waspaRksvgCI77qskwT1lDmeEYFkh/pERd
J2CmYFRbbXpw\nnmPon7swJ0otblyOjeGYyZnVuC71X3YCeIb1GgrkCgYEAKTrWdrZZ9wOmtQ
u4oiS4\nn5k9VhezmaexT12Rt3Ft/B43PZMNaNaze3pdZOzkyRSaJip4fhUIfOwsfrgTi16i
\n03Sb8mPrgInyH0GavJTLmu3ieGtXSSRpPxheJ1M1r3+2y1JMqzxb6sZ9eqBHQzuV\nn44wz
QVQrO6fFWViEo3bCWc8=\n-----END PRIVATE KEY-----\n",
  "client_email": "firebase-adminsdk-7tbql@pwa-for-
ode.iam.gserviceaccount.com",
  "client_id": "101438182080892277501",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-
7tbql%40pwa-for-ode.iam.gserviceaccount.com"
}
```



# Capítulo V:

## Codificación del cliente

# 1. Archivos públicos

## 1.1. Punto de entrada

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="./images/iconb1024.png"/>

    <link rel="stylesheet" href="./bootstrap.css"/>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" integrity="sha512-
iBBXm8fw90+nuLcSK1bmrPcLa0T92x01BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHlhr35VZc
2oM/gI1w==" crossorigin="anonymous" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="description" content="PWA for Ordinary Differential Equations
Solving"/>
    <meta name="theme-color" content="#000000"/>
    <link rel="apple-touch-icon" href="./images/iconb1024.png"/>
    <link rel="manifest" href="./manifest.json"/>

    <title>PWA for ODE</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>

    <!-- Main Content -->
    <div id="root"></div>

    <!-- Service worker -->
    <script>
      if ("serviceWorker" in navigator) {
        window.addEventListener("load", () => {
          navigator.serviceWorker.register("./serviceworker.js")
            .then((reg) => console.log("Success: ", reg.scope))
            .catch((err) => console.log("Failure: ", err));
        });
      }
    </script>
  </body>
</html>

if 'x' in str(functionG):
  return False
```

## 1.2. Punto de entrada offline

```

<html>
<head>
  <style> ... </style>
  <title>PWA for ODE</title>
</head>
<body>
  <div class="jumbotron">
    <div class="container">
      <div class="d-flex justify-content-center">
        <div class="d-flex flex-column">
          <div class="p-2">
            <h1>App Offline</h1><br>
          </div>
          <div class="p-2">
            <svg viewBox="0 0 100 140"
xmlns="http://www.w3.org/2000/svg" width="256" height="256">
<ellipse rx="10.496939" ry="14.497783" cx="-55" cy="104" class="back"
fill="#751B19" stroke="black" transform="rotate(-45)" />
<path stroke="black" stroke-width="1" fill="#9B211F" class="body" d="M
28.312621 120.675565
                                A 20 44.7213 0 0 0 39.287710 67.972109
                                A 20 44.7213 0 0 1 68.578440 17.242373
                                A 20 44.7213 0 0 0 59.287710 67.972109
                                A 20 44.7213 0 0 1 42.084597 124.477772
                                A 10.496939 14.497783 -45 0 1 28.312621
120.675565"/>

                                <path stroke="black" stroke-width="1" fill="#C12825" class="belly"
                                d="M 57.733382 100.679356
                                    A 20 44.7213 0 0 0 40.866617 35.320643
                                    A 20 44.7213 0 0 0 57.733382 100.679356"/>
                                <ellipse rx="10.496939" ry="14.497783" cx="27" cy="61"
                                class="head" fill="#C12825" stroke="black"
                                transform="rotate(-
                                45)"/>
                                <ellipse rx="2" ry="5" cx="61" cy="22" class="eye" />
                                <path stroke-width="1" stroke="black" class="tongue"
                                d="M 70 36 A 10 10 0 0 0 74 39
                                    A 10 10 0 0 1 70 36
                                    A 10 10 0 0 1 69 43
                                    A 10 10 0 0 0 70 36"/>
                                </svg>
                              </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>

```

## 1.3. Registro de cache de PWA

```
const CACHE_NAME = "version-1"
const urlsToCache = [ "index.html", "offline.html" ];

const self = this;

// Install SW

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log("Opened cache");
        return cache.addAll(urlsToCache);
      })
  );
});

// Listen for request

self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request)
      .then(() => {
        return fetch(event.request)
          .catch(() => caches.match("offline.html"));
      })
  );
});

// Activate the SW

self.addEventListener("activate", (event) => {
  const cachWhitelist = [];
  cachWhitelist.push(CACHE_NAME);

  event.waitUntil(
    caches.keys().then((cacheNames) =>{
      // eslint-disable-next-line array-callback-return
      Promise.all(cacheNames.map((cacheName) => {
        if (!cachWhitelist.includes(cacheName)) {
          return caches.delete(cacheName);
        }
      })))
  );
});
```

## 2. Alertas

---

### 2.1. Dialogo Base

```

    <div class="form-check">
import React from 'react'
import './Popup.css'

function Popup(props) {
  return (props.trigger) ? (
    <div className="popup">
      <div className="popup-inner">
        <button className="close-btn btn btn-danger" onClick={() =>
props.setTrigger(false)}>
          close
        </button>
        { props.children }
      </div>
    </div>
  ) : "";
}

export default Popup

```

### 2.2. Dialogo de Espera

```

import React from "react";
import Popup from "../screens/global_components/Popup";

export class WaitingScreen {
  constructor(isWaiting, setIsWaiting) {
    this.isWaiting = isWaiting;
    this.setIsWaiting = setIsWaiting;
  }

  display = () => {
    return (
      <Popup trigger={ this.isWaiting } setTrigger={ this.setIsWaiting }>
        <h3 style={{ color:"black" }}> Solving Equation </h3>
        <p style={{ color:"black" }}> We're solving your equation </p>
      </Popup>
    );
  }
}

```

## 3. Anomalías

---

### 3.1. Dialogo de Anomalia de Clasificacion

```
import React from "react";
import Popup from "../screens/global_components/Popup";
export class ClassificationAnomaly {
  constructor (message, alertPopup, setAlertPopup) {
    this.message = message;
    this.alertPopup = alertPopup;
    this.setAlertPopup = setAlertPopup;
  }
  display = () => {
    return (
      <Popup trigger={ this.alertPopup } setTrigger={ this.setAlertPopup }>
        <h3 style={{ color:"black" }}> Classification Error </h3>
        <p style={{ color:"black" }}> { this.message } </p>
      </Popup>
    )
  }
};
```

### 3.2. Dialogo de Anomalia de Completitud

```
import React from "react";
import Popup from "../screens/global_components/Popup";
export class CompletenessAnomaly {
  constructor (message, alertPopup, setAlertPopup) {
    this.message = message;
    this.alertPopup = alertPopup;
    this.setAlertPopup = setAlertPopup;
  }
  display = () => {
    return (
      <Popup trigger={ this.alertPopup } setTrigger={ this.setAlertPopup }>
        <h3 style={{ color:"black" }}> Completeness Error </h3>
        <p style={{ color:"black" }}> { this.message } </p>
      </Popup>
    )
  }
};
```

### 3.3. Dialogo de Anomalia Generica

```
import React from "react";
import Popup from "../screens/global_components/Popup";
export class GenericAnomaly {
  constructor (message, alertPopup, setAlertPopup) {
    this.message = message;
    this.alertPopup = alertPopup;
    this.setAlertPopup = setAlertPopup;
  }
  display = () => {
    return (
      <Popup trigger={ this.alertPopup } setTrigger={ this.setAlertPopup }>
        <h3 style={{ color:"black" }}> Unexpected Error </h3>
        <p style={{ color:"black" }}> { this.message } </p>
      </Popup>
    )
  }
};
```

### 3.4. Dialogo de Anomalia de Interpretacion

```
import React from "react";
import Popup from "../screens/global_components/Popup";
export class InterpretationAnomaly {
  constructor (message, alertPopup, setAlertPopup) {
    this.message = message;
    this.alertPopup = alertPopup;
    this.setAlertPopup = setAlertPopup;
  }
  display = () => {
    return (
      <Popup trigger={ this.alertPopup } setTrigger={ this.setAlertPopup }>
        <h3 style={{ color:"black" }}> Interpretation Error </h3>
        <p style={{ color:"black" }}> { this.message } </p>
      </Popup>
    )
  }
};
```

## 3.5. Dialogo de Anomalia de Tiempo

```
import React from "react";
import Popup from "../screens/global_components/Popup";

export class SolutionTimeoutAnomaly {
  constructor (alertPopup, setAlertPopup) {
    this.alertPopup = alertPopup;
    this.setAlertPopup = setAlertPopup;
  }

  display = () => {
    return (
      <Popup trigger={ this.alertPopup } setTrigger={
this.setAlertPopup }>
        <h3 style={{ color:"black" }}> Timeout Error </h3>
        <p style={{ color:"black" }}> There was no answer to the
equation in 30 seconds </p>
      </Popup>
    )
  }
};
```



## 4. Pantallas principales

---

### 4.1. Pantalla de Inicio de sesion

```
import React, { useState, useCallback, useContext } from "react";
import { withRouter, Redirect } from "react-router";
import app from "../base";
import { AuthContext } from "../auth";
import Popup from "../global_components/Popup";
import "../Login.css";

const Login = ( {history} ) => {

  const [alertPopup, setAlertPopup] = useState(false);
  const [ currentError, setCurrentError ] = useState(null);

  const handleLogin = useCallback(
    async event => {
      event.preventDefault();
      const { email, password } = event.target.elements;
      try {
        await app.auth().signInWithEmailAndPassword(email.value,
password.value);
        history.push("/");
      } catch (error) {
        setAlertPopup(true)
        setCurrentError(error.message)
      }
    },
    [history]
  );
```

```

return (
  <div className="jumbotron">
    <form onSubmit={handleLogin}>
      <fieldset>
        <legend>Log In</legend>
        <div className="form-group">
          <label>Email address</label>
          <input name="email" type="email" className="form-
control" id="exampleInputEmail1" aria-describedby="emailHelp"
placeholder="Enter email"/>
          <small id="emailHelp" className="form-text text-
muted">We'll never share your email with anyone else.</small>
        </div>
        <div className="form-group">
          <label>Password</label>
          <input name="password" type="password" className="form-
control" id="exampleInputPassword1" placeholder="Password"/>
        </div>
        <div className="btn-group-vertical">
          <button type="submit" className="btn btn-
primary">Submit</button><br></div>
          <button onClick={() => history.push("/signup")}
type="button" className="btn btn-secondary">Sign Up</button>
        </div>
      </fieldset>
    </form>
    <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
      <h3 style={{ color:"black" }}> Error </h3>
      <p style={{ color:"black" }}> { currentError } </p>
    </Popup>
  </div>
);
};

export default withRouter(Login);

```

## 4.2. Pantalla de Crear cuenta

```
import React, { useState, useCallback } from "react";
import { withRouter } from "react-router";
import app from "../base";
import Popup from "../global_components/Popup";

const SignUp = ({ history }) => {

  const [alertPopup, setAlertPopup] = useState(false);
  const [ currentError, setCurrentError ] = useState(null);

  const handleSignUp = useCallback(
    async event => {
      event.preventDefault();
      const { email, password, confirmPassword, userType, isSubscribed }
= event.target.elements;

      const addUser = async (userType, isSubscribed, uid) => {
        const db = app.firestore();
        var isTrueSet = isSubscribed.checked;

        await db.collection('users').doc(uid).set({
          email: email.value,
          isSubscribed: isTrueSet,
          type: userType.value,
        });
      }

      function validatePassword(newPassword) {
        var minNumberOfChars = 6;
        var maxNumberOfChars = 16;
        var regularExpression = /^[a-zA-Z0-9]{6,16}$/;

        if (newPassword.length < minNumberOfChars ||
newPassword.length > maxNumberOfChars){
          setAlertPopup(true)
          setCurrentError("Bad lenght")
          return false;
        }
        if (!regularExpression.test(newPassword)) {
          setAlertPopup(true)
          setCurrentError("Bad chars")
          return false;
        }
        return true;
      }

      if (!validatePassword(password.value)) {
        setAlertPopup(true)
        setCurrentError("Invalid password")
        return;
      }
    }
  );
}
```

```

        if (password.value !== confirmPassword.value) {
            showAlertPopup(true)
            setCurrentError("Passwords dont match")
            return;
        }
        else {
            try {
                const newUser = await
app.auth().createUserWithEmailAndPassword(email.value, password.value);
                await addUser(userType, isSubscribed, newUser.user.uid);
                history.push("/");
            } catch (error) {
                showAlertPopup(true)
                setCurrentError(error.message)
            }
        }
    }, [history]
);
return (
    <div className="jumbotron">
        <form onSubmit={handleSignUp}>
            <fieldset>
                <legend>Sign Up</legend>
                <div className="form-group">
                    <label>Email address</label>
                    <input name="email" type="email" className="form-
control" id="exampleInputEmail1" aria-describedby="emailHelp"
placeholder="Enter email" />
                    <small id="emailHelp" className="form-text text-
muted">We'll never share your email with anyone else.</small>
                </div>
                <div className="form-group">
                    <label>Password</label>
                    <input name="password" type="password"
className="form-control" id="exampleInputPassword1" placeholder="Password" />
                </div>
                <div className="form-group">
                    <label>Confirm Password</label>
                    <input name="confirmPassword" type="password"
className="form-control" id="exampleInputPassword1" placeholder="Password" />
                </div>
                <div class="form-group">
                    <label>User type</label>
                    <select name="userType" class="custom-select d-block
w-100" id="inputGroupSelect" required>
                        <option value="student">student</option>
                        <option value="teacher">teacher</option>
                    </select>
                </div>
            </fieldset>
        </form>
    </div>

```

```

    <div class="form-check">
      <input name="isSubscribed" class="form-check-input"
type="checkbox" id="defaultCheck1"></input>
      <label>I want to receive updates in my email</label>
    </div>
    <br><br>
    <div className="btn-group-vertical">
      <button type="submit" className="btn btn-
primary">Submit</button><br><br>
      <button onClick={() => history.push("/login")}
type="button" className="btn btn-secondary">Log In</button>
    </div>
  </fieldset>
</form>
  <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
    <h3 style={{ color:"black" }}> Error </h3>
    <p style={{ color:"black" }}> { currentError } </p>
  </Popup>
</div>
  );
};

export default withRouter(SignUp);

```

## 4.3. Componente de Barra de navegacion

```
import React from "react";
import app from "../../base";
import "./Navbar.css";
import { useState } from "react";
const Navbar = () => {
  const [ toggle, setToggle ] = useState(true);
  const toggleState = () => {
    setToggle(!toggle);
    const doc = document.getElementById("navbarElements");
    if (toggle) {
      doc.classList.remove("collapse");
    } else {
      doc.classList.add("collapse");
    }
  }
  return (
    <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
      <button className="navbar-toggler collapsed" type="button"
onClick={toggleState}>
        <span className="navbar-toggler-icon"></span>
      </button>
      <div className="collapse navbar-collapse" id="navbarElements">
        <ul className="navbar-nav mr-auto">
          <li className="nav-item">
            <a className="nav-link" href="/">Home
              <span className="sr-only">(current)</span> </a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/draw">Draw</a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/keyboard">Keyboard</a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/picture">Picture</a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/record">Record</a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/settings">Settings</a>
          </li>
          <li className="nav-item">
            <a className="nav-link" href="/about">About</a>
          </li>
        </ul>
        <button onClick={() => app.auth().signOut()} className="btn btn-danger
my-2 my-lg-0">Sign Out </button>
      </div>
    </nav>);
}
export default Navbar;
```

## 4.4. Pantalla de Bienvenida

```
import React from "react";
import "../Home.css"

import Navbar from "../menu_components/Navbar";

const Home = () => {
  return (
    <div>
      <Navbar />
      <div className="jumbotron">
        <h1 className="display-3">Welcome to PWA for ODE!</h1>
        <p className="lead">Solve your differential equations
with us and get amazing presentations of your solutions</p>
        <hr className="my-4"/>
        <p>Get the steps to solve higher order linear,
homogeneous, exact, and separable ordinary differential equations.</p>
        <p className="lead">
          <a className="btn btn-primary btn-lg"
href="/keyboard" role="button">Let's solve it!</a>
        </p>
      </div>
    </div>
  );
}

export default Home;
```

## 4.5. Componente de Ecuacion

```
import React, { useContext, useState } from "react";
import { Link } from 'react-router-dom';
import { XIcon, PinIcon } from '@primer/octicons-react'
import app from "../../../base"
import "../Ecuacion.css"
import { AuthContext } from "../../../auth";
import Popup from "../../global_components/Popup";

const MathJax = require('react-mathjax')

const Ecuacion = ({ item }) => {
  const [ alertPopup, setAlertPopup ] = useState(false);
  const [ currentTitle, setCurrentTitle ] = useState(null);
  const [ currentError, setCurrentError ] = useState(null);
  const eq = item.equation;
  const date = item.date
  const { currentUser } = useContext(AuthContext);
  const [pinned, setPinned] = useState(item.pinned)
  const onDelete = async () => {
    setAlertPopup(true)
    setCurrentTitle("Success")
    setCurrentError("Equation eliminated successfully")
    const db = app.firestore()
    await
    db.collection('users').doc(currentUser.uid).collection("equations").doc(
    item.id).delete()
    window.location.reload(false);
  }
  const onUpdate = async () => {
    const db = app.firestore()
    var newPinned = !pinned
    setPinned(newPinned)
    await
    db.collection('users').doc(currentUser.uid).collection("equations").doc(
    item.id).set({
      date: item.date,
      equation: item.equation,
      pinned: newPinned
    })
  }
  const renderColor = () => {
    if(pinned){
      return "#5d5d5a"
    }
    else{
      return "#303030"
    }
  }
}
```



```

return (
  <Link to={`/preview/${eq}`} class="list-group-item list-group-item-
action flex-column align-items-start " style={{background:
renderColor()}}>
    <div class="d-flex w-100 justify-content-between">
      <MathJax.default.Provider>
        <MathJax.default.Node inline formula={eq} />
      </MathJax.default.Provider>
      <small>
        <Link to={`/record`}>
          <button class="btn bg-transparent" onClick={onDelete}>
            <XIcon size={24} />
          </button>
        </Link>
        <Link to={`/record`}>
          <button class="btn bg-transparent" onClick={onUpdate}>
            <PinIcon size={24} />
          </button>
        </Link>
      </small>
    </div>
    <p class="mb-1">{date}</p>
    <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
      <h3 style={{ color:"black" }}> { currentTitle } </h3>
      <p style={{ color:"black" }}> { currentError } </p>
    </Popup>
  </Link>
)
}

export default Ecuation;

```

## 4.6. Pantalla de Historial

```
import React, { useState, useEffect, useContext } from "react";
import app from "../../base";
import { AuthContext } from "../../auth"
import "../Record.css"

import Navbar from "../menu_components/Navbar";
import Ecuation from "../menu_screens/record_components/Ecuation";

const Record = () => {
  const [equations, setEquations] = useState([]);
  const { currentUser } = useContext(AuthContext);

  useEffect(() => {
    const fetchData = async () => {
      const db = app.firestore()

      db.collection("users").doc(currentUser.uid).collection("equations").onSnapshot(
        (querySnapshot) => {
          const docs = []
          querySnapshot.forEach((doc) => {
            docs.push({ ...doc.data(), id: doc.id });
          })
          setEquations(docs)
        }
      )
      fetchData()
    }, [currentUser.uid]);
  });
}
```

```

const renderRecord = () => {
  const db = app.firestore()

  if (equations.length === 0) {
    return (
      <>
      <p className="lead">You haven't solved any differential
equations yet. Try some to generate a history</p>
      </>
    );
  } else {
    var equationsPinnedSorted = []
    var equationsNoPinnedSorted = []
    var equationsSorted = []

    for (var i = 0; i < equations.length; i++) {
      if (equations[i].pinned) {
        equationsPinnedSorted.push(equations[i]);
      } else {
        equationsNoPinnedSorted.push(equations[i]);
      }
    }

    equationsPinnedSorted = sortByDate(equationsPinnedSorted);
    equationsNoPinnedSorted = sortByDate(equationsNoPinnedSorted);

    for (i = 0; i < equationsPinnedSorted.length; i++) {
      equationsSorted.push(equationsPinnedSorted[i]);
    }

    for (i = 0; i < equationsNoPinnedSorted.length; i++) {
      equationsSorted.push(equationsNoPinnedSorted[i]);
    }

    const today = new Date();
    for (i = equationsSorted.length - 1; i >= 0; i--) {
      // remove out date from database
      var checkDate = Date.parse(equationsSorted[i].date)

      // 60_000 = 1 minute
      // 3_600_000 = 1 hour
      // 86_400_000 = 1 day
      // 2_592_000_000 = 30 days
      if (((today - checkDate) > 2_592_000_000) &&
!equationsSorted[i].pinned) {

db.collection('users').doc(currentUser.uid).collection('equations').doc(equati
onsSorted[i].id).delete()
      equationsSorted.splice(i, 1)
    }
  }
}

```

```

const MAX_ELEMENTS = 30;

    try {
      if (equationsSorted.length > MAX_ELEMENTS) {
        // remove extras from database
        for (var j = MAX_ELEMENTS; j < equationsSorted.length;
j++) {

db.collection('users').doc(currentUser.uid).collection("equations").doc(equati
onsSorted[j].id).delete()
        }

        while (equationsSorted.length > MAX_ELEMENTS) {
          equationsSorted.pop();
        }
      } catch (exception) {

    }

    return equationsSorted.map(item => (
      <Equation item={ item } />
    ));
  }
}

const sortByDate = (array) => {
  var di, dj, aux;
  for (var i = 0; i < array.length; i++) {
    for (var j = i+1; j < array.length; j++) {
      di = Date.parse(array[i].date);
      dj = Date.parse(array[j].date);
      if (di < dj) {
        aux = array[i];
        array[i] = array[j];
        array[j] = aux;
      }
    }
  }
  return array;
}

return (
  <div>
    <Navbar />
    <div className="jumbotron">
      <h1>Record</h1><br></br>
      <div class="list-group">
        { renderRecord() }
      </div>
    </div>
  </div>
);
}

```

## 4.7. Pantalla de Configuración

```
import firebase from "firebase/app";
import React, { useEffect, useCallbck, useContext, useState } from "react";
import app from "../../base";
import "../Settings.css";
import { AuthContext } from "../../auth";
import Navbar from "../../menu_components/Navbar";
import Popup from "../../global_components/Popup";

const Settings = ({ item }) => {

  const [alertPopup, setAlertPopup] = useState(false);
  const [currentError, setCurrentError] = useState(null);
  const [currentTitle, setCurrentTitle] = useState(null);

  const { currentUser } = useContext(AuthContext);
  const [currentUserType, setCurrentUserType] = useState(null);
  const [currentSubscribed, setCurrentSubscribed] = useState(null);

  const uid = currentUser.uid;

  useEffect(() => {
    const fetchData = async () => {
      const db = app.firestore()
      db.collection("users").onSnapshot((querySnapshot) => {
        const docs = []
        querySnapshot.forEach((doc) => {
          if (doc.id === currentUser.uid) {
            docs.push({ ...doc.data(), id: doc.id });
          }
        });
      });

      try {
        setCurrentUserType(docs[0].type);
        setCurrentSubscribed(docs[0].isSubscribed);
      } catch (someError) {
        setAlertPopup(true)
        setCurrentTitle("Error")
        setCurrentError(someError.message)
      }
    }
    fetchData()
  }, [currentUser.uid]);
}
```

```

const renderCurrentValues = () => {
  if (currentUserType === "teacher" && currentSubscribed) {
    return (
      <>
      <div class="form-group">
        <label>User type</label>
        <select name="userType" class="custom-select d-block
w-100" id="inputGroupSelect" required>
          <option value="student">student</option>
          <option value="teacher" selected>teacher</option>
        </select>
      </div>
      <div class="form-check">
        <input name="isSubscribed" class="form-check-input"
type="checkbox" id="defaultCheck1" defaultChecked></input>
        <label>I want to receive updates in my email</label>
      </div>
      </>
    );
  } else if (currentUserType === "teacher" && !currentSubscribed) {
    return (
      <>
      <div class="form-group">
        <label>User type</label>
        <select name="userType" class="custom-select d-block
w-100" id="inputGroupSelect" required>
          <option value="student">student</option>
          <option value="teacher" selected>teacher</option>
        </select>
      </div>
      <div class="form-check">
        <input name="isSubscribed" class="form-check-input"
type="checkbox" id="defaultCheck1"></input>
        <label>I want to receive updates in my email</label>
      </div>
      </>
    );
  } else if (currentUserType === "student" && currentSubscribed) {
    return (
      <>
      <div class="form-group">
        <label>User type</label>
        <select name="userType" class="custom-select d-block
w-100" id="inputGroupSelect" required>
          <option value="student" selected>student</option>
          <option value="teacher">teacher</option>
        </select>
      </div>
      <div class="form-check">
        <input name="isSubscribed" class="form-check-input"
type="checkbox" id="defaultCheck1" defaultChecked></input>
        <label>I want to receive updates in my email</label>
      </div>
      </>
    );
  } else {

```

```

return (
    <>
    <div class="form-group">
        <label>User type</label>
        <select name="userType" class="custom-select d-block w-100" id="inputGroupSelect" required>
            <option value="student" selected>student</option>
            <option value="teacher">teacher</option>
        </select>
    </div>
    <div class="form-check">
        <input name="isSubscribed" class="form-check-input" type="checkbox" id="defaultCheck1"></input>
        <label>I want to receive updates in my email</label>
    </div>
    </>
);
}
}

const handleSettings = useCallback(
    async event => {
        event.preventDefault();
        const { password, newPassword, confirmNewPassword, userType, isSubscribed } = event.target.elements;

        var credentials = firebase.auth.EmailAuthProvider.credential(
            currentUser.email,
            password.value
        );
    }
);

```

```

try {
    await currentUser.reauthenticateWithCredential(credentials);
  } catch (passError) {
    showAlertPopup(true)
    setCurrentTitle("Error")
    setCurrentError("Password is incorrect")
    return;
  }

  if (password.value === newPassword.value) {
    showAlertPopup(true)
    setCurrentTitle("Error")
    setCurrentError("New password can not be the current
password")
    return;
  }

  function validatePassword(newPassword) {
    var minNumberOfChars = 6;
    var maxNumberOfChars = 16;
    var regularExpression = /^[a-zA-Z0-9]{6,16}$/;

    if (newPassword.length < minNumberOfChars ||
newPassword.length > maxNumberOfChars) {
      showAlertPopup(true)
      setCurrentTitle("Error")
      setCurrentError("Bad length")
      return false;
    }

    if (!regularExpression.test(newPassword)) {
      showAlertPopup(true)
      setCurrentTitle("Error")
      setCurrentError("Bad chars")
      return false;
    }

    return true;
  }
}

```



```

    if (newPassword.value !== "" && !validatePassword(newPassword.value)) {
      showAlertPopup(true)
      setCurrentTitle("Error")
      setCurrentError("Invalid new password")
      return;
    }

    if (newPassword.value !== "" && newPassword.value !==
confirmNewPassword.value) {
      showAlertPopup(true)
      setCurrentTitle("Error")
      setCurrentError("New Passwords dont match")
      return;
    }

    const updateUser = async (userTypeStr, isSubscribed) => {
      const db = app.firestore();
      var isTrueSet = isSubscribed.checked;

      await db.collection('users').doc(uid).set({
        email: currentUser.email,
        isSubscribed: isTrueSet,
        type: userTypeStr,
      });

      if (newPassword.value !== "") {
        await currentUser.updatePassword(newPassword.value);
      }
    }

    await updateUser(userType.value, isSubscribed);
    password.value = "";
    newPassword.value = "";
    confirmNewPassword.value = "";
    showAlertPopup(true)
    setCurrentTitle("Success")
    setCurrentError("The changes were successful")

  }, [item]
);

```

```

return (
  <div>
    <Navbar />
    <div className="jumbotron">
      <h1>Settings</h1>
      <form onSubmit={handleSettings}>
        <fieldset>
          <div class="form-group">
            <label>Actual Password</label>
            <input name="password" type="password"
class="form-control" id="exampleInputPassword1"
placeholder="Password"></input>
          </div>
          <div class="form-group">
            <label>New Password (leave empty to not
change)</label>
            <input name="newPassword" type="password"
class="form-control" id="exampleInputPassword1"
placeholder="Password"></input>
          </div>
          <div class="form-group">
            <label>Confirm New Password (leave empty to not
change)</label>
            <input name="confirmNewPassword" type="password"
class="form-control" id="exampleInputPassword1"
placeholder="Password"></input>
          </div>

          { renderCurrentValues() }

          <div className="btn-group-vertical">
            <button type="submit" className="btn btn-
primary">Submit</button><br></div>
          </fieldset>
        </form>
      </div>
      <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
        <h3 style={{ color:"black" }}> { currentTitle } </h3>
        <p style={{ color:"black" }}> { currentError } </p>
      </Popup>
    </div>
  </div>
);
}

export default Settings;

```

## 5. Pantallas de entrada

### 5.1. Componente de Teclado

```
import React, { useState } from 'react';
import { InterpretationAnomaly } from
'../../../../../anomalies/InterpretationAnomaly';
import { IP_SERVER } from '../../../../../global/consts';
const request = require("request-promise");
const MathKeyboard = ({ history }) => {
  const [alertPopup, setAlertPopup] = useState(false);
  const [ currentError, setCurrentError ] = useState(null);
  const writeChar = (char) => {
    return () => {
      const equationBox = document.getElementById("equationBox");
      var index = equationBox.value.indexOf("|");
      var left = equationBox.value.substring(0, index);
      var right = equationBox.value.substring(index,
equationBox.value.length);
      equationBox.value = left + char + right;
    }
  }
  const clear = () => {
    const equationBox = document.getElementById("equationBox");
    equationBox.value = "|";
  }
  const remove = () => {
    const equationBox = document.getElementById("equationBox");
    var index = equationBox.value.indexOf("|");
    if (index > 0) {
      var left = equationBox.value.substring(0, index-1);
      var right = equationBox.value.substring(index,
equationBox.value.length);
      equationBox.value = left + right;
    }
  }
  const moveLeft = () => {
    const equationBox = document.getElementById("equationBox");

    var index = equationBox.value.indexOf("|");
    if (index > 0) {
      var left = equationBox.value.substring(0, index - 1);
      var right = equationBox.value.substring(index - 1,
equationBox.value.length).replace("|", "");
      equationBox.value = left + "|" + right;
    }
  }
}
```

```

const moveRight = () => {
  const equationBox = document.getElementById("equationBox");

  var index = equationBox.value.indexOf("|");
  if (index < equationBox.value.length-1) {
    var left = equationBox.value.substring(0, index + 2).replace("|",
    "");
    var right = equationBox.value.substring(index + 2);
    equationBox.value = left + "|" + right;
  }
}

const handleSubmit = async () => {
  const equationBox = document.getElementById("equationBox");
  const equation = equationBox.value.replace("|", "");

  const options = {
    method: "POST",
    uri: `https://${IP_SERVER}/parse/latex`,
    body: { equation: equation },
    json: true
  };

  console.log(options);

  const res = await request(options);
  if (res.status !== 'ok') {
    showAlertPopup(true)
    setCurrentError(res.status)
  } else {
    history.push(`/preview/${res.equation}`);
  }
}

```

```

const keydownCallback = e => {
  const keyName = e.key;
  switch(keyName) {
    case "0" : writeChar("0")(); break;
    case "1" : writeChar("1")(); break;
    case "2" : writeChar("2")(); break;
    case "3" : writeChar("3")(); break;
    case "4" : writeChar("4")(); break;
    case "5" : writeChar("5")(); break;
    case "6" : writeChar("6")(); break;
    case "7" : writeChar("7")(); break;
    case "8" : writeChar("8")(); break;
    case "9" : writeChar("9")(); break;

    case "(" : writeChar("(")(); break;
    case ")" : writeChar(")")(); break;
    case "," : writeChar(",")(); break;
    case "." : writeChar(".")(); break;
    case "D" : writeChar("D")(); break;
    case "Backspace" : remove(); break;
    case "^" : writeChar("^")(); break;
    case "+" : writeChar("+")(); break;
    case "-" : writeChar("-")(); break;
    case "*" : writeChar("*")(); break;
    case "/" : writeChar("/")(); break;
    case "=" : writeChar("=")(); break;

    case "x" : writeChar("x")(); break;
    case "y" : writeChar("y")(); break;
    case "E" : writeChar("E")(); break;
    case "p" : writeChar("pi")(); break;

    case "s" : writeChar("sin")(); break;
    case "S" : writeChar("sec")(); break;
    case "c" : writeChar("cos")(); break;
    case "C" : writeChar("csc")(); break;
    case "t" : writeChar("tan")(); break;
    case "T" : writeChar("cot")(); break;
    case "l" : writeChar("ln")(); break;
    case "r" : writeChar("sqrt")(); break;

    case "ArrowLeft" : moveLeft(); break;
    case "ArrowRight" : moveRight(); break;
    case "Delete" : clear(); break;
    case "Enter" : handleSubmit(); break;

    default: break;
  }
}

```

```

    return (
      <div onKeyDown={ keydownCallback }>
        <input defaultValue="|" disabled type="text" className="form-
control" placeholder="Equation" name="equationBox" id="equationBox"
        onClick = { () => {
document.getElementById("equationBox").addEventListener("keydown",
keydownCallback) } }
        /><br/>
        <div className="row">
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-primary btn-lg
btn-block" onClick={ clear }>Clear</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar("7") }>7</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar("8") }>8</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar("9") }>9</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg btn-
block" onClick={ writeChar("sin") }>sin</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-info btn-lg btn-
block" onClick={ writeChar("+") }>+</button>
          </div>
        </div><br/>
        <div className="row">
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-primary btn-lg
btn-block" onClick={ remove }>Remove</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar("4") }>4</button>
          </div>
          <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar("5") }>5</button>
          </div>
        </div>
      </div>
    )
  
```

```

        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg btn-block"
onClick={ writeChar("6") }>6</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg btn-block"
onClick={ writeChar("cos") }>cos</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-info btn-lg btn-block"
onClick={ writeChar("-") }>-</button>
        </div>
    </div><br/>
    </div><br/>
    <div className="row">
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-primary btn-lg btn-block"
onClick={ moveLeft }>
                <i className="fa fa-arrow-left"></i>
            </button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg btn-block"
onClick={ writeChar("1") }>1</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg btn-block"
onClick={ writeChar("2") }>2</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg btn-block"
onClick={ writeChar("3") }>3</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg btn-block"
onClick={ writeChar("tan") }>tan</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-info btn-lg btn-block"
onClick={ writeChar("*") }>*</button>
        </div>
    </div><br/>
    <div className="row">
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-primary btn-lg btn-block"
onClick={ moveRight }>
                <i className="fa fa-arrow-right"></i>
            </button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg btn-block"
onClick={ writeChar("0") }>0</button>
        </div>
    </div>

```

```

        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-danger btn-lg
btn-block" onClick={ writeChar(".") }>.</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-secondary btn-lg
btn-block" onClick={ writeChar("D") }>D</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg
btn-block" onClick={ writeChar("sec") }>sec</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-info btn-lg btn-
block" onClick={ writeChar("/") }>/</button>
        </div>
    </div><br/>
    <div className="row">
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-warning btn-lg
btn-block" onClick={ writeChar(",") }>,</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-success btn-lg
btn-block" onClick={ writeChar("x") }>x</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-success btn-lg
btn-block" onClick={ writeChar("E") }>E</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-secondary btn-lg
btn-block" onClick={ writeChar("ln") }>ln</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg
btn-block" onClick={ writeChar("csc") }>csc</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-info btn-lg btn-
block" onClick={ writeChar("^") }>^</button>
        </div>
    </div><br/>
    <div className="row">
        <div className="col-6 col-sm-1">
            <button type="button" className="btn btn-warning btn-lg
btn-block" onClick={ writeChar("(") }>(</button>
        </div>
        <div className="col-6 col-sm-1">
            <button type="button" className="btn btn-warning btn-lg
btn-block" onClick={ writeChar(")") }>)</button>
        </div>
    </div>

```



```

        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-success btn-lg
btn-block" onClick={ writeChar("y") }>y</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-success btn-lg
btn-block" onClick={ writeChar("pi") }>pi</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-secondary btn-lg
btn-block" onClick={ writeChar("sqrt") }>sqrt</button>
        </div>
        <div className="col-6 col-sm-2">
            <button type="button" className="btn btn-light btn-lg btn-
block" onClick={ writeChar("cot") }>cot</button>
        </div>
        <div className="col-6 col-sm-1">
            <button type="button" className="btn btn-info btn-lg btn-
block" onClick={ writeChar("=") }>=</button>
        </div>
        <div className="col-6 col-sm-1">
            <button type="button" className="btn btn-primary btn-lg
btn-block" onClick={ handleSubmit }>
                <i className="fa fa-check"></i>
            </button>
        </div>
    </div><br/>
    { new InterpretationAnomaly(currentError, alertPopup,
setAlertPopup).display() }
</div>
    );
}

export default MathKeyboard;

```

## 5.2. Componente de Canvas

```
import React, { useEffect, useState, useRef, useCallback } from "react";
import { IP_SERVER } from "../../global/consts";
import Popup from "../../global_components/Popup";

const request = require("request-promise");

const Canvas = ({ history }) => {

  const [alertPopup, setAlertPopup] = useState(false);
  const [currentError, setCurrentError] = useState(null);

  async function postData(url = '', data = {}) {
    const response = await fetch(url, {
      method: 'POST',
      mode: 'cors',
      cache: 'no-cache',
      credentials: 'same-origin',
      headers: {
        'Content-Type': 'application/json'
      },
      redirect: 'follow',
      referrerPolicy: 'no-referrer',
      body: JSON.stringify(data)
    });
    return response.json();
  }

  const [isDrawing, setIsDrawing] = useState(false)
  const [ canvasHistory, setCanvasHistory ] = useState([new
ImageData(window.innerWidth, window.innerHeight)]);

  const canvasRef = useRef(null);
  const contextRef = useRef(null);

  const recursiveOffsetLeftTop = (element) => {
    var offsetLeft = 0;
    var offsetTop = 0;
    while (element) {
      offsetLeft = element.offsetLeft;
      offsetTop = element.offsetTop;
      element = element.offsetParent;
    }

    return {
      offsetLeft,
      offsetTop
    };
  }
}
```

```

useEffect(() => {
  const canvas = canvasRef.current;
  canvas.width = window.innerWidth;
  canvas.height = window.innerHeight;
  canvas.style.width = `${window.innerWidth/2}px`;
  canvas.style.height = `${window.innerHeight/2}px`;
  canvas.style.background = "white";

  const context = canvas.getContext("2d")

  context.scale(2, 2);
  context.lineCap = "round";
  context.strokeStyle = "black";
  context.lineWidth = 2;
  contextRef.current = context;
}, []);

const startDrawing = ({ nativeEvent }) => {
  if (isDrawing) {
    setIsDrawing(false);
    return;
  }

  const { offsetX, offsetY } = nativeEvent;
  contextRef.current.beginPath();
  contextRef.current.moveTo(offsetX, offsetY);
  setIsDrawing(true);
};

const startDrawingWithTouch = ({ nativeEvent }) => {
  if (isDrawing) {
    setIsDrawing(false);
    return;
  }

  const { clientX, clientY } = nativeEvent.touches[0];

  const canvas = canvasRef.current;
  const { offsetLeft, offsetTop } = recursiveOffsetLeftTop(canvas);

  contextRef.current.beginPath();
  contextRef.current.moveTo(clientX - offsetLeft, clientY - offsetTop);
  setIsDrawing(true);
}

```

```

const finishDrawing = () => {
  contextRef.current.closePath();
  const canvas = canvasRef.current;
  const context = canvas.getContext("2d")

  var data = context.getImageData(0, 0, canvas.width, canvas.height);
  canvasHistory.push(data);
  console.log(canvasHistory.length);

  setIsDrawing(false);
};

const draw = ({ nativeEvent }) => {
  if (!isDrawing) {
    return;
  }
  const { offsetX, offsetY } = nativeEvent;
  contextRef.current.lineTo(offsetX, offsetY);
  contextRef.current.stroke();
};

const drawWithTouch = ({ nativeEvent }) => {
  if (!isDrawing) {
    return;
  }

  const { clientX, clientY } = nativeEvent.touches[0];
  const canvas = canvasRef.current;
  const { offsetLeft, offsetTop } = recursiveOffsetLeftTop(canvas);

  contextRef.current.lineTo(clientX - offsetLeft, clientY - offsetTop);
  contextRef.current.stroke();
}

const handleUndo = () => {
  if (canvasHistory.length === 1) {
    return;
  }

  var prevState = canvasHistory[canvasHistory.length - 2];
  canvasHistory.pop();

  const canvas = canvasRef.current;
  const context = canvas.getContext("2d");
  context.putImageData(prevState, 0, 0);
}

```

```

const handleDelete = () => {
  setCanvasHistory([new ImageData(window.innerWidth,window.innerHeight)]);
  const canvas = canvasRef.current;
  const context = canvas.getContext("2d");
  context.putImageData(canvasHistory[0], 0, 0);
}
const handleSend = useCallback(() => {
  var canvas = document.getElementById('canvas');
  var imgData = canvas.toDataURL();
  imgData = imgData.split("data:image/png;base64,")[1]
  const url = `https://${IP_SERVER}/image/text`;
  postData(url, { image: imgData }).then(async data => {
    console.log(data);
    if (data.status === 'ok') {const requestPreview = async () => {
      const options = {
        method: "POST",
        uri: `https://${IP_SERVER}/parse/latex`,
        body: { equation: data.text },
        json: true};
      const res = await request(options);
      if (res.status !== 'ok') {
        showAlertPopup(true)
        setCurrentError(res.status)
      } else { history.push(`/preview/${res.equation}`);
      }}
      await requestPreview();}});}, [history]);

return (
  <>
    <canvas
      onMouseDown={startDrawing}
      onTouchStart={startDrawingWithTouch}
      onMouseUp={finishDrawing}
      onTouchEnd={finishDrawing}
      onMouseMove={draw}
      onTouchMove={drawWithTouch}
      ref={canvasRef}
      id="canvas"
    /><br><br><br><br>
    <div class="row justify-content-start">
      <div class="col-6 col-sm-1">
        <button className="btn btn-primary" onClick={ handleUndo }>Undo</button> </div>
        <div class="col-6 col-sm-1">
          <button className="btn btn-danger" onClick={ handleDelete }>Clear</button> </div>
          <div class="col-6 col-sm-1">
            <button className="btn btn-success" onClick={ handleSend }>Send</button></div>
        </div>
        <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
          <h3 style={{ color:"black" }}> Error </h3>
          <p style={{ color:"black" }}> { currentError } </p>
        </Popup>
      </>
    </>
  );
}
export default Canvas;

```

## 5.3. Pantalla de Dibujo

```
import React from "react";
import "../Draw.css"

import Navbar from "../../menu_components/Navbar";
import Canvas from "../Canvas";

const Solve = ({ history }) => {
  return (
    <div>
      <Navbar />
      <div className="jumbotron">
        <h1>Solve by Drawing</h1>
        <Canvas history={ history }/>
      </div>
    </div>
  );
}

export default Solve;
```

## 5.4. Pantalla de Teclado

```
import React from "react";
import Navbar from "../../menu_components/Navbar";
import MathKeyboard from "../solve_components/MathKeyboard";

const Keyboard = ({ history }) => {
  return (
    <div>
      <Navbar />
      <div className="jumbotron">
        <h1>Solve by Keyboard</h1><br></br>
        <MathKeyboard history={ history }/>
      </div>
    </div>
  );
}

export default Keyboard;
```

## 5.5. Pantalla de Fotografia

```
import React, { useState, useCallback, useEffect, useRef } from "react";
import Navbar from "../../menu_components/Navbar";
import Popup from "../../global_components/Popup";
import { IP_SERVER } from "../../global/consts";

const request = require("request-promise");

async function postData(url = '', data = {}) {
  const response = await fetch(url, {
    method: 'POST',
    mode: 'cors',
    cache: 'no-cache',
    credentials: 'same-origin',
    headers: {
      'Content-Type': 'application/json'
    },
    redirect: 'follow',
    referrerPolicy: 'no-referrer',
    body: JSON.stringify(data)
  });
  return response.json();
}

const Picture = ({ history }) => {

  const [alertPopup, setAlertPopup] = useState(false);
  const [currentError, setCurrentError] = useState(null);

  const videoRef = useRef(null);
  const photoRef = useRef(null);
  var hasPic = false;

  const getVideo = () => {
    try {
      navigator.mediaDevices.getUserMedia({
        video: { width: 250, height: 250 }
      }).then(stream => {
        let video = videoRef.current;
        video.srcObject = stream;
        video.play();
      }).catch(err => {
        console.error(err);
      })
    } catch (e) {
      console.log(e.message);
    }
  }
}
```

```

useEffect(() => {
  getVideo();
}, [videoRef]);

const handleSendPicture = useCallback(
  async event => {
    event.preventDefault();
    try {
      var imageFile = document.getElementById("file");
      if (imageFile.files.length > 0) {
        var reader = new FileReader();
        reader.onload = (e) => {
          var image64 = e.target.result;
          image64 = image64.split("data:image/png;base64,")[1]
const url = `https://${IP_SERVER}/image/text`;
          postData(url, { image: image64 })
            .then(async data => {
              console.log(data);
              if (data.status === 'ok') {
                const requestPreview = async () => {
                  const options = {
                    method: "POST",
                    uri:
`https://${IP_SERVER}/parse/latex`,
                    body: { equation: data.text },
                    json: true
                  };
                  const res = await request(options);
                  if (res.status !== 'ok') {
                    showAlertPopup(true)
                    setCurrentError(res.status)
                  } else {
                    history.push(`/preview/${res.equation}`);
                  }
                }
                await requestPreview();
              }
            });
        }
      } else {
        var canvas = document.getElementById('picture');
        var imgData = canvas.toDataURL();
        imgData = imgData.split("data:image/png;base64,")[1]
        if (!hasPic) {
          showAlertPopup(true)
          setCurrentError("Please upload a file or take a picture")
          return;
        }
      }
    }
  }
);

```



```

const url = `https://${IP_SERVER}/image/text`;
    postData(url, { image: imgData })
    .then(async data => {
        console.log(data);
        if (data.status === 'ok') {
            const requestPreview = async () => {
                const options = {
                    method: "POST",
                    uri: `https://${IP_SERVER}/parse/latex`,
                    body: { equation: data.text },
                    json: true
                };
                const res = await request(options);
                if (res.status !== 'ok') {
                    showAlertPopup(true)
                    setCurrentError(res.status)
                } else {
                    history.push(`/preview/${res.equation}`);
                }
            }
            await requestPreview();
        }
    });
}
} catch (error) {
    showAlertPopup(true)
    setCurrentError(error)
}
}, [history, hasPic]
)
const takePhoto = () => {
    const width = 250;
    const height = 250;

    let video = videoRef.current;
    let photo = photoRef.current;

    photo.width = width;
    photo.height = height;

    let context = photo.getContext('2d');
    context.drawImage(video, 0, 0, width, height);

    hasPic = true;
}

```

```

return (
  <div>
    <Navbar />
    <div className="jumbotron">
      <h1>Solve by Picture</h1>
      <form onSubmit={handleSendPicture}>
        <fieldset>
          <div className="form-group">
            <label>Select picture</label>
            <input type="file" className="form-control-file"
name="image" aria-describedby="fileHelp" id="file" />
            <small id="fileHelp" className="form-text text-muted">
              Please select an image from the gallery or take a
photo to continue
            </small><br></div>
            <video ref={videoRef}></video><br></div>
            <button type="button" className="btn btn-secondary"
onClick={takePhoto}>Take Picture</button><br></div><br></div>
            <canvas id="picture"
ref={photoRef}></canvas><br></div><br></div>
            <button type="submit" className="btn btn-primary">Send
image</button>
          </div>
        </fieldset>
      </form>
    </div>
    <Popup trigger={alertPopup} setTrigger={setAlertPopup}>
      <h3 style={{ color: "black" }}>Error </h3>
      <p style={{ color: "black" }}> {currentError} </p>
    </Popup>
  </div>
)
}

export default Picture;

```

## 6. Pantalla de solucion

---

```
import React, { useState, useContext } from 'react'
import { useParams } from 'react-router'

import Navbar from "../../menu_components/Navbar";
import app from "../../base";
import { AuthContext } from "../../auth";
import { WaitingScreen } from "../../alerts/WaitingScreen";
import { GenericAnomaly } from "../../anomalies/GenericAnomaly";
import { SolutionTimeoutAnomaly } from "../../anomalies/SolutionTimeoutAnomaly";
import { CompletenessAnomaly } from "../../anomalies/CompletenessAnomaly";
import { ClassificationAnomaly } from "../../anomalies/ClassificationAnomaly";
import { IP_SERVER } from "../../global/consts";

const request = require("request-promise");
const MathJax = require('react-mathjax');

const Preview = () => {
  const { currentUser } = useContext(AuthContext);
  const { equation } = useParams();
  const [ solution, setSolution ] = useState([]);
  const [ showLatex, setShowLatex ] = useState(false);

  const [ isWaiting, setIsWaiting ] = useState(false);
  const [ hasGenericAnomaly, setHasGenericAnomaly ] = useState(false);
  const [ hasCompletenessAnomaly, setHasCompletenessAnomaly ] = useState(false);
  const [ hasClassificationAnomaly, setHasClassificationAnomaly ] =
    useState(false);

  const [ genericAnomalyText, setGenericAnomalyText ] = useState(null);
  const [ completenessAnomalyText, setCompletenessAnomalyText ] = useState(null);
  const [ classificationAnomalyText, setClassificationAnomalyText ] =
    useState(null);
  const [ isOnTimeout, setIsOnTimeout ] = useState(false);
  const [ downloadButtonVisibility, setDownloadButtonVisibility ] =
    useState(false);
  const [ pdfContent, setPdfContent ] = useState(null);
  const [ latexForRender ] = useState([]);
```

```

var userKind = null;

const send = equation.replace("%2F", "/");

const addEquation = async (equation) => {
  const db = app.firestore();
  const uid = currentUser.uid;

  const newEquation = {
    date: new Date().toString().split("GMT")[0],
    equation: equation,
    pinned: false
  }

  const fetchData = async () => {
    db.collection("users").onSnapshot((querySnapshot) => {
      const docs = []
      querySnapshot.forEach((doc) => {
        if (doc.id === uid) {
          docs.push({ ...doc.data(), id: doc.id });
        }
      });

      try {
        userKind = docs[0].type;
      } catch (someError) {
        setHasGenericAnomaly(true)
        setGenericAnomalyText("Error getting the user type")
      }
    })
  }

  await fetchData()
  await db.collection('users').doc(uid).collection('equations').doc().set({
    ...newEquation
  });
}

```

```

const handleSubmit = async () => {
  await addEquation(send)
  const options = {
    method: "POST",
    uri: `https://${IP_SERVER}/solve`,
    body: { equation: send, type: userKind },
    json: true
  };
  console.log(options)
  var flagWait = true;
  setIsWaiting(true);
  const res = request(options);

  setTimeout(() => {
    if (flagWait) {
      res.abort()
      setIsWaiting(false)
      setIsOnTimeout(true)
    }
  }, 60000)
  // Catch the response from the server in body object
  res.then(body => {
    // Update waiting status flags
    setIsWaiting(false)
    flagWait = false
    if (body.status !== 'ok') {
      // Classification error
      if (body.exception === 'classification') {
        // Anomaly flags and messages
        setClassificationAnomalyText(body.status)
        setIsOnTimeout(false);
        setHasGenericAnomaly(false);
        setHasCompletenessAnomaly(false);
        setHasClassificationAnomaly(true);
        // Set partial solution
        deserializeSolution(body.solution)
      } // Completeness error
      else if (body.exception === 'completeness') {
        // Anomaly flags and messages
        setCompletenessAnomalyText(body.status);
        setIsOnTimeout(false);
        setHasGenericAnomaly(false);
        setHasCompletenessAnomaly(true);
        setHasClassificationAnomaly(false);
        // Set partial solution
        deserializeSolution(body.solution)
      } // Unexpected error
      else if (body.exception === 'generic') {
        // Anomaly flags and messages
        setGenericAnomalyText(body.status);
        setIsOnTimeout(false);
        setHasGenericAnomaly(true);
        setHasCompletenessAnomaly(false);
        setClassificationAnomalyText(false);
      }
    }
  })
}

```

```

// No Anomaly in solution
    } else {
        // Set solution
        deserializeSolution(body.solution)
    }
  })
}

const getText = (input) => {
  return input.replace("-", " ").replace("\\\\ \\", "")
}

const getLaTeX = (array) => {
  var latex = "";
  for (let i = 0; i < array.length; i++) {
    latex += getText(array[i].header) + "\\n";
    latex += array[i].latex + "\\n";
  }
  return latex;
}

const printLaTeX = () => {
  setShowLatex(!showLatex)
}

const renderLaTeXBox = () => {
  if (showLatex) {
    return (
      <div>
        <textarea readOnly className="form-control" style={{ height:
"300px" }}>{getLaTeX(latexForRender)}</textarea>
        <br></br>
      </div>
    )
  }
}

```

```

const deserializeSolution = (solution) => {
  var solutionAux = []
  const aux_01 = solution.replaceAll("'", "\'")
  const aux_02 = aux_01.replaceAll("\\""", "\\\"")

  const jsonSolve = JSON.parse(aux_02)
  for (let index = 0; index < jsonSolve.length; index++) {
    const step = jsonSolve[index];
    const stepHeader = step[0]
    const imgBase64Array = []
    var stepLatex = ""
    var stepLatexForRender = ""
    if (getText(stepHeader) !== " Graphs") {
      for (let j = 0; j < step[1].length; j++) {
        stepLatexForRender += step[1][j];
        var stepAux = step[1][j]
        if (!step[1][j].includes('$')) {
          stepAux = stepAux.replaceAll("\\", "")
          stepAux = `\\mathtt{\\text{${stepAux}}}`
        }
        console.log(stepAux)
        stepLatex += stepAux;
      }
    } else {
      for (let j = 0; j < step[1].length; j++) {
        imgBase64Array.push(step[1][j])
      }
    }
    const stepObject = { latex: stepLatex, header: stepHeader, images:
imgBase64Array }
    const stepForRenderObject = { latex: stepLatexForRender, header:
stepHeader }

    solutionAux.push(stepObject);
    latexForRender.push(stepForRenderObject);
  }

  setSolution(solutionAux);
}

const renderGraphs = (images) => {
  if (images.length === 0) {
    return (<></>)
  } else {
    return images.map(image => (
      <div>
        <img width="400" src={"data:image/png;base64," + image}
alt="Graph"
        style={{marginBottom: 15}}></img>
      </div>
    ));
  }
}

```

```

const renderLatexStep = (latex) => {
  console.log()
  return latex.replaceAll("$", "");
}
const renderSolve = () => {
  if (solution.length !== 0) {
    var i = 1;
    return solution.map(step => (
      <div>
        <br></br>
        <h4>
          Step {i}:
          {getText(step.header)}
          {(() => { i++ })()}
        </h4><br></br>
        <MathJax.default.Provider options={{ tex2jax: { processEscapes:
true,
  inlineMath: [["\\(", "\\)"]], displayMath: [["\\[", "\\]"] ] },
"HTML-CSS": {
    fonts: ["TeX"]
  } }}>
          <MathJax.default.Node
            inline={true}
            formula={renderLatexStep(step.latex)}
            onRender={() => console.log("Step " + (i-1) + "
rendered")}}
          />
        </MathJax.default.Provider>
        { renderGraphs(step.images) }
      </div>
    ))
  }
}
const renderLaTeXButton = () => {
  if (solution.length > 0) {
    if (!showLatex) {
      return (
        <div>
          <button onClick={printLaTeX} type="button" className="btn
btn-success">Get LaTeX</button>
          <br></br>
          <br></br>
        </div>
      )
    }
    return (
      <div>
        <button onClick={printLaTeX} type="button" className="btn btn-
danger">Hide LaTeX</button>
        <br></br>
        <br></br>
      </div>
    )
  }
}

```



```

const renderLaTeXPdfButton = () => {
  if (solution.length > 0) {
    return (
      <div>
        <button onClick={handleLatexToPdf} type="button" className="btn
btn-warning">Export to PDF</button>
        <br></br>
        <br></br>
      </div>
    )
  }
}

const handleLatexToPdf = () => {
  const options = {
    method: "POST",
    uri: `https://${IP_SERVER}/pdf`,
    body: { latex: getLaTeX(latexForRender) },
    json: true
  };
  var flagWait = true;
  const res = request(options);
  setTimeout(() => {
    if (flagWait) {
      res.abort()
      setIsWaiting(false)
      setIsOnTimeout(true)
    }
  }, 30000)
  // Catch the response from the server in body object
  res.then(body => {
    console.log("Pdf content: ")
    console.log(body.text)
    flagWait = false
    setPdfContent(body.text);
    setDownloadButtonVisibility(true);
  })
}

const renderDownloadButton = () => {
  if (downloadButtonVisibility) {
    const day = new Date();
    const pdfName = `${day.getFullYear()}-${day.getMonth()}-${
day.getDate()}-${day.getHours()}-${day.getMinutes()}-${day.getSeconds()}.pdf`
    return (
      <a download={ pdfName } href={
`data:application/pdf;base64,${pdfContent}` } title='Download pdf document'
      className="btn btn-primary">
        Download PDF Document
      </a>
    )
  } else {
    return <></>
  }
}

```

```

return (
  <div>
    <Navbar />
    <div className="jumbotron">
      <h1>Equation Preview</h1>
      <br></br>
      <h3>
        <MathJax.default.Provider>
          <MathJax.default.Node inline formula={send} />
        </MathJax.default.Provider>
      </h3><br></br>
      <button onClick={handleSubmit} type="button" className="btn btn-
primary">Solve</button><br></br>
      <br></br>
      <div id="solve">
        {renderSolve()}
      </div>
      {renderLaTeXButton()}
      {renderLaTeXBox()}
      {renderLaTeXPdfButton()}
      {renderDownloadButton()}
    </div>
    { new WaitingScreen(isWaiting, setIsWaiting).display() }
    { new GenericAnomaly(genericAnomalyText, hasGenericAnomaly,
setHasGenericAnomaly).display() }
    { new SolutionTimeoutAnomaly(isOnTimeout, setIsOnTimeout).display() }
    { new ClassificationAnomaly(classificationAnomalyText,
hasClassificationAnomaly, setHasClassificationAnomaly).display() }
    { new CompletenessAnomaly(completenessAnomalyText,
hasCompletenessAnomaly, setHasCompletenessAnomaly).display() }
  </div>
)
}

export default Preview;

```

# Capítulo VI:

## Pruebas y Evaluación

# 1. ODEs separables

---

## Prueba PODES-01

**Tipo de prueba:** Funcionalidad de ODE separable

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y,x) = x$$

**Equivalente:**

$$\frac{dy}{dx} = x$$

**Tiempos (3):** 00:02.413, 00:01.993, 00:01.498

**Tiempo Promedio:** 00:01.968

**Dificultad Global:** 390

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Si

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \frac{1}{2}x^2 + C$$

## Prueba PODES-02

**Tipo de prueba:** Funcionalidad de ODE separable

**Tipo de entrada:** Teclado

**Entrada:**

$$x \cdot D(y, x) = y^2$$

**Equivalente:**

$$x \frac{dy}{dx} = y^2$$

**Tiempos (3):** 00:02.614, 00:02.039, 00:01.654

**Tiempo Promedio:** 00:02.102

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Si

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \frac{1}{C - \log(x)}$$

## Prueba PODES-03

**Tipo de prueba:** Funcionalidad de ODE separable

**Tipo de entrada:** Teclado

**Entrada:**

$$(x^2+1)*D(y,x)=1/\sin(y)$$

**Equivalente:**

$$(x^2 + 1) \frac{dy}{dx} = \frac{1}{\sin(y)}$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = -\arccos(C - \arctan(x)) + 2\pi$$

## Prueba PODES-04

**Tipo de prueba:** Funcionalidad de ODE separable

**Tipo de entrada:** Teclado

**Entrada:**

$$(x^5)*(E^{(3*x)})=y*D(y,x)$$

**Equivalente:**

$$e^{3x}x^5 = y \frac{dy}{dx}$$

**Tiempos (3):** 00:19.972, 00:18.397, 00:17.671

**Tiempo Promedio:** 00:18.680

**Dificultad Global:** 770

**Máxima Dificultad de Paso Integral:** 140

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \pm \frac{\sqrt{-1485C + 486x^5e^{3x} - 810x^4e^{3x} + 1080x^3e^{3x} - 1080x^2e^{3x} + 720xe^{3x} - 240e^{3x}}}{27}$$

## Prueba PODES-05

**Tipo de prueba:** Funcionalidad de ODE separable

**Tipo de entrada:** Teclado

**Entrada:**

$$(E^x) * (\sin(x))^2 = \cos(y) * D(y, x)$$

**Equivalente:**

$$(x^2 + 1) \frac{dy}{dx} = \frac{1}{\sin(y)}$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = -\arccos(C - \arctan(x)) + 2\pi$$



## 2. ODEs homogéneas

---

### Prueba PODES-06

**Tipo de prueba:** Funcionalidad de ODE homogénea

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x) * x * (y^2) = (y^3 - x^3)$$

**Equivalente:**

$$\frac{dy}{dx} x y^2 = y^3 - x^3$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \sqrt[3]{-3C - 3 \log(x)}$$

## Prueba PODES-07

**Tipo de prueba:** Funcionalidad de ODE homogénea

**Tipo de entrada:** Teclado

**Entrada:**

$$x^2 + y^2 = x \cdot y \cdot D(y, x)$$

**Equivalente:**

$$(x^2 + y^2) = xy \frac{dy}{dx}$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = -x\sqrt{-2C + 2\log(x)}$$

# Prueba PODES-08

**Tipo de prueba:** Funcionalidad de ODE homogénea

**Tipo de entrada:** Teclado

**Entrada:**

$$(D(y,x)=(y^2-x^2)/(3*x*y))$$

**Equivalente:**

$$\frac{dy}{dx} = \frac{y^2 - x^2}{3xy}$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = -\frac{x\sqrt{-2 + 2e^{-4C}x^{-\frac{4}{3}}}}{2}$$

## 3. ODEs Lineares

---

### Prueba PODES-09

**Tipo de prueba:** Funcionalidad de ODE linear de orden 1

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x) + 5y = 10x - 3$$

**Equivalente:**

$$\frac{dy}{dx} + 5y = 10x - 3$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = Ce^{-5x} + 2x - 1$$

# Prueba PODES-10

**Tipo de prueba:** Funcionalidad de ODE linear de orden 1

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x) - x*y = (1 - x)*(E^x)$$

**Equivalente:**

$$\frac{dy}{dx} - xy = (1 - x)e^x$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = Ce^{\frac{x^2}{2}} + e^x$$

# Prueba PODES-11

**Tipo de prueba:** Funcionalidad de ODE linear de orden 1

**Tipo de entrada:** Teclado

**Entrada:**

$$x \cdot D(y, x) + 2 \cdot y = E^x$$

**Equivalente:**

$$x \frac{dy}{dx} + 2y = e^x$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \frac{C + (x - 1)e^x}{x^2}$$

## 4. ODEs Reducibles a Linear

---

### Prueba PODES-12

**Tipo de prueba:** Funcionalidad de ODE reducible a linear

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x) - y = 3 * (E^x) * (y^2)$$

**Equivalente:**

$$\frac{dy}{dx} - y = 3e^x y^2$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \frac{1}{(C + 3x)e^x}$$

## Prueba PODES-13

**Tipo de prueba:** Funcionalidad de ODE reducible a linear

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x) - x/y = 2*y$$

**Equivalente:**

$$\frac{dy}{dx} - \frac{x}{y} = 2y$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = \sqrt{C e^{-4x} - \frac{x}{2} + \frac{1}{8}}$$



## 5. ODEs Exactas

---

### Prueba PODES-14

**Tipo de prueba:** Funcionalidad de ODE exacta

**Tipo de entrada:** Teclado

**Entrada:**

$$(2xy - 3x^2y^2) + (x^2 - 2x^3y) \frac{dy}{dx} = 0$$

**Equivalente:**

$$2xy - 3x^2y^2 + (x^2 - 2x^3y) \frac{dy}{dx} = 0$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$0 = C - x^3y^2 + x^2y$$

## Prueba PODES-15

**Tipo de prueba:** Funcionalidad de ODE exacta

**Tipo de entrada:** Teclado

**Entrada:**

$$(E^{x+y}) + (2+x+y \cdot E^y) \cdot D(y, x) = 0$$

**Equivalente:**

$$e^x + y + (2 + x + ye^y) \frac{dy}{dx} = 0$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$0 = C + xy + 2y + (y - 1)e^y + e^x$$

## 6. ODEs Orden Superior

---

### Prueba PODES-16

**Tipo de prueba:** Funcionalidad de ODE orden superior

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x, 2) = -5y$$

**Equivalente:**

$$\frac{d^2y}{dx^2} = -5y$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = K \sin(\sqrt{5}x) + C \cos(\sqrt{5}x)$$

## Prueba PODES-17

**Tipo de prueba:** Funcionalidad de ODE orden superior

**Tipo de entrada:** Teclado

**Entrada:**

$$D(y, x, 2) - y = x$$

**Equivalente:**

$$\frac{d^2 y}{dx^2} - y = x$$

**Tiempos (3):** 00:07.575, 00:05.041, 00:04.760

**Tiempo Promedio:** 00:05.792

**Dificultad Global:** 400

**Máxima Dificultad de Paso Integral:** 60

**Datos de Investigador:** Error en críticos

**Gráfica:** Si

**PDF de solución:** Si

**Salida:**

$$y = x + C_1 e^x + C_0 e^{-x}$$

# Conclusiones

# 1. Situación del proyecto

---

Actualmente el proyecto se ha empezado a divulgar en un círculo cercano, en el que se ha puesto a prueba tanto la funcionalidad como la utilidad del sistema; hasta el momento toda la retroalimentación ha sido positiva, los objetivos estipulados por el proyecto se han cumplido y, como se esperaba, el sistema ofrece una interpretación y explicación única para la solución de ecuaciones diferenciales.

Pese a que la solución aún puede ser desglosada en pasos más específicos y el rango de tipos de ecuaciones diferenciales es aún limitado, resulta suficiente para estudiantes de universidad de ingeniería y algunas ciencias aplicadas de forma que les brinda soporte y ayuda de manera satisfactoria en su estudiar diario.

Así mismo los métodos de ingreso brindan la flexibilidad al usuario para la fácil y accesible resolución de un problema determinado, volviendo el uso de la aplicación ideal para estudiantes que buscan mejorar y superarse a sí mismos.

## 2. Futuro del proyecto

---

Todo el diseño del proyecto fue pensado para que fuera fácilmente escalable, de igual forma sus modelos de licenciamientos están pensados para que cualquier estudiante pueda aprender del código implementado y si lo desea aportar para añadir soporte a nuevos tipos de ecuaciones diferenciales o mejorar los procesos ya existentes de solución.

De igual forma los catálogos de integrales fueron implementados de forma que sea sencillo actualizarlos para brindar soporte a nuevas clases de integrales en el futuro o en todo volver más detallado el proceso de solución y brindar un análisis con mayor escrutinio.

El documentado del sistema, así como su implementación está hecha por módulos de forma que es posible trabajar sobre uno en particular sin necesidad de entender completamente lo que sucede en otros módulos, en caso de que el proyecto se volviera excesivamente grande distintos equipos podrán trabajar sin ningún problema en cada uno de los módulos de forma independiente

La implementación de APIs externas también esta implementada de una forma modular por lo que remplazarla por una tecnología desarrollada por el mismo equipo no debería presentar un problema mayor. Esta última tarea es algo que ya se está trabajando y está en fase de desarrollo se implementara en el futuro tras la entrega de la primera versión de este proyecto.

El proyecto tiene todas las características necesarias para poder seguir creciendo junto con la comunidad de software abierto, y poder cumplir su meta principal que es el de brindar apoyo y facilitar el estudio de las matemáticas en general **no solo las ecuaciones diferenciales.**

## 3. Futuras líneas

---

Los catálogos de integrales fueron implementados de forma que sea sencillo actualizarlos para brindar soporte a nuevas clases de integrales en el futuro o en todo volver más detallado el proceso de solución y brindar un análisis con mayor escrutinio.

La implementación de APIs externas también está implementada de una forma modular por lo que remplazarla por una tecnología desarrollada por el mismo equipo no debería presentar un problema mayor. Esta última tarea es algo que ya se está trabajando y está en fase de desarrollo se implementara en el futuro tras la entrega de la primera versión de este proyecto.



## 4. Lecciones del proyecto

---

Durante el proyecto se pusieron a prueba muchas aptitudes que debe poseer un programador más allá del conocimiento teórico se destaca la importancia de desarrollar paciencia, pensamiento crítico y habilidades de comunicación para el desarrollo de un proyecto en equipo.

En particular destacamos la habilidad de poder implementar tecnologías nuevas y nunca vista para atacar un problema particular. En este proyecto se implementaron muchas tecnologías con las que nunca habían trabajado los integrantes del equipo, sin embargo, el equipo encontró las semejanzas con lo aprendido y tuvo la capacidad de usar los conocimientos previos para implementar las nuevas tecnologías. En ese aspecto el equipo se ha superado a si mismo descubriendo muchas funcionalidades y nuevas herramientas que estarán a la disposición para atacar a un rango más amplio de problemas en el futuro.

## 5. Lecciones sobre el desarrollo

---

El equipo aprendió a manejar los problemas que tienen que ver con la ejecución de lo planeado o parte teórica del proyecto, esto es, hacer pruebas del sistema de manera eficiente, resolver problemas y debugear el proyecto de manera sistemática al ir codificando.

Otra parte importante fue el diseño estructurado para permitir el funcionamiento modular del proyecto de forma que el análisis, escalamiento y pruebas del proyecto se realizaran de forma sencilla y sin tener que reestructurar el mismo.

# Referencias

# 1. Referencias de Python

---

[1] Kuhlman, D. (2013). **A Python Book: Beginning Python**, Advanced Python, and Python Exercises.

[2] Chabot-Leclerc, A. (2017). **MATLAB® to Python: A Migration Guide**. 52.

[3] GaneshL6Follow. (2017). **Call MATLAB Script and Function from Python**. Instructables.  
From: <https://www.instructables.com/id/Call-MATLAB-Script-and-Function-From-Python/>

[4] **Solve Differential Equation**—MATLAB & Simulink—MathWorks América Latina. (2017).  
From: <https://la.mathworks.com/help/symbolic/solve-a-single-differential-equation.html>

[5] **Solve Differential Equations with ODEINT**. (2017).  
From: <https://apmonitor.com/pdc/index.php/Main/SolveDifferentialEquations>

[6] **An introduction to Python on Android**. (2017, marzo 31). Android Authority.  
From: <https://www.androidauthority.com/an-introduction-to-python-on-android-759685/>

[6] Toro, L. (2017, septiembre 7). **Kivy: Un framework para Python que permite desarrollar aplicaciones de manera rápida**. Desde Linux.  
From: <https://blog.desdelinux.net/kivy-framework-para-python/>

[7] **ODE — SymPy 1.5.1 documentation**. (2018).  
From: <https://docs.sympy.org/latest/modules/solvers/ode.html>

[10] Saha, S. (2018, December 17). **A Comprehensive Guide to Convolutional Neural Networks—The ELI5 way**. Medium.  
From: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[11] **Learn Programming Main/Solve Differential Equations in MATLAB.**(2019).

From: <http://apmonitor.com/che263/index.php/Main/MatlabDynamicSim>

[12] **Handwritten Equation Solver in Python.** (2019, Jun 26). GeeksforGeeks.

From: <https://www.geeksforgeeks.org/handwritten-equation-solver-in-python>

[13] Gad, A. (2020). **Ahmedfgad/NumPyCNNAndroid [Python].**

From: <https://github.com/ahmedfgad/NumPyCNNAndroid>

(Original work published 2018)

## 2. Referencias de React

---

[14] **Mathjax-react-npm**

From: <https://www.npmjs.com/package/mathjax-react>

[15] **Using MathJax with React** - Louis Li

From: <https://louisrli.github.io/blog/2020/06/04/react-mathjax/>

[16] **Mathjax-react-npm**

From: <https://github.com/wko27/react-mathjax>

[17] **Using MathJax v3 in React** · GitHub – Gist

From: <https://gist.github.com/GiacoSorsiglia/1619828473f4b34d3d914a16fcbf10f3>

[18] GitHub - asnunes/mathjax3-react: **React component to load.**

From: <https://www.npmjs.com/package/mathjax-react>

[19] **How to display Math Symbols in React** – Kindacode

From: <https://www.kindacode.com/article/displaying-math-symbols-in-react/>

### 3. Referencias de ODE & Vision

---

[20] **Numerical Methods | Unit I: First Order Differential Equations** | Differential Equations | Mathematics | MIT Open Courseware. (2016).  
From: <https://ocw.mit.edu/courses/mathematics/18-03sc-differential-equations-fall-2011/unit-i-first-order-differential-equations/numerical-methods/>

[21] Orozco, B. (2019, December 13). **Converting Handwritten Math Symbols into Text Using Random Forest. Medium.**  
From: <https://medium.com/@biorozco3/converting-handwritten-math-symbols-into-text-using-random-forest-26fd5517682a>

[22] Flôr, A. (2019, December 25). **Handwritten Text Recognition using TensorFlow 2.0.** Medium.  
From: <https://medium.com/@arthurflor23/handwritten-text-recognition-using-tensorflow-2-0-f4352b7afe16>

[23] **Android QuickStart | TensorFlow Lite.** (2020). TensorFlow.  
From: <https://www.tensorflow.org/lite/guide/android?hl=es>

[24] **Convolutional Neural Network (CNN)** | TensorFlow Core. (2020). TensorFlow.  
From: <https://www.tensorflow.org/tutorials/images/cnn?hl=es>

[25] **Create a package for Android**—Kivy 1.11.1 documentation. (2020).  
From: <https://kivy.org/doc/stable/guide/packaging-android.html>

[26] **Google Collaboratory | Convolutional Neural Network** (2020).  
From: [https://colab.research.google.com/github/Hvass-Labs/TensorFlow-Tutorials/blob/master/02\\_Convolutional\\_Neural\\_Network.ipynb](https://colab.research.google.com/github/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb)

[27] **Google Collaboratory | Deep Learning** (2020).  
From: [https://colab.research.google.com/github/lexfridman/mit-deep-learning/blob/master/tutorial\\_deep\\_learning\\_basics/deep\\_learning\\_basicsb.ipynb](https://colab.research.google.com/github/lexfridman/mit-deep-learning/blob/master/tutorial_deep_learning_basics/deep_learning_basicsb.ipynb)

[28] **Handwritten Math Recognition in Windows**—Wolfram Language Documentation.(2020).

From: <https://reference.wolfram.com/language/tutorial/HandwrittenMathRecognition.html>

[29] **TensorFlow/examples**. (2020). GitHub.

From: <https://github.com/tensorflow/examples>

[30] Gad, A. (2020, February 19). **Image Classification for Android Devices using NumPy and Kivy**. Medium.

From: <https://heartbeat.fritz.ai/image-classification-for-android-devices-using-numpy-and-kivy-587f65e7e99a>

[31] **Kivy/Kivy**. (2020). [Python]. Kivy.

From: <https://github.com/kivy/kivy> (Original work published 2010)

[32] Schechter, A., Borus, N., & Bakst, W. (2017). **Converting Handwritten Mathematical Expressions into LATEX**. Stanford, California, USA: Stanford University.

[33] More, Avinash, "**IMAGE TO LATEX VIA NEURAL NETWORKS**" (2018). Master's Projects. 602. DOI: <https://doi.org/10.31979/etd.b52e-g3e7>.

From: [https://scholarworks.sjsu.edu/etd\\_projects/602](https://scholarworks.sjsu.edu/etd_projects/602)

[34] Swartz, B. (2001). **Numerical Methods for Differential Equations**. Oxford, United Kingdom: Oxford University Press.

[35] Butcher, J. (2005). **Runge–Kutta methods for ordinary differential equations**. Auckland, New Zealand: The University of Auckland New Zealand.

[36] Wang, Z., & Liu, J.-C. (2002). **Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training**. Texas, Texas: University College Station.



# Apéndices

# 1. Manual de Usuario

---

## 1.1. Creación de Cuenta

En caso de no poseer una cuenta creada, presione el botón "Sign up", será dirigido a una forma para crear su cuenta, en esta le serán solicitados los siguientes datos:

Email: Ingrese un email valido, debe poseer el carácter arroba '@' seguido de las extensiones yahoo, gmail, hotmail, amazon. De no ser de esta forma le será negado el registro del correo.

Contraseña: Ingrese una contraseña para proteger su cuenta, tome en cuenta que le será solicitada para iniciar sesión cada vez que quiera acceder a la plataforma por lo que debe ser una contraseña que le sea fácil recordar.

Corroborar contraseña: Vuelva a ingresar la contraseña de su elección para confirmar su decisión. De no coincidir ambas contraseñas aparecerá un mensaje que le notificará de este error y no le será permitido registrar la contraseña.

Tipo de Usuario: Este campo posee dos opciones, **estudiante** o **profesor** la diferencia radica en los detalles que se darán sobre la solución de las ecuaciones a resolver. Nota: Si le interesa únicamente conocer la solución de la ecuación se recomienda el modo **estudiante**.

Tras terminar de llenar los campos oprima el botón "Sign up" y será iniciado a su sesión automáticamente, la próxima vez que intente ingresar a la plataforma siga los pasos descritos en la sección **Inicio de Sesión**.

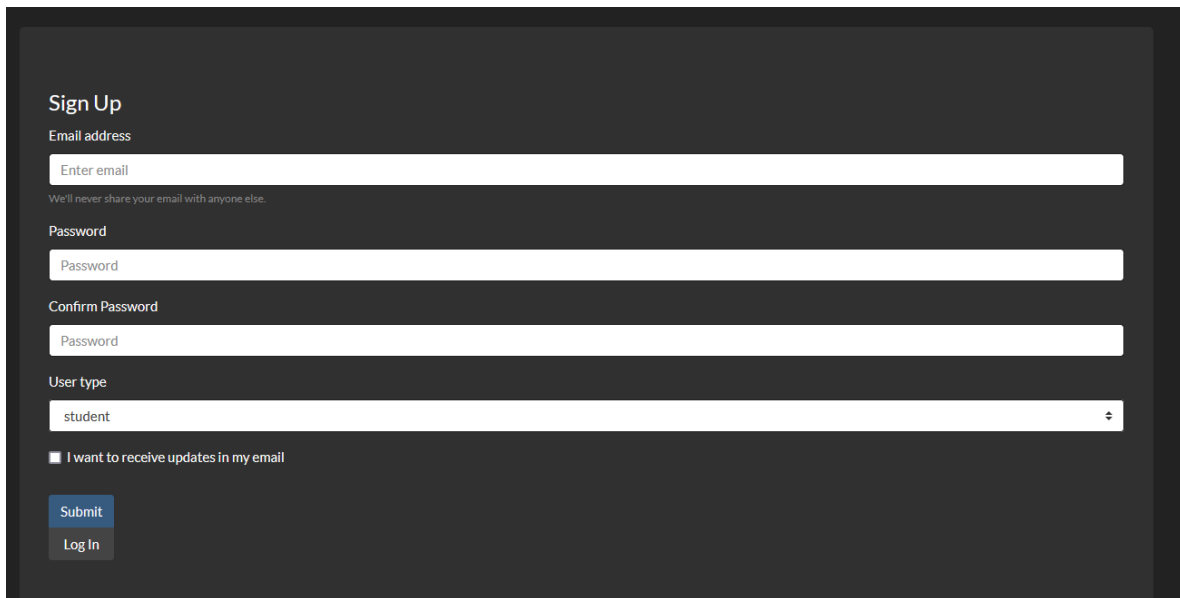
A screenshot of a 'Sign Up' form on a dark background. The form includes a title 'Sign Up', an 'Email address' label with a text input field containing 'Enter email', a small note 'We'll never share your email with anyone else.', a 'Password' label with a text input field containing 'Password', a 'Confirm Password' label with a text input field containing 'Password', a 'User type' label with a dropdown menu showing 'student', a checkbox labeled 'I want to receive updates in my email', and two buttons: 'Submit' and 'Log In'.

Figura 29: Demostración de Pantalla de crear cuenta

## 1.2. Inicio de Sesión

A screenshot of a 'Log In' form on a dark background. The form includes a title 'Log In', an 'Email address' label with a text input field containing 'Enter email', a small note 'We'll never share your email with anyone else.', a 'Password' label with a text input field containing 'Password', and two buttons: 'Submit' and 'Sign Up'.

Figura 30: Demostración de Pantalla de inicio de sesión

Si usted ya posee una cuenta, en la vista de inicio de sesión ingrese los datos correspondientes en la forma. Estos últimos son:

Email: Ingrese el email que haya registrado al crear su cuenta (descrito en la sección **Creación de Cuenta**).

Password: Ingrese la contraseña o password que haya registrado asociado a su email al crear su cuenta (descrito en la sección **Creación de Cuenta**).

Una vez ingresados los datos anteriores presione el botón "Login" para ingresar a la plataforma, en caso de que sus datos sean incorrectos (correo o contraseña) se le notificará en un mensaje.

## 1.3. Pantalla de Bienvenida

Al ingresar a su cuenta se encontrará en la página de inicio o "home", aquí encontrará un pequeño mensaje de bienvenida a la aplicación. En la parte superior de la pantalla encontrará una barra de navegación por la cual podrá navegar entre las distintas opciones de la aplicación. Estas son:

**Home:** pestaña actual

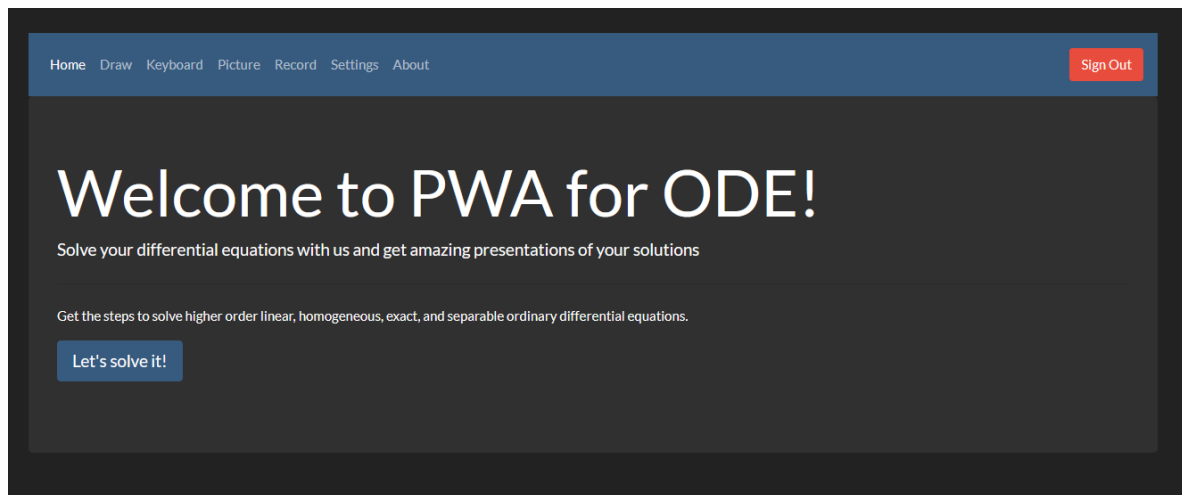


Figura 31: Demostración de Pantalla inicio

**Draw:** Ingrese la ecuación a resolver dibujándola en la pantalla de su computadora.

**Picture:** Ingrese la ecuación a resolver por medio de una imagen almacenada en su computadora o directamente tomándola con la cámara de su computadora/teléfono.

**Keyboard:** Ingrese manualmente la ecuación carácter por carácter con un teclado mostrado en pantalla.

**History:** Historial de ecuaciones; contiene las ultimas ecuaciones resueltas por el sistema en su cuenta.

**Settings:** Opciones para actualizar o confirmar los datos de su cuenta de usuario.

**About:** Información general de la aplicación.

Cada una de las anteriores opciones será descrita a detalle en las secciones correspondientes.

## 1.4. Pantalla de Dibujo

Al ser dirigido a esta pestaña encontrará un pequeño canvas (área de dibujo), representado como un recuadro de color blanco donde podrá dibujar con su ratón. Para realizar una línea coloque el ratón dentro del área delimitada por el recuadro blanco y presione el botón izquierdo mientras mueve el ratón para trazar una línea, para dejar de trazar la línea simplemente suelte el botón izquierdo del ratón.

Debajo del recuadro blanco encontrará tres botones.

El primer botón le permitirá deshacer el último trazo realizado por el ratón (puede presionarlo más de una vez para deshacer trazos anteriores), pero tenga cuidado pues no puede deshacer este cambio. Presione el botón derecho del ratón sobre este botón para habilitar la opción.

El segundo botón le permite cambiar a modo de borrador, de forma que los trazos que haga ahora con el ratón de la misma manera en la que ya se explicó anteriormente se borrarán los trazos por donde el ratón pase. Presione el botón derecho del ratón sobre este botón para habilitar la opción.

El tercer botón le permite quitar el modo borrador para volver a rayar en el recuadro blanco previamente. Presione el botón derecho del ratón sobre este botón para habilitar la opción.

Encontrará un botón adicional "Send" (Enviar) que evaluará la imagen que haya dibujado, en caso de que su dibujo no pueda ser procesado como una ecuación válida se mostrará un mensaje de error. En caso de no haber error se le dirigirá a la pestaña de **Solve**.

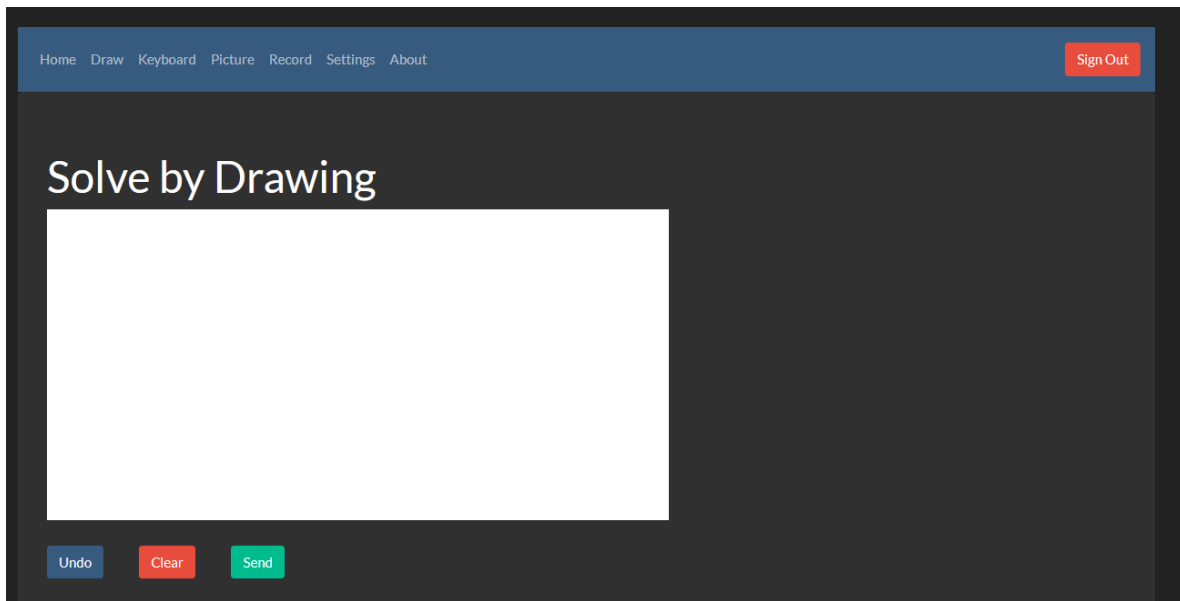


Figura 32: Demostración de Pantalla de dibujo

## 1.5. Pantalla de Fotografía

Al ser redirigido a esta página encontrará un botón blanco que dice “Attach image”, presione el botón derecho del ratón sobre este botón para habilitar la opción. Al presionar el botón aparecerá una ventana emergente con el administrador de archivos, navegue entre sus carpetas hasta encontrar la imagen que desea evaluar, de seleccione la imagen y presione el botón “aceptar” en la esquina inferior derecha del administrador de archivos para subir la imagen. Una vez adjuntado el archivo presione el botón **“Send”** para evaluar la ecuación, en caso de la imagen no pueda ser procesada como una ecuación valida se mostrará un mensaje de error. En caso de no haber error se le dirigirá a la pestaña de **Solve**.

Debajo del botón “Attach image”, encontrará un recuadro que muestra lo que ve la cámara de su ordenador o teléfono. Para adjuntar una imagen por la vía de fotografía posicione la ecuación diferencial a resolver escrita de forma clara en una hoja de papel o cuaderno enfrente de la cámara. Presione el botón “Take photo” cuando en el recuadro de la cámara que muestra la vista previa de la foto aparezca claramente la ecuación que pretende resolver. Una vez adjuntado el archivo presione el botón **“Send”** para evaluar la ecuación, en caso de la imagen no pueda ser procesada como una ecuación valida se mostrará un mensaje de error. En caso de no haber error se le dirigirá a la pestaña de **Solve**.

Nota: Debe dar permiso a la aplicación para utilizar la cámara, cuando su navegador predeterminado le pregunte si desea otorgar el permiso presione el botón "**Allow**" o "**Permitir**".

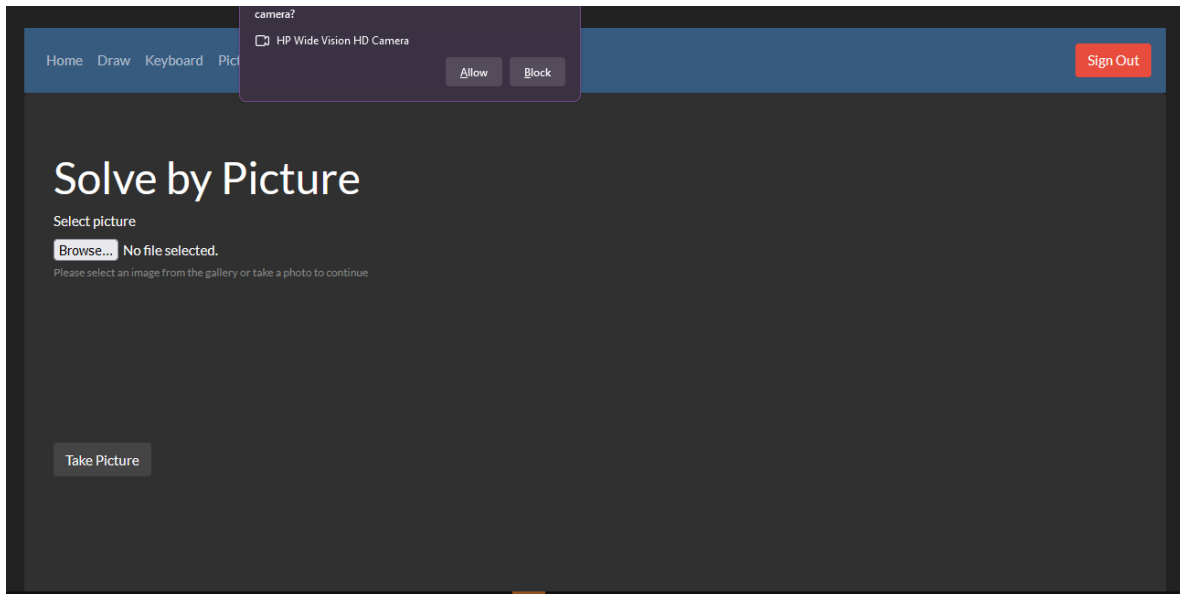


Figura 33: Demostración de Pantalla de fotografía

## 1.6. Pantalla de Teclado

Al dirigirse a esta pestaña se le mostrará un teclado que abarca toda la pantalla, presione el botón derecho del ratón sobre cada una de las teclas para incluir el carácter en la línea superior donde se irán acumulando los caracteres presionados. Escriba de esta forma la ecuación que desea evaluar. Al terminar presione el botón marcado con "**Send**" para evaluar la ecuación, en caso de la imagen no pueda ser procesada como una ecuación válida se mostrará un mensaje de error. En caso de no haber error se le dirigirá a la pestaña de **Solve**.

Nota: Tome en cuenta las reglas de sintaxis para la escritura de la ecuación diferencial a evaluar, descritas en la sección "**Reglas de Sintaxis para ODE**"

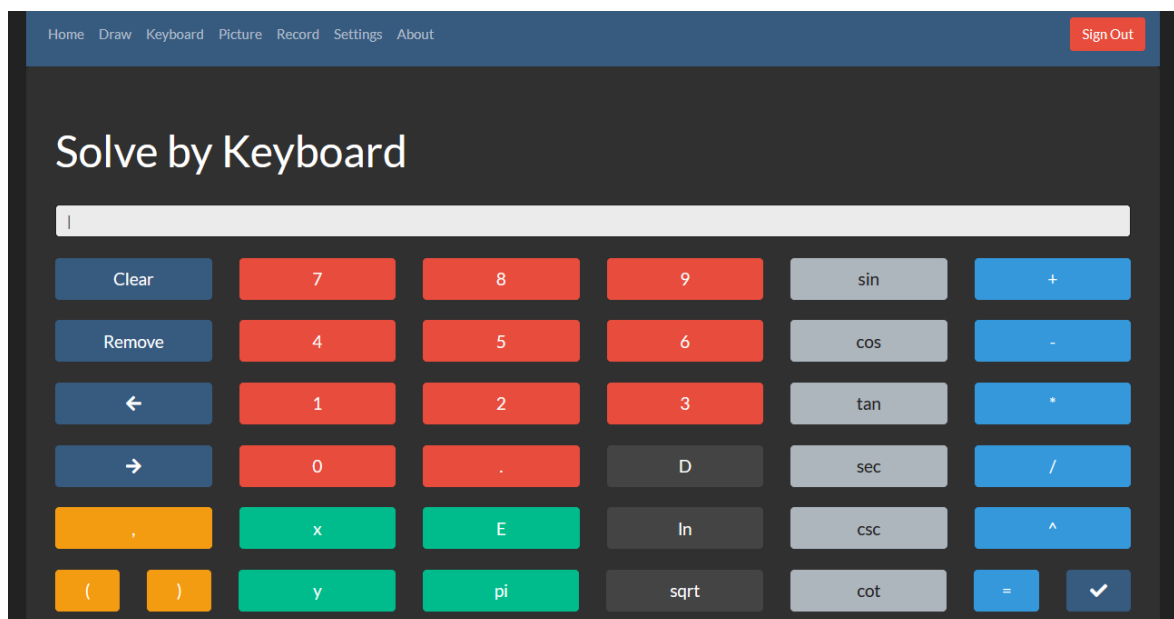


Figura 34: Demostración de Pantalla de teclado

## 1.7. Pantalla de Vista Previa y Solución

Al dirigirse a esta pestaña se le mostrará una vista previa de la ecuación que usted escribió, compruebe que efectivamente sea la ecuación que desea resolver. Si esta seguro que la vista previa representa la ecuación que desea resolver presione el botón etiquetado con “**Solve**” para iniciar el proceso de solución.

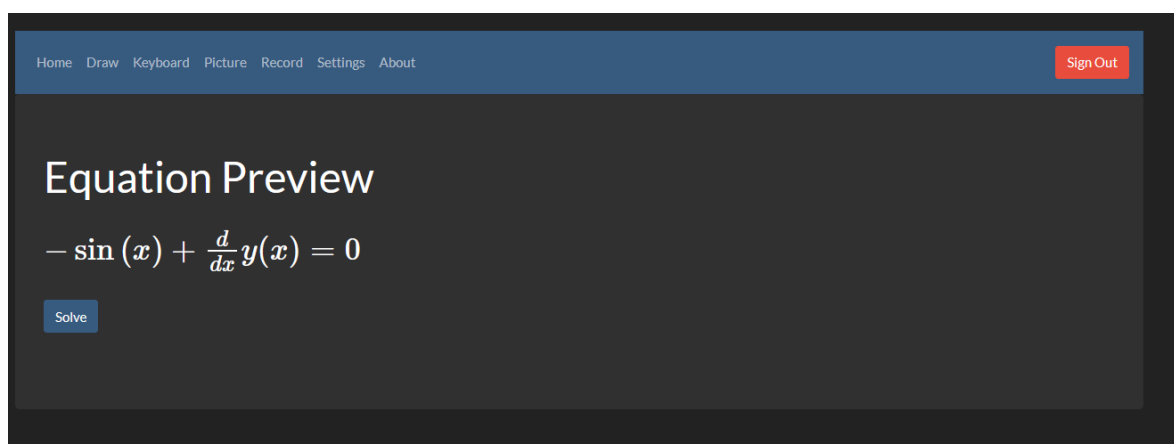


Figura 35: Demostración de Pantalla de vista previa



En caso de que ningún error se suscite debajo de la vista previa se mostrarán la lista de pasos que se realizaron para resolver la ecuación diferencial. Utilice la rueda del ratón para desplazarse por la página verticalmente y ver completamente la solución.

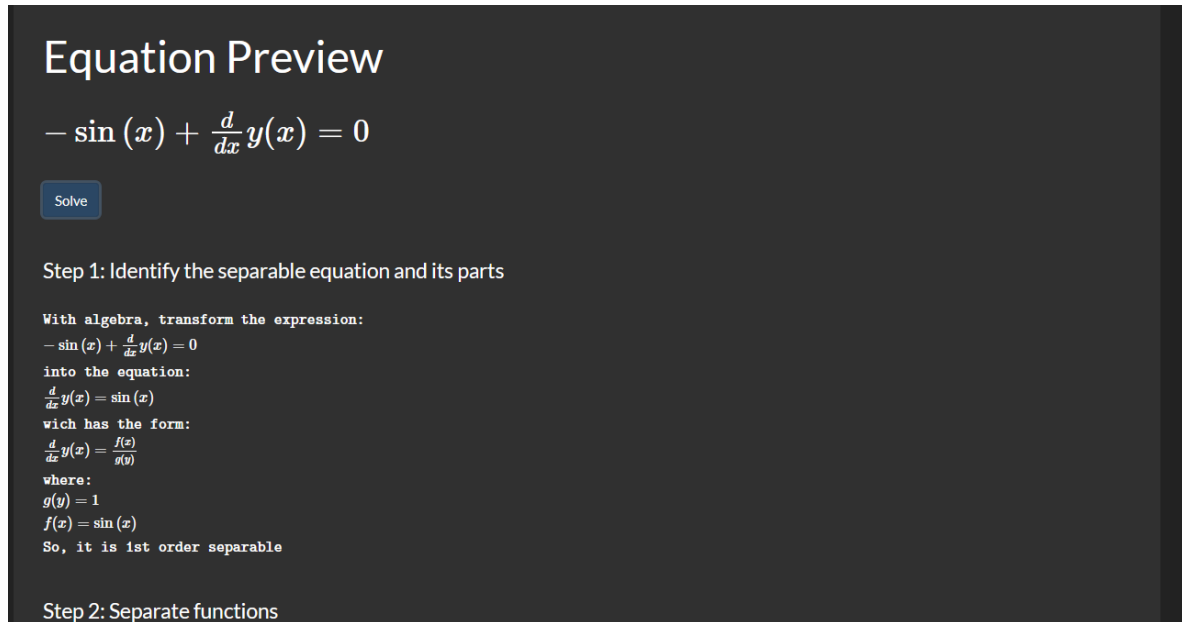


Figura 36: Demostración de Pantalla de solución

Tras la solución encontrara dos botones etiquetados como **GetLatex** y **GetPdf**.

**GetLatex:** Al presionar el botón izquierdo del ratón sobre este botón, aparecerá un recuadro debajo de este con el texto en *Latex* de la solución que acaba de obtener. Puede copiar este texto y pegarlo en un compilador de *Latex* para trasladar la solución a otro documento. **GetPdf:** Al presionar el botón izquierdo del ratón sobre este botón, se descargará un archivo pdf que contiene la solución de la ecuación solicitada. La aplicación mostrara una ventana emergente del administrador de archivos antes de descargar el pdf, elija la locación donde desea guardar el archivo y presione el botón aceptar en la esquina inferior derecha de la ventana emergente para concluir con el proceso de descarga.



Figura 37: Demostración de Botones para exportar solución

Si es usted un usuario de tipo profesor se mostrarán además de la lista de pasos ya mencionada, los puntos críticos y los puntos de inflexión de la solución, así como un gráfico en donde se muestran un par de familias de la solución general en un rango predeterminado.

Durante la ejecución de la solución puede que se de alguno de los siguientes errores:

**Error de completitud:** Este error se da cuando la ecuación se ha podido clasificar exitosamente sin embargo el tiempo de ejecución y la complejidad de la solución se volvió muy alto, por lo que el proceso de solución se interrumpe. El sistema mostrara la solución parcial que logro determinar, así como la solución final calculada por medio de algoritmos numéricos (sin pasos).

**Error de Clasificación:** Este error se da cuando la ecuación no se ha logrado clasificar con éxito, o cuando la ecuación que se evaluado no tiene soporte en el sistema. El sistema mostrara la solución final calculada por medio de algoritmos numéricos (sin pasos).

Para ingresar una nueva ecuación simplemente regrese a otra de las pestañas mediante la barra de navegación superior.

## 1.8. Pantalla de Historial

En esta pestaña se mostrarán las ultimas ecuaciones realizadas, ordenadas por la fecha en las que fueron evaluadas.

Cada ecuación disponible en la lista pose un botón con la imagen de un **pin** en este, presione el botón para priorizar dicha ecuación, moviéndola hasta el tope de la lista. Para retirar el **pin** de una ecuación simplemente vuelva a presionar el botón de nuevo. Todas las ecuaciones con **pin** serán ordenadas por fecha y serán ordenadas por fecha las ecuaciones sin **pin**.

Esta lista de ecuaciones tiene un máximo de 100 ecuaciones (las 100 más recientes), ecuaciones de fechas anteriores serán olvidadas por el sistema automáticamente ya sé que tengan **pin** o no.

Al presionar el botón derecho del ratón sobre cualquiera de las ecuaciones del historial, esta ecuación se cargará en la pestaña de **Solve** automáticamente.

Cada ecuación disponible en la lista posee un botón con la imagen de un bote de basura, presione el botón para remover dicha ecuación del historial. Tenga cuidado pues este proceso es irreversible.

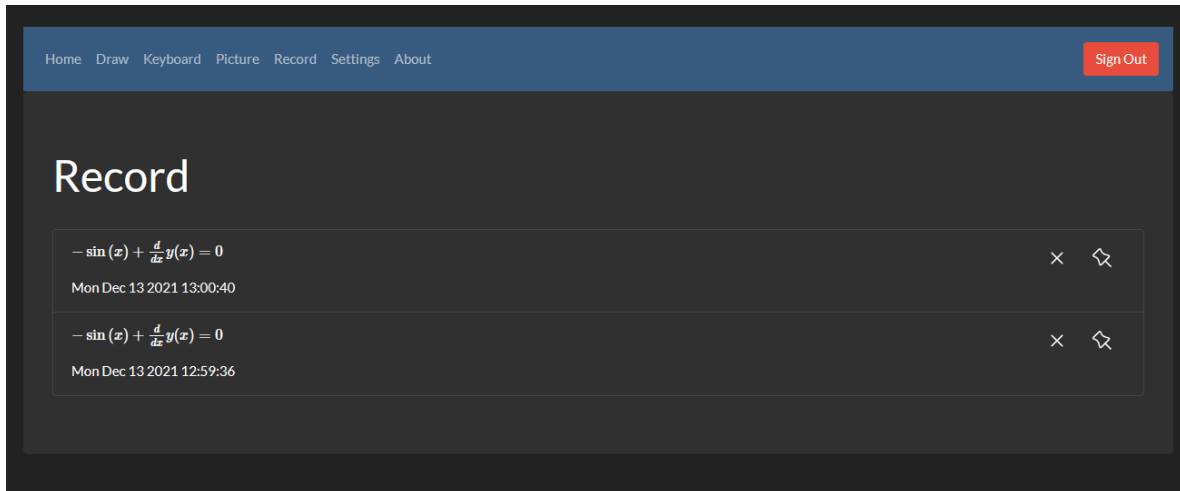


Figura 38: Demostración de Pantalla de historial

## 1.9. Pantalla de Configuración

En esta pestaña encontrara los datos de su cuenta, Email, Password y tipo de usuario. En caso de que desee cambiar alguno de estos datos, haga los cambios en sus respectivos campos y presione el botón "**Save**", para efectuar la actualización de los datos.

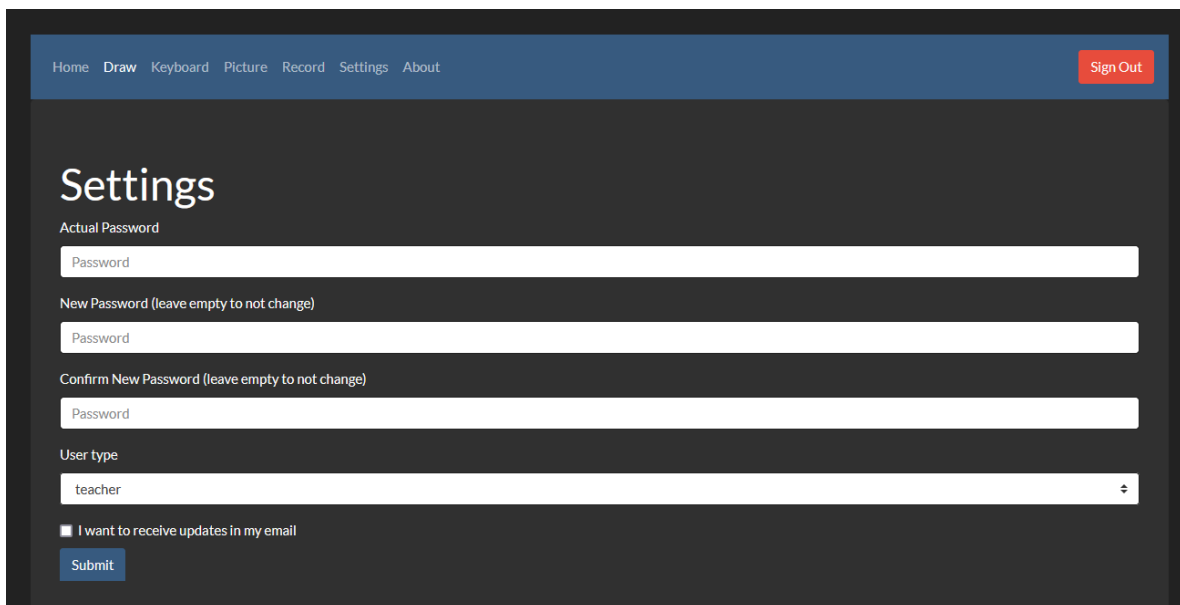


Figura 39: Demostración de Pantalla de configuración

## 1.10. Pantalla de Información

Posee información explícita de las reglas de sintaxis para el ingreso de ecuaciones diferenciales en el sistema, así como una lista de ejemplos que le serán útiles para comprender como el sistema hace la lectura de las ecuaciones ingresadas.

También se presenta una lista de atajos en el teclado para escribir las ecuaciones de una manera más rápida y fluida en el teclado implementado por el sistema.

Se lee recomienda al lector ávidamente que de lectura a esta pestaña antes de comenzar a usar el sistema para evitar cualquier confusión o mal funcionamiento.

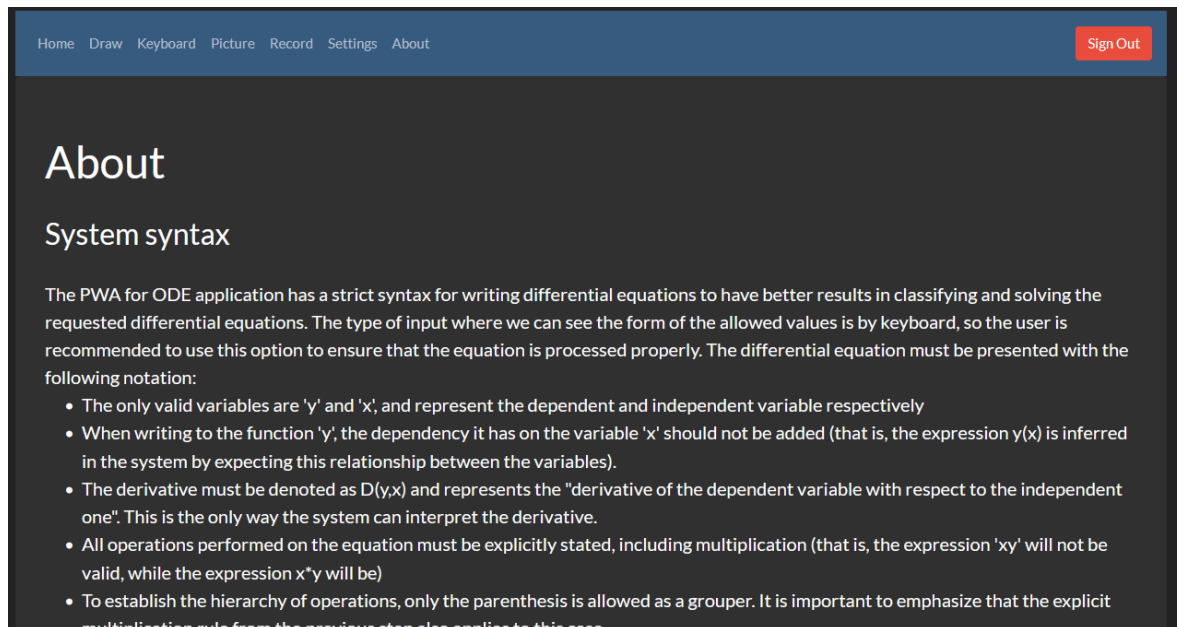


Figura 40: Demostración de Pantalla de información

## 2. Mantenimiento

---

Como parte del monitoreo de la aplicación será importante vigilar el flujo de información que es requerida por los clientes pues la cantidad de espacio que el sistema tiene disponible no es ilimitada, y fue desarrollado únicamente bajo prestaciones gratuitas de otros servicios. Es por esto último que será importante vigilar el estado de estos servicios y contratar un mayor espacio si este es requerido (ya hablamos en futuras líneas sobre remplazar estos servicios por desarrollos propios de forma que el sistema no dependa de ellos).

Debido a que implementa tecnologías y frameworks modernos el equipo checara constantemente si hay cambios en las librerías implementadas de forma que el sistema continúe funcionando correctamente pese a las actualizaciones que pueda haber en estas tecnologías.

Pese a que hemos ya corrido varias pruebas sobre el sistema ningún chequeo es completamente eficaz y puede que algunos errores continúen divagando por lo que el equipo continuara realizando pruebas sobre versiones no publicadas para arreglar imperfecciones, así mismo el equipo estará atento a cualquier queja o sugerencia de los usuarios del sistema.

Debido a la naturaleza del proyecto no existe un factor temporal que pueda afectar la calidad del producto excepto el mencionado en el primer punto de este apartado. Sin embargo, se recalca que la estructura modular del sistema le permite al equipo encontrar y arreglar errores de manera eficaz, así como expandir las funcionalidades de este sin tener que destruir el código previo.

## 3. Administración

---

Debido a que buscamos que el sistema sea una solución para la resolución algebraica de ecuaciones diferenciales y posiblemente en un futuro de varios problemas matemáticos, requeriremos de un equipo de trabajo que ayude a realizar estos avances posibles.

Uno de los objetivos de este proyecto es ayudar a otros estudiantes a aprender no solo de las soluciones que el sistema entrega, pero también del sistema en sí. Es por esto por lo que el licenciamiento del proyecto es de código abierto con ciertas restricciones para el control de su crecimiento.

Todas las ideas son bienvenidas y buscamos que la comunidad de desarrolladores de código abierto se una para contribuir a este proyecto y ayudar a más personas a aprender y desarrollar sus habilidades matemáticas y de programación.

# Anexos

# 1. Integrales Atómicas

---

# Updated in 14/11/2021 21:26:42

```
from sympy import *
def build_integrals(symbol):
    dx = Symbol('d' + str(symbol))
    x = Symbol(str(symbol))
    global BASIC
    global TEXT
    global SOLVE
    global EXCEPTIONS
    BASIC = []
    TEXT = []
    SOLVE = []
    EXCEPTIONS = []
    HINTS = []

    BASIC_0 = dx
    TEXT_0 = "Some text"
    SOLVE_0 = x
    BASIC.append(BASIC_0)
    TEXT.append(TEXT_0)
    SOLVE.append(SOLVE_0)
    HINT_0 = [[]]
    HINTS.append(HINT_0)
    EXCEPTION_0 = []
    EXCEPTIONS.append(EXCEPTION_0)

    BASIC_1 = Mul(dx, Pow(x, Symbol('n')))
    TEXT_1 = "Some text"
    SOLVE_1 = Mul(Pow(x, Add(Symbol('n'), Integer(1))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)))
    BASIC.append(BASIC_1)
    TEXT.append(TEXT_1)
    SOLVE.append(SOLVE_1)
    HINT_1 = [{"symbol":Symbol('n'), "value":1}]
    HINTS.append(HINT_1)
    EXCEPTION_1 = [{"symbol":Symbol('n'), "value":Integer(-1),
"type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_1)
```



```

BASIC_10 = Mul(dx, cot(Mul(Symbol('a'), x)), csc(Mul(Symbol('a'), x)))
    TEXT_10 = "Some text"
    SOLVE_10 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
csc(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_10)
    TEXT.append(TEXT_10)
    SOLVE.append(SOLVE_10)
    HINT_10 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_10)
    EXCEPTION_10 = []
    EXCEPTIONS.append(EXCEPTION_10)

    BASIC_100 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Integer(2),
Symbol('a'), x), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
    TEXT_100 = "Some text"
    SOLVE_100 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Mul(Pow(x, Integer(-1)), Add(Mul(Integer(2), Symbol('a'))),
Mul(Integer(-1), x))), Rational(1, 2)))
    BASIC.append(BASIC_100)
    TEXT.append(TEXT_100)
    SOLVE.append(SOLVE_100)
    HINT_100 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_100)
    EXCEPTION_100 = [{"symbol":Symbol('a'), "value":Integer(0),
"type":"g"}]
    EXCEPTIONS.append(EXCEPTION_100)

    BASIC_101 = Mul(dx, sinh(Mul(Symbol('a'), x)))
    TEXT_101 = "Some text"
    SOLVE_101 = Mul(Pow(Symbol('a'), Integer(-1)),
cosh(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_101)
    TEXT.append(TEXT_101)
    SOLVE.append(SOLVE_101)
    HINT_101 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_101)
    EXCEPTION_101 = []
    EXCEPTIONS.append(EXCEPTION_101)

```

```

BASIC_102 = Mul(dx, cosh(Mul(Symbol('a'), x)))
    TEXT_102 = "Some text"
    SOLVE_102 = Mul(Pow(Symbol('a'), Integer(-1)),
sinh(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_102)
    TEXT.append(TEXT_102)
    SOLVE.append(SOLVE_102)
    HINT_102 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_102)
    EXCEPTION_102 = []
    EXCEPTIONS.append(EXCEPTION_102)

    BASIC_103 = Mul(dx, Pow(sinh(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_103 = "Some text"
    SOLVE_103 = Add(Mul(Integer(-1), Rational(1, 2), x),
Mul(Rational(1, 4), Pow(Symbol('a'), Integer(-1)), sinh(Mul(Integer(2),
Symbol('a'), x))))
    BASIC.append(BASIC_103)
    TEXT.append(TEXT_103)
    SOLVE.append(SOLVE_103)
    HINT_103 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_103)
    EXCEPTION_103 = []
    EXCEPTIONS.append(EXCEPTION_103)

    BASIC_104 = Mul(dx, Pow(cosh(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_104 = "Some text"
    SOLVE_104 = Add(Mul(Rational(1, 2), x), Mul(Rational(1, 4),
Pow(Symbol('a'), Integer(-1)), sinh(Mul(Integer(2), Symbol('a'), x))))
    BASIC.append(BASIC_104)
    TEXT.append(TEXT_104)
    SOLVE.append(SOLVE_104)
    HINT_104 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_104)
    EXCEPTION_104 = []
    EXCEPTIONS.append(EXCEPTION_104)
BASIC_105 = Mul(dx, x, sinh(Mul(Symbol('a'), x)))
    TEXT_105 = "Some text"
    SOLVE_105 = Add(Mul(Pow(Symbol('a'), Integer(-1)),
cosh(Mul(Symbol('a'), x))), Mul(Integer(-1), Pow(Symbol('a'), Integer(-2)),
sinh(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_105)
    TEXT.append(TEXT_105)
    SOLVE.append(SOLVE_105)
    HINT_105 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_105)
    EXCEPTION_105 = []
    EXCEPTIONS.append(EXCEPTION_105)

```

```

BASIC_106 = Mul(dx, x, cosh(Mul(Symbol('a'), x)))
TEXT_106 = "Some text"
SOLVE_106 = Add(Mul(Pow(Symbol('a'), Integer(-1)),
sinh(Mul(Symbol('a'), x))), Mul(Integer(-1), Pow(Symbol('a'), Integer(-2)),
cosh(Mul(Symbol('a'), x))))
BASIC.append(BASIC_106)
TEXT.append(TEXT_106)
SOLVE.append(SOLVE_106)
HINT_106 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_106)
EXCEPTION_106 = []
EXCEPTIONS.append(EXCEPTION_106)

BASIC_107 = Mul(dx, tanh(Mul(Symbol('a'), x)))
TEXT_107 = "Some text"
SOLVE_107 = Mul(Pow(Symbol('a'), Integer(-1)),
log(cosh(Mul(Symbol('a'), x))))
BASIC.append(BASIC_107)
TEXT.append(TEXT_107)
SOLVE.append(SOLVE_107)
HINT_107 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_107)
EXCEPTION_107 = []
EXCEPTIONS.append(EXCEPTION_107)

BASIC_108 = Mul(dx, coth(Mul(Symbol('a'), x)))
TEXT_108 = "Some text"
SOLVE_108 = Mul(Pow(Symbol('a'), Integer(-1)),
log(sinh(Mul(Symbol('a'), x))))
BASIC.append(BASIC_108)
TEXT.append(TEXT_108)
SOLVE.append(SOLVE_108)
HINT_108 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_108)
EXCEPTION_108 = []
EXCEPTIONS.append(EXCEPTION_108)

BASIC_109 = Mul(dx, Pow(tanh(Mul(Symbol('a'), x)), Integer(2)))
TEXT_109 = "Some text"
SOLVE_109 = Add(x, Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
tanh(Mul(Symbol('a'), x))))
BASIC.append(BASIC_109)
TEXT.append(TEXT_109)
SOLVE.append(SOLVE_109)
HINT_109 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_109)
EXCEPTION_109 = []
EXCEPTIONS.append(EXCEPTION_109)

```

```

BASIC_11 = Mul(dx, tan(Mul(Symbol('a'), x)))
    TEXT_11 = "Some text"
    SOLVE_11 = Mul(Pow(Symbol('a'), Integer(-1)),
log(sec(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_11)
    TEXT.append(TEXT_11)
    SOLVE.append(SOLVE_11)
    HINT_11 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_11)
    EXCEPTION_11 = []
    EXCEPTIONS.append(EXCEPTION_11)

    BASIC_110 = Mul(dx, Pow(coth(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_110 = "Some text"
    SOLVE_110 = Add(x, Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
coth(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_110)
    TEXT.append(TEXT_110)
    SOLVE.append(SOLVE_110)
    HINT_110 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_110)
    EXCEPTION_110 = []
    EXCEPTIONS.append(EXCEPTION_110)

    BASIC_111 = Mul(dx, sech(Mul(Symbol('a'), x)))
    TEXT_111 = "Some text"
    SOLVE_111 = Mul(Pow(Symbol('a'), Integer(-1)),
asin(tanh(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_111)
    TEXT.append(TEXT_111)
    SOLVE.append(SOLVE_111)
    HINT_111 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_111)
    EXCEPTION_111 = []
    EXCEPTIONS.append(EXCEPTION_111)

    BASIC_112 = Mul(dx, csch(Mul(Symbol('a'), x)))
    TEXT_112 = "Some text"
    SOLVE_112 = Mul(Pow(Symbol('a'), Integer(-1)),
log(tanh(Mul(Rational(1, 2), Symbol('a'), x))))
    BASIC.append(BASIC_112)
    TEXT.append(TEXT_112)
    SOLVE.append(SOLVE_112)
    HINT_112 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_112)
    EXCEPTION_112 = []
    EXCEPTIONS.append(EXCEPTION_112)

```

```

BASIC_113 = Mul(dx, Pow(sech(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_113 = "Some text"
    SOLVE_113 = Mul(Pow(Symbol('a'), Integer(-1)),
tanh(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_113)
    TEXT.append(TEXT_113)
    SOLVE.append(SOLVE_113)
    HINT_113 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_113)
    EXCEPTION_113 = []
    EXCEPTIONS.append(EXCEPTION_113)
    BASIC_114 = Mul(dx, Pow(csch(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_114 = "Some text"
    SOLVE_114 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
coth(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_114)
    TEXT.append(TEXT_114)
    SOLVE.append(SOLVE_114)
    HINT_114 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_114)
    EXCEPTION_114 = []
    EXCEPTIONS.append(EXCEPTION_114)
    BASIC_115 = Mul(dx, tanh(Mul(Symbol('a'), x)),
Pow(sech(Mul(Symbol('a'), x)), Symbol('n')))
    TEXT_115 = "Some text"
    SOLVE_115 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), Pow(sech(Mul(Symbol('a'), x)), Symbol('n')))
    BASIC.append(BASIC_115)
    TEXT.append(TEXT_115)
    SOLVE.append(SOLVE_115)
    HINT_115 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('n'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
    HINTS.append(HINT_115)
    EXCEPTION_115 = []
    EXCEPTIONS.append(EXCEPTION_115)
    BASIC_116 = Mul(dx, cot(Mul(Symbol('a'), x)),
Pow(csch(Mul(Symbol('a'), x)), Symbol('n')))
    TEXT_116 = "Some text"
    SOLVE_116 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), Pow(csch(Mul(Symbol('a'), x)), Symbol('n')))
    BASIC.append(BASIC_116)
    TEXT.append(TEXT_116)
    SOLVE.append(SOLVE_116)
    HINT_116 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('n'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
    HINTS.append(HINT_116)
    EXCEPTION_116 = []
    EXCEPTIONS.append(EXCEPTION_116)

```

```

for hint_doc in hint_docs:
    BASIC_117 = Mul(dx, exp(Mul(Symbol('a'), x)), sinh(Mul(Symbol('b'), x)))
    TEXT_117 = "Some text"
    SOLVE_117 = Mul(Rational(1, 2), Add(Mul(Pow(Add(Symbol('a'),
Symbol('b')), Integer(-1)), exp(Mul(Symbol('b'), x))), Mul(Integer(-1),
Pow(Add(Symbol('a'), Mul(Integer(-1), Symbol('b'))), Integer(-1)),
exp(Mul(Integer(-1), Symbol('b'), x)))), exp(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_117)
    TEXT.append(TEXT_117)
    SOLVE.append(SOLVE_117)
    HINT_117 = [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}]]
    HINTS.append(HINT_117)
    EXCEPTION_117 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_117)

    BASIC_118 = Mul(dx, exp(Mul(Symbol('a'), x)), cosh(Mul(Symbol('b'),
x)))
    TEXT_118 = "Some text"
    SOLVE_118 = Mul(Rational(1, 2), Add(Mul(Pow(Add(Symbol('a'),
Symbol('b')), Integer(-1)), exp(Mul(Symbol('b'), x))),
Mul(Pow(Add(Symbol('a'), Mul(Integer(-1), Symbol('b'))), Integer(-1)),
exp(Mul(Integer(-1), Symbol('b'), x)))), exp(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_118)
    TEXT.append(TEXT_118)
    SOLVE.append(SOLVE_118)
    HINT_118 = [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}]]
    HINTS.append(HINT_118)
    EXCEPTION_118 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_118)

    BASIC_119 = Mul(dx, sin(Mul(Symbol('a'), x)), cos(Mul(Symbol('a'),
x)))
    TEXT_119 = "Some text"
    SOLVE_119 = Mul(Integer(-1), Rational(1, 4), Pow(Symbol('a'),
Integer(-1)), cos(Mul(Integer(2), Symbol('a'), x)))
    BASIC.append(BASIC_119)
    TEXT.append(TEXT_119)
    SOLVE.append(SOLVE_119)
    HINT_119 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_119)
    EXCEPTION_119 = []
    EXCEPTIONS.append(EXCEPTION_119)

```

```

BASIC_12 = Mul(dx, cot(Mul(Symbol('a'), x)))
TEXT_12 = "Some text"
SOLVE_12 = Mul(Pow(Symbol('a'), Integer(-1)), log(sin(Mul(Symbol('a'),
x))))
BASIC.append(BASIC_12)
TEXT.append(TEXT_12)
SOLVE.append(SOLVE_12)
HINT_12 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_12)
EXCEPTION_12 = []
EXCEPTIONS.append(EXCEPTION_12)
BASIC_13 = Mul(dx, sinh(Mul(Symbol('a'), x)))
TEXT_13 = "Some text"
SOLVE_13 = Mul(Pow(Symbol('a'), Integer(-1)), cosh(Mul(Symbol('a'), x)))
BASIC.append(BASIC_13)
TEXT.append(TEXT_13)
SOLVE.append(SOLVE_13)
HINT_13 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_13)
EXCEPTION_13 = []
EXCEPTIONS.append(EXCEPTION_13)
BASIC_14 = Mul(dx, cosh(Mul(Symbol('a'), x)))
TEXT_14 = "Some text"
SOLVE_14 = Mul(Pow(Symbol('a'), Integer(-1)), sinh(Mul(Symbol('a'), x)))
BASIC.append(BASIC_14)
TEXT.append(TEXT_14)
SOLVE.append(SOLVE_14)
HINT_14 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_14)
EXCEPTION_14 = []
EXCEPTIONS.append(EXCEPTION_14)
BASIC_15 = Mul(dx, Pow(Add(Pow(Symbol('a'), Integer(2)), Mul(Integer(-
1), Pow(x, Integer(2)))), Rational(-1, 2)))
TEXT_15 = "Some text"
SOLVE_15 = asin(Mul(Pow(Symbol('a'), Integer(-1)), x))
BASIC.append(BASIC_15)
TEXT.append(TEXT_15)
SOLVE.append(SOLVE_15)
HINT_15 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_15)
EXCEPTION_15 = []
EXCEPTIONS.append(EXCEPTION_15)
BASIC_16 = Mul(dx, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x,
Integer(2))), Integer(-1)))
TEXT_16 = "Some text"
SOLVE_16 = Mul(Pow(Symbol('a'), Integer(-1)), atan(Mul(Pow(Symbol('a'),
Integer(-1)), x)))
BASIC.append(BASIC_16)
TEXT.append(TEXT_16)
SOLVE.append(SOLVE_16)
HINT_16 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_16)
EXCEPTION_16 = []
EXCEPTIONS.append(EXCEPTION_16)

```

```

BASIC_17 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(-1, 2)))
TEXT_17 = "Some text"
SOLVE_17 = Mul(Pow(Symbol('a'), Integer(-1)),
asec(Mul(Pow(Symbol('a'), Integer(-1)), x)))
BASIC.append(BASIC_17)
TEXT.append(TEXT_17)
SOLVE.append(SOLVE_17)
HINT_17 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_17)
EXCEPTION_17 = []
EXCEPTIONS.append(EXCEPTION_17)

BASIC_18 = Mul(dx, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x,
Integer(2))), Rational(-1, 2)))
TEXT_18 = "Some text"
SOLVE_18 = asinh(Mul(Pow(Symbol('a'), Integer(-1)), x))
BASIC.append(BASIC_18)
TEXT.append(TEXT_18)
SOLVE.append(SOLVE_18)
HINT_18 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_18)
EXCEPTION_18 = []
EXCEPTIONS.append(EXCEPTION_18)

BASIC_19 = Mul(dx, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Rational(-1, 2)))
TEXT_19 = "Some text"
SOLVE_19 = acosh(Mul(Pow(Symbol('a'), Integer(-1)), x))
BASIC.append(BASIC_19)
TEXT.append(TEXT_19)
SOLVE.append(SOLVE_19)
HINT_19 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_19)
EXCEPTION_19 = [{"symbol":Symbol('a'), "value":Integer(0),
"type":"g"}]
EXCEPTIONS.append(EXCEPTION_19)

BASIC_2 = Mul(dx, Pow(x, Integer(-1)))
TEXT_2 = "Some text"
SOLVE_2 = log(x)
BASIC.append(BASIC_2)
TEXT.append(TEXT_2)
SOLVE.append(SOLVE_2)
HINT_2 = []
HINTS.append(HINT_2)
EXCEPTION_2 = []
EXCEPTIONS.append(EXCEPTION_2)

```



```

BASIC_20 = Mul(dx, Pow(Add(Mul(Symbol('a'), x), Symbol('b')), Symbol('n'))))
TEXT_20 = "Some text"
SOLVE_20 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Add(Symbol('n'),
Integer(1)), Integer(-1)), Pow(Add(Mul(Symbol('a'), x), Symbol('b')),
Add(Symbol('n'), Integer(1))))
BASIC.append(BASIC_20)
TEXT.append(TEXT_20)
SOLVE.append(SOLVE_20)
HINT_20 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_20)
EXCEPTION_20 = [{"symbol":Symbol('n'), "value":Integer(-1),
"type":"neq"}]
EXCEPTIONS.append(EXCEPTION_20)
BASIC_21 = Mul(dx, x, Pow(Add(Mul(Symbol('a'), x), Symbol('b')),
Symbol('n'))))
TEXT_21 = "Some text"
SOLVE_21 = Mul(Pow(Symbol('a'), Integer(-2)),
Pow(Add(Mul(Symbol('a'), x), Symbol('b')), Add(Symbol('n'), Integer(1))),
Add(Mul(Integer(-1), Symbol('b'), Pow(Add(Symbol('n'), Integer(1)),
Integer(-1))), Mul(Pow(Add(Symbol('n'), Integer(2)), Integer(-1)),
Add(Mul(Symbol('a'), x), Symbol('b'))))))
BASIC.append(BASIC_21)
TEXT.append(TEXT_21)
SOLVE.append(SOLVE_21)
HINT_21 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_21)
EXCEPTION_21 = [{"symbol":Symbol('n'), "value":Integer(-1),
"type":"neq"}, {"symbol":Symbol('n'), "value":Integer(-2), "type":"neq"}]
EXCEPTIONS.append(EXCEPTION_21)
BASIC_22 = Mul(dx, Pow(Add(Mul(Symbol('a'), x), Symbol('b')),
Integer(-1)))
TEXT_22 = "Some text"
SOLVE_22 = Mul(Pow(Symbol('a'), Integer(-1)),
log(Add(Mul(Symbol('a'), x), Symbol('b'))))
BASIC.append(BASIC_22)
TEXT.append(TEXT_22)
SOLVE.append(SOLVE_22)
HINT_22 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_22)
EXCEPTION_22 = []
EXCEPTIONS.append(EXCEPTION_22)
BASIC_23 = Mul(dx, x, Pow(Add(Mul(Symbol('a'), x), Symbol('b')),
Integer(-1)))
TEXT_23 = "Some text"
SOLVE_23 = Add(Mul(Pow(Symbol('a'), Integer(-1)), x), Mul(Integer(-
1), Pow(Symbol('a'), Integer(-2)), Symbol('b'), log(Add(Mul(Symbol('a'),
x), Symbol('b')))))
BASIC.append(BASIC_23)
TEXT.append(TEXT_23)
SOLVE.append(SOLVE_23)
HINT_23 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_23)
EXCEPTION_23 = []
EXCEPTIONS.append(EXCEPTION_23)

```

```

BASIC_24 = Mul(dx, x, Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Integer(-2)))
TEXT_24 = "Some text"
SOLVE_24 = Mul(Pow(Symbol('a'), Integer(-2)), Add(Mul(Symbol('b')),
Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Integer(-1))),
log(Add(Mul(Symbol('a')), x), Symbol('b'))))
BASIC.append(BASIC_24)
TEXT.append(TEXT_24)
SOLVE.append(SOLVE_24)
HINT_24 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_24)
EXCEPTION_24 = []
EXCEPTIONS.append(EXCEPTION_24)
BASIC_25 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Integer(-1)))
TEXT_25 = "Some text"
SOLVE_25 = Mul(Pow(Symbol('b'), Integer(-1)), log(Mul(x,
Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Integer(-1)))))
BASIC.append(BASIC_25)
TEXT.append(TEXT_25)
SOLVE.append(SOLVE_25)
HINT_25 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_25)
EXCEPTION_25 = []
EXCEPTIONS.append(EXCEPTION_25)
BASIC_26 = Mul(dx, Pow(Add(Mul(Symbol('a')), x), Symbol('b')),
Mul(Rational(1, 2), Symbol('n'))))
TEXT_26 = "Some text"
SOLVE_26 = Mul(Integer(2), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(2)), Integer(-1)), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Add(Mul(Rational(1, 2), Symbol('n')), Integer(1))))
BASIC.append(BASIC_26)
TEXT.append(TEXT_26)
SOLVE.append(SOLVE_26)
HINT_26 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_26)
EXCEPTION_26 = [{"symbol":Symbol('n'), "value":Integer(-2),
"type":"neq"}]
EXCEPTIONS.append(EXCEPTION_26)

BASIC_27 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Rational(-1, 2)))
TEXT_27 = "Some text"
SOLVE_27 = Mul(Pow(Symbol('b'), Rational(-1, 2)),
log(Mul(Add(Mul(Integer(-1), Pow(Symbol('b'), Rational(1, 2))),
Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Rational(1, 2))),
Pow(Add(Pow(Symbol('b'), Rational(1, 2)), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Rational(1, 2))), Integer(-1)))))
BASIC.append(BASIC_27)
TEXT.append(TEXT_27)
SOLVE.append(SOLVE_27)
HINT_27 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_27)
EXCEPTION_27 = []
EXCEPTIONS.append(EXCEPTION_27)

```

```

BASIC_28 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Symbol('a')), x),
Mul(Integer(-1), Symbol('b'))), Rational(-1, 2)))
    TEXT_28 = "Some text"
    SOLVE_28 = Mul(Integer(2), Pow(Symbol('b'), Rational(-1, 2)),
atan(Pow(Mul(Pow(Symbol('b'), Integer(-1)), Add(Mul(Symbol('a')), x),
Mul(Integer(-1), Symbol('b'))), Rational(1, 2))))
    BASIC.append(BASIC_28)
    TEXT.append(TEXT_28)
    SOLVE.append(SOLVE_28)
    HINT_28 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_28)
    EXCEPTION_28 = []
    EXCEPTIONS.append(EXCEPTION_28)
BASIC_29 = Mul(dx, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x,
Integer(2))), Integer(-2)))
    TEXT_29 = "Some text"
    SOLVE_29 = Add(Mul(Rational(1, 2), Pow(Symbol('a'), Integer(-2)),
x, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x, Integer(2))), Integer(-
1))), Mul(Rational(1, 2), Pow(Symbol('a'), Integer(-3)),
atan(Mul(Pow(Symbol('a'), Integer(-1)), x))))
    BASIC.append(BASIC_29)
    TEXT.append(TEXT_29)
    SOLVE.append(SOLVE_29)
    HINT_29 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_29)
    EXCEPTION_29 = []
    EXCEPTIONS.append(EXCEPTION_29)
BASIC_3 = Mul(dx, exp(x))
    TEXT_3 = "Some text"
    SOLVE_3 = exp(x)
    BASIC.append(BASIC_3)
    TEXT.append(TEXT_3)
    SOLVE.append(SOLVE_3)
    HINT_3 = [[]]
    HINTS.append(HINT_3)
    EXCEPTION_3 = []
    EXCEPTIONS.append(EXCEPTION_3)
BASIC_30 = Mul(dx, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x,
Integer(2))), Rational(1, 2)))
    TEXT_30 = "Some text"
    SOLVE_30 = Add(Mul(Rational(1, 2), Pow(Symbol('a'), Integer(2)),
log(Add(x, Pow(Add(Pow(Symbol('a'), Integer(2)), Pow(x, Integer(2))),
Rational(1, 2)))), Mul(Rational(1, 2), x, Pow(Add(Pow(Symbol('a'),
Integer(2)), Pow(x, Integer(2))), Rational(1, 2))))
    BASIC.append(BASIC_30)
    TEXT.append(TEXT_30)
    SOLVE.append(SOLVE_30)
    HINT_30 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_30)
    EXCEPTION_30 = []
    EXCEPTIONS.append(EXCEPTION_30)

```

```

BASIC_31 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_31 = "Some text"
SOLVE_31 = Add(Mul(Integer(-1), Rational(1, 8), Pow(Symbol('a'),
Integer(4)), log(Add(x, Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x,
Integer(2))), Rational(1, 2))))) , Mul(Rational(1, 8), x,
Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x, Integer(2))), Rational(1, 2)),
Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(2), Pow(x, Integer(2)))))
BASIC.append(BASIC_31)
TEXT.append(TEXT_31)
SOLVE.append(SOLVE_31)
HINT_31 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_31)
EXCEPTION_31 = []
EXCEPTIONS.append(EXCEPTION_31)

BASIC_32 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_32 = "Some text"
SOLVE_32 = Add(Mul(Integer(-1), Symbol('a'), log(Mul(Pow(x,
Integer(-1)), Add(Symbol('a'), Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x,
Integer(2))), Rational(1, 2))))) , Pow(Add(Pow(Symbol('a')), Integer(2)),
Pow(x, Integer(2))), Rational(1, 2)))
BASIC.append(BASIC_32)
TEXT.append(TEXT_32)
SOLVE.append(SOLVE_32)
HINT_32 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_32)
EXCEPTION_32 = []
EXCEPTIONS.append(EXCEPTION_32)

BASIC_33 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_33 = "Some text"
SOLVE_33 = Add(log(Add(x, Pow(Add(Pow(Symbol('a')), Integer(2)),
Pow(x, Integer(2))), Rational(1, 2))), Mul(Integer(-1), Pow(x, Integer(-
1)), Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x, Integer(2))), Rational(1,
2))))
BASIC.append(BASIC_33)
TEXT.append(TEXT_33)
SOLVE.append(SOLVE_33)
HINT_33 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_33)
EXCEPTION_33 = []
EXCEPTIONS.append(EXCEPTION_33)

```

```

BASIC_34 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(-1, 2)))
TEXT_34 = "Some text"
SOLVE_34 = Add(Mul(Integer(-1), Rational(1, 2), Pow(Symbol('a'),
Integer(2)), log(Add(x, Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x,
Integer(2))), Rational(1, 2))))), Mul(Rational(1, 2), x,
Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x, Integer(2))), Rational(1,
2))))
BASIC.append(BASIC_34)
TEXT.append(TEXT_34)
SOLVE.append(SOLVE_34)
HINT_34 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_34)
EXCEPTION_34 = []
EXCEPTIONS.append(EXCEPTION_34)
BASIC_35 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(-1, 2)))
TEXT_35 = "Some text"
SOLVE_35 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
log(Mul(Pow(x, Integer(-1)), Add(Symbol('a'), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(1, 2))))))
BASIC.append(BASIC_35)
TEXT.append(TEXT_35)
SOLVE.append(SOLVE_35)
HINT_35 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_35)
EXCEPTION_35 = []
EXCEPTIONS.append(EXCEPTION_35)
BASIC_36 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Pow(x, Integer(2))), Rational(-1, 2)))
TEXT_36 = "Some text"
SOLVE_36 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-2)), Pow(x,
Integer(-1)), Pow(Add(Pow(Symbol('a')), Integer(2)), Pow(x, Integer(2))),
Rational(1, 2)))
BASIC.append(BASIC_36)
TEXT.append(TEXT_36)
SOLVE.append(SOLVE_36)
HINT_36 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_36)
EXCEPTION_36 = []
EXCEPTIONS.append(EXCEPTION_36)
BASIC_37 = Mul(dx, Pow(Add(Pow(Symbol('a')), Integer(2)),
Mul(Integer(-1), Pow(x, Integer(2))), Integer(-1))),
TEXT_37 = "Some text"
SOLVE_37 = Mul(Rational(1, 2), Pow(Symbol('a'), Integer(-1)),
log(Mul(Pow(Add(Mul(Integer(-1), Symbol('a')), x), Integer(-1)),
Add(Symbol('a'), x))))
BASIC.append(BASIC_37)
TEXT.append(TEXT_37)
SOLVE.append(SOLVE_37)
HINT_37 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_37)
EXCEPTION_37 = []
EXCEPTIONS.append(EXCEPTION_37)

```

```

BASIC_38 = Mul(dx, Pow(Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-1), Pow(x,
Integer(2)))), Integer(-2)))
TEXT_38 = "Some text"
SOLVE_38 = Add(Mul(Rational(1, 2), Pow(Symbol('a')), Integer(-2)), x,
Pow(Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))),
Integer(-1))), Mul(Rational(1, 4), Pow(Symbol('a')), Integer(-3)),
log(Mul(Pow(Add(Mul(Integer(-1), Symbol('a')), x), Integer(-1)),
Add(Symbol('a'), x))))))
BASIC.append(BASIC_38)
TEXT.append(TEXT_38)
SOLVE.append(SOLVE_38)
HINT_38 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_38)
EXCEPTION_38 = []
EXCEPTIONS.append(EXCEPTION_38)
BASIC_39 = Mul(dx, Pow(Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-
1), Pow(x, Integer(2)))), Rational(1, 2)))
TEXT_39 = "Some text"
SOLVE_39 = Add(Mul(Rational(1, 2), Pow(Symbol('a')), Integer(2)),
asin(Mul(Pow(Symbol('a')), Integer(-1)), x))), Mul(Rational(1, 2), x,
Pow(Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))),
Rational(1, 2)))
BASIC.append(BASIC_39)
TEXT.append(TEXT_39)
SOLVE.append(SOLVE_39)
HINT_39 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_39)
EXCEPTION_39 = []
EXCEPTIONS.append(EXCEPTION_39)
BASIC_4 = Mul(Pow(Symbol('a'), x), dx)
TEXT_4 = "Some text"
SOLVE_4 = Mul(Pow(Symbol('a'), x), Pow(log(Symbol('a')), Integer(-1)))
BASIC.append(BASIC_4)
TEXT.append(TEXT_4)
SOLVE.append(SOLVE_4)
HINT_4 = [[]]
HINTS.append(HINT_4)
EXCEPTION_4 = [{"symbol":Symbol('a'), "value":Integer(1),
"type":"neq"}, {"symbol":Symbol('a'), "value":Integer(0), "type":"g"}]
EXCEPTIONS.append(EXCEPTION_4)
BASIC_40 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Pow(Symbol('a')), Integer(2)),
Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2)))
TEXT_40 = "Some text"
SOLVE_40 = Add(Mul(Rational(1, 8), Pow(Symbol('a')), Integer(4)),
asin(Mul(Pow(Symbol('a')), Integer(-1)), x))), Mul(Integer(-1), Rational(1, 8),
x, Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-1), Integer(2), Pow(x,
Integer(2)))), Pow(Add(Pow(Symbol('a')), Integer(2)), Mul(Integer(-1), Pow(x,
Integer(2)))), Rational(1, 2)))
BASIC.append(BASIC_40)
TEXT.append(TEXT_40)
SOLVE.append(SOLVE_40)
HINT_40 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_40)
EXCEPTION_40 = []
EXCEPTIONS.append(EXCEPTION_40)

```

```

BASIC_41 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Pow(Symbol('')),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2)))
    TEXT_41 = "Some text"
    SOLVE_41 = Add(Mul(Integer(-1), Symbol('a'), log(Mul(Pow(x,
Integer(-1)), Add(Symbol('a'), Pow(Add(Pow(Symbol('a'), Integer(2)),
Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2)))))),
Pow(Add(Pow(Symbol('a'), Integer(2)), Mul(Integer(-1), Pow(x,
Integer(2)))), Rational(1, 2)))
    BASIC.append(BASIC_41)
    TEXT.append(TEXT_41)
    SOLVE.append(SOLVE_41)
    HINT_41 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_41)
    EXCEPTION_41 = []
    EXCEPTIONS.append(EXCEPTION_41)

    BASIC_42 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2)))
    TEXT_42 = "Some text"
    SOLVE_42 = Add(Mul(Integer(-1), asin(Mul(Pow(Symbol('a'), Integer(-
1)), x))), Mul(Integer(-1), Pow(x, Integer(-1)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2))))
    BASIC.append(BASIC_42)
    TEXT.append(TEXT_42)
    SOLVE.append(SOLVE_42)
    HINT_42 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_42)
    EXCEPTION_42 = []
    EXCEPTIONS.append(EXCEPTION_42)

    BASIC_43 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Pow(Symbol('a')),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
    TEXT_43 = "Some text"
    SOLVE_43 = Add(Mul(Rational(1, 2), Pow(Symbol('a'), Integer(2)),
asin(Mul(Pow(Symbol('a'), Integer(-1)), x))), Mul(Integer(-1), Rational(1,
2), x, Pow(Add(Pow(Symbol('a'), Integer(2)), Mul(Integer(-1), Pow(x,
Integer(2)))), Rational(1, 2))))
    BASIC.append(BASIC_43)
    TEXT.append(TEXT_43)
    SOLVE.append(SOLVE_43)
    HINT_43 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_43)
    EXCEPTION_43 = []
    EXCEPTIONS.append(EXCEPTION_43)

```



```

BASIC_44 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Pow(Symbol('a'),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
TEXT_44 = "Some text"
SOLVE_44 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
log(Mul(Pow(x, Integer(-1)), Add(Symbol('a'), Pow(Add(Pow(Symbol('a'),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2))))))
BASIC.append(BASIC_44)
TEXT.append(TEXT_44)
SOLVE.append(SOLVE_44)
HINT_44 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_44)
EXCEPTION_44 = []
EXCEPTIONS.append(EXCEPTION_44)
BASIC_45 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Pow(Symbol('a'),
Integer(2)), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
TEXT_45 = "Some text"
SOLVE_45 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-2)), Pow(x,
Integer(-1)), Pow(Add(Pow(Symbol('a'), Integer(2)), Mul(Integer(-1), Pow(x,
Integer(2)))), Rational(1, 2)))
BASIC.append(BASIC_45)
TEXT.append(TEXT_45)
SOLVE.append(SOLVE_45)
HINT_45 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_45)
EXCEPTION_45 = []
EXCEPTIONS.append(EXCEPTION_45)
BASIC_46 = Mul(dx, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_46 = "Some text"
SOLVE_46 = Add(Mul(Integer(-1), Rational(1, 2), Pow(Symbol('a'),
Integer(2))), log(Add(x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Rational(1, 2))))), Mul(Rational(1, 2),
x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Rational(1, 2))))
BASIC.append(BASIC_46)
TEXT.append(TEXT_46)
SOLVE.append(SOLVE_46)
HINT_46 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_46)
EXCEPTION_46 = []
EXCEPTIONS.append(EXCEPTION_46)
BASIC_47 = Mul(dx, x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Mul(Rational(1, 2), Symbol('n'))))
TEXT_47 = "Some text"
SOLVE_47 = Mul(Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Add(Mul(Rational(1, 2), Symbol('n')),
Integer(1))), Pow(Add(Symbol('n'), Integer(2)), Integer(-1)))
BASIC.append(BASIC_47)
TEXT.append(TEXT_47)
SOLVE.append(SOLVE_47)
HINT_47 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_47)
EXCEPTION_47 = []
EXCEPTIONS.append(EXCEPTION_47)

```



```

BASIC_48 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_48 = "Some text"
SOLVE_48 = Add(Mul(Integer(-1), Rational(1, 8), Pow(Symbol('a'),
Integer(4)), log(Add(x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Rational(1, 2)))), Mul(Rational(1, 8), x,
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))),
Rational(1, 2)), Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))),
Mul(Integer(2), Pow(x, Integer(2))))))
BASIC.append(BASIC_48)
TEXT.append(TEXT_48)
SOLVE.append(SOLVE_48)
HINT_48 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_48)
EXCEPTION_48 = []
EXCEPTIONS.append(EXCEPTION_48)
BASIC_49 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_49 = "Some text"
SOLVE_49 = Add(Mul(Integer(-1), Symbol('a'), asec(Mul(Pow(Symbol('a'),
Integer(-1)), x))), Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))),
Pow(x, Integer(2))), Rational(1, 2)))
BASIC.append(BASIC_49)
TEXT.append(TEXT_49)
SOLVE.append(SOLVE_49)
HINT_49 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_49)
EXCEPTION_49 = []
EXCEPTIONS.append(EXCEPTION_49)
BASIC_5 = Mul(dx, sin(Mul(Symbol('a'), x)))
TEXT_5 = "Some text"
SOLVE_5 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
cos(Mul(Symbol('a'), x)))
BASIC.append(BASIC_5)
TEXT.append(TEXT_5)
SOLVE.append(SOLVE_5)
HINT_5 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_5)
EXCEPTION_5 = []
EXCEPTIONS.append(EXCEPTION_5)
BASIC_50 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(1, 2)))
TEXT_50 = "Some text"
SOLVE_50 = Add(log(Add(x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Rational(1, 2))), Mul(Integer(-1), Pow(x,
Integer(-1), Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Rational(1, 2))))
BASIC.append(BASIC_50)
TEXT.append(TEXT_50)
SOLVE.append(SOLVE_50)
HINT_50 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_50)
EXCEPTION_50 = []
EXCEPTIONS.append(EXCEPTION_50)

```

```

BASIC_51 = Mul(dx, Pow(x, Integer(2)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(-1, 2)))
    TEXT_51 = "Some text"
    SOLVE_51 = Add(Mul(Rational(1, 2), Pow(Symbol('a'), Integer(2)),
log(Add(x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Rational(1, 2))))), Mul(Rational(1, 2), x,
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Rational(1, 2))))
    BASIC.append(BASIC_51)
    TEXT.append(TEXT_51)
    SOLVE.append(SOLVE_51)
    HINT_51 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_51)
    EXCEPTION_51 = []
    EXCEPTIONS.append(EXCEPTION_51)
    BASIC_52 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(-1, 2)))
    TEXT_52 = "Some text"
    SOLVE_52 = Mul(Pow(Symbol('a'), Integer(-1)),
asec(Mul(Pow(Symbol('a'), Integer(-1)), x)))
    BASIC.append(BASIC_52)
    TEXT.append(TEXT_52)
    SOLVE.append(SOLVE_52)
    HINT_52 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_52)
    EXCEPTION_52 = []
    EXCEPTIONS.append(EXCEPTION_52)
    BASIC_53 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))), Rational(-1, 2)))
    TEXT_53 = "Some text"
    SOLVE_53 = Mul(Pow(Symbol('a'), Integer(-2)), Pow(x, Integer(-1)),
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Rational(1, 2)))
    BASIC.append(BASIC_53)
    TEXT.append(TEXT_53)
    SOLVE.append(SOLVE_53)
    HINT_53 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_53)
    EXCEPTION_53 = []
    EXCEPTIONS.append(EXCEPTION_53)
    BASIC_54 = Mul(dx, Pow(sin(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_54 = "Some text"
    SOLVE_54 = Add(Mul(Rational(1, 2), x), Mul(Integer(-1), Rational(1,
4), Pow(Symbol('a'), Integer(-1)), sin(Mul(Integer(2), Symbol('a'), x))))
    BASIC.append(BASIC_54)
    TEXT.append(TEXT_54)
    SOLVE.append(SOLVE_54)
    HINT_54 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_54)
    EXCEPTION_54 = []
    EXCEPTIONS.append(EXCEPTION_54)

```

```

BASIC_55 = Mul(dx, Pow(cos(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_55 = "Some text"
    SOLVE_55 = Add(Mul(Rational(1, 2), x), Mul(Rational(1, 4),
Pow(Symbol('a'), Integer(-1)), sin(Mul(Integer(2), Symbol('a'), x))))
    BASIC.append(BASIC_55)
    TEXT.append(TEXT_55)
    SOLVE.append(SOLVE_55)
    HINT_55 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_55)
    EXCEPTION_55 = []
    EXCEPTIONS.append(EXCEPTION_55)

    BASIC_56 = Mul(dx, sin(Mul(Symbol('a'), x)), cos(Mul(Symbol('b'),
x)))
    TEXT_56 = "Some text"
    SOLVE_56 = Add(Mul(Integer(-1), x, Pow(Add(Mul(Integer(2),
Symbol('a')), Mul(Integer(2), Symbol('b'))), Integer(-1)),
cos(Add(Symbol('a'), Symbol('b')))), Mul(Integer(-1), x,
Pow(Add(Mul(Integer(2), Symbol('a')), Mul(Integer(-1), Integer(2),
Symbol('b'))), Integer(-1)), cos(Add(Symbol('a'), Mul(Integer(-1),
Symbol('b'))))))))
    BASIC.append(BASIC_56)
    TEXT.append(TEXT_56)
    SOLVE.append(SOLVE_56)
    HINT_56 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_56)
    EXCEPTION_56 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_56)

    BASIC_57 = Mul(dx, sin(Mul(Symbol('a'), x)), sin(Mul(Symbol('b'),
x)))
    TEXT_57 = "Some text"
    SOLVE_57 = Add(Mul(Integer(-1), x, Pow(Add(Mul(Integer(2),
Symbol('a')), Mul(Integer(2), Symbol('b'))), Integer(-1)),
sin(Add(Symbol('a'), Symbol('b')))), Mul(x, Pow(Add(Mul(Integer(2),
Symbol('a')), Mul(Integer(-1), Integer(2), Symbol('b'))), Integer(-1)),
sin(Add(Symbol('a'), Mul(Integer(-1), Symbol('b'))))))))
    BASIC.append(BASIC_57)
    TEXT.append(TEXT_57)
    SOLVE.append(SOLVE_57)
    HINT_57 = [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}], [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}]
    HINTS.append(HINT_57)
    EXCEPTION_57 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_57)

```

```

BASIC_58 = Mul(dx, cos(Mul(Symbol('a'), x)), cos(Mul(Symbol('b'), x)))
    TEXT_58 = "Some text"
    SOLVE_58 = Add(Mul(x, Pow(Add(Mul(Integer(2), Symbol('a')),
Mul(Integer(2), Symbol('b'))), Integer(-1)), sin(Add(Symbol('a'),
Symbol('b')))), Mul(x, Pow(Add(Mul(Integer(2), Symbol('a')), Mul(Integer(-
1), Integer(2), Symbol('b'))), Integer(-1)), sin(Add(Symbol('a'),
Mul(Integer(-1), Symbol('b'))))))
    BASIC.append(BASIC_58)
    TEXT.append(TEXT_58)
    SOLVE.append(SOLVE_58)
    HINT_58 = [[{"symbol":Symbol('a'),
"value":1},{ "symbol":Symbol('b'), "value":1}], [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}]]
    HINTS.append(HINT_58)
    EXCEPTION_58 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_58)

    BASIC_59 = Mul(dx, Pow(sin(Mul(Symbol('a'), x)), Symbol('n')),
cos(Mul(Symbol('a'), x)))
    TEXT_59 = "Some text"
    SOLVE_59 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Add(Symbol('n'),
Integer(1)), Integer(-1)), Pow(sin(Mul(Symbol('a'), x)), Add(Symbol('n'),
Integer(1))))
    BASIC.append(BASIC_59)
    TEXT.append(TEXT_59)
    SOLVE.append(SOLVE_59)
    HINT_59 = [[{"symbol":Symbol('a'),
"value":1},{ "symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_59)
    EXCEPTION_59 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_59)
BASIC_6 = Mul(dx, cos(Mul(Symbol('a'), x)))
    TEXT_6 = "Some text"
    SOLVE_6 = Mul(Pow(Symbol('a'), Integer(-1)), sin(Mul(Symbol('a'),
x)))
    BASIC.append(BASIC_6)
    TEXT.append(TEXT_6)
    SOLVE.append(SOLVE_6)
    HINT_6 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_6)
    EXCEPTION_6 = []
    EXCEPTIONS.append(EXCEPTION_6)

```

```

BASIC_60 = Mul(dx, sin(Mul(Symbol('a'), x)), Pow(cos(Mul(Symbol('a'), x)),
Symbol('n')))
    TEXT_60 = "Some text"
    SOLVE_60 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), Pow(cos(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(1))))
    BASIC.append(BASIC_60)
    TEXT.append(TEXT_60)
    SOLVE.append(SOLVE_60)
    HINT_60 = [{"symbol":Symbol('a'),
"value":1},{ "symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_60)
    EXCEPTION_60 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('a'), Integer(2)), "type":"neq"}]
    EXCEPTIONS.append(EXCEPTION_60)

    BASIC_61 = Mul(dx, sin(Mul(Symbol('a'), x)),
Pow(cos(Mul(Symbol('a'), x)), Integer(-1)))
    TEXT_61 = "Some text"
    SOLVE_61 = Mul(Pow(Symbol('a'), Integer(-1)),
log(sec(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_61)
    TEXT.append(TEXT_61)
    SOLVE.append(SOLVE_61)
    HINT_61 = [{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_61)
    EXCEPTION_61 = []
    EXCEPTIONS.append(EXCEPTION_61)

    BASIC_62 = Mul(dx, Pow(sin(Mul(Symbol('a'), x)), Integer(-1)),
cos(Mul(Symbol('a'), x)))
    TEXT_62 = "Some text"
    SOLVE_62 = Mul(Pow(Symbol('a'), Integer(-1)),
log(sin(Mul(Symbol('a'), x))))
    BASIC.append(BASIC_62)
    TEXT.append(TEXT_62)
    SOLVE.append(SOLVE_62)
    HINT_62 = [{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_62)
    EXCEPTION_62 = []
    EXCEPTIONS.append(EXCEPTION_62)

```

```

BASIC_63 = Mul(dx, Pow(Add(Symbol('b'), Mul(Symbol('c'),
sin(Mul(Symbol('a'), x)))), Integer(-1)))
    TEXT_63 = "Some text"
    SOLVE_63 = Mul(Integer(-1), Integer(2), Pow(Symbol('a'), Integer(-
1)), Pow(Add(Pow(Symbol('b'), Integer(2)), Mul(Integer(-1),
Pow(Symbol('c'), Integer(2)))), Rational(-1, 2)),
atan(Mul(Pow(Mul(Add(Symbol('b'), Mul(Integer(-1), Symbol('c'))),
Pow(Add(Symbol('b'), Symbol('c')), Integer(-1))), Rational(1, 2)),
cot(Add(Mul(Rational(1, 2), Symbol('a'), x), Mul(Rational(1, 4), pi))))))
    BASIC.append(BASIC_63)
    TEXT.append(TEXT_63)
    SOLVE.append(SOLVE_63)
    HINT_63 = [[{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}], [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('c'), "value":1}]]
    HINTS.append(HINT_63)
    EXCEPTION_63 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('c'), Integer(2)), "type":"g"}]
    EXCEPTIONS.append(EXCEPTION_63)

    BASIC_64 = Mul(dx, Pow(Add(Symbol('b'), Mul(Symbol('c'),
sin(Mul(Symbol('a'), x)))), Integer(-1)))
    TEXT_64 = "Some text"
    SOLVE_64 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Mul(Integer(-1), Pow(Symbol('b'), Integer(2))), Pow(Symbol('c'),
Integer(2))), Rational(-1, 2)), log(Mul(Pow(Add(Symbol('b'),
Mul(Symbol('c'), sin(Mul(Symbol('a'), x)))), Integer(-1)),
Add(Mul(Symbol('b'), sin(Mul(Symbol('a'), x))), Symbol('c'),
Mul(Pow(Add(Mul(Integer(-1), Pow(Symbol('b'), Integer(2))),
Pow(Symbol('c'), Integer(2))), Rational(1, 2)), cos(Mul(Symbol('a'),
x)))))))
    BASIC.append(BASIC_64)
    TEXT.append(TEXT_64)
    SOLVE.append(SOLVE_64)
    HINT_64 = [[{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}], [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('c'), "value":1}]]
    HINTS.append(HINT_64)
    EXCEPTION_64 = [{"symbol":Pow(Symbol('c'), Integer(2)),
"value":Pow(Symbol('b'), Integer(2)), "type":"g"}]
    EXCEPTIONS.append(EXCEPTION_64)

    BASIC_65 = Mul(dx, Pow(Add(sin(Mul(Symbol('a'), x))), Integer(1)),
Integer(-1)))
    TEXT_65 = "Some text"
    SOLVE_65 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
cot(Add(Mul(Rational(1, 2), Symbol('a'), x), Mul(Rational(1, 4), pi))))
    BASIC.append(BASIC_65)
    TEXT.append(TEXT_65)
    SOLVE.append(SOLVE_65)
    HINT_65 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_65)
    EXCEPTION_65 = []
    EXCEPTIONS.append(EXCEPTION_65)

```

```

BASIC_66 = Mul(dx, Pow(Add(Integer(1), Mul(Integer(-1), sin(Mul(Symbol('a'),
x)))), Integer(-1)))
    TEXT_66 = "Some text"
    SOLVE_66 = Mul(Pow(Symbol('a'), Integer(-1)),
tan(Add(Mul(Rational(1, 2), Symbol('a'), x), Mul(Rational(1, 4), pi))))
    BASIC.append(BASIC_66)
    TEXT.append(TEXT_66)
    SOLVE.append(SOLVE_66)
    HINT_66 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_66)
    EXCEPTION_66 = []
    EXCEPTIONS.append(EXCEPTION_66)

    BASIC_67 = Mul(dx, Pow(Add(Symbol('b'), Mul(Symbol('c'),
cos(Mul(Symbol('a'), x)))), Integer(-1)))
    TEXT_67 = "Some text"
    SOLVE_67 = Mul(Integer(2), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Pow(Symbol('b'), Integer(2)), Mul(Integer(-1), Pow(Symbol('c'),
Integer(2)))), Rational(-1, 2)), cot(Mul(Pow(Mul(Add(Symbol('b'),
Mul(Integer(-1), Symbol('c'))), Pow(Add(Symbol('b'), Symbol('c')), Integer(-
1))), Rational(1, 2)), tan(Mul(Rational(1, 2), Symbol('a'), x))))
    BASIC.append(BASIC_67)
    TEXT.append(TEXT_67)
    SOLVE.append(SOLVE_67)
    HINT_67 = [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}]
    HINTS.append(HINT_67)
    EXCEPTION_67 = [{"symbol":Pow(Symbol('b'), Integer(2)),
"value":Pow(Symbol('c'), Integer(2)), "type":"g"}]
    EXCEPTIONS.append(EXCEPTION_67)

    BASIC_68 = Mul(dx, Pow(Add(Symbol('b'), Mul(Symbol('c'),
cos(Mul(Symbol('a'), x)))), Integer(-1)))
    TEXT_68 = "Some text"
    SOLVE_68 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Add(Mul(Integer(-
1), Pow(Symbol('b'), Integer(2))), Pow(Symbol('c'), Integer(2))), Rational(-
1, 2)), log(Mul(Pow(Add(Symbol('b'), Mul(Symbol('c'), cos(Mul(Symbol('a'),
x)))), Integer(-1)), Add(Mul(Symbol('b'), cos(Mul(Symbol('a'), x))),
Symbol('c'), Mul(Pow(Add(Mul(Integer(-1), Pow(Symbol('b'), Integer(2))),
Pow(Symbol('c'), Integer(2))), Rational(1, 2)), sin(Mul(Symbol('a'),
x)))))))
    BASIC.append(BASIC_68)
    TEXT.append(TEXT_68)
    SOLVE.append(SOLVE_68)
    HINT_68 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('c'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}]
    HINTS.append(HINT_68)
    EXCEPTION_68 = [{"symbol":Pow(Symbol('c'), Integer(2)),
"value":Pow(Symbol('b'), Integer(2)), "type":"g"}]
    EXCEPTIONS.append(EXCEPTION_68)

```



```

BASIC_69 = Mul(dx, Pow(Add(Mul(Symbol('c'), cos(Mul(Symbol('a'), x))),
Integer(1)), Integer(-1)))
TEXT_69 = "Some text"
SOLVE_69 = Mul(Pow(Symbol('a'), Integer(-1)), tan(Mul(Rational(1,
2), Symbol('a'), x)))
BASIC.append(BASIC_69)
TEXT.append(TEXT_69)
SOLVE.append(SOLVE_69)
HINT_69 = [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('c'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}]
HINTS.append(HINT_69)
EXCEPTION_69 = []
EXCEPTIONS.append(EXCEPTION_69)

BASIC_7 = Mul(dx, Pow(sec(Mul(Symbol('a'), x)), Integer(2)))
TEXT_7 = "Some text"
SOLVE_7 = Mul(Pow(Symbol('a'), Integer(-1)), tan(Mul(Symbol('a'),
x)))
BASIC.append(BASIC_7)
TEXT.append(TEXT_7)
SOLVE.append(SOLVE_7)
HINT_7 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_7)
EXCEPTION_7 = []
EXCEPTIONS.append(EXCEPTION_7)

BASIC_70 = Mul(dx, Pow(Add(Mul(Symbol('c'), cos(Mul(Symbol('a'),
x))), Integer(1)), Integer(-1)))
TEXT_70 = "Some text"
SOLVE_70 = Mul(Pow(Symbol('a'), Integer(-1)), cot(Mul(Rational(1,
2), Symbol('a'), x)))
BASIC.append(BASIC_70)
TEXT.append(TEXT_70)
SOLVE.append(SOLVE_70)
HINT_70 = [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('c'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('c'), "value":1}]
HINTS.append(HINT_70)
EXCEPTION_70 = []
EXCEPTIONS.append(EXCEPTION_70)

BASIC_71 = Mul(dx, x, sin(Mul(Symbol('a'), x)))
TEXT_71 = "Some text"
SOLVE_71 = Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)), x,
cos(Mul(Symbol('a'), x))), Mul(Pow(Symbol('a'), Integer(-2)),
sin(Mul(Symbol('a'), x))))
BASIC.append(BASIC_71)
TEXT.append(TEXT_71)
SOLVE.append(SOLVE_71)
HINT_71 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_71)
EXCEPTION_71 = []
EXCEPTIONS.append(EXCEPTION_71)

```



```

BASIC_72 = Mul(dx, x, cos(Mul(Symbol('a'), x)))
TEXT_72 = "Some text"
SOLVE_72 = Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)), x,
sin(Mul(Symbol('a'), x))), Mul(Pow(Symbol('a'), Integer(-2)), cos(Mul(Symbol('a'),
x))))
    BASIC.append(BASIC_72)
    TEXT.append(TEXT_72)
    SOLVE.append(SOLVE_72)
    HINT_72 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_72)
    EXCEPTION_72 = []
    EXCEPTIONS.append(EXCEPTION_72)

    BASIC_73 = Mul(dx, tan(Mul(Symbol('a'), x)))
    TEXT_73 = "Some text"
    SOLVE_73 = Mul(Pow(Symbol('a'), Integer(-1)), log(sec(Mul(Symbol('a'),
x))))
        BASIC.append(BASIC_73)
        TEXT.append(TEXT_73)
        SOLVE.append(SOLVE_73)
        HINT_73 = [[{"symbol":Symbol('a'), "value":1}]]
        HINTS.append(HINT_73)
        EXCEPTION_73 = []
        EXCEPTIONS.append(EXCEPTION_73)

        BASIC_74 = Mul(dx, cot(Mul(Symbol('a'), x)))
        TEXT_74 = "Some text"
        SOLVE_74 = Mul(Pow(Symbol('a'), Integer(-1)), log(sin(Mul(Symbol('a'),
x))))
            BASIC.append(BASIC_74)
            TEXT.append(TEXT_74)
            SOLVE.append(SOLVE_74)
            HINT_74 = [[{"symbol":Symbol('a'), "value":1}]]
            HINTS.append(HINT_74)
            EXCEPTION_74 = []
            EXCEPTIONS.append(EXCEPTION_74)

            BASIC_75 = Mul(dx, Pow(tan(Mul(Symbol('a'), x)), Integer(2)))
            TEXT_75 = "Some text"
            SOLVE_75 = Add(Mul(Integer(-1), x), Mul(Pow(Symbol('a'), Integer(-1)),
tan(Mul(Symbol('a'), x))))
                BASIC.append(BASIC_75)
                TEXT.append(TEXT_75)
                SOLVE.append(SOLVE_75)
                HINT_75 = [[{"symbol":Symbol('a'), "value":1}]]
                HINTS.append(HINT_75)
                EXCEPTION_75 = []
                EXCEPTIONS.append(EXCEPTION_75)
exception_ref = cat_ref.document(int_doc.id).collection('exceptions')
    exception_docs = exception_ref.stream()

    exception_text = ""
    for exception_doc in exception_docs:
        exception_dict = exception_doc.to_dict()

```

```

BASIC_76 = Mul(dx, Pow(cot(Mul(Symbol('a'), x)), Integer(2)))
TEXT_76 = "Some text"
SOLVE_76 = Add(Mul(Integer(-1), x), Mul(Integer(-1), Pow(Symbol('a'),
Integer(-1)), cot(Mul(Symbol('a'), x))))
BASIC.append(BASIC_76)
TEXT.append(TEXT_76)
SOLVE.append(SOLVE_76)
HINT_76 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_76)
EXCEPTION_76 = []
EXCEPTIONS.append(EXCEPTION_76)
BASIC_77 = Mul(dx, sec(Mul(Symbol('a'), x)))
TEXT_77 = "Some text"
SOLVE_77 = Mul(Pow(Symbol('a'), Integer(-1)), log(Add(tan(Mul(Symbol('a'),
x)), sec(Mul(Symbol('a'), x)))))
BASIC.append(BASIC_77)
TEXT.append(TEXT_77)
SOLVE.append(SOLVE_77)
HINT_77 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_77)
EXCEPTION_77 = []
EXCEPTIONS.append(EXCEPTION_77)
BASIC_78 = Mul(dx, csc(Mul(Symbol('a'), x)))
TEXT_78 = "Some text"
SOLVE_78 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
log(Add(cot(Mul(Symbol('a'), x)), csc(Mul(Symbol('a'), x)))))
BASIC.append(BASIC_78)
TEXT.append(TEXT_78)
SOLVE.append(SOLVE_78)
HINT_78 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_78)
EXCEPTION_78 = []
EXCEPTIONS.append(EXCEPTION_78)
BASIC_79 = Mul(dx, Pow(sec(Mul(Symbol('a'), x)), Integer(2)))
TEXT_79 = "Some text"
SOLVE_79 = Mul(Pow(Symbol('a'), Integer(-1)), tan(Mul(Symbol('a'), x)))
BASIC.append(BASIC_79)
TEXT.append(TEXT_79)
SOLVE.append(SOLVE_79)
HINT_79 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_79)
EXCEPTION_79 = []
EXCEPTIONS.append(EXCEPTION_79)
BASIC_8 = Mul(dx, Pow(csc(Mul(Symbol('a'), x)), Integer(2)))
TEXT_8 = "Some text"
SOLVE_8 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
cot(Mul(Symbol('a'), x)))
BASIC.append(BASIC_8)
TEXT.append(TEXT_8)
SOLVE.append(SOLVE_8)
HINT_8 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_8)
EXCEPTION_8 = []
EXCEPTIONS.append(EXCEPTION_8)

```

```

BASIC_80 = Mul(dx, Pow(csc(Mul(Symbol('a'), x)), Integer(2)))
    TEXT_80 = "Some text"
    SOLVE_80 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
cot(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_80)
    TEXT.append(TEXT_80)
    SOLVE.append(SOLVE_80)
    HINT_80 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_80)
    EXCEPTION_80 = []
    EXCEPTIONS.append(EXCEPTION_80)

    BASIC_81 = Mul(dx, tan(Mul(Symbol('a'), x)),
Pow(sec(Mul(Symbol('a'), x)), Symbol('n')))
    TEXT_81 = "Some text"
    SOLVE_81 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Symbol('n'),
Integer(-1)), Pow(sec(Mul(Symbol('a'), x)), Integer(2)))
    BASIC.append(BASIC_81)
    TEXT.append(TEXT_81)
    SOLVE.append(SOLVE_81)
    HINT_81 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_81)
    EXCEPTION_81 = []
    EXCEPTIONS.append(EXCEPTION_81)

    BASIC_82 = Mul(dx, cot(Mul(Symbol('a'), x)),
Pow(csc(Mul(Symbol('a'), x)), Symbol('n')))
    TEXT_82 = "Some text"
    SOLVE_82 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), Pow(csc(Mul(Symbol('a'), x)), Integer(2)))
    BASIC.append(BASIC_82)
    TEXT.append(TEXT_82)
    SOLVE.append(SOLVE_82)
    HINT_82 = [[{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
    HINTS.append(HINT_82)
    EXCEPTION_82 = []
    EXCEPTIONS.append(EXCEPTION_82)
BASIC_83 = Mul(dx, asin(Mul(Symbol('a'), x)))
    TEXT_83 = "Some text"
    SOLVE_83 = Add(Mul(x, asin(Mul(Symbol('a'), x))),
Mul(Pow(Symbol('a'), Integer(-1)), Pow(Add(Mul(Integer(-1),
Pow(Symbol('a'), Integer(2)), Pow(x, Integer(2))), Integer(1)), Rational(1,
2))))
    BASIC.append(BASIC_83)
    TEXT.append(TEXT_83)
    SOLVE.append(SOLVE_83)
    HINT_83 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_83)
    EXCEPTION_83 = []
    EXCEPTIONS.append(EXCEPTION_83)

```

```

BASIC_84 = Mul(dx, acos(Mul(Symbol('a'), x)))
TEXT_84 = "Some text"
SOLVE_84 = Add(Mul(x, acos(Mul(Symbol('a'), x))), Mul(Integer(-1),
Pow(Symbol('a'), Integer(-1)), Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2)), Pow(x, Integer(2))), Integer(1)), Rational(1, 2))))
BASIC.append(BASIC_84)
TEXT.append(TEXT_84)
SOLVE.append(SOLVE_84)
HINT_84 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_84)
EXCEPTION_84 = []
EXCEPTIONS.append(EXCEPTION_84)

BASIC_85 = Mul(dx, atan(Mul(Symbol('a'), x)))
TEXT_85 = "Some text"
SOLVE_85 = Add(Mul(x, atan(Mul(Symbol('a'), x))), Mul(Rational(1,
2), Pow(Symbol('a'), Integer(-1)), log(Add(Mul(Pow(Symbol('a'),
Integer(2)), Pow(x, Integer(2))), Integer(1)))))
BASIC.append(BASIC_85)
TEXT.append(TEXT_85)
SOLVE.append(SOLVE_85)
HINT_85 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_85)
EXCEPTION_85 = []
EXCEPTIONS.append(EXCEPTION_85)

BASIC_86 = Mul(dx, exp(Mul(Symbol('a'), x)))
TEXT_86 = "Some text"
SOLVE_86 = Mul(Pow(Symbol('a'), Integer(-1)), exp(Mul(Symbol('a'),
x)))
BASIC.append(BASIC_86)
TEXT.append(TEXT_86)
SOLVE.append(SOLVE_86)
HINT_86 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_86)
EXCEPTION_86 = []
EXCEPTIONS.append(EXCEPTION_86)

BASIC_87 = Mul(Pow(Symbol('b'), Mul(Symbol('a'), x)), dx)
TEXT_87 = "Some text"
SOLVE_87 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Symbol('b'),
Mul(Symbol('a'), x)), Pow(log(Symbol('b')), Integer(-1)))
BASIC.append(BASIC_87)
TEXT.append(TEXT_87)
SOLVE.append(SOLVE_87)
HINT_87 = [[{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}]]
HINTS.append(HINT_87)
EXCEPTION_87 = []
EXCEPTIONS.append(EXCEPTION_87)

```

```

BASIC_88 = Mul(dx, x, exp(Mul(Symbol('a'), x)))
    TEXT_88 = "Some text"
    SOLVE_88 = Mul(Pow(Symbol('a'), Integer(-2)), Add(Mul(Symbol('a'),
x), Integer(-1)), exp(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_88)
    TEXT.append(TEXT_88)
    SOLVE.append(SOLVE_88)
    HINT_88 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_88)
    EXCEPTION_88 = []
    EXCEPTIONS.append(EXCEPTION_88)
    BASIC_89 = Mul(dx, exp(Mul(Symbol('a'), x)), sin(Mul(Symbol('b'),
x)))
    TEXT_89 = "Some text"
    SOLVE_89 = Mul(Pow(Add(Pow(Symbol('a'), Integer(2)),
Pow(Symbol('b'), Integer(2))), Integer(-1)), Add(Mul(Symbol('a'),
sin(Mul(Symbol('b'), x))), Mul(Integer(-1), Symbol('b'),
cos(Mul(Symbol('b'), x)))), exp(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_89)
    TEXT.append(TEXT_89)
    SOLVE.append(SOLVE_89)
    HINT_89 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('b'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}]
    HINTS.append(HINT_89)
    EXCEPTION_89 = []
    EXCEPTIONS.append(EXCEPTION_89)
    BASIC_9 = Mul(dx, tan(Mul(Symbol('a'), x)), sec(Mul(Symbol('a'),
x)))
    TEXT_9 = "Some text"
    SOLVE_9 = Mul(Pow(Symbol('a'), Integer(-1)), sec(Mul(Symbol('a'),
x)))
    BASIC.append(BASIC_9)
    TEXT.append(TEXT_9)
    SOLVE.append(SOLVE_9)
    HINT_9 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_9)
    EXCEPTION_9 = []
    EXCEPTIONS.append(EXCEPTION_9)
    BASIC_90 = Mul(dx, exp(Mul(Symbol('a'), x)), cos(Mul(Symbol('b'),
x)))
    TEXT_90 = "Some text"
    SOLVE_90 = Mul(Pow(Add(Pow(Symbol('a'), Integer(2)),
Pow(Symbol('b'), Integer(2))), Integer(-1)), Add(Mul(Symbol('a'),
cos(Mul(Symbol('b'), x))), Mul(Symbol('b'), sin(Mul(Symbol('b'), x)))),
exp(Mul(Symbol('a'), x)))
    BASIC.append(BASIC_90)
    TEXT.append(TEXT_90)
    SOLVE.append(SOLVE_90)
    HINT_90 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('b'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('b'), "value":1}]
    HINTS.append(HINT_90)
    EXCEPTION_90 = []
    EXCEPTIONS.append(EXCEPTION_90)

```

```

BASIC_91 = Mul(dx, log(Mul(Symbol('a'), x)))
TEXT_91 = "Some text"
SOLVE_91 = Add(Mul(x, log(Mul(Symbol('a'), x))), Mul(Integer(-1),
x))
BASIC.append(BASIC_91)
TEXT.append(TEXT_91)
SOLVE.append(SOLVE_91)
HINT_91 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_91)
EXCEPTION_91 = []
EXCEPTIONS.append(EXCEPTION_91)

BASIC_92 = Mul(dx, Pow(x, Integer(-1)), Pow(log(Mul(Symbol('a'),
x)), Symbol('n')))
TEXT_92 = "Some text"
SOLVE_92 = Mul(Pow(Add(Symbol('n'), Integer(1)), Integer(-1)),
Pow(log(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(1))))
BASIC.append(BASIC_92)
TEXT.append(TEXT_92)
SOLVE.append(SOLVE_92)
HINT_92 = [{"symbol":Symbol('a'),
"value":1}, [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
HINTS.append(HINT_92)
EXCEPTION_92 = [{"symbol":Symbol('n'), "value":Integer(-1),
"type":"neq"}]
EXCEPTIONS.append(EXCEPTION_92)

BASIC_93 = Mul(dx, Pow(x, Integer(-1)), Pow(log(Mul(Symbol('a'),
x)), Integer(-1)))
TEXT_93 = "Some text"
SOLVE_93 = log(log(Mul(Symbol('a'), x)))
BASIC.append(BASIC_93)
TEXT.append(TEXT_93)
SOLVE.append(SOLVE_93)
HINT_93 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_93)
EXCEPTION_93 = []
EXCEPTIONS.append(EXCEPTION_93)

BASIC_94 = Mul(dx, Pow(Add(Mul(Integer(2), Symbol('a'), x),
Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
TEXT_94 = "Some text"
SOLVE_94 = asin(Mul(Pow(Symbol('a'), Integer(-1)),
Add(Mul(Integer(-1), Symbol('a')), x)))
BASIC.append(BASIC_94)
TEXT.append(TEXT_94)
SOLVE.append(SOLVE_94)
HINT_94 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_94)
EXCEPTION_94 = [{"symbol":Symbol('a'), "value":Integer(0),
"type":"g"}]
EXCEPTIONS.append(EXCEPTION_94)

```

```

BASIC_95 = Mul(dx, Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2))))) Rational(1, 2)))
TEXT_95 = "Some text"
SOLVE_95 = Add(Mul(Rational(1, 2), Pow(Symbol('a'), Integer(2)), asin(Mul(Pow(Symbol('a'), Integer(-1)), Add(Mul(Integer(-1), Symbol('a')), x))), Mul(Rational(1, 2), Add(Mul(Integer(-1), Symbol('a')), x), Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2))))) Rational(1, 2))))
BASIC.append(BASIC_95)
TEXT.append(TEXT_95)
SOLVE.append(SOLVE_95)
HINT_95 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_95)
EXCEPTION_95 = [{"symbol":Symbol('a'), "value":Integer(0), "type":"g"}]
EXCEPTIONS.append(EXCEPTION_95)

BASIC_96 = Mul(dx, x, Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2))))) Rational(1, 2)))
TEXT_96 = "Some text"
SOLVE_96 = asin(Mul(Pow(Symbol('a'), Integer(-1)), Add(Mul(Integer(-1), Symbol('a')), x)))
BASIC.append(BASIC_96)
TEXT.append(TEXT_96)
SOLVE.append(SOLVE_96)
HINT_96 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_96)
EXCEPTION_96 = [{"symbol":Symbol('a'), "value":Integer(0), "type":"g"}]
EXCEPTIONS.append(EXCEPTION_96)

BASIC_97 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2))))) Rational(1, 2)))
TEXT_97 = "Some text"
SOLVE_97 = Add(Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2))))) Rational(1, 2)), asin(Mul(Pow(Symbol('a'), Integer(-1)), Add(Mul(Integer(-1), Symbol('a')), x))))
BASIC.append(BASIC_97)
TEXT.append(TEXT_97)
SOLVE.append(SOLVE_97)
HINT_97 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_97)
EXCEPTION_97 = [{"symbol":Symbol('a'), "value":Integer(0), "type":"g"}]
EXCEPTIONS.append(EXCEPTION_97)

```

```

BASIC_98 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Mul(Integer(2),
Symbol('a'), x), Mul(Integer(-1), Pow(x, Integer(2)))), Rational(1, 2)))
    TEXT_98 = "Some text"
    SOLVE_98 = Add(Mul(Integer(-1), Integer(2), Pow(Mul(Pow(x,
Integer(-1)), Add(Mul(Integer(2), Symbol('a')), Mul(Integer(-1), x))),
Rational(1, 2))), Mul(Integer(-1), asin(Mul(Pow(Symbol('a'), Integer(-1)),
Add(Mul(Integer(-1), Symbol('a')), x)))))
    BASIC.append(BASIC_98)
    TEXT.append(TEXT_98)
    SOLVE.append(SOLVE_98)
    HINT_98 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_98)
    EXCEPTION_98 = [{"symbol":Symbol('a'), "value":Integer(0),
"type":"g"}]
    EXCEPTIONS.append(EXCEPTION_98)

    BASIC_99 = Mul(dx, x, Pow(Add(Mul(Integer(2), Symbol('a'), x),
Mul(Integer(-1), Pow(x, Integer(2)))), Rational(-1, 2)))
    TEXT_99 = "Some text"
    SOLVE_99 = Add(Mul(Symbol('a'), asin(Mul(Pow(Symbol('a'), Integer(-
1)), Add(Mul(Integer(-1), Symbol('a')), x)))), Mul(Integer(-1),
Pow(Add(Mul(Integer(2), Symbol('a'), x), Mul(Integer(-1), Pow(x,
Integer(2)))), Rational(1, 2))))
    BASIC.append(BASIC_99)
    TEXT.append(TEXT_99)
    SOLVE.append(SOLVE_99)
    HINT_99 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_99)
    EXCEPTION_99 = [{"symbol":Symbol('a'), "value":Integer(0),
"type":"g"}]
    EXCEPTIONS.append(EXCEPTION_99)
index = 0
    for HINT in HINTS:
        for STATE in HINT:
            if len(STATE) > 0:
                new_bas = BASIC[index]
                new_solve = SOLVE[index]
                for CHANGE in STATE:
                    new_bas =
new_bas.subs(CHANGE['symbol'], CHANGE['value'])
                    new_solve =
new_solve.subs(CHANGE['symbol'], CHANGE['value'])
                BASIC.append(new_bas)
                SOLVE.append(new_solve)
            index = index + 1

```



## 2. Integrales Recursivas

```

from sympy import *
def build_integrals(symbol):
    dx = Symbol('d' + str(symbol))
    x = Symbol(str(symbol))
    global RECURSIVE
    global TEXT
    global PARTIAL_SOLVE
    global NEW_INTEGRAL
    global EXCEPTIONS
    RECURSIVE = []
    TEXT = []
    PARTIAL_SOLVE = []
    NEW_INTEGRAL = []
    EXCEPTIONS = []
    HINTS = []
    RECURSIVE_0 = Mul(dx, Pow(x, Integer(-1)), Pow(Add(Mul(Symbol('a')),
x), Symbol('b')), Rational(1, 2)))
    TEXT_0 = "Some text"
    PARTIAL_SOLVE_0 = Mul(Integer(2), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Rational(1, 2)))
    NEW_INTEGRAL_0 = Mul(Symbol('b'), dx, Pow(x, Integer(-1)),
Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Rational(-1, 2)))
    RECURSIVE.append(RECURSIVE_0)
    TEXT.append(TEXT_0)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_0)
    NEW_INTEGRAL.append(NEW_INTEGRAL_0)
    HINT_0 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_0)
    EXCEPTION_0 = []
    EXCEPTIONS.append(EXCEPTION_0)
    RECURSIVE_1 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Mul(Symbol('a')), x),
Symbol('b')), Rational(1, 2)))
    TEXT_1 = "Some text"
    PARTIAL_SOLVE_1 = Mul(Integer(-1), Pow(x, Integer(-1)),
Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Rational(1, 2)))
    NEW_INTEGRAL_1 = Mul(Rational(1, 2), Symbol('a'), dx, Pow(x,
Integer(-1)), Pow(Add(Mul(Symbol('a')), x), Symbol('b')), Rational(-1, 2)))
    RECURSIVE.append(RECURSIVE_1)
    TEXT.append(TEXT_1)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_1)
    NEW_INTEGRAL.append(NEW_INTEGRAL_1)
    HINT_1 = [{"symbol":Symbol('a'), "value":1}]
    HINTS.append(HINT_1)
    EXCEPTION_1 = []
    EXCEPTIONS.append(EXCEPTION_1)

```

```

RECURSIVE_10 = Mul(dx, Pow(tan(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_10 = "Some text"
PARTIAL_SOLVE_10 = Mul(Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(tan(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_10 = Mul(Integer(-1), dx, Pow(tan(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_10)
TEXT.append(TEXT_10)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_10)
NEW_INTEGRAL.append(NEW_INTEGRAL_10)
HINT_10 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_10)
EXCEPTION_10 = []
EXCEPTIONS.append(EXCEPTION_10)

RECURSIVE_11 = Mul(dx, Pow(cot(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_11 = "Some text"
PARTIAL_SOLVE_11 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(cot(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_11 = Mul(Integer(-1), dx, Pow(cot(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_11)
TEXT.append(TEXT_11)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_11)
NEW_INTEGRAL.append(NEW_INTEGRAL_11)
HINT_11 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_11)
EXCEPTION_11 = []
EXCEPTIONS.append(EXCEPTION_11)

RECURSIVE_12 = Mul(dx, Pow(sec(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_12 = "Some text"
PARTIAL_SOLVE_12 = Mul(Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), tan(Mul(Symbol('a'), x)),
Pow(sec(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-2))))
NEW_INTEGRAL_12 = Mul(dx, Add(Symbol('n'), Integer(-2)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(sec(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_12)
TEXT.append(TEXT_12)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_12)
NEW_INTEGRAL.append(NEW_INTEGRAL_12)
HINT_12 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_12)
EXCEPTION_12 = []
EXCEPTIONS.append(EXCEPTION_12)

```

```

RECURSIVE_13 = Mul(dx, Pow(csc(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_13 = "Some text"
PARTIAL_SOLVE_13 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), cot(Mul(Symbol('a'), x)),
Pow(csc(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-2))))
NEW_INTEGRAL_13 = Mul(dx, Add(Symbol('n'), Integer(-2)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(csc(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_13)
TEXT.append(TEXT_13)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_13)
NEW_INTEGRAL.append(NEW_INTEGRAL_13)
HINT_13 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_13)
EXCEPTION_13 = []
EXCEPTIONS.append(EXCEPTION_13)
RECURSIVE_14 = Mul(dx, Pow(x, Symbol('n')), asin(Mul(Symbol('a'),
x)))
TEXT_14 = "Some text"
PARTIAL_SOLVE_14 = Mul(Pow(x, Add(Symbol('n'), Integer(1))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), asin(Mul(Symbol('a'), x)))
NEW_INTEGRAL_14 = Mul(Integer(-1), Symbol('a'), dx, Pow(x,
Add(Symbol('n'), Integer(1))), Pow(Add(Symbol('n'), Integer(1)), Integer(-
1)), Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2)), Pow(x,
Integer(2))), Integer(1)), Rational(-1, 2)))
RECURSIVE.append(RECURSIVE_14)
TEXT.append(TEXT_14)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_14)
NEW_INTEGRAL.append(NEW_INTEGRAL_14)
HINT_14 = [[{"symbol":Symbol('n'),
"value":1}, {"symbol":Symbol('a'), "value":1}], [{"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_14)
EXCEPTION_14 = []
EXCEPTIONS.append(EXCEPTION_14)
RECURSIVE_15 = Mul(dx, Pow(x, Symbol('n')), acos(Mul(Symbol('a'),
x)))
TEXT_15 = "Some text"
PARTIAL_SOLVE_15 = Mul(Pow(x, Add(Symbol('n'), Integer(1))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), acos(Mul(Symbol('a'), x)))
NEW_INTEGRAL_15 = Mul(Symbol('a'), dx, Pow(x, Add(Symbol('n'),
Integer(1))), Pow(Add(Symbol('n'), Integer(1)), Integer(-1)),
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2)), Pow(x, Integer(2))),
Integer(1)), Rational(-1, 2)))
RECURSIVE.append(RECURSIVE_15)
TEXT.append(TEXT_15)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_15)
NEW_INTEGRAL.append(NEW_INTEGRAL_15)
HINT_15 = [[{"symbol":Symbol('n'),
"value":1}, {"symbol":Symbol('a'), "value":1}], [{"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_15)
EXCEPTION_15 = []
EXCEPTIONS.append(EXCEPTION_15)

```

```

    RECURSIVE_16 = Mul(dx, Pow(x, Symbol('n')), atan(Mul(Symbol('a'), x)))
    TEXT_16 = "Some text"
    PARTIAL_SOLVE_16 = Mul(Pow(x, Add(Symbol('n'), Integer(1))),
    Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), atan(Mul(Symbol('a'), x)))
    NEW_INTEGRAL_16 = Mul(Integer(-1), Symbol('a'), dx, Pow(x,
    Add(Symbol('n'), Integer(1))), Pow(Add(Symbol('n'), Integer(1)), Integer(-1)),
    Pow(Add(Mul(Pow(Symbol('a'), Integer(2)), Pow(x, Integer(2))), Integer(1)),
    Integer(-1)))
    RECURSIVE.append(RECURSIVE_16)
    TEXT.append(TEXT_16)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_16)
    NEW_INTEGRAL.append(NEW_INTEGRAL_16)
    HINT_16 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}]]
    HINTS.append(HINT_16)
    EXCEPTION_16 = []
    EXCEPTIONS.append(EXCEPTION_16)
    RECURSIVE_17 = Mul(dx, Pow(x, Symbol('n')), exp(Mul(Symbol('a'), x)))
    TEXT_17 = "Some text"
    PARTIAL_SOLVE_17 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(x,
    Symbol('n')), exp(Mul(Symbol('a'), x)))
    NEW_INTEGRAL_17 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)), dx,
    Symbol('n'), Pow(x, Add(Symbol('n'), Integer(-1))), exp(Mul(Symbol('a'), x)))
    RECURSIVE.append(RECURSIVE_17)
    TEXT.append(TEXT_17)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_17)
    NEW_INTEGRAL.append(NEW_INTEGRAL_17)
    HINT_17 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}]]
    HINTS.append(HINT_17)
    EXCEPTION_17 = []
    EXCEPTIONS.append(EXCEPTION_17)
    RECURSIVE_18 = Mul(Pow(Symbol('b'), Mul(Symbol('a'), x)), dx, Pow(x,
    Symbol('n')))
    TEXT_18 = "Some text"
    PARTIAL_SOLVE_18 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Symbol('b'),
    Mul(Symbol('a'), x)), Pow(x, Symbol('n')), Pow(log(Symbol('b')), Integer(-1)))
    NEW_INTEGRAL_18 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
    Pow(Symbol('b'), Mul(Symbol('a'), x)), dx, Symbol('n'), Pow(x, Add(Symbol('n'),
    Integer(-1))), Pow(log(Symbol('b')), Integer(-1)))
    RECURSIVE.append(RECURSIVE_18)
    TEXT.append(TEXT_18)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_18)
    NEW_INTEGRAL.append(NEW_INTEGRAL_18)
    HINT_18 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}]]
    HINTS.append(HINT_18)
    EXCEPTION_18 = []
    EXCEPTIONS.append(EXCEPTION_18)

```

```

RECURSIVE_19 = Mul(dx, Pow(x, Symbol('n')), Pow(log(Mul(Symbol('a'), x)),
Symbol('b'))))
    TEXT_19 = "Some text"
    PARTIAL_SOLVE_19 = Mul(Pow(x, Add(Symbol('n'), Integer(1))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), Pow(log(Mul(Symbol('a'),
x)), Symbol('b'))))
    NEW_INTEGRAL_19 = Mul(Integer(-1), Symbol('b'), dx, Pow(x,
Symbol('n')), Pow(Add(Symbol('n'), Integer(1)), Integer(-1)),
Pow(log(Mul(Symbol('a'), x)), Add(Symbol('b'), Integer(-1))))
    RECURSIVE.append(RECURSIVE_19)
    TEXT.append(TEXT_19)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_19)
    NEW_INTEGRAL.append(NEW_INTEGRAL_19)
    HINT_19 = [[{"symbol":Symbol('n'),
"value":1}, {"symbol":Symbol('a'), "value":1}, {"symbol":Symbol('b'),
"value":1}], [{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('b'),
"value":1}], [{"symbol":Symbol('a'), "value":1}, {"symbol":Symbol('b'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('b'), "value":1}]]
    HINTS.append(HINT_19)
    EXCEPTION_19 = []
    EXCEPTIONS.append(EXCEPTION_19)

    RECURSIVE_2 = Mul(dx, Pow(x, Integer(-2)), Pow(Add(Mul(Symbol('a'),
x), Symbol('b')), Rational(-1, 2)))
    TEXT_2 = "Some text"
    PARTIAL_SOLVE_2 = Mul(Integer(-1), Pow(Symbol('b'), Integer(-1)),
Pow(x, Integer(-1)), Pow(Add(Mul(Symbol('a'), x), Symbol('b')), Rational(1,
2)))
    NEW_INTEGRAL_2 = Mul(Integer(-1), Rational(1, 2), Symbol('a'),
Pow(Symbol('b'), Integer(-1)), dx, Pow(x, Integer(-1)),
Pow(Add(Mul(Symbol('a'), x), Symbol('b')), Rational(-1, 2)))
    RECURSIVE.append(RECURSIVE_2)
    TEXT.append(TEXT_2)
    PARTIAL_SOLVE.append(PARTIAL_SOLVE_2)
    NEW_INTEGRAL.append(NEW_INTEGRAL_2)
    HINT_2 = [[{"symbol":Symbol('a'), "value":1}]]
    HINTS.append(HINT_2)
    EXCEPTION_2 = []
    EXCEPTIONS.append(EXCEPTION_2)

```

```

RECURSIVE_20 = Mul(dx, Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1),
Pow(x, Integer(2)))), Mul(Rational(1, 2), Symbol('n'))))
TEXT_20 = "Some text"
PARTIAL_SOLVE_20 = Mul(Add(Mul(Integer(-1), Symbol('a')), x),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), Pow(Add(Mul(Integer(2), Symbol('a')),
x), Mul(Integer(-1), Pow(x, Integer(2)))), Mul(Rational(1, 2), Symbol('n'))))
NEW_INTEGRAL_20 = Mul(Pow(Symbol('a'), Integer(2)), dx, Symbol('n'),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)), Pow(Add(Mul(Integer(2), Symbol('a')),
x), Mul(Integer(-1), Pow(x, Integer(2)))), Add(Mul(Rational(1, 2), Symbol('n')),
Integer(-1))))
RECURSIVE.append(RECURSIVE_20)
TEXT.append(TEXT_20)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_20)
NEW_INTEGRAL.append(NEW_INTEGRAL_20)
HINT_20 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_20)
EXCEPTION_20 = []
EXCEPTIONS.append(EXCEPTION_20)
RECURSIVE_21 = Mul(dx, Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-
1), Pow(x, Integer(2)))), Mul(Integer(-1), Rational(1, 2), Symbol('n'))))
TEXT_21 = "Some text"
PARTIAL_SOLVE_21 = Mul(Pow(Symbol('a'), Integer(-2)), Add(Mul(Integer(-1),
Symbol('a')), x), Pow(Add(Symbol('n'), Integer(-2)), Integer(-1)),
Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2)))),
Add(Integer(1), Mul(Integer(-1), Rational(1, 2), Symbol('n')))))
NEW_INTEGRAL_21 = Mul(Pow(Symbol('a'), Integer(-2)), dx, Add(Symbol('n'),
Integer(-3)), Pow(Add(Symbol('n'), Integer(-2)), Integer(-1)),
Pow(Add(Mul(Integer(2), Symbol('a')), x), Mul(Integer(-1), Pow(x, Integer(2)))),
Add(Integer(1), Mul(Integer(-1), Rational(1, 2), Symbol('n')))))
RECURSIVE.append(RECURSIVE_21)
TEXT.append(TEXT_21)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_21)
NEW_INTEGRAL.append(NEW_INTEGRAL_21)
HINT_21 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_21)
EXCEPTION_21 = []
EXCEPTIONS.append(EXCEPTION_21)
RECURSIVE_22 = Mul(dx, Pow(sinh(Mul(Symbol('a')), x)), Symbol('n'))
TEXT_22 = "Some text"
PARTIAL_SOLVE_22 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(Symbol('n'),
Integer(-1)), Pow(sinh(Mul(Symbol('a')), x)), Add(Symbol('n'), Integer(-1))),
cosh(Mul(Symbol('a')), x)))
NEW_INTEGRAL_22 = Mul(Integer(-1), dx, Pow(Symbol('n'), Integer(-1)),
Add(Symbol('n'), Integer(-1)), Pow(sinh(Mul(Symbol('a')), x)), Add(Symbol('n'),
Integer(-2))))
RECURSIVE.append(RECURSIVE_22)
TEXT.append(TEXT_22)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_22)
NEW_INTEGRAL.append(NEW_INTEGRAL_22)
HINT_22 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_22)
EXCEPTION_22 = []
EXCEPTIONS.append(EXCEPTION_22)

```

```

RECURSIVE_23 = Mul(dx, Pow(cosh(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_23 = "Some text"
PARTIAL_SOLVE_23 = Mul(Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), sinh(Mul(Symbol('a'), x)),
Pow(cosh(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_23 = Mul(dx, Pow(Symbol('n'), Integer(-1)),
Add(Symbol('n'), Integer(-1)), Pow(cosh(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_23)
TEXT.append(TEXT_23)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_23)
NEW_INTEGRAL.append(NEW_INTEGRAL_23)
HINT_23 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_23)
EXCEPTION_23 = []
EXCEPTIONS.append(EXCEPTION_23)
RECURSIVE_24 = Mul(dx, Pow(x, Symbol('n')), sinh(Mul(Symbol('a'),
x)))
TEXT_24 = "Some text"
PARTIAL_SOLVE_24 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(x,
Symbol('n')), cosh(Mul(Symbol('a'), x)))
NEW_INTEGRAL_24 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
dx, Symbol('n'), Pow(x, Add(Symbol('n'), Integer(-1))),
cosh(Mul(Symbol('a'), x)))
RECURSIVE.append(RECURSIVE_24)
TEXT.append(TEXT_24)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_24)
NEW_INTEGRAL.append(NEW_INTEGRAL_24)
HINT_24 = [[{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
HINTS.append(HINT_24)
EXCEPTION_24 = []
EXCEPTIONS.append(EXCEPTION_24)
RECURSIVE_25 = Mul(dx, Pow(x, Symbol('n')), cosh(Mul(Symbol('a'),
x)))
TEXT_25 = "Some text"
PARTIAL_SOLVE_25 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(x,
Symbol('n')), sinh(Mul(Symbol('a'), x)))
NEW_INTEGRAL_25 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
dx, Symbol('n'), Pow(x, Add(Symbol('n'), Integer(-1))),
sinh(Mul(Symbol('a'), x)))
RECURSIVE.append(RECURSIVE_25)
TEXT.append(TEXT_25)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_25)
NEW_INTEGRAL.append(NEW_INTEGRAL_25)
HINT_25 = [[{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}, [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}]]
HINTS.append(HINT_25)
EXCEPTION_25 = []
EXCEPTIONS.append(EXCEPTION_25)

```



```

RECURSIVE_26 = Mul(dx, Pow(tanh(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_26 = "Some text"
PARTIAL_SOLVE_26 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(tanh(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_26 = Mul(dx, Pow(tanh(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_26)
TEXT.append(TEXT_26)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_26)
NEW_INTEGRAL.append(NEW_INTEGRAL_26)
HINT_26 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_26)
EXCEPTION_26 = []
EXCEPTIONS.append(EXCEPTION_26)

RECURSIVE_27 = Mul(dx, Pow(coth(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_27 = "Some text"
PARTIAL_SOLVE_27 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(coth(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_27 = Mul(dx, Pow(coth(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_27)
TEXT.append(TEXT_27)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_27)
NEW_INTEGRAL.append(NEW_INTEGRAL_27)
HINT_27 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_27)
EXCEPTION_27 = []
EXCEPTIONS.append(EXCEPTION_27)

RECURSIVE_28 = Mul(dx, Pow(sech(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_28 = "Some text"
PARTIAL_SOLVE_28 = Mul(Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), tanh(Mul(Symbol('a'), x)),
Pow(sech(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-2))))
NEW_INTEGRAL_28 = Mul(dx, Add(Symbol('n'), Integer(-2)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(sech(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_28)
TEXT.append(TEXT_28)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_28)
NEW_INTEGRAL.append(NEW_INTEGRAL_28)
HINT_28 = [{"symbol":Symbol('a'), "value":1}]
HINTS.append(HINT_28)
EXCEPTION_28 = []
EXCEPTIONS.append(EXCEPTION_28)

```



```

RECURSIVE_29 = Mul(dx, Pow(csch(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_29 = "Some text"
PARTIAL_SOLVE_29 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), coth(Mul(Symbol('a'), x)),
Pow(csch(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-2))))
NEW_INTEGRAL_29 = Mul(Integer(-1), dx, Add(Symbol('n'), Integer(-2)),
Pow(Add(Symbol('n'), Integer(-1)), Integer(-1)), Pow(csch(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_29)
TEXT.append(TEXT_29)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_29)
NEW_INTEGRAL.append(NEW_INTEGRAL_29)
HINT_29 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_29)
EXCEPTION_29 = []
EXCEPTIONS.append(EXCEPTION_29)
RECURSIVE_3 = Mul(dx, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Mul(Rational(1, 2), Symbol('n'))))
TEXT_3 = "Some text"
PARTIAL_SOLVE_3 = Mul(x, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Mul(Rational(1, 2), Symbol('n'))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)))
NEW_INTEGRAL_3 = Mul(Integer(-1), Pow(Symbol('a'), Integer(2)), dx,
Symbol('n'), Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x,
Integer(2))), Add(Mul(Rational(1, 2), Symbol('n')), Integer(-1))),
Pow(Add(Symbol('n'), Integer(1)), Integer(-1)))
RECURSIVE.append(RECURSIVE_3)
TEXT.append(TEXT_3)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_3)
NEW_INTEGRAL.append(NEW_INTEGRAL_3)
HINT_3 = [[]]
HINTS.append(HINT_3)
EXCEPTION_3 = []
EXCEPTIONS.append(EXCEPTION_3)
RECURSIVE_4 = Mul(dx, Pow(Add(Mul(Integer(-1), Pow(Symbol('a'),
Integer(2))), Pow(x, Integer(2))), Mul(Integer(-1), Rational(1, 2),
Symbol('n'))))
TEXT_4 = "Some text"
PARTIAL_SOLVE_4 = Mul(Pow(Symbol('a'), Integer(-2)), x,
Pow(Add(Integer(2), Mul(Integer(-1), Symbol('n'))), Integer(-1)),
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))),
Add(Integer(1), Mul(Integer(-1), Rational(1, 2), Symbol('n')))))
NEW_INTEGRAL_4 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-2)), dx,
Pow(Add(Mul(Integer(-1), Pow(Symbol('a'), Integer(2))), Pow(x, Integer(2))),
Add(Integer(1), Mul(Integer(-1), Rational(1, 2), Symbol('n'))), Add(Symbol('n'),
Integer(-3)), Pow(Add(Symbol('n'), Integer(-2)), Integer(-1)))
RECURSIVE.append(RECURSIVE_4)
TEXT.append(TEXT_4)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_4)
NEW_INTEGRAL.append(NEW_INTEGRAL_4)
HINT_4 = [[]]
HINTS.append(HINT_4)
EXCEPTION_4 = []
EXCEPTIONS.append(EXCEPTION_4)

```

```

RECURSIVE_5 = Mul(dx, Pow(sin(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_5 = "Some text"
PARTIAL_SOLVE_5 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), Pow(sin(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-1))), cos(Mul(Symbol('a'), x)))
NEW_INTEGRAL_5 = Mul(dx, Pow(Symbol('n'), Integer(-1)),
Add(Symbol('n'), Integer(-1)), Pow(sin(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_5)
TEXT.append(TEXT_5)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_5)
NEW_INTEGRAL.append(NEW_INTEGRAL_5)
HINT_5 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_5)
EXCEPTION_5 = []
EXCEPTIONS.append(EXCEPTION_5)
RECURSIVE_6 = Mul(dx, Pow(cos(Mul(Symbol('a'), x)), Symbol('n')))
TEXT_6 = "Some text"
PARTIAL_SOLVE_6 = Mul(Pow(Symbol('a'), Integer(-1)),
Pow(Symbol('n'), Integer(-1)), sin(Mul(Symbol('a'), x)),
Pow(cos(Mul(Symbol('a'), x)), Add(Symbol('n'), Integer(-1))))
NEW_INTEGRAL_6 = Mul(dx, Pow(Symbol('n'), Integer(-1)),
Add(Symbol('n'), Integer(-1)), Pow(cos(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))))
RECURSIVE.append(RECURSIVE_6)
TEXT.append(TEXT_6)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_6)
NEW_INTEGRAL.append(NEW_INTEGRAL_6)
HINT_6 = [[{"symbol":Symbol('a'), "value":1}]]
HINTS.append(HINT_6)
EXCEPTION_6 = []
EXCEPTIONS.append(EXCEPTION_6)
RECURSIVE_8 = Mul(dx, Pow(x, Symbol('n')), sin(Mul(Symbol('a'), x)))
TEXT_8 = "Some text"
PARTIAL_SOLVE_8 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(x, Symbol('n')), cos(Mul(Symbol('a'), x)))
NEW_INTEGRAL_8 = Mul(Pow(Symbol('a'), Integer(-1)), dx,
Symbol('n'), Pow(x, Add(Symbol('n'), Integer(-1))), cos(Mul(Symbol('a'),
x)))
RECURSIVE.append(RECURSIVE_8)
TEXT.append(TEXT_8)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_8)
NEW_INTEGRAL.append(NEW_INTEGRAL_8)
HINT_8 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}]]
HINTS.append(HINT_8)
EXCEPTION_8 = []
EXCEPTIONS.append(EXCEPTION_8)

```

```

RECURSIVE_7 = Mul(dx, Pow(sin(Mul(Symbol('a'), x)), Symbol('n')),
Pow(cos(Mul(Symbol('a'), x)), Symbol('c')))
TEXT_7 = "Some text"
PARTIAL_SOLVE_7 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
Pow(Add(Symbol('c'), Symbol('n')), Integer(-1)), Pow(sin(Mul(Symbol('a'),
x)), Add(Symbol('n'), Integer(-1))), Pow(cos(Mul(Symbol('a'), x)),
Add(Symbol('c'), Integer(1))))
NEW_INTEGRAL_7 = Mul(dx, Pow(Add(Symbol('c'), Symbol('n')),
Integer(-1)), Add(Symbol('n'), Integer(-1)), Pow(sin(Mul(Symbol('a'), x)),
Add(Symbol('n'), Integer(-2))), Pow(cos(Mul(Symbol('a'), x)), Symbol('c')))
RECURSIVE.append(RECURSIVE_7)
TEXT.append(TEXT_7)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_7)
NEW_INTEGRAL.append(NEW_INTEGRAL_7)
HINT_7 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('c'),
"value":1}], [{"symbol":Symbol('a'), "value":1}, {"symbol":Symbol('c'),
"value":1}], [{"symbol":Symbol('a'), "value":1}, {"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('c'), "value":1}], [{"symbol":Symbol('n'),
"value":1}], [{"symbol":Symbol('a'), "value":1}], [{"symbol":Symbol('a'),
"value":1}, {"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('c'),
"value":1}]]
HINTS.append(HINT_7)
EXCEPTION_7 = []
EXCEPTIONS.append(EXCEPTION_7)
RECURSIVE_9 = Mul(dx, Pow(x, Symbol('n')), cos(Mul(Symbol('a'), x)))
TEXT_9 = "Some text"
PARTIAL_SOLVE_9 = Mul(Pow(Symbol('a'), Integer(-1)), Pow(x,
Symbol('n')), sin(Mul(Symbol('a'), x)))
NEW_INTEGRAL_9 = Mul(Integer(-1), Pow(Symbol('a'), Integer(-1)),
dx, Symbol('n'), Pow(x, Add(Symbol('n'), Integer(-1))),
sin(Mul(Symbol('a'), x)))
RECURSIVE.append(RECURSIVE_9)
TEXT.append(TEXT_9)
PARTIAL_SOLVE.append(PARTIAL_SOLVE_9)
NEW_INTEGRAL.append(NEW_INTEGRAL_9)
HINT_9 = [[{"symbol":Symbol('n'), "value":1}, {"symbol":Symbol('a'),
"value":1}], [{"symbol":Symbol('n'), "value":1}], [{"symbol":Symbol('a'),
"value":1}]]
HINTS.append(HINT_9)
EXCEPTION_9 = []
EXCEPTIONS.append(EXCEPTION_9)

```

```

index = 0
    for HINT in HINTS:
        for STATE in HINT:
            if len(STATE) > 0:
                new_bas = RECURSIVE[index]
                new_solve = PARTIAL_SOLVE[index]
                new_int = NEW_INTEGRAL[index]
                for CHANGE in STATE:
                    new_bas =
new_bas.subs(CHANGE['symbol'], CHANGE['value'])
                    new_solve =
new_solve.subs(CHANGE['symbol'], CHANGE['value'])
                    new_int =
new_int.subs(CHANGE['symbol'], CHANGE['value'])
                    RECURSIVE.append(new_bas)
                    PARTIAL_SOLVE.append(new_solve)
                    NEW_INTEGRAL.append(new_int)
            index = index + 1

```

### 3. Actualizador de Integrales

Realiza una solicitud al servidor de flask que rescribe el catálogo de integrales tanto atómicas como recursivas. Se corre como un cliente aislado con esta única función.

```
const request = require('request-promise');
const express = require('express');
const IP_SERVER = "https://pwa-ode-project-server.herokuapp.com";
const updateAtmIntegrals = () =>
{
  const options =
  {
    method: "POST",
    uri: `${IP_SERVER}/update/atm`,
    body: {},
    json: true
  };

  const res = request(options);
  res.then(status =>
  {
    if (status === 200)
    {
      setTimeout(updateRecIntegrals, 60_000);
      console.log("Atomic Integrals were Updated")
    }
  })
}

const updateRecIntegrals = () =>
{
  const options =
  {
    method: "POST",
    uri: `${IP_SERVER}/update/rec`,
    body: {},
    json: true
  };

  const res = request(options);
  res.then(status =>
  {
    if (status === 200)
    {
      console.log("Recursive Integrals were Updated")
      setTimeout(updateAtmIntegrals, 86_400_000);
    }
  })
}
```

## 4. Escritor de catálogos

Rescribe el catálogo de integrales tanto atómicas como recursivas.

```
def write_atm_integrals(db):
    filepath = f'integrals/atomic_integrals_temp.py'
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")

    with open(filepath, "w") as f:
        f.write(f"# Updated in {dt_string}\n\n")
        f.write("from sympy import *\n")
        f.write("def build_integrals(symbol):\n")
        f.write("\tdx = Symbol('d' + str(symbol))\n")
        f.write("\tx = Symbol(str(symbol))\n")
        f.write("\tglobal BASIC\n")
        f.write("\tglobal TEXT\n")
        f.write("\tglobal SOLVE\n")
        f.write("\tglobal EXCEPTIONS\n")
        f.write("\tBASIC = []\n")
        f.write("\tTEXT = []\n")
        f.write("\tSOLVE = []\n")
        f.write("\tEXCEPTIONS = []\n")
        f.write("\tHINTS = []\n\n")

    cat_ref = db.collection('atomic')
    int_docs = cat_ref.stream()
    for int_doc in int_docs:
        int_dict = int_doc.to_dict()
        basic_sym = int_dict['basic'].replace("Symbol('dx')", "dx")
        basic_sym = basic_sym.replace("Symbol('x')", "x")
        f.write(f"\tBASIC_{int_doc.id} = {basic_sym}\n")
        text_sym = int_dict['text']
        f.write(f"\tTEXT_{int_doc.id} = \"{text_sym}\"\n")

        solve_sym = int_dict['solve'].replace("Symbol('x')", "x")
        f.write(f"\tSOLVE_{int_doc.id} = {solve_sym}\n")
        f.write(f"\tBASIC.append(BASIC_{int_doc.id})\n")
        f.write(f"\tTEXT.append(TEXT_{int_doc.id})\n")
        f.write(f"\tSOLVE.append(SOLVE_{int_doc.id})\n")

        f.write(f"\tHINT_{int_doc.id} = [")
        hint_ref = cat_ref.document(int_doc.id).collection('hints')
        hint_docs = hint_ref.stream()

        hint_text = ""
```

```

for hint_doc in hint_docs:
    hint_text = hint_text + "["
    changes_ref =
hint_ref.document(hint_doc.id).collection('changes')
    changes_doc = changes_ref.stream()

    change_text = ""
    for change_doc in changes_doc:
        change_dict = change_doc.to_dict()

        change_text = change_text + "{"
        change_text = change_text + '"symbol":'
        change_text = change_text + change_dict['symbol']
        change_text = change_text + ", "
        change_text = change_text + '"value":'
        change_text = change_text + change_dict['value']
        change_text = change_text + "},"

    change_text = change_text[:-1]
    hint_text = hint_text + change_text
    hint_text = hint_text + "],"
else:
    if hint_text == "":
        f.write("[]")

if hint_text != "":
    hint_text = hint_text[:-1]
    f.write(hint_text)

f.write(f"]\n")
f.write(f"\tHINTS.append(HINT_{int_doc.id})\n")

f.write(f"\tEXCEPTION_{int_doc.id} = [")
exception_ref =
cat_ref.document(int_doc.id).collection('exceptions')
exception_docs = exception_ref.stream()

exception_text = ""
for exception_doc in exception_docs:
    exception_dict = exception_doc.to_dict()

    exception_text = exception_text + "{"
    exception_text = exception_text + '"symbol":'
    exception_text = exception_text + exception_dict['symbol']
    exception_text = exception_text + ", "
    exception_text = exception_text + '"value":'
    exception_text = exception_text + exception_dict['value']
    exception_text = exception_text + ", "
    exception_text = exception_text + '"type":'
    exception_text = exception_text + '"" +
exception_dict['type'] + '""
    exception_text = exception_text + "},"
if exception_text != "":
    exception_text = exception_text[:-1]

```

```

f.write(f"\tindex = 0\n")
    f.write(f"\tfor HINT in HINTS:\n")
    f.write(f"\t\tfor STATE in HINT:\n")
    f.write(f"\t\t\tif len(STATE) > 0:\n")
    f.write(f"\t\t\t\tnew_bas = BASIC[index]\n")
    f.write(f"\t\t\t\tnew_solve = SOLVE[index]\n")
    f.write(f"\t\t\t\tfor CHANGE in STATE:\n")
    f.write(f"\t\t\t\t\tnew_bas = new_bas.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\t\tnew_solve = new_solve.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\tBASIC.append(new_bas)\n")
    f.write(f"\t\t\t\tSOLVE.append(new_solve)\n")
    f.write(f"\t\tindex = index + 1\n")

    move(f'integrals/atomic_integrals_temp.py',
f'integrals/atomic_integrals.py')

def write_rec_integrals(db):
    filepath = f'integrals/recursive_integrals_temp.py'
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")

    with open(filepath, "w") as f:
        f.write(f"# Updated in {dt_string}\n\n")
        f.write("from sympy import *\n")
        f.write("def build_integrals(symbol):\n")
        f.write("\tdx = Symbol('d' + str(symbol))\n")
        f.write("\tx = Symbol(str(symbol))\n")
        f.write("\tglobal RECURSIVE\n")
        f.write("\tglobal TEXT\n")
        f.write("\tglobal PARTIAL_SOLVE\n")
        f.write("\tglobal NEW_INTEGRAL\n")
        f.write("\tglobal EXCEPTIONS\n")
        f.write("\tRECURSIVE = []\n")
        f.write("\tTEXT = []\n")
        f.write("\tPARTIAL_SOLVE = []\n")
        f.write("\tNEW_INTEGRAL = []\n")
        f.write("\tEXCEPTIONS = []\n")
        f.write("\tHINTS = []\n\n")
    cat_ref = db.collection('recursive')
    int_docs = cat_ref.stream()
    for int_doc in int_docs:
        int_dict = int_doc.to_dict()

        recursive_sym = int_dict['recursive'].replace("Symbol('dx')",
"dx")

        recursive_sym = recursive_sym.replace("Symbol('x')", "x")
        f.write(f"\tRECURSIVE_{int_doc.id} = {recursive_sym}\n")

        text_sym = int_dict['text']
        f.write(f"\tTEXT_{int_doc.id} = \"{text_sym}\"\n")

        solve_sym = int_dict['solve'].replace("Symbol('x')", "x")
        f.write(f"\tPARTIAL_SOLVE_{int_doc.id} = {solve_sym}\n")

```



```

f.write(f"\tindex = 0\n")
    f.write(f"\tfor HINT in HINTS:\n")
    f.write(f"\t\tfor STATE in HINT:\n")
    f.write(f"\t\t\tif len(STATE) > 0:\n")
    f.write(f"\t\t\t\tnew_bas = BASIC[index]\n")
    f.write(f"\t\t\t\tnew_solve = SOLVE[index]\n")
    f.write(f"\t\t\t\tfor CHANGE in STATE:\n")
    f.write(f"\t\t\t\t\tnew_bas = new_bas.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\t\tnew_solve = new_solve.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\tBASIC.append(new_bas)\n")
    f.write(f"\t\t\t\tSOLVE.append(new_solve)\n")
    f.write(f"\t\tindex = index + 1\n")

    move(f'integrals/atomic_integrals_temp.py',
f'integrals/atomic_integrals.py')

def write_rec_integrals(db):
    filepath = f'integrals/recursive_integrals_temp.py'
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")

    with open(filepath, "w") as f:
        f.write(f"# Updated in {dt_string}\n\n")
        f.write("from sympy import *\n")
        f.write("def build_integrals(symbol):\n")
        f.write("\tdx = Symbol('d' + str(symbol))\n")
        f.write("\tx = Symbol(str(symbol))\n")
        f.write("\tglobal RECURSIVE\n")
        f.write("\tglobal TEXT\n")
        f.write("\tglobal PARTIAL_SOLVE\n")
        f.write("\tglobal NEW_INTEGRAL\n")
        f.write("\tglobal EXCEPTIONS\n")
        f.write("\tRECURSIVE = []\n")
        f.write("\tTEXT = []\n")
        f.write("\tPARTIAL_SOLVE = []\n")
        f.write("\tNEW_INTEGRAL = []\n")
        f.write("\tEXCEPTIONS = []\n")
        f.write("\tHINTS = []\n\n")
    cat_ref = db.collection('recursive')
    int_docs = cat_ref.stream()
    for int_doc in int_docs:
        int_dict = int_doc.to_dict()

        recursive_sym = int_dict['recursive'].replace("Symbol('dx')",
"dx")

        recursive_sym = recursive_sym.replace("Symbol('x')", "x")
        f.write(f"\tRECURSIVE_{int_doc.id} = {recursive_sym}\n")

        text_sym = int_dict['text']
        f.write(f"\tTEXT_{int_doc.id} = \"{text_sym}\"\n")

        solve_sym = int_dict['solve'].replace("Symbol('x')", "x")
        f.write(f"\tPARTIAL_SOLVE_{int_doc.id} = {solve_sym}\n")

```

```

        new_integral_sym =
int_dict['new_integral'].replace("Symbol('dx')", "dx")
        new_integral_sym = new_integral_sym.replace("Symbol('x')", "x")
        f.write(f"\tNEW_INTEGRAL_{int_doc.id} = {new_integral_sym}\n")

        f.write(f"\tRECURSIVE.append(RECURSIVE_{int_doc.id})\n")
        f.write(f"\tTEXT.append(TEXT_{int_doc.id})\n")

f.write(f"\tPARTIAL_SOLVE.append(PARTIAL_SOLVE_{int_doc.id})\n")
        f.write(f"\tNEW_INTEGRAL.append(NEW_INTEGRAL_{int_doc.id})\n")

        f.write(f"\tHINT_{int_doc.id} = [")
        hint_ref = cat_ref.document(int_doc.id).collection('hints')
        hint_docs = hint_ref.stream()

        hint_text = ""
        for hint_doc in hint_docs:
            hint_text = hint_text + "["
            changes_ref =
hint_ref.document(hint_doc.id).collection('changes')
            changes_doc = changes_ref.stream()

            change_text = ""
            for change_doc in changes_doc:
                change_dict = change_doc.to_dict()

                change_text = change_text + "{"
                change_text = change_text + '"symbol":'
                change_text = change_text + change_dict['symbol']
                change_text = change_text + ", "
                change_text = change_text + '"value":'
                change_text = change_text + change_dict['value']
                change_text = change_text + "},"

            change_text = change_text[:-1]
            hint_text = hint_text + change_text
            hint_text = hint_text + "],"
        else:
            if hint_text == "":
                f.write("[]")

        if hint_text != "":
            hint_text = hint_text[:-1]
            f.write(hint_text)
f.write(f"]\n")
        f.write(f"\tHINTS.append(HINT_{int_doc.id})\n")

        f.write(f"\tEXCEPTION_{int_doc.id} = [")

```

```

exception_ref = cat_ref.document(int_doc.id).collection('exceptions')
exception_docs = exception_ref.stream()

exception_text = ""
for exception_doc in exception_docs:
    exception_dict = exception_doc.to_dict()

    exception_text = exception_text + "{"
    exception_text = exception_text + '"symbol":'
    exception_text = exception_text + exception_dict['symbol']
    exception_text = exception_text + ", "
    exception_text = exception_text + '"value":'
    exception_text = exception_text + exception_dict['value']
    exception_text = exception_text + ", "
    exception_text = exception_text + '"type":'
    exception_text = exception_text + '"' +
exception_dict['type'] + '"'
    exception_text = exception_text + "},"
    if exception_text != "":
        exception_text = exception_text[:-1]
f.write(f"{exception_text}]\n")
    f.write(f"\tEXCEPTIONS.append(EXCEPTION_{int_doc.id})\n\n")

    f.write(f"\tindex = 0\n")
    f.write(f"\tfor HINT in HINTS:\n")
    f.write(f"\t\tfor STATE in HINT:\n")
    f.write(f"\t\t\tif len(STATE) > 0:\n")
    f.write(f"\t\t\t\tnew_bas = RECURSIVE[index]\n")
    f.write(f"\t\t\t\tnew_solve = PARTIAL_SOLVE[index]\n")
    f.write(f"\t\t\t\tnew_int = NEW_INTEGRAL[index]\n")
    f.write(f"\t\t\t\tfor CHANGE in STATE:\n")
    f.write(f"\t\t\t\t\tnew_bas = new_bas.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\t\tnew_solve = new_solve.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\t\tnew_int = new_int.subs(CHANGE['symbol'],
CHANGE['value'])\n")
    f.write(f"\t\t\t\t\tRECURSIVE.append(new_bas)\n")
    f.write(f"\t\t\t\t\tPARTIAL_SOLVE.append(new_solve)\n")
    f.write(f"\t\t\t\t\tNEW_INTEGRAL.append(new_int)\n")
    f.write(f"\t\t\t\t\tindex = index + 1\n")

    move(f'integrals/recursive_integrals_temp.py',
f'integrals/recursive_integrals.py')

```