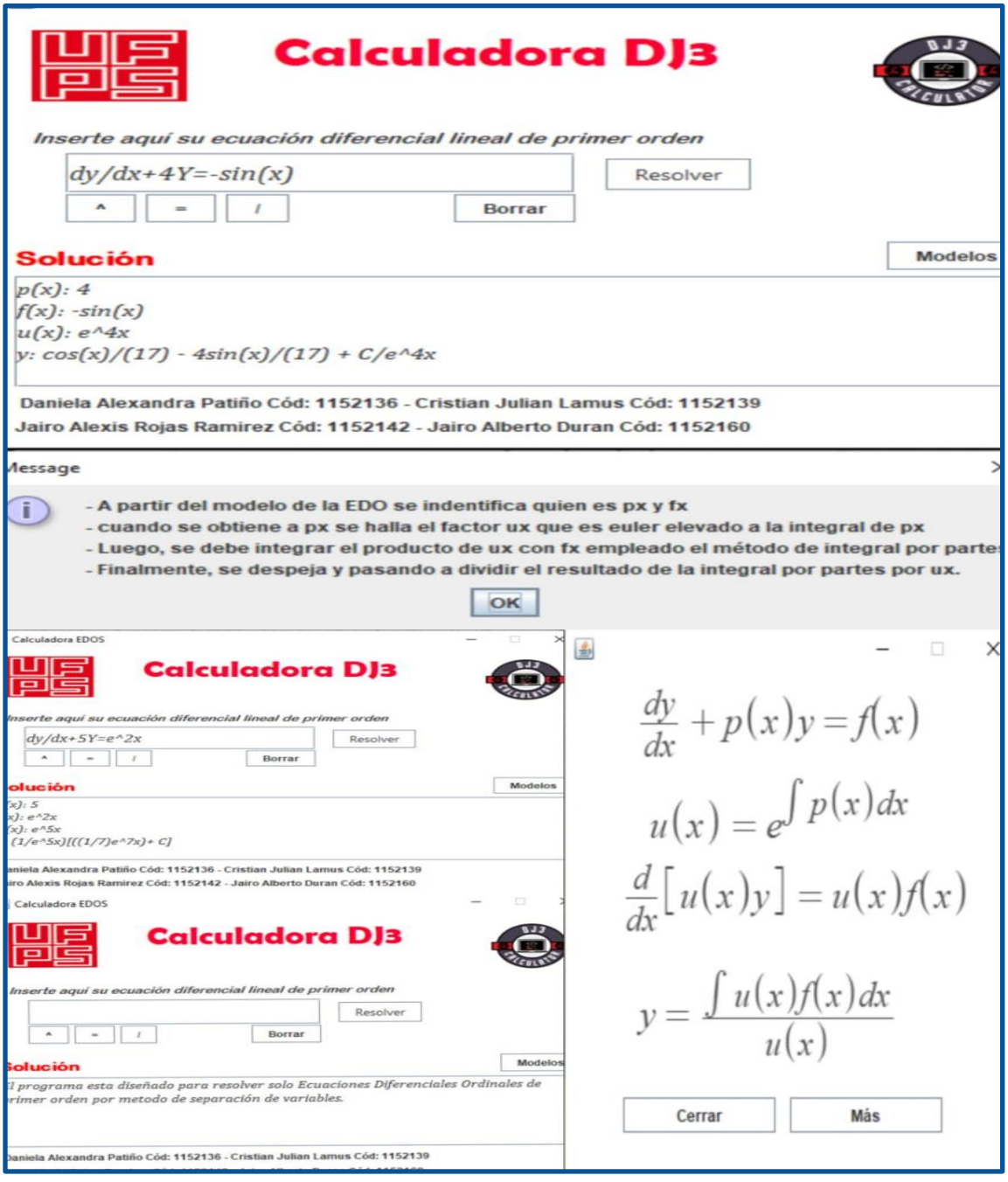
CALCULADORA

MATERIA: ECUACIONES DIFERENCIALES 1110401-C

CALCULADORA DE ECUACIONES

DIFERENCIALES

DJ3 CALCULATOR



Introducción

El objetivo de esta calculadora es proporcionar a los alumnos que se encuentren viendo ecuaciones diferenciales una herramienta que les permita comprobar por sí mismos los cálculos y operaciones de ecuaciones diferenciales por factor integrante siguiendo unas normas preestablecidas que se han realizado a mano. Tiene como objetivo facilitar el aprendizaje cuando se aborda el tema de ecuaciones diferenciales y no se tienen buenas bases, ya que aunque los alumnos aprenden pronto, es posible que no estén seguros de lo que hicieron o mantengan dudas de signos, expresiones tales como (Euler, logaritmos, exponentes negativos, fracciones y raíces, etc) por esto es común que muchos estudiantes se desanimen porque aunque sepan la forma y manera de resolver las ecuaciones es común que se cometan errores en operaciones con lo que el resultado no será el esperado.

La calculadora realiza el análisis de si la función que ingrese el usuario cumple con alguna de las formas establecidas, siguiente a esto empieza la solución de la ecuación diferencial, donde realiza el proceso de resolución de derivadas, integrales y el modelo matemático de solución, mostrando el resultado esperado.
El entorno de desarrollo utilizado de este proyecto es netbeans versión 8.2, lenguaje de programación java, jdk 17 y puede ser utilizada fácilmente como complemento en páginas web que incluyan unidades didácticas sobre ecuaciones o cualquier otro tema relacionado; para ello simplemente hay que incorporar en un marco o en una ventana el archivo "calculadora.html" con todas las imágenes utilizadas en ella.

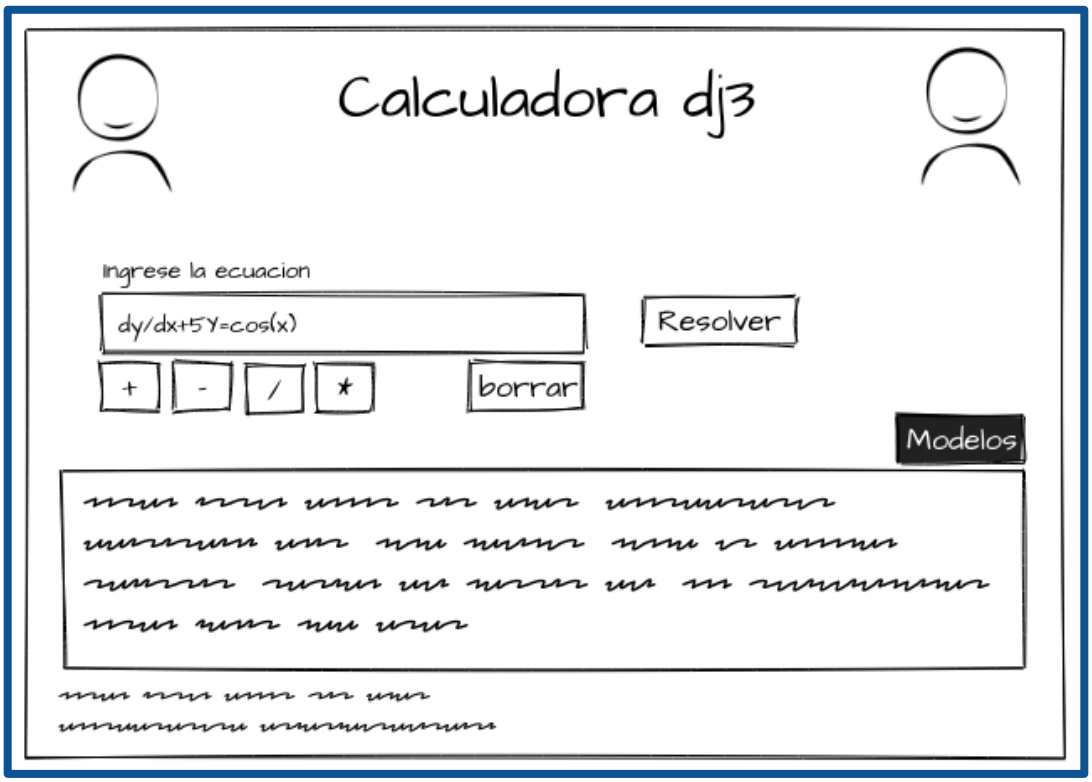
Esta calculadora ha sido nuestra práctica de nuestro curso de ecuaciones

diferenciales representando a nuestra carrera y la manera de incorporar ambas fue en la creación de una ayuda didáctica para esta materia por medio de las herramientas que nos brinda estudiar ingeniería de sistemas.

Tabla de Contenidos

1. Explicación del mockup

Interfaz gráfica de usuario.



En la realización del estilo de la calculadora, nos basamos en un modelo sencillo, tipo minimalista, siendo esencial y fácil de utilizar. Como se puede observar se tiene una pantalla con un input para colocar la ecuación que se quiere resolver, el botón de borrado, botones de apoyo para insertar símbolos de utilidad para escribir la ecuación, botón de modelos el cual nos va suministrar información de la calculadora y que modelos son los que se evalúan para resolverla y un campo de texto donde se dará el resultado de la ecuación.

Como desarrolladores delimitamos los alcances que debe tener el software.

2. Explicación del entorno de desarrollo, y librerías

El entorno de desarrollo utilizado para este proyecto es netbeans versión 8.2

Lenguaje de programación java, jdk 17.

Paradigma de programación orientada a objetos.

Se utilizó una serie de librerías para la parte matemática, para las derivadas se usó la librería jep (Java Equation Parse) Análisis de ecuaciones de Java.

En la parte de integrales, desarrollamos el código desde cero por el grupo de trabajo, ya que no había una librería que nos permitiera obtener respuesta a los distintos tipos de integrales y dar una solución efectiva a las ecuaciones planteadas.

Para la interfaz gráfica, trabajamos con el paquete estándar de Java, que nos proporciona la biblioteca swing para la creación de interfaces gráficas de usuario.

No hizo falta importar ningún fichero adicional en nuestro proyecto.

3. Explicación del código fuente de la calculadora

Métodos de la clase integral:

```
private String integralDirecta(String fx) {
if (fx.contains("e^")) {
    if (!fx.equals("e^x")) {
        int k = Integer.parseInt(fx.substring(fx.indexOf("^") + 1, fx.length() - 1));
        if (k == 0) {
            fx = "x";
        } else {
            fx = "(1/" + k + ")e^" + k + "x";
        }
    }
} else if (fx.equals("sin(x)")) {
    fx = "-cos(x)";
} else if (fx.equals("cos(x)")) {
    fx = "sin(x)";
} else if (fx.equals("-sin(x)")) {
    fx = "cos(x)";
} else if (fx.equals("-cos(x)")) {
    fx = "-sin(x)";
} else if (fx.contains("ln(x)")) {
    fx = "xln-x";
```

```
   } else if (fx.contains("x^")) {
       String coeficiente = fx.substring(0, fx.indexOf("x"));
       int n = Integer.parseInt(fx.substring(fx.indexOf("^") + 1, fx.length()));
       if (n == -1) {
           fx = coeficiente + "ln(x)";
       } else {
           fx = coeficiente + "(1/" + (n + 1) + ")x^" + (n + 1);
       }
   } else if (fx.equals("x")) {
       fx = "(1/2)x^2";
   } else if (fx.equals("1")) {
       fx = "x";
   } else if (fx.contains("/x")) {
       fx = fx.substring(0, fx.indexOf("/")) + "ln(x)";
   } else if (!fx.contains("x")) {
       fx = fx + "x";
   } else if (fx.contains("x") && !fx.equals("x")) {
       fx = "(1/2)" + fx.substring(0, fx.indexOf("x")) + "x^2";
   } else {
       fx = "Nada";
   }
   return fx;
}
```

La creación de la clase Integral comienza por un método especializado en integrales básicas faltos a una libreria que las solucione, `integralDirecta(fx)` verifica con una serie de condicionales para identificar los caracteres de la ecuación, este recibe un `String fx` con la función a integrar.

```
        static int ILATE(String fx) {
        int n = 0;
        String function = "";
        if (fx.contains("arc")) {
            function = "inversa";
            n = 5;
        } else if (fx.contains("ln")) {
            function = "logaritmica";
            n = 4;
        } else if ((fx.contains("x") || fx.contains("x^")) && (!(fx.contains("sin") ||
fx.contains("cos") || fx.contains("tan") || fx.contains("sec") || fx.contains("cot")
|| fx.contains("csc") || fx.contains("ln") || fx.contains("e^")))) {
            function = "algebraica";
            n = 3;
        } else if (fx.contains("sin") || fx.contains("cos") || fx.contains("tan") ||
fx.contains("sec") || fx.contains("cot") || fx.contains("csc")) {
            function = "trigonometrica";
            n = 2;
        } else if (fx.contains("e")) {
            function = "exponencial";
            n = 1;
        }
        return n;
    }
```

`ILATE(fx)` otro método sistemático, este algoritmo identifica caracteres de siguiendo la línea del uso de `.contains()`, en una serie de if y else que reúnen las clasificaciones dadas por el acrónimo ILATE, `ILATE(fx)` determina el tipo de ecuación mediante una comparación numérica como se ve en la propiedad `function`.

```
    public String integral_siclica(String fx, String gx) {
        String I = "";
        String ux = fx;
        int k = Integer.parseInt(ux.substring(ux.indexOf("^") + 1, ux.indexOf("x")));
        if (gx.equals("sin(x)")) {
            I = "-cos(x)/(" + (1 + k * k) + ") + " + k + "sin(x)/(" + (1 + k * k) + ")
+ C/e^" + k + "x";
        } else if (gx.equals("cos(x)")) {
            I = "sin(x)/(" + (1 + k * k) + ") + " + k + "cos(x)/(" + (1 + k * k) + ")
+ C/e^" + k + "x";
        } else if (gx.equals("-sin(x)")) {
            I = "cos(x)/(" + (1 + k * k) + ") - " + k + "sin(x)/(" + (1 + k * k) + ")
+ C/e^" + k + "x";
        } else if (gx.equals("-cos(x)")) {
            I = "-sin(x)/(" + (1 + k * k) + ") - " + k + "cos(x)/(" + (1 + k * k) + ")
+ C/e^" + k + "x";
        }
        return I;
    }
```

`integral_siclica()` es un algoritmo que realiza integrales del método por partes integrando dando énfasis principalmente a las funciones trigonométricas de manera cíclica, `integral_siclica()` retorna un `String I` con la solución.

```
private void componentes(String fx, String gx) {
    int f1 = ILATE(fx);
    int f2 = ILATE(gx);
    if (f1 > f2) {
        u = fx;
        dv = gx;
    } else if (f1 < f2) {
        u = gx;
        dv = fx;
    } else {
        if (fx.contains("/")) {
            u = gx;
            dv = fx;
        } else {
            u = fx;
            dv = gx;
        }
    }
    Derivada derivada = new Derivada();
    derivada.setFuncionADerivar(u);
    derivada.derivar();
    du = derivada.getFuncionDerivada(); //Se obtiene du
    v = integralDirecta(dv);
}//end void componentes.
```

`componentes(fx,gx)`, este algoritmo identifica caracteres usando `.contains()`, en este caso se especifica que parte de la ecuación será reemplazada usando `ILATE(fx)`, comparando los valores numéricos `f1` y `f2`. Usando métodos de la librería dJep y el método `integralDirecta(dv)`, se hallan *du*, *v*.

```
public String integral_algebraica_trigonometrica(String fx, String gx) {
    String ux = fx;
    String I = "";
    int n = Integer.parseInt(ux.substring(ux.indexOf("^") + 1, ux.indexOf("^") + 2));
    if (gx.equals("sin(x)")) {
        I = coseno_gxsinx(n) + " + " + seno_gxsinx(n) + "+ C";
    } else if (gx.equals("cos(x)")) {
        I = seno_gxcosx(n) + " + " + coseno_gxcosx(n) + "+ C";
    } else if (gx.equals("-cos(x)")) {
        I = "-" + seno_gxcosx(n) + " - " + coseno_gxcosx(n) + "+ C";
    } else if (gx.equals("-sin(x)")) {
        I = "-" + coseno_gxsinx(n) + " - " + seno_gxsinx(n) + "+ C";
    }
    return I;
}
```

`integral_algebraica_trigonometrica(fx,gx)` soluciona la forma de integrales cos(x).cos(x) teniendo en cuenta las pautas para esto, usando el método `coseno_gxcosx()` y sus variantes, que es donde se encuentran los pasos a realizar, retorna `I` como la solución.

`exponencial(gx,fx)` realiza las integrales que contengan *Euler(ℰ)*, luego de verificar que si este se encuentra contenido hace uso de `.substring` para obtener el dato del exponente, retorna `I`.

```
public String exponecial(String gx, String fx) {
        String I = "";
        int n = 0;
        int k = 0;
        int m = 0;
        boolean b = false;
        try {
            k = Integer.parseInt(fx);
            b = true;
        } catch (NumberFormatException e) {
            b = false;
        }
        if (gx.contains("e^") && b) {
            if (k != 0) {
                n = Integer.parseInt(gx.substring(gx.indexOf("^") + 1, gx.indexOf("^") + 2));
                I = k + "/" + n + " + " + " C/e^" + n + "x";
            } else {
                n = Integer.parseInt(gx.substring(gx.indexOf("^") + 1, gx.indexOf("^") + 2));
                I = "Ce^-" + n + "x";
            }
        }
        else if (gx.contains("e^") && fx.contains("e^")) {
            n = Integer.parseInt(gx.substring(gx.indexOf("^") + 1, gx.indexOf("^") + 2));
            m = Integer.parseInt(fx.substring(fx.indexOf("^") + 1, fx.indexOf("^") + 2));
            I = "(1/" + (n + m) + ")e^" + m + "x + Ce^-" + n + "x";
        } else if (gx.contains("x^") && b) {
            if (k != 0) {
                n = Integer.parseInt(gx.substring(gx.indexOf("^") + 1, gx.indexOf("^") + 2));
                I = k + "x/" + (n + 1) + " + " + " Cx^-" + n;
            } else {
                n = Integer.parseInt(gx.substring(gx.indexOf("^") + 1, gx.indexOf("^") + 2));
                I = "Cx^-" + n;
            }
        }
        return I;
    }
```

4. Novedades de la calculadora

En este trabajo lo novedoso que se implementa, cuando se realizó una revisión de librerías disponibles para resolver las ecuaciones diferenciales.

Se concluye de que no existe una librería de integrales originaria de JAVA que fuese de utilidad y/o maneje varios de los tipos de integrales necesarias para llegar a la resolución de estas ecuaciones diferenciales, ya que para la parte de derivadas si existe una cantidad considerable de librerías de apoyo de la cual usamos la librería jep (Java Equation Parse), pero en integrales se implementa la lógica matemática y bases de programación en java para crear una base de integrales las cuales resuelve integrales básicas, el método de integración por partes (ILATE), integrales trigonométricas, integral algebraica-trigonométrica, integrales de productos y exponenciales.

5. Condiciones de uso

Formas de ecuaciones solucionables

Para usar la calculadora diseñada para la solución de ecuaciones lineales de primer orden por el método de se deben respetar las siguientes restricciones:

Para Ingresar una Ecuación solucionable se debe escribir de la forma:

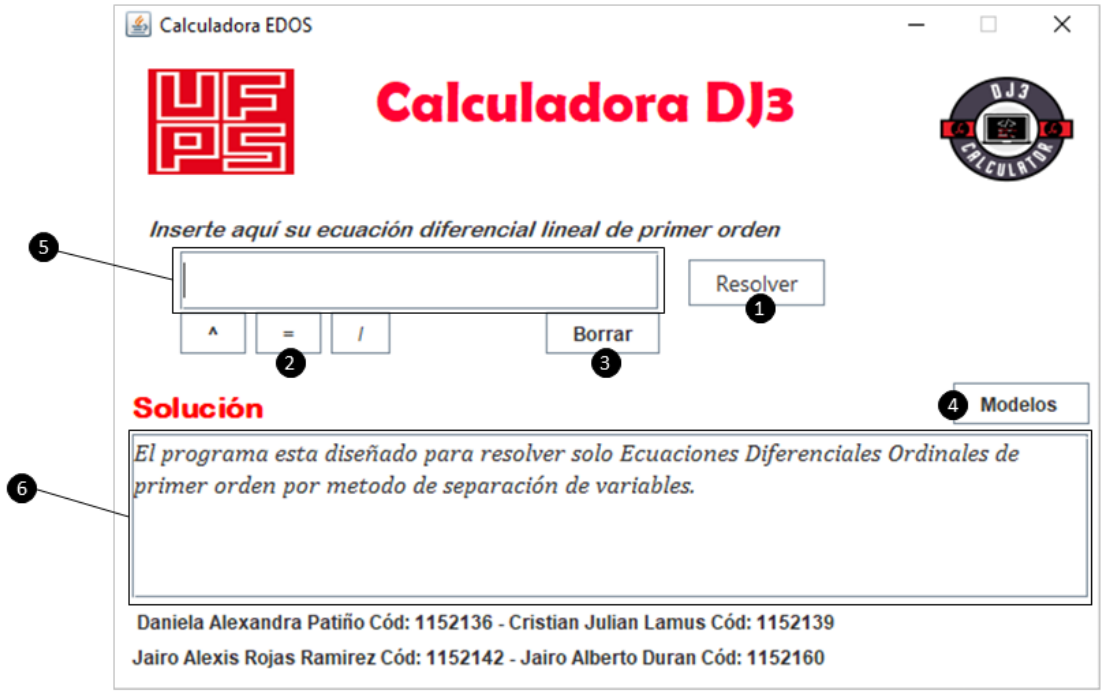
$$dy/dx+k/xY=P(x)$$

Se debe respetar la mayúscula "Y" como la variable "y" de la ecuación, y no se deben dejar espacios ni tabulaciones entre caracteres, ni antes o después de la ecuación. Entre las variantes solucionables por la calculadora, se presentan y se escriben de la siguiente manera:

1. *dy/dx+k/xY=sin(x)*
2. *dy/dx+k/xY=cos(x)*
3. *dy/dx+ky=sin(x)*
4. *dy/dx+ky=cos(x)*
5. *dy/dx+ky=-cos(x)*
6. *dy/dx+ky= -sin(x)*
7. *dy/dx+kY=e^mx*
8. *dy/dx+kY= n*
9. *dy/dx+k/xY= n*

Donde $9 \geq k > 0$, donde $m > 0$, y n es igual a todos los enteros positivos hasta el infinito.

Manual de Usuario



1. **Botón Resolver:** Soluciona la Ecuación ingresada en el campo de texto (5) y muestra su resultado en el cuadro de texto (6).[1]
2. **Botones Auxiliares:** Añade a la ecuación (5) uno de los símbolos (Exponente, Igual, División o Fracción respectivamente).
3. **Botón Borrar:** Limpia por completo la ecuación (5).
4. **Boton Modelos:** Muestra el modelo matemático usado para la solución de las ecuaciones diferenciales.
5. **Campo de Texto Ecuación:** Aquí puede escribir la ecuación que desee resolver.
6. **Cuadro de Texto Solución:** Aquí aparece la solución de la ecuación ingresada en el campo de texto(5)

[1] *Tenga en cuenta las formas de ecuaciones solucionables*