<!--Manual técnico-->

DataForge {

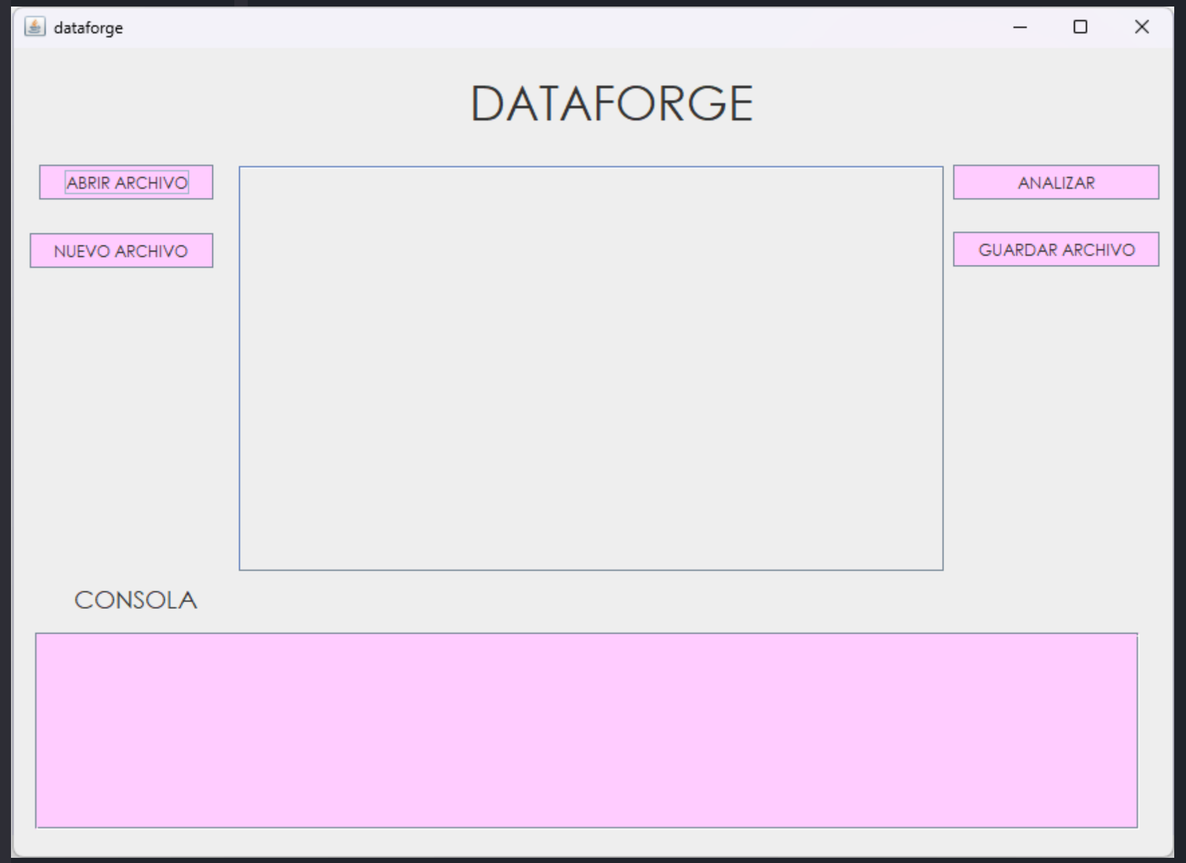
<Por="Gladys Ajuchán"/>

}

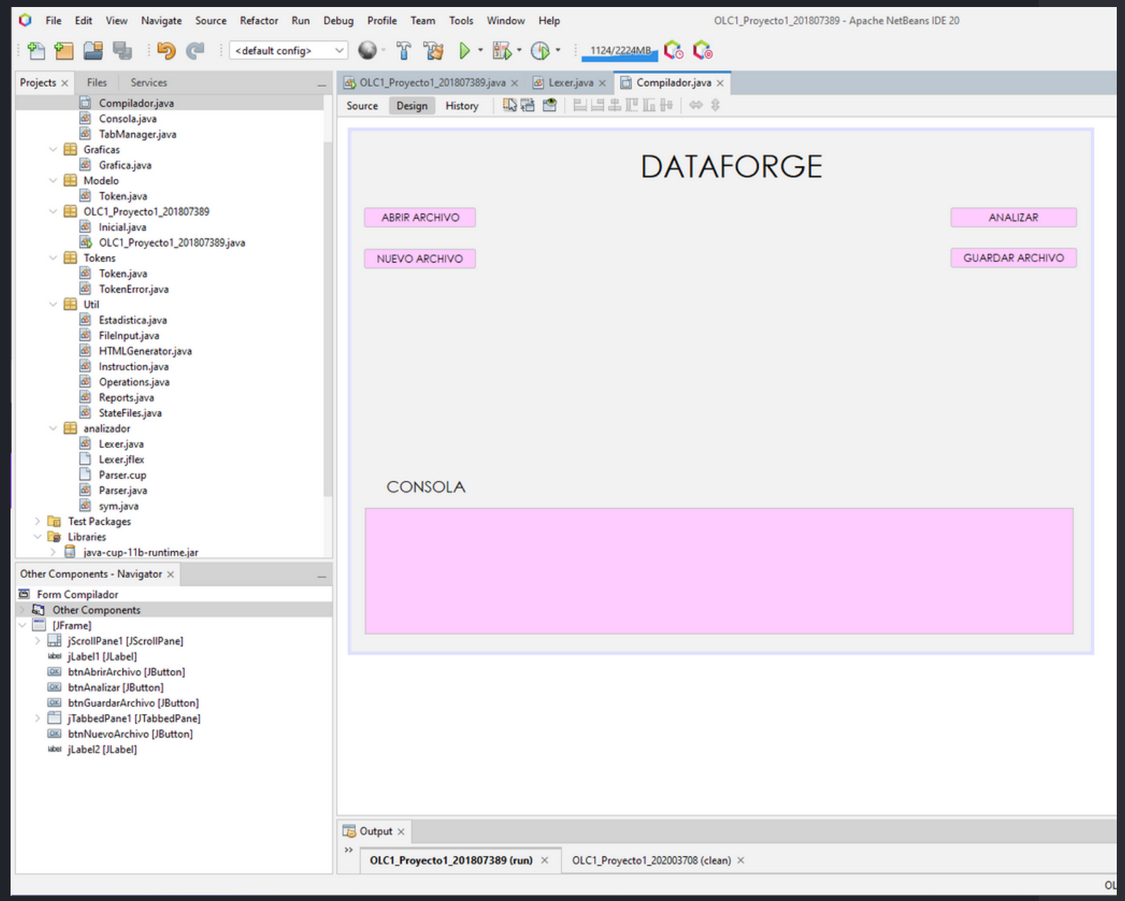
Contenido

Dataforge {

Datafoge es un sistema capaz de realizar operaciones aritméticas y estadísticas, además de poder generar diversos gráficos a partir de una colección de datos.



}



Requisitos mínimos {

- Java Development Kit (JDK
- Ambiente de desarrollo integrado (IDE)
- Técnología jFlex y Cup

Dataforge es desarrollado en el IDE Apache Netbeans IDE 20.

Desarrollo del programa {

- Diseño con JavaFX
  Se utiliza desing del IDE para la creación de la interfaz gráfica, ventana, botones y cajas de texto.

- Archivos
  Se implementa lógica de manejo de archivos para las acciones de abrir, crear y guardar archivos utilizando clases de entrada y salida. Por ejemplo File, FileReader, FileWriter.

```
public static void generateHTML(LinkedList<Token> tokens, String titulo, String nameF
    try {
        File file = new File(nameFile+".html");
        BufferedWriter writer = new BufferedWriter(new FileWriter(file));

        // Escribir el encabezado del HTML con Bootstrap
        writer.write("<html>\n<head>\n<title>"+titulo+"</title>\n"
                + "<link rel=\"stylesheet\" href=\"https://maxcdn.bootstrapcdn.com/bo
                + "</head>\n<body>\n<div class=\"container\">\n"
                + "<h1><CENTER>"+titulo+"</CENTER></h1>\n<table class=\"table table-c
                + "<thead>\n<tr>\n<th>No.</th>\n<th>Token</th>\n<th>Tipo</th>\n<th>Fi

        int counter = 1;
        for (Token token : tokens) {
            writer.write("<tr>\n<td>" + counter + "</td>\n"
                    + "<td>" + token.getToken() + "</td>\n"
                    + "<td>" + token.getTipo() + "</td>\n"
                    + "<td>" + token.getFila() + "</td>\n"
                    + "<td>" + token.getColumna() + "</td>\n</tr>\n");
```

}

```
package analizador;
import java_cup.runtime.*;


%%
//-------> Directivas (No tocar)

%public
%class Lexer
%cup
%char
%column
%line
%unicode
%ignorecase

%init{
    yyline = 1;
    yycolumn = 1;
%init}

// ------> Expresiones Regulares

entero = [0-9]+
decimal = [0-9]+("."[0-9]+)?
cadena = [\"][^\n\"]*[\"]
id = [a-zA-Z][a-zA-Z0-9_]*
```

- Análisis léxico con jFlex

Se crea un archivo con extensión .flex en el directorio ‘jflex’ que contenga las reglas léxicas para el lenguaje a implementar. Al ejecutar el programa se genera el analizador léxico “Lexer.java”

- Análisis sintáctico con Cup

Se crea un archivo con extensión .cup en el directorio ‘cup’ que contenga las reglas gramaticales y acciones semánticas. Al ejecutar el programa se genera el analizador sintáctico “Parser.java” y “sym.java”.

- Generación de reportes

Se implementa funciones para generar los reportes de las tablas de símbolos, tokens y errores.

- Interconexión léxica y sintáctica

Integra el analizador léxico y sintáctico en el programa, todo ello para la ejecución secuencial de ambos para realizar el análisis completo.

- Interfaz para el usuario

Conecta las acciones de los botones y elementos de la interfaz con las funciones correspondientes del programa.

```
private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {

    Util.Reports.listaTokens.clear();
    Consola.setText("");
    Analisis.analizarContenido(jTabbedPanel, Consola);


    Util.HTMLGenerator.generateHTML(Util.Reports.listaTokens, "Lista de Tokens", "TokensCorre
    Util.Reports.listaTokens.clear();


    Util.HTMLGenerator.generateHTMLErr(Util.Reports.listaErroresTok, "Lista de Errores", "Rep
    Util.Reports.listaErroresTok.clear();


    Util.HTMLGenerator.generateHTMLTabSimb(Util.Instruction.tablaVariables, "Tabla de Simbolo
    Util.Instruction.tablaVariables.clear();
```

}

```java
public class TokenError {

    private String lexema;
    private int fila;
    private int columna;

    public TokenError(String lexema, int fila, int columna) {
        this.lexema = lexema;
        this.fila = fila;
        this.columna = columna;
    }

    public String getLexema() {
        return lexema;
    }

    public void setLexema(String lexema) {
        this.lexema = lexema;
    }

    public int getFila() {
        return fila;
    }
}
```

Consideraciones {

- Gestión de errores
  Se implementa manejo de errores para una posible solución durante la ejecucion del programa.

- Pruebas
  Realizar pruebas al programa con archivos de entrada para la verificación de reportes y así probar las funcionalidades.