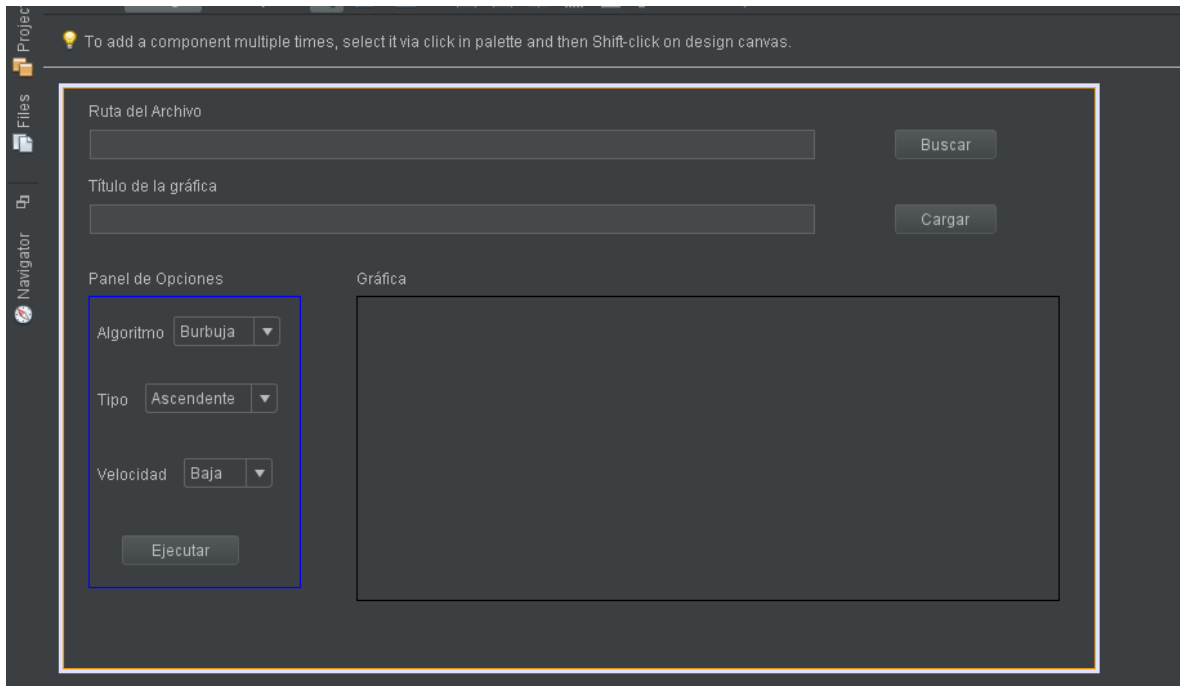


# **Manual Técnico**

**Por: Mario Ernesto Marroquín Pérez--202110509**

Dentro de la clase “Milnicio” que es un form se encuentra el main de ejecución. La clase consta del diseño de la interfaz para la elección del archivo csv, el método para realizar la gráfica, el método para la elección de opciones para ordenar los datos y la opción de colocar el título de la gráfica.

Diseño interfaz:



Código generación de la gráfica:

```

353 private void BotonCargarActionPerformed(java.awt.event.ActionEvent evt) {
354     GraficaBarras.TituloGrafica(NombreGrafica.getText());
355     DefaultCategoryDataset datos = new DefaultCategoryDataset();
356
357     for (datos dato : getAlmacenDatos()) {
358         if (dato != null) {
359
360             int numero1 = dato.getNotas();
361             String numero2 = String.valueOf(numero1);
362             datos.setValue(numero1, dato.getCursos(), "");
363             notas1+=" " + numero2 + "</th>"; 364             cursos1+=" " + dato.getCursos() + "</th>"; 365 366         } 367     } 368 369     grafico_Barras = ChartFactory.createBarChart(NombreGrafica.getText(), AlmacenDatos.encabezado1, AlmacenDatos.encabezado2, datos, PlotOrientation.VERTICAL, 370 //JFreeChart grafico_Barras = ChartFactory.createBarChart(nombreGraf, nombreGraf, nombreGraf, datos, PlotOrientation.HORIZONTAL, rootPaneCheckingEnabled, false); 371 ChartPanel panel = new ChartPanel(grafico_Barras); 372 panel.setMouseWheelEnabled(true); 373 panel.setPreferredSize(new Dimension(546, 237)); 374 panelGrafica.setLayout(new BorderLayout()); 375 376 panelGrafica.add(panel, BorderLayout.NORTH); 377 378 pack(); 379 repaint(); 380 381 generarImg1(); 382 383 } | |
```

En la clase “RecibirCsv” se recibe la ruta del archivo csv, se lee el archivo y se almacena en los diferentes arreglos, así mismo en esta clase es donde se ordenan según el algoritmo seleccionado por el usuario, ya sea de forma ascendente o ascendente.

Lectura del csv:

```
23 public static void recibir(String ruta){
24     String path = ruta;
25     String linea = "";
26     String encabezado;
27     String encabezado1;
28     String [] encabezado2;
29
30     //int i=0;
31
32     try {
33         BufferedReader br = new BufferedReader (new FileReader (path));
34         encabezado = br.readLine();
35         encabezado1 = encabezado.replace(';',' ');
36         encabezado2 = encabezado1.split(",");
37         while ((linea = br.readLine()) != null ){
38
39
40             String linea1 = linea.replace(';',' ');
41             String Linea = linea1.replaceAll(System.getProperty("line.separator"), "");
42             String[] linea2 = Linea.split("\r\n");
43
44             for(String linea3: linea2){
45                 String [] atributos = Linea.trim().split(",") ;
46                 int notas = Integer.parseInt(atributos[1]);
47                 String cursos = String.valueOf(atributos[0]);
48                 if(notas != 0){
49                     notas1[I] = notas;
50                     CURSOS[I] = cursos;
51                     datos nuevoDato = new datos (atributos [0], notas);
52                     AlmacenDatos.colocarDatos(nuevoDato);
53                     I++;
54                 }
55             }
56         }
57     }
58 }
```

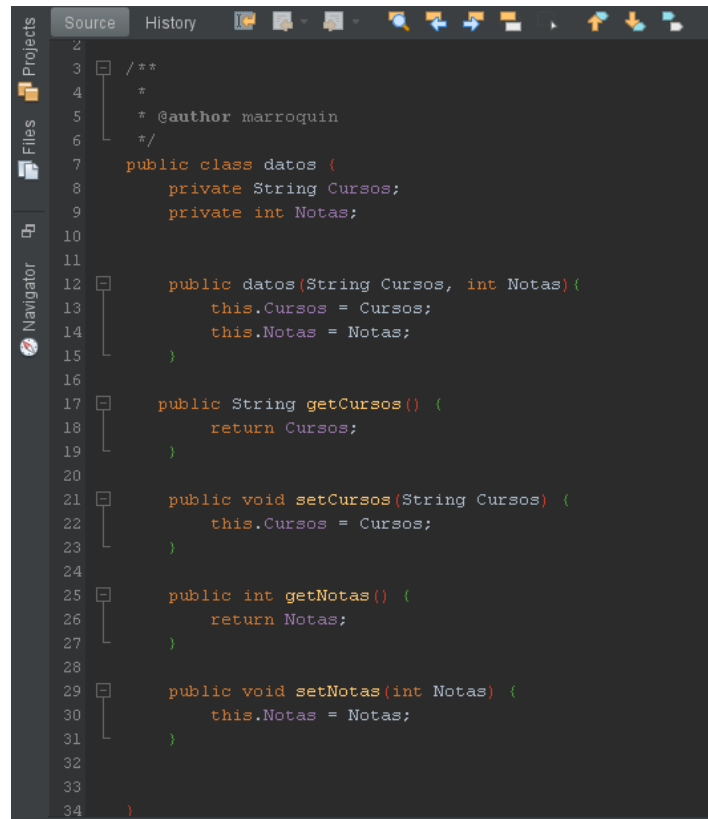
Ordenamientos:

```
public static void bur(){
    int aux;
    boolean cambio = false;
    String cur;

    while(true){
        cambio = false;

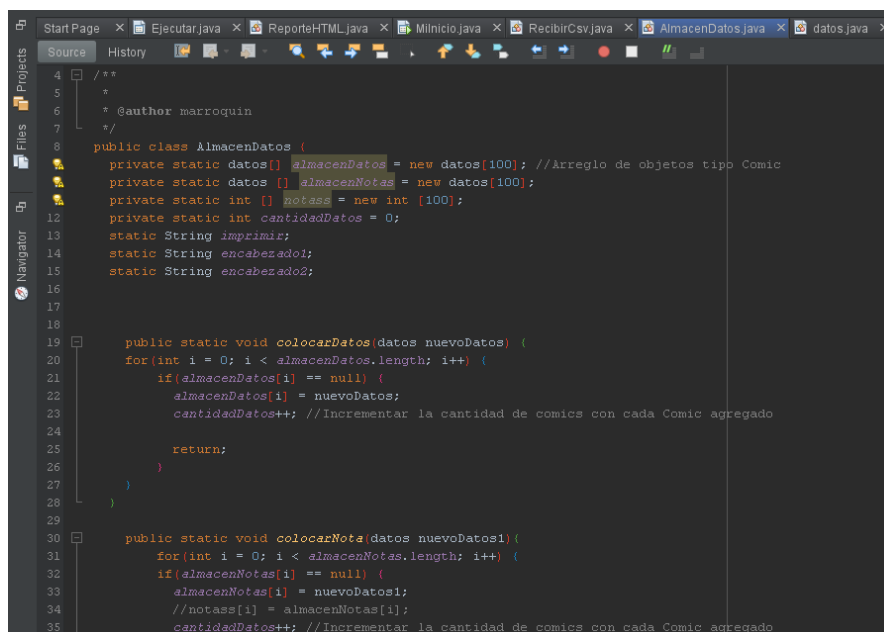
        for(int i=0; i<notas1.length-1; i++){
            for(int k=0; k<notas1.length-1; k++){
                if(notas1[k]>notas1[k+1]){
                    aux = notas1[k];
                    notas1[k] = notas1 [k+1];
                    notas1[k+1] = aux;
                    pasos++;
                    cambio = true;
                }
            }
        }
        if(cambio == false)
            break;
    }
}
```

En la clase “datos” se encuentran los métodos setter and getters, en esta clase se envían los datos leídos del csv.



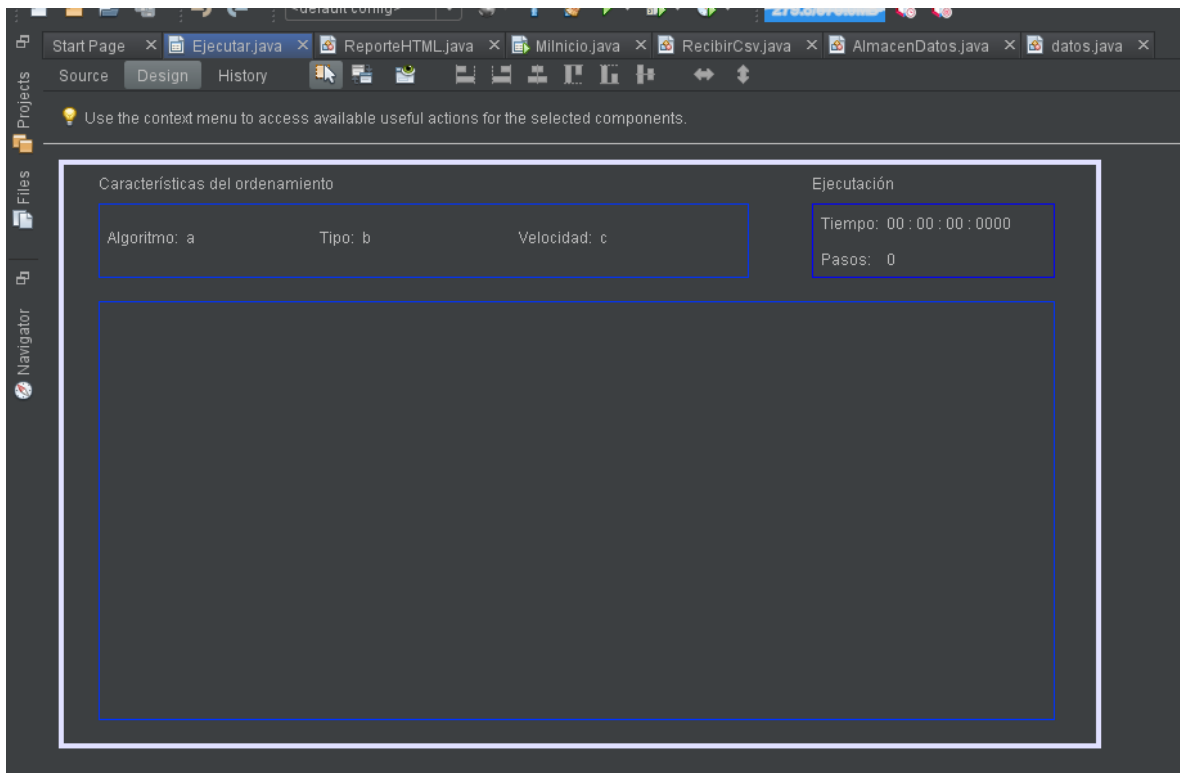
```
2
3  /**
4   *
5   * @author marroquin
6   */
7  public class datos {
8      private String Cursos;
9      private int Notas;
10
11
12      public datos(String Cursos, int Notas) {
13          this.Cursos = Cursos;
14          this.Notas = Notas;
15      }
16
17      public String getCursos() {
18          return Cursos;
19      }
20
21      public void setCursos(String Cursos) {
22          this.Cursos = Cursos;
23      }
24
25      public int getNotas() {
26          return Notas;
27      }
28
29      public void setNotas(int Notas) {
30          this.Notas = Notas;
31      }
32
33
34  }
```

En la clase “AlmacenDatos” se reciben los datos del método setter and getter y se asignan en los arreglos establecidos, en esta clase también se reinician los arreglos para ingresar los nuevos datos ya ordenados



```
4  /**
5   *
6   * @author marroquin
7   */
8  public class AlmacenDatos {
9      private static datos[] almacenDatos = new datos[100]; //Arreglo de objetos tipo Comic
10     private static datos[] almacenNotas = new datos[100];
11     private static int[] notas = new int [100];
12     private static int cantidadDatos = 0;
13     static String imprimir;
14     static String encabezado1;
15     static String encabezado2;
16
17
18
19     public static void colocarDatos(datos nuevoDatos) {
20         for(int i = 0; i < almacenDatos.length; i++) {
21             if(almacenDatos[i] == null) {
22                 almacenDatos[i] = nuevoDatos;
23                 cantidadDatos++; //Incrementar la cantidad de comics con cada Comic agregado
24             }
25         }
26         return;
27     }
28
29
30     public static void colocarNota(datos nuevoDatos1){
31         for(int i = 0; i < almacenNotas.length; i++) {
32             if(almacenNotas[i] == null) {
33                 almacenNotas[i] = nuevoDatos1;
34                 //notas[i] = almacenNotas[i];
35                 cantidadDatos++; //Incrementar la cantidad de comics con cada Comic agregado
36             }
37         }
38     }
39 }
```

En la clase “Ejecutar” se diseña la segunda ventana grafica del programa, en donde se visualiza el ordenamiento de los datos, el tiempo, pasos y características del ordenamiento, en esta clase también se ejecutan los hilos para el tiempo y ordenamiento.



El hilo de tiempo se ejecuta durante todo el ordenamiento hasta que este termina:

```
4 public void tiempo1(){
5
6     Thread cronometro = new Thread(){
7         public void run(){
8             for(;;){
9                 if(estado == true){
10                     try{
11                         Thread.sleep(1);
12                         if(mili>=1000){
13                             mili = 0;
14                             seg++;
15                         }
16                         if(seg >= 60){
17                             mili=0;
18                             seg=0;
19                             min++;
20                         }
21                         if(min>=60){
22                             mili=0;
23                             seg=0;
24                             min=0;
25                             hora++;
26                         }
27                         mili+=6;
28                         Tiempo.setText(hora+ ":" +min+ ":" +seg+"."+mili);
29                     } catch(Exception e){
30                     }
31                 }else{
32                     break;
33                 }
34             }
35         }
36     };
37 }
```

Así mismo con el hilo del ordenamiento de la gráfica, este hilo empieza desde que el usuario cliquee el botón “ejecutar” y termina cuando finaliza el ordenamiento.

```
public void burbuja() {
    Thread bur = new Thread() {
        public void run() {
            int aux;
            boolean cambio = false;
            int pasos = 0;
            DefaultCategoryDataset datos = new DefaultCategoryDataset();
            try {
                for (datos dato : getAlmacenDatos()) {
                    if (dato != null) {
                        int numero1 = dato.getNotas();
                        String curs = dato.getCursos();
                        String numero2 = String.valueOf(numero1);
                        notas1+=" " + numero2 + "</th>";                         cursos1+=" " + curs + "</th>";                         datos.setValue(numero1, curs, "Orden" + " " + MiInicio.tip);                         barras = ChartFactory.createBarChart(GraficaBarras.tilGraf(), AlmacenDatos.encabezado1, AlmacenDatos.encabezado2, datos, PlotOrientation.VERTICAL, false, true, false, false);                         ChartPanel panel = new ChartPanel(barras);                         panel.setMouseWheelEnabled(true);                         panel.setPreferredSize(new Dimension(546, 237));                         panelGrafica.setLayout(new BorderLayout());                         panelGrafica.add(panel, BorderLayout.NORTH);                         pack();                         repaint();                         MiInicio inicio = new MiInicio();                         if (inicio.sendVelocidad().equals("Rápida")) {                             Thread.sleep(100);                         }                          if (inicio.sendVelocidad().equals("Media")) {                             Thread.sleep(500);                         }                          if (inicio.sendVelocidad().equals("Baja")) {                             Thread.sleep(800);                         }                          ReporteHTML reporte = new ReporteHTML();                         paso++;                         pass = String.valueOf(paso);                         labelPasos.setText(pass);                         retPasos(pass);                     }                 }             } catch (InterruptedException ex) {                 Logger.getLogger(Ejecutar.class.getName()).log(Level.SEVERE, null, ex);             }         }     };     bur.start(); } | |
```

En esta clase también se encuentra la función para generar la imagen de la grafica y guardarla en la carpeta del archivo.

```
220
221 public void generarImg(){
222     try{
223         final File file = new File("grafica2.png");
224         ChartUtils.saveChartAsPNG(file, barras, 546, 237);
225         //this.dispose();
226
227     }catch(IOException e){
228         System.out.println("Erro en la img");
229     }
230 }
```