

# Calificación de películas Códigos del proyecto

Emmanuel Peto Gutiérrez  
Rodrigo Fernando Velázquez Cruz

IIMAS  
UNAM

7 de diciembre de 2022

# Main.run

Calificación de  
películas  
Códigos del proyecto

```
src / Main.java / Main / start(Stage)
```

```
@Override
public void start(Stage primaryStage) {
    anchoVentana = 800;
    altoVentana = 700;
    escenario = primaryStage;

    caja = new VBox();

    labelSelect = new Label(" Indique las columnas a seleccionar. ");
    inputSelect = new TextField();
    inputSelect.setPrefWidth(anchoVentana-100);

    labelWhere = new Label(" Escriba las condiciones de filtrado en forma normal disyuntiva. ");
    inputWhere = new TextField();
    inputWhere.setPrefWidth(anchoVentana-100);

    GraficaBarras grafica = new GraficaBarras();

    ejecuta = new Button(" Ejecutar consulta ");
    ejecuta.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            boolean valida = ejecutaConsulta();
            if (valida) {
                grafica.reset();
            }
        }
    });

    avanzaGrafica = new Button("Avanzar");
    avanzaGrafica.setOnAction(new EventHandler<ActionEvent>() {
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Main.run

Calificación de  
películas  
Códigos del proyecto

```
avanzaGrafica = new Button("Avanzar");
avanzaGrafica.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        grafica.avanza();
    }
});

retrocedeGrafica = new Button("Retroceder");
retrocedeGrafica.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        grafica.retrocede();
    }
});

HBox cajaBotones = new HBox();
cajaBotones.setSpacing(20);
cajaBotones.getChildren().addAll(ejecuta, retrocedeGrafica, avanzaGrafica);
caja.getChildren().addAll(labelSelect, inputSelect, labelWhere, inputWhere, cajaBotones);
caja.setSpacing(10);

VBox cajaGraph = new VBox(grafica.getBarChart());
cajaGraph.setMaxWidth(1500);
cajaGraph.setAlignment(Pos.TOP_CENTER);
caja.getChildren().add(cajaGraph);

Scene scene = new Scene(caja, anchoVentana, altoVentana);
escenario.setTitle("Consulta");
escenario.setScene(scene);
escenario.show();
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Main.getNumHilos

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * @return El número de hilos a usar es 4 veces el número de CPU's según la JVM.  
 */  
public int getNumHilos() {  
    int CPUs = Runtime.getRuntime().availableProcessors();  
    return CPUs*4;  
}
```

## Main.ejecutaConsulta

## Calificación de películas Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```

/**
 * Revisa las entradas del usuario.
 * Realiza la división de archivos.
 * Llama al manager para realizar el filtrado.
 * @return true si se pudo realizar la consulta, false en otro caso.
 */
public boolean ejecutaConsulta(){
    String select = inputSelect.getText();
    String expr = inputWhere.getText();

    if (!Parser.revisaCorrectas(select)) {
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Columnas no validas.");
        alert.setContentText("Ingresa solo columnas validas: \n movieId, title, year, genres, rating, imdb, themoviedb, age, name, lastname");
        alert.showAndWait();
        return false;
    }
    ArrayList<ArrayList<Expresion>> expresiones = null;
    try {
        // Interprete envia las cláusulas de filtrado
        expresiones = Parser.analiza(expr);
    } catch (Exception e) {
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Consulta no valida.");
        alert.setContentText("Ese comparador no está en nuestra gramática o la sintaxis de la consulta es incorrecta");
        alert.showAndWait();
        return false;
    }
    if (select.strip().equals("*")) {
        select = "movieId,title,year,genres,rating,imdb,themoviedb,age,name,lastname";
    }
}

```

# Main.ejecutaConsulta

```
    alert.setContentText("Ese comparador no está en nuestra gramática o la sintaxis");
    alert.showAndWait();
    return false;
}
if (select.strip().equals("*")) {
    select = "movieId,title,year,genres,rating,imdb,themoviedb,age,name,lastname";
}
if (!select.contains("title")) {
    select += ", title";
}
if (!select.contains("rating")) {
    select += ", rating";
}

int numHilos = getNumHilos();
// Probando con un archivo de pocos registros
String direccion = "data/out-users-8000_v3.csv"; // ya vamos por la versión 3
Divisor.divideArchivos(numHilos, direccion);
// Realiza el filtrado sobre los workers
Manager.filtrarInformacion(numHilos, expresiones, select);

Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Query");
alert.setHeaderText("Consulta finalizada.");
alert.setContentText("Revisar archivo data/resultado.csv");
alert.showAndWait();

inputSelect.setText("");
inputWhere.setText("");
return true;
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Manager.filtralInformacion

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Crea un pool de hilos y obtén la información de cada hilo.  
 * @param numHilos hilos a crear en el pool.  
 * @param expresiones filtros para los registros.  
 * @param seleccionadas columnas seleccionadas  
 */  
public static void filtraInformacion(int numHilos, ArrayList<ArrayList<Expresion>> expresiones, String se'  
    // El buffer donde los hilos escriben  
    Worker.inicializaBufferConcurrente(seleccionadas);  
  
    ExecutorService poolWorkers = Executors.newFixedThreadPool(numHilos);  
  
    for (int i = 0; i < numHilos; i++) {  
        String subarchivo = DIR_SUBARCHIVOS + Integer.toString(i+1) + ".csv";  
        poolWorkers.execute(new Worker(subarchivo, expresiones, seleccionadas));  
    }  
  
    int cantidadWorkers = Thread.activeCount() - 1;  
    System.out.println("Threads actuales: " + cantidadWorkers);  
  
    poolWorkers.shutdown();  
    while (! poolWorkers.isTerminated()) {  
    }  
  
    // Cierra el buffer cuando todos los Threads terminaron su tarea  
    Worker.cierraBufferConcurrente();  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

```
/**  
 * Recibe el archivo sobre el cual va a trabajar.  
 * @param archivo el archivo a realizar la operación de filtrado.  
 * @param expresiones las listas de expresiones para filtrar el archivo.  
 * @param seleccion arreglo con las columnas seleccionadas.  
 */  
public Worker(String archivo, ArrayList<ArrayList<Expresion>> expresiones, String seleccion) {  
    this.archivo = archivo;  
    this.expresiones = expresiones;  
    colSelect = seleccion;  
}
```

# Worker.inicializaBufferConcurrente

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Inicializa el buffer de escritura concurrente para los hilos.  
 * @param columnas las columnas a agregar al archivo de resultados.  
 */  
static void inicializaBufferConcurrente(String columnas) {  
    try {  
        File file = new File(DIR_RESULTADO);  
        file.createNewFile();  
        FileWriter fw = new FileWriter(DIR_RESULTADO);  
        bufferResultado = new BufferedWriter(fw);  
        fw.write(columnas + "\n");  
    } catch (IOException e) {  
        System.out.printf("No se puede escribir en el archivo %s%n", DIR_RESULTADO);  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Worker.cierraBufferConcurrente

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Una vez que los hilos hayan terminado su tarea, cierra el buffer.  
 */  
static void cierraBufferConcurrente() {  
    try {  
        bufferResultado.close();  
    } catch (IOException e) {  
        System.out.println("No se pudo cerrar el buffer de escritura compartida");  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
@Override  
public void run() {  
    manejaArchivo();  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Worker.seleccionaColumnas

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Hace una proyección de un registro, dejando solamente las columnas seleccionadas.  
 * @param registro  
 * @param indSelec lista de índices de las columnas seleccionadas.  
 * @return registro sin algunas columnas.  
 */  
private String seleccionaColumnas(String registro, ArrayList<Integer> indSelec){  
    String salida = "";  
    String[] regSplit = registro.split(",");  
    for(int i=0; i<indSelec.size()-1; i++){  
        int indiceCol = indSelec.get(i);  
        salida += regSplit[indiceCol]+",";  
    }  
    salida += regSplit[indSelec.get(indSelec.size()-1)];  
    return salida;  
}
```

# Worker.manejaArchivo

Calificación de  
películas  
Códigos del proyecto

```
/*
 * Hace el filtrado de información del subarchivo que le tocó.
 */
private void manejaArchivo() {
    System.out.println(">>>> Worker " + this.getName() + " trabajando con " + this.archivo);
    File subarchivo = new File (this.archivo);
    try{ FileReader fr = new FileReader(subarchivo);
        BufferedReader br = new BufferedReader(fr); }
        String registro = br.readLine();
        // En la primera fila estan las columnas
        String[] columnaSubarchivo = registro.split(",");
        ArrayList<Integer> indicesColumnas = new ArrayList<>();
        HashMap<String, Integer> tablaNombres = new HashMap<>();
        for(int i=0; i<columnaSubarchivo.length; i++){
            tablaNombres.put(columnaSubarchivo[i], i);
        }

        String [] proyeccion; //va a contener solamente las columnas seleccionadas.
        if(colSelect.equals("*")){
            proyeccion = columnaSubarchivo;
        }else{
            proyeccion = colSelect.split(",");
            for(int k=0; k<proyeccion.length; k++){
                proyeccion[k] = proyeccion[k].strip();
            }
        }

        for(String cs : proyeccion){
            indicesColumnas.add(tablaNombres.get(cs));
        }
        while ((registro = br.readLine()) != null) {
            if(expresiones.isEmpty()){
                System.out.println("No se cumplió la condición");
            }else{
                String[] linea = registro.split(",");
                int[] indices = new int[indicesColumnas.size()];
                for(int i=0; i<indicesColumnas.size(); i++){
                    indices[i] = linea[indicesColumnas.get(i)];
                }
                String resultado = expresiones.evaluar(indices);
                System.out.println(resultado);
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Worker.manejaArchivo

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
        }
    }else{
        proyeccion = colSelect.split(",");
        for(int k=0; k<proyeccion.length; k++){
            proyeccion[k] = proyeccion[k].strip();
        }
    }

    for(String cs : proyeccion){
        indicesColumnas.add(tablaNombres.get(cs));
    }
    while ((registro = br.readLine()) != null) {
        if(expresiones.isEmpty()){
            escribeArchivo(registro+"\n");
        }else{
            for (ArrayList<Expresion> listaExpresiones : expresiones) {
                // Filtra las columnas
                if (satisfaceCondiciones(registro, listaExpresiones, tablaNombres)) {
                    String proyectado = seleccionaColumnas(registro, indicesColumnas);
                    escribeArchivo(proyectado + "\n");
                    break; // verificar si no sale un bug por esta linea
                }
            }
        }
    }
} catch (FileNotFoundException e) {
    System.out.printf("%s no pudo escribir en el archivo %s porque no existe", this.getName(), this.
} catch (Exception e) {
    System.out.println(e);
}
}
```

# Worker.escribeArchivo

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Escribe de manera concurrente un registro en un archivo específico para guardar los resultados.  
 * @param registro el registro a guardar en el archivo general.  
 */  
private synchronized void escribeArchivo(String registro) {  
    try {  
        bufferResultado.write(registro);  
        bufferResultado.flush();  
    } catch (IOException e) {  
        System.out.printf("%s no pudo escribir en el archivo %s%n", this.getName(), DIR_RESULTADO);  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}
```

# Worker.tipoColumna

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Le asigna un número dependiendo del tipo de columna  
 * @param nombreColumna  
 * @return 0 si es genres, 1 si es tipo double, 2 si es entero, 3 en otro caso.  
 */  
private int tipoColumna(String nombreColumna){  
    switch(nombreColumna){  
        case "genres": return 0;  
        case "rating": return 1;  
        case "idRating":  
        case "userId":  
        case "movieId":  
        case "timestamp":  
        case "year":  
        case "age": return 2;  
        default: return 3;  
    }  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Worker.satisfaceCondiciones

Calificación de  
películas  
Códigos del proyecto

```
/**  
 * Revisa si el registro dado satisface las condiciones.  
 * @param registro  
 * @param listaExpresiones  
 * @param tablaNombres  
 */  
private boolean satisfaceCondiciones(String registro, ArrayList<Expresion> listaExpresiones, HashSet<String> registroSeparado = registro.split(",");  
    // Como es una lista de conjunciones, se devuelve falso al primero que  
    // no cumpla con la condición.  
    for(Expresion expr : listaExpresiones){  
        String valorEsperado = expr.getValor();  
        int indCol = tablaNombres.get(expr.getVariable());  
        String valorReal = registroSeparado[indCol];  
        Comparable vrComp;  
        Comparable veComp;  
        switch(tipoColumna(expr.getVariable())){  
            case 1 : vrComp = Double.valueOf(valorReal);  
                      | veComp = Double.valueOf(valorEsperado);  
            break;  
            case 2 : vrComp = Integer.valueOf(valorReal);  
                      | veComp = Integer.valueOf(valorEsperado);  
            break;  
            default: vrComp = valorReal;  
                      | veComp = valorEsperado;  
        }  
  
        if(tipoColumna(expr.getVariable()) == 0){  
            String[] generos = valorReal.split("\\|");  
            valorReal = valorReal.toUpperCase();  
            valorEsperado = valorEsperado.toUpperCase();  
        }  
    }  
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Worker.satisfaceCondiciones

Calificación de  
películas  
Códigos del proyecto

```
valorReal = valorReal.toUpperCase();
valorEsperado = valorEsperado.toUpperCase();
switch(expr.getComparador()){
    case IGUALDAD:
        if(!valorReal.contains(valorEsperado)){
            return false;
        }
        break;
    case DIFERENTE:
        if(valorReal.contains(valorEsperado)){
            return false;
        }
        break;
    case MAYOR_IGUAL:
        for(String gen : generos){
            if(gen.compareTo(valorEsperado) < 0){
                return false;
            }
        }
        break;
    case MENOR_IGUAL:
        for(String gen : generos){
            if(gen.compareTo(valorEsperado) > 0){
                return false;
            }
        }
        break;
    case MAYOR:
        for(String gen : generos){
            if(gen.compareTo(valorEsperado) < 0
                ||gen.equals(valorEsperado)){
                return false;
            }
        }
        break;
```

Emmanuel Peto  
Gutiérrez

Rodrigo Fernando  
Velázquez Cruz

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Worker.satisfaceCondiciones

```
        break;
    case MAYOR:
        for(String gen : generos){
            if(gen.compareTo(valorEsperado) < 0
               ||gen.equals(valorEsperado)){
                return false;
            }
        }
        break;
    case MENOR:
        for(String gen : generos){
            if(gen.compareTo(valorEsperado) > 0
               ||gen.equals(valorEsperado)){
                return false;
            }
        }
    }
} else{
    switch(expr.getComparador()){
        case IGUALDAD:
            if(!vrComp.equals(veComp)){
                return false;
            }
            break;
        case DIFERENTE:
            if(vrComp.equals(veComp)){
                return false;
            }
            break;
        case MAYOR_IGUAL:
            if(vrComp.compareTo(veComp) < 0){
                return false;
            }
    }
}
```

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Worker.satisfaceCondiciones

Calificación de  
películas  
Códigos del proyecto

```
        }
        break;
    case DIFERENTE:
        if(vrComp.equals(veComp)){
            return false;
        }
        break;
    case MAYOR_IGUAL:
        if(vrComp.compareTo(veComp) < 0){
            return false;
        }
        break;
    case MENOR_IGUAL:
        if(vrComp.compareTo(veComp) > 0){
            return false;
        }
        break;
    case MAYOR:
        if(vrComp.compareTo(veComp) <= 0){
            return false;
        }
        break;
    case MENOR:
        if(vrComp.compareTo(veComp) >= 0){
            return false;
        }
        break;
    }
}
}

return true;
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# ComparadorEnum

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Enumerador para el tipo de comparaciones  
 */  
public enum ComparadorEnum {  
    IGUALDAD,  
    MENOR_IGUAL,  
    MAYOR_IGUAL,  
    MENOR,  
    MAYOR,  
    DIFERENTE  
}
```

# Divisor.divideArchivos

Calificación de  
películas  
Códigos del proyecto

```
/**  
 * Divide el archivo especificado en la ruta en multiples subarchivos,  
 * tantos como haya hilos.  
 * @param numHilos el numero de subarchivos.  
 * @param direccion ruta del archivo fuente.  
 */  
public static void divideArchivos(int numHilos, String direccion) {  
    // Cantidad de registros para cada subarchivo  
    int num_ratings_hilo = NUM_RATINGS / numHilos;  
    int residuo_ratings_hilo = NUM_RATINGS % numHilos;  
  
    // Linea sobre la cual debe empezar a leer cada registro  
    int linea_inicial = 1;  
    // El i-esimo archivo en crearse  
    int archivos = 1;  
  
    while (numHilos != 0) {  
        // Registros a guardar en un subarchivo  
        ArrayList<String> registrosHilo = new ArrayList<String>();  
        // Lee el archivo y obtien una cantidad determinada de registros a partir de una linea del arco  
        leerArchivo(direccion, num_ratings_hilo, linea_inicial, registrosHilo);  
        linea_inicial += num_ratings_hilo;  
        numHilos--;  
        // Añade los registros faltantes al ultimo hilo  
        if (numHilos == 1) {  
            num_ratings_hilo += residuo_ratings_hilo;  
        }  
        escribeArchivo(registrosHilo, archivos);  
        archivos++;  
    }  
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Divisor.leeArchivo

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
private static void leerArchivo(String direccion, int lineas, int i, ArrayList<String> registros) {  
  
    File archivo = new File (direccion);  
    try( FileReader fr = new FileReader(archivo);  
        BufferedReader br = new BufferedReader(fr);) {  
        int linea_actual = 1;  
        // Agrega las columnas  
        String columnas = br.readLine();  
        registros.add(columnas);  
        // Llega hasta la linea donde tiene que iniciar la lectura  
        while (linea_actual != i) {  
            br.readLine();  
            linea_actual += 1;  
        }  
  
        // Agrega tantas lineas como se indica  
        while (lineas != 0) {  
            String registro = br.readLine();  
            registros.add(registro);  
            lineas -= 1;  
        }  
  
    } catch (FileNotFoundException e) {  
        System.out.printf("No se puede leer el archivo %s%n", direccion);  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}
```

# Divisor.escribeArchivo

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/*
 * Dada una lista de registros, guardarlos en un archivo con un nombre identificado con un número.
 * @param registros los registros a guardar.
 * @param numArchivo identificador para el nombre del archivo.
 */
private static void escribeArchivo(ArrayList<String> registros, int numArchivo) {
    File dir = new File(WRITE_DIR);
    if (!dir.exists()) {
        dir.mkdirs();
    }
    String nombreArchivo = "subarchivo-" + numArchivo + ".csv";
    File file = new File(WRITE_DIR + nombreArchivo);
    try( FileWriter fw = new FileWriter(file);
        BufferedWriter bw = new BufferedWriter(fw);
        PrintWriter pw = new PrintWriter(bw)){{
        for(String registro : registros) {
            pw.write(registro + "\n");
        }
    } catch(IOException e) {
        System.out.printf("No se puede escribir en %s%n", WRITE_DIR + nombreArchivo);
    } catch(Exception e) {
        System.out.println(e);
    }
}
```

# Expresion.builder

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
public Expresion(ComparadorEnum comparador, String variable, String valor) {  
    this.comparador = comparador;  
    this.variable = variable;  
    this.valor = valor;  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Parser.splitConektor

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/*
 * Separa una expresión dado un conector.
 * @param expr expresión
 * @param conector AND u OR
 * @return lista de cada token
 */
private static ArrayList<String> splitConektor(String expr, String conector){
    String[] separado = expr.split(conector);
    for(int i=0; i<separado.length; i++){
        String ct = separado[i].strip();
        int indexIni = (ct.charAt(0) == '(')? 1 : 0;
        int indexFin = (ct.charAt(ct.length()-1) == ')')? ct.length()-1 : ct.length();
        separado[i] = ct.substring(indexIni, indexFin);
    }
    ArrayList<String> salida = new ArrayList<>(Arrays.asList(separado));
    return salida;
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Parser.fnd

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Dado un string en forma normal disyuntiva, lo separa en tokens.  
 * @param expr expresión a separar.  
 * @return lista de listas de comparaciones.  
 */  
private static ArrayList<ArrayList<String>> fnd(String expr){  
    ArrayList<String> sinOr = splitConector(expr, conector: "OR");  
    ArrayList<ArrayList<String>> sinAnd = new ArrayList<>();  
    for(var cadena : sinOr){  
        sinAnd.add(splitConector(cadena, conector: "AND"));  
    }  
    return sinAnd;  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Parser.extraeOperador

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Dada una expresión con un único comparador, devuelve el comparador.  
 * @param expr expresión en string.  
 * @return el operador dentro de la expresión.  
 * @throws UnsupportedOperationException  
 */  
private static String extraeOperador(String expr) throws UnsupportedOperationException {  
    if(expr.contains(">=")){  
        return ">=";  
    }else if(expr.contains("<=")){  
        return "<=";  
    }else if(expr.contains("<>")){  
        return "<>";  
    }else if(expr.contains(">")){  
        return ">";  
    }else if(expr.contains("<")){  
        return "<";  
    }else if(expr.contains("!=")){  
        return "!=";  
    }else{  
        throw new UnsupportedOperationException("Ese comparador no está en nuestra gramática.");  
    }  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Parser.buildExpr

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Construye una expresión (de la clase Expresion) dado un string con exactamente un comparador.  
 * @param expr expresión con un comparador.  
 * @return Expresion construida a partir de un String.  
 * @throws UnsupportedOperationException  
 */  
private static Expresion buildExpr(String expr) throws UnsupportedOperationException{  
    String op = extraeOperador(expr);  
    String[] separado = expr.split(op);  
    ComparadorEnum oe = ComparadorEnum.IGUALDAD;  
    switch(op){  
        case "<=" : oe = ComparadorEnum.MENOR_IGUAL;  
        break;  
        case ">=" : oe = ComparadorEnum.MAYOR_IGUAL;  
        break;  
        case "<>" : oe = ComparadorEnum.DIFERENTE;  
        break;  
        case "<" : oe = ComparadorEnum.MENOR;  
        break;  
        case ">" : oe = ComparadorEnum.MAYOR;  
        break;  
        case "=" : oe = ComparadorEnum.IGUALDAD;  
        break;  
    }  
    if (!revisaCorrectas(separado[0].strip())) {  
        throw new UnsupportedOperationException("Esa variable no es nombre valido de columna.");  
    }  
    revisaSemantica(oe, separado[0].strip(), separado[1].strip());  
    Expresion ne = new Expresion(oe, separado[0].strip(), separado[1].strip());  
    return ne;  
}
```

# Parser.revisaSemantica

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Revisa si las operaciones descritas por el operando, la columna y el valor son validas.  
 * @param op operador.  
 * @param columna la columna a revisar.  
 * @param valor el valor dado por el usuario.  
 */  
public static void revisaSemantica(ComparadorEnum op, String columna, String valor) throws Unsuppo  
    if (op != ComparadorEnum.IGUALDAD) {  
        if (!valor.matches("[0-9]+")) {  
            throw new UnsupportedOperationException("Tienes que tener un valor numérico.");  
        }  
    } else {  
        if (columna.equals("movieId") || columna.equals("year")) {  
            if (!valor.matches("[0-9]+")) {  
                throw new UnsupportedOperationException("Tienes que tener un valor numérico.");  
            }  
        }  
    }  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# Parser.analiza

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/*
 * Convierte una expresión en forma normal disyuntiva a una lista de listas de expresiones.
 * @param expr expresión en fnd.
 * @return expresiones separadas.
 */
public static ArrayList<ArrayList<Expresion>> analiza(String expr){
    if(expr.equals("*")){
        return new ArrayList<>();
    }
    ArrayList<ArrayList<String>> fndExp = fnd(expr);
    ArrayList<ArrayList<Expresion>> expresiones = new ArrayList<>();
    for(ArrayList<String> lista : fndExp){
        ArrayList<Expresion> listaExpr = new ArrayList<>();
        for(String token : lista){
            listaExpr.add(buildExpr(token));
        }
        expresiones.add(listaExpr);
    }
    return expresiones;
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# Parser.revisaCorrectas

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Revisa si las columnas dadas por el usuario son correctas.  
 * @param select la entrada del usuario con las columnas.  
 * @return true si la cadena tiene columnas validas, false en otro caso.  
 */  
public static boolean revisaCorrectas(String select) {  
    if(select.strip().equals("")){  
        return true;  
    }  
    String[] columnas = select.split(",");  
    String[] totales = {"idRating", "userId", "movieId", "rating", "timestamp", "title", "year", "genres"  
    List<String> listaColumnas = Arrays.asList(totales);  
    for(String c : columnas) {  
        String columna = c.strip();  
        if (!listaColumnas.contains(columna)) {  
            return false;  
        }  
    }  
    return true;  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# EstadisticaRating

Calificación de  
películas  
Códigos del proyecto

```
class EstadisticaRating implements Comparable<EstadisticaRating> {
    private final String titulo;
    private final float estadistica;
    private final String parametros;

    public EstadisticaRating(String titulo, float estadistica, String parametros){
        this.titulo = titulo;
        this.estadistica = estadistica;
        this.parametros = parametros;
    }

    @Override
    public int compareTo(EstadisticaRating valorRating) {
        return this.titulo.strip().compareTo(valorRating.getTitulo().strip());
    }

    public String getTitulo () {
        return this.titulo;
    }

    public float getEstadistica() {
        return this.estadistica;
    }

    public String getParametros() {
        return parametros;
    }
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GeneraEstadisticas.estadisticasRatingPromedio

Calificación de  
películas  
Códigos del proyecto

```
public static ArrayList<EstadisticaRating> estadisticasRatingPromedio() {  
    ArrayList<EstadisticaRating> estadisticaRating = new ArrayList<>();  
    File archivo = new File (Worker.DIR_RESULTADO);  
    try( FileReader fr = new FileReader(archivo);  
        BufferedReader br = new BufferedReader(fr)) {  
        String[] cols = br.readLine().split(",");  
        for (int i = 0; i < cols.length; i++) {  
            cols[i] = cols[i].strip();  
        }  
        ArrayList<String> columnas = new ArrayList<>(Arrays.asList(cols));  
        String registro;  
        ArrayList<String> titulos = new ArrayList<>();  
        ArrayList<Float> promedios = new ArrayList<>();  
        ArrayList<Integer> frecuencias = new ArrayList<>();  
        ArrayList<String> parametros = new ArrayList<>();  
  
        while ((registro = br.readLine()) != null) {  
            String[] valores = registro.split(",");  
            int index = columnas.indexOf("title");  
            String titulo = valores[index];  
            if (!titulos.contains(titulo)) {  
                titulos.add(titulo);  
                int indexRating = columnas.indexOf("rating");  
                float rating = Float.parseFloat(valores[indexRating].strip());  
                promedios.add(rating);  
                frecuencias.add(1);  
                String params = titulo;  
                for (int i = 0; i < columnas.size(); i++) {  
                    String columna = columnas.get(i);  
                    if (!columna.equals("title") && !columna.equals("rating") &&  
                        !columna.equals("age") && !columna.equals("name") &&  
                        !columna.equals("lastname")) {  
                        params += ", " + valores[i];  
                    }  
                }  
                parametros.add(params);  
            }  
        }  
    }  
}
```

Go to Line/Column

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GeneraEstadisticas.estadisticasRatingPromedio

Calificación de  
películas  
Códigos del proyecto

```
    .addParametros("ultimo"));
    params += ", " + valores[i];
}
parametros.add(params);
} else {
    int indexTitulo = titulos.indexOf(titulo);
    int freq = frecuencias.get(indexTitulo);
    float promedio = promedios.get(indexTitulo);
    frecuencias.set(indexTitulo, freq + 1);
    int indexRating = columnas.indexOf("rating");
    float rating = Float.parseFloat(valores[indexRating].strip());
    promedios.set(indexTitulo, promedio + rating);
}
}

for (int i = 0; i < titulos.size() ;i++) {
    String titulo = titulos.get(i);
    int frecuencia = frecuencias.get(i);
    float suma = promedios.get(i);
    float promedio = suma / frecuencia;
    promedios.set(i, promedio);
    String valores = parametros.get(i);
    estadisticaRating.add(new EstadisticaRating(titulo, promedio, valores));
}
} catch (FileNotFoundException e) {
    System.out.printf("No se puede leer el archivo %s%n", Worker.DIR_RESULTADO);
} catch (Exception e) {
    System.out.println(e);
    e.printStackTrace();
}
Collections.sort(estadisticaRating);
return estadisticaRating;
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GeneraEstadisticas.estadisticasRatingMinimo

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
public static ArrayList<EstadisticaRating> estadisticasRatingMinimo(boolean calculaMinimo) {  
    ArrayList<EstadisticaRating> estadisticaRating = new ArrayList<>();  
    File archivo = new File (Worker.DIR_RESULTADO);  
    try( FileReader fr = new FileReader(archivo);  
        BufferedReader br = new BufferedReader(fr)) {  
        String[] cols = br.readLine().split(",");  
        for (int i = 0; i < cols.length; i++) {  
            cols[i] = cols[i].strip();  
        }  
        ArrayList<String> columnas = new ArrayList<>(Arrays.asList(cols));  
        String registro;  
        ArrayList<String> titulos = new ArrayList<>();  
        ArrayList<Float> minimos = new ArrayList<>();  
        ArrayList<String> parametros = new ArrayList<>();  
  
        while ((registro = br.readLine()) != null) {  
            String[] valores = registro.split(",");  
            int index = columnas.indexOf("title");  
            String titulo = valores[index];  
            if (!titulos.contains(titulo)) {  
                titulos.add(titulo);  
                int indexRating = columnas.indexOf("rating");  
                float rating = Float.parseFloat(valores[indexRating].strip());  
                minimos.add(rating);  
                String params = titulo;  
                for (int i = 0; i < columnas.size(); i++) {  
                    String columna = columnas.get(i);  
                    if (!columna.equals("title") && !columna.equals("rating") &&  
                        !columna.equals("age") && !columna.equals("name") &&  
                        !columna.equals("lastname")) {  
                        params += ", " + valores[i];  
                    }  
                }  
                parametros.add(params);  
            }  
        }  
    }  
}
```

# GeneraEstadisticas.estadisticasRatingMinimo

Calificación de  
películas  
Códigos del proyecto

```
        params += ", " + valores[i];
    }
    parametros.add(params);
} else {
    int indexTitulo = titulos.indexOf(titulo);
    Float min = minimos.get(indexTitulo);
    int indexRating = columnas.indexOf("rating");
    float rating = Float.parseFloat(valores[indexRating].strip());
    if (calculaMinimo) {
        if (rating < min) {
            | minimos.set(indexTitulo, rating);
        }
    } else {
        if (rating > min) {
            | minimos.set(indexTitulo, rating);
        }
    }
}
for (int i = 0; i < titulos.size() ;i++) {
    String titulo = titulos.get(i);
    float minimo = minimos.get(i);
    String valores = parametros.get(i);
    estadisticaRating.add(new EstadisticaRating(titulo, minimo, valores));
}
} catch (FileNotFoundException e) {
    | System.out.printf("No se puede leer el archivo %s%n", Worker.DIR_RESULTADO);
} catch (Exception e) {
    | e.printStackTrace();
}
Collections.sort(estadisticaRating);
return estadisticaRating;
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

## GeneraEstadisticas.estadisticasRatingMediana

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
**EstadisticaRating**  
GraficaBarras

```
public static ArrayList<EstadisticaRating> estadisticasRatingMediana() {
    ArrayList<EstadisticaRating> estadisticaRating = new ArrayList<>();
    File archivo = new File (Worker.DIR_RESULTADO);
    try( FileReader fr = new FileReader(archivo);
        BufferedReader br = new BufferedReader(fr)) {
        String[] cols = br.readLine().split(",");
        for (int i = 0; i < cols.length; i++) {
            cols[i] = cols[i].strip();
        }
        ArrayList<String> columnas = new ArrayList<>(Arrays.asList(cols));
        String registro;
        ArrayList<String> titulos = new ArrayList<>();
        ArrayList<ArrayList<Float>> ratingsPorTitulo = new ArrayList<>();
        ArrayList<String> parametros = new ArrayList<>();

        while ((registro = br.readLine()) != null) {
            String[] valores = registro.split(",");
            int index = columnas.indexOf("title");
            String titulo = valores[index];
            if (!titulos.contains(titulo)) {
                titulos.add(titulo);
                int indexRating = columnas.indexOf("rating");
                float rating = Float.parseFloat(valores[indexRating].strip());
                ArrayList<Float> ratingsTitulo = new ArrayList<>();
                ratingsTitulo.add(rating);
                ratingsPorTitulo.add(ratingsTitulo);
                String params = titulo;
                for (int i = 0; i < columnas.size(); i++) {
                    String columna = columnas.get(i);
                    if (!columna.equals("title") && !columna.equals("rating") &&
                        !columna.equals("age") && !columna.equals("name") &&
                        !columna.equals("lastname")) {
                        params += ", " + valores[i];
                    }
                }
                estadisticaRating.add(new EstadisticaRating(params, rating));
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

# GeneraEstadisticas.estadisticasRatingMediana

Calificación de  
películas  
Códigos del proyecto

```
        for (int i = 0; i < columnas.size(); i++) {
            String columna = columnas.get(i);
            if (!columna.equals("title") && !columna.equals("rating") &&
                !columna.equals("age") && !columna.equals("name") &&
                !columna.equals("lastname")) {
                params += ", " + valores[i];
            }
        }
        parametros.add(params);
    } else {
        int indexTitulo = titulos.indexOf(title);
        int indexRating = columnas.indexOf("rating");
        float rating = Float.parseFloat(valores[indexRating].strip());
        ArrayList<Float> ratings = ratingsPorTitulo.get(indexTitulo);
        ratings.add(rating);
        ratingsPorTitulo.set(indexTitulo, ratings);
    }
}
for (int i = 0; i < titulos.size() ;i++) {
    String titulo = titulos.get(i);
    ArrayList<Float> ratings = ratingsPorTitulo.get(i);
    Collections.sort(ratings);
    float media = ratings.get(ratings.size()/2);
    String valores = parametros.get(i);
    estadisticaRating.add(new EstadisticaRating(titulo, media, valores));
}
} catch (FileNotFoundException e) {
    System.out.printf("No se puede leer el archivo %s%n", Worker.DIR_RESULTADO);
} catch (Exception e) {
    e.printStackTrace();
}
Collections.sort(estadisticaRating);
return estadisticaRating;
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

# GraficaBarras.builder

Calificación de  
películas  
Códigos del proyecto

```
public class GraficaBarras {  
  
    private final static int ELEMENTOS_PAGINA = 5;  
  
    private int numResultados;  
  
    private int indexInclusive;  
  
    private int indexExclusive;  
  
    private ArrayList<EstadisticaRating> promediosRating;  
    private ArrayList<EstadisticaRating> minimosRating;  
    private ArrayList<EstadisticaRating> maximosRating;  
    private ArrayList<EstadisticaRating> medianasRating;  
  
    private BarChart barChart;  
  
    /*  
     * Crea la grafica de barras tomando un rango de datos.  
     */  
    public GraficaBarras() {  
        NumberAxis xAxis = new NumberAxis();  
        CategoryAxis yAxis = new CategoryAxis();  
        this.indexInclusive = -ELEMENTOS_PAGINA;  
        this.indexExclusive = 0;  
        this.barChart = new BarChart<>(xAxis, yAxis);  
        this.barChart.setAnimated(false);  
    }  
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GraficaBarras.avanza

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

```
/**  
 * Muestra los siguientes N registros.  
 */  
public void avanza() {  
    if (this.promediosRating == null) {  
        return;  
    }  
    if (this.promediosRating.isEmpty()) {  
        return;  
    }  
    this.indexInclusive += ELEMENTOS_PAGINA;  
    this.indexExclusive += ELEMENTOS_PAGINA;  
    if (this.indexExclusive > this.promediosRating.size()) {  
        this.indexExclusive = this.promediosRating.size();  
        this.indexInclusive = (this.promediosRating.size() / ELEMENTOS_PAGINA) * ELEMENTOS_PAGINA;  
        if (this.indexInclusive == this.indexExclusive) {  
            this.indexInclusive -= ELEMENTOS_PAGINA;  
            if (this.indexInclusive < 0) {  
                this.indexInclusive = 0;  
            }  
        }  
    }  
    actualizaDatos();  
}
```

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GraficaBarras.retrocede

Calificación de  
películas  
Códigos del proyecto

```
/**  
 * Muestra los N registros anteriores.  
 */  
public void retrocede() {  
    if (this.promediosRating == null) {  
        return;  
    }  
    if (this.promediosRating.isEmpty()) {  
        return;  
    }  
    if (this.indexExclusive % ELEMENTOS_PAGINA != 0) {  
        this.indexExclusive -= this.indexExclusive % ELEMENTOS_PAGINA;  
        this.indexInclusive = this.indexExclusive - ELEMENTOS_PAGINA;  
    } else {  
        this.indexExclusive -= ELEMENTOS_PAGINA;  
        this.indexInclusive -= ELEMENTOS_PAGINA;  
        if (this.indexInclusive < 0) {  
            this.indexInclusive = 0;  
            if (ELEMENTOS_PAGINA > this.promediosRating.size()) {  
                this.indexExclusive = this.promediosRating.size();  
            } else {  
                this.indexExclusive = ELEMENTOS_PAGINA;  
            }  
        }  
    }  
    actualizaDatos();  
}
```

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

# GraficaBarras.reset

Calificación de  
películas  
Códigos del proyecto

Emmanuel Peto  
Gutiérrez  
Rodrigo Fernando  
Velázquez Cruz

Main  
Manager  
Worker  
ComparadorEnum  
Divisor  
Expresion  
Parser  
EstadisticaRating  
GraficaBarras

```
/**  
 * Calcula otro archivo.  
 */  
public void reset() {  
    this.promediosRating = GeneraEstadisticas.estadisticasRatingPromedio();  
    this.medianasRating = GeneraEstadisticas.estadisticasRatingMediana();  
    this.maximosRating = GeneraEstadisticas.estadisticasRatingMinimo(calculaMinimo: false);  
    this.minimosRating = GeneraEstadisticas.estadisticasRatingMinimo(calculaMinimo: true);  
    int elementos = promediosRating.size();  
    if (elementos == 0) {  
        this.indexInclusive = -ELEMENTOS_PAGINA;  
        this.indexExclusive = 0;  
        this.barChart.setTitle("Sin resultados");  
        this.barChart.setData(FXCollections.observableArrayList());  
        return;  
    }  
    if (medianasRating.size() != elementos ||  
        maximosRating.size() != elementos ||  
        minimosRating.size() != elementos) {  
        return;  
    }  
    this.numResultados = elementos;  
    this.indexInclusive = -ELEMENTOS_PAGINA;  
    this.indexExclusive = 0;  
    avanza();  
}
```

Main

Manager

Worker

ComparadorEnum

Divisor

Expresion

Parser

EstadisticaRating

GraficaBarras

```
private ObservableList<XYChart.Series<Double, String>> obtenDatos() {
    ArrayList<EstadisticaRating> topPromedio = new ArrayList<> (this.promediosRating.subList(this.indexIncluir));
    ArrayList<EstadisticaRating> topMediana = new ArrayList<> (this.medianasRating.subList(this.indexIncluir));
    ArrayList<EstadisticaRating> topMinimo = new ArrayList<> (this.minimosRating.subList(this.indexIncluir));
    ArrayList<EstadisticaRating> topMaximo = new ArrayList<> (this.maximosRating.subList(this.indexIncluir));

    ObservableList<XYChart.Series<Double, String>> data = FXCollections.observableArrayList();
    XYChart.Series<Double, String> seriePromedio = new XYChart.Series<>();
    XYChart.Series<Double, String> serieMediana = new XYChart.Series<>();
    XYChart.Series<Double, String> serieMinimo = new XYChart.Series<>();
    XYChart.Series<Double, String> serieMaximo = new XYChart.Series<>();

    seriePromedio.setName("Rating promedio");
    serieMediana.setName("Rating mediana");
    serieMinimo.setName("Rating minimo");
    serieMaximo.setName("Rating maximo");

    for (EstadisticaRating promedioRating : topPromedio) {
        seriePromedio.getData().add(new XYChart.Data<> ((double) promedioRating.getEstadistica(), promedioRating.getRating()));
    }
    for (EstadisticaRating medianaRating : topMediana) {
        serieMediana.getData().add(new XYChart.Data<> ((double) medianaRating.getEstadistica(), medianaRating.getRating()));
    }
    for (EstadisticaRating minimoRating : topMinimo) {
        serieMinimo.getData().add(new XYChart.Data<> ((double) minimoRating.getEstadistica(), minimoRating.getRating()));
    }
    for (EstadisticaRating maximoRating : topMaximo) {
        serieMaximo.getData().add(new XYChart.Data<> ((double) maximoRating.getEstadistica(), maximoRating.getRating()));
    }
    data.addAll(seriePromedio, serieMediana, serieMinimo, serieMaximo);
    return data;
}
```