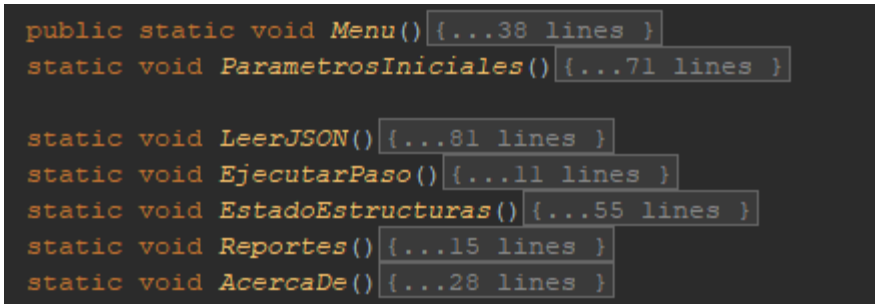
Manual Técnico

Requerimientos:

- **OS:** Windows 10 Pro 21H2
- **Procesador:** Intel(R) Core (TM) i5-4310M CPU @ 2.70GHz
- **RAM:** 8GB

Funcionamiento del Programa:

1. **Cosas Generales:** Se hizo uso de Programación Orientada a Objetos así como Programación Funcional mediante el uso de Métodos y Funciones para lograr la ejecución óptima del proyecto, estos son los métodos utilizados:



Y estas son las clases utilizadas:

Como se puede observar es una clase por cada nodo y una por cada estructura utilizada.

2. **Menús:** Todos los menús del proyecto fueron implementados de la misma forma, desplegando las opciones en pantalla seguidas de un Scanner que lee la opción del usuario y un switch que le dice al programa qué método ejecutar:

```
System.out.println("6) Salir");

Scanner LeerOpcionMenu = new Scanner (System.in);
OpcionMenuP = LeerOpcionMenu.nextInt();
switch(OpcionMenuP){
    case 1:
        ParametrosIniciales();
        break;
    case 2:
        EjecutarPaso();
        break;
    case 3:
        EstadoEstructuras();
        break;
    case 4:
        Reportes();
        break;
    case 5:
        AcercaDe();
        break;
    case 6:
        System.exit(0);
        break;
    default:
        System.out.println("Solo puede elegir entre las opciones que ve en pantalla, Intentelo de nuevo");
        System.out.println(""+"");
        Menu();
        break;
}
```

3. **Parámetros Iniciales:** Solo se muestra el menú correspondiente, en esencia, funciona igual al Menú Principal:

```
static void ParametrosIniciales(){
    //Línea Extra para más orden
    System.out.println("");

    int OpcionMenuPIniciales;
    System.out.println("Parámetros Iniciales, Ingrese la opción que desee:");
    System.out.println("1) Carga Masiva de Clientes");
    System.out.println("2) Numero de Ventanillas");
    System.out.println("3) Volver al Menú Principal");

    //Crear Scanner
    Scanner LeerOpcionMenu = new Scanner (System.in);
    OpcionMenuPIniciales = LeerOpcionMenu.nextInt();

    switch(OpcionMenuPIniciales){
        case 1:
            LeerJSON();
            break;
        case 2:
            //Línea Extra para más orden
            System.out.println("");
```

4. **Leer JSON:** Para poder leer el archivo JSON se utilizó la librería JSON Simple, la cual está instalada en las dependencias del programa:

Debido a la dificultad de trabajar con Maven se creó una dependencia con un nombre arbitrario, pero el jar que contiene es la librería JSON Simple la cuál nos permite tener métodos que nos facilitan la lectura del archivo JSON, funciona de manera similar a usar ElementTree para leer archivos XML en Python:

```
//Leer el JSON
JSONParser parser = new JSONParser();

try{
    Object obj = parser.parse(new FileReader(RutaJSON));
    JSONObject jsonObject = (JSONObject) obj;
    System.out.println("Este es el JSON Leido: " + jsonObject);

    //Línea en blanco para más orden
    System.out.println("");

    //Obtener las etiquetas del JSON
    for (int i = 1; i <= jsonObject.size(); i++) {
        //Crear Objeto Cliente del JSON
        JSONObject ClienteActual = (JSONObject) jsonObject.get("Cliente" + i);

        //Obtener Parametros del Cliente

        //Crear Variables que guarden los parametros del cliente
        int id_clienteA = 0;
        String nombre_clienteA = new String();
        int img_colorA = 0;
        int img_bwA=0;

        //Guardar los datos del cliente actual en las Variables
        id_clienteA = Integer.valueOf((String)ClienteActual.get("id_cliente"));
        nombre_clienteA = String.valueOf(ClienteActual.get("nombre_cliente"));
        img_colorA = Integer.valueOf((String)ClienteActual.get("img_color"));
        img_bwA = Integer.valueOf((String)ClienteActual.get("img_bw"));
```

Una vez leídas las etiquetas se almacenan en la cola de Recepción, la cual veremos a fondo más adelante, así se almacenan estos datos en la lista:

```
//Guardar los datos en la lista
ColaRecepcion.push(id_clienteA, nombre_clienteA, img_colorA, img_bwA);
System.out.println("Datos Guardados en la Cola de Recepción");
```

5. **Número de Ventanillas:** Para esta función se utilizó el mismo switch del menú de Parámetros Iniciales debido a que es una función corta que solo le asigna el valor que ingrese el usuario a una Variable Global llamada “NumeroVentanillas” y crea la cantidad de nodos ingresada en la lista de ventanillas. Cabe mencionar que cada Lista de Ventanillas tiene una Pila de Imágenes vacía para contener las imágenes que el cliente desee imprimir.

```
case 2:
    //Línea Extra para más orden
    System.out.println("");

    System.out.println("Ingrese la cantidad de Ventanillas que desea: ");
    Scanner SCVentanillas = new Scanner(System.in);
    NumeroVentanillas = SCVentanillas.nextInt();

    System.out.println("La Cantidad de Ventanillas ingresda es: " + NumeroVentanillas);

    //Crear ese numero de ventanillas en la Lista de Ventanillas
    for (int i = 1; i <= NumeroVentanillas; i++) {

        PImgVentanilla Pilai = new PImgVentanilla();
        ListaVentanillas.AgregarAlFinal(i, 0, false, Pilai, null);

    }

    //Confirmación para volver al menú principal
    System.out.println("\nIngrese cualquier valor para volver al menú de Parámetros Iniciales: ");

    //Leer la confirmación
    Scanner LeerVCVentanillas = new Scanner (System.in);
    String OLV = LeerVCVentanillas.nextLine();

    //Volver al Menú
    ParametrosIniciales();

    break;
```

6. **Estado Estructuras:** Este método genera un grafo de las estructuras en tiempo real, por conveniencia el recorrer las listas es un método que se encuentra en las listas en sí, esto es debido a que por la complejidad del programa iba a ser muy laborioso y confuso recorrerlas todas en el mismo método, entonces cada lista solo genera subgrafos que contienen el cluster de sus nodos, y este método, EstadoEstructuras, solo genera las partes de código faltantes para generar el grafo completo. Así mismo genera el archivo .dot y la imagen .jpg en la dirección del proyecto:

```
static void EstadoEstructuras(){
    //Línea Extra para más orden
    System.out.println("");

    //Informar de Generación de Grafo
    System.out.println("Generando Grafo de las Estructuras...");

    //Iniciar el Codigo de GraphViz
    String CodigoGraphViz = "digraph G {\n";

    //Agregar los codigos de las listas
    CodigoGraphViz = CodigoGraphViz + ColaRecepcion.GenerarCodigoGraphViz();

    CodigoGraphViz = CodigoGraphViz + ListaVentanillas.GenerarCodigoGraphViz();

    //Terminar el Codigo de GraphViz
    CodigoGraphViz = CodigoGraphViz + "\n}";
```

Y así se generan el archivo y la imagen:

```
//Generar el Archivo del grafo
try {
    //Craer Archivo
    FileWriter ArchivoGraphViz = new FileWriter("GrafoEstadoMemoria.dot");
    System.out.println("Archivo Creado");
    //Escribir en el archivo
    ArchivoGraphViz.write(CodigoGraphViz);
    ArchivoGraphViz.close();
    System.out.println("Se ha escrito en el archivo de GraphViz correctamente");
} catch (IOException e) {
    System.out.println("Ha ocurrido un error :'v");
    e.printStackTrace();
}

//Convertir de dot a jpg
try {
    //Crear la imagen a partir del archivo dot
    Process p = Runtime.getRuntime().exec("dot -Tjpg GrafoEstadoMemoria.dot -o GrafoEstadoMemoria.jpg");

} catch (IOException e) {
    System.out.println("Ha ocurrido un error :(");
    e.printStackTrace();
}

System.out.println("La imágen del grafo se ha generado");
```

7. **Acerca De:** Solo despliega los datos del creador con sout:

```
static void AcercaDe(){
    //Línea Extra para más orden
    System.out.println("");

    //Crear Línea de Arriba
    for (int i=0;i<45;i++){
        System.out.print("-");
    }
    System.out.println("\n|Nombre: Bryan Steve Montepeque Santos      |");
    System.out.println("|Carnet: 201700375                          |");
    System.out.println("|Curso: Estructuras de Datos                |");
    System.out.println("|Sección: C                                 |");

    //Crear Línea de Abajo
    for (int i=0;i<45;i++){
        System.out.print("-");
    }

    //Confirmación para volver al menú principal
    System.out.println("\nIngrese cualquier valor para volver al menú principal: ");

    //Leer la confirmación
    Scanner LeerVMenu = new Scanner (System.in);
    String OVM = LeerVMenu.nextLine();

    //Volver al Menú
    Menu();
}
```

8. **Salir:** Cierra el programa con system.exit(0);

```
case 6:
    System.exit(0);
    break;
```

9. **Lista Simple:** Debemos empezar con los nodos, los nodos de todas las estructuras funcionan igual, son un objeto que guarda datos, debido a esto todas las clases nodo tienen la misma estructura, solo cambian sus parámetros, aquí pueden ver el nodo de la clase CRecepcion (Es decir, “ColaRecepcion”) Esta es la clase NodoCRecepcion:

```
public class NodoCRecepcion {
    private int id_cliente;
    private String nombre_cliente;
    private int img_color;
    private int img_bw;
    private NodoCRecepcion Siguiente;

    public NodoCRecepcion(int id_cliente, String nombre_cliente, int img_color, int img_bw) {
        this.id_cliente = id_cliente;
        this.nombre_cliente = nombre_cliente;
        this.img_color = img_color;
        this.img_bw = img_bw;
    }

    /**
     * @return the id_cliente
     */
    public int getId_cliente() {
        return id_cliente;
    }
```

Todo lo demás de la clase de los nodos son sets y gets para poder manejarlos desde otras clases o utilizarlos con mayor facilidad.

10.Cola Recepción: Esta es la clase ColaRecepcion:

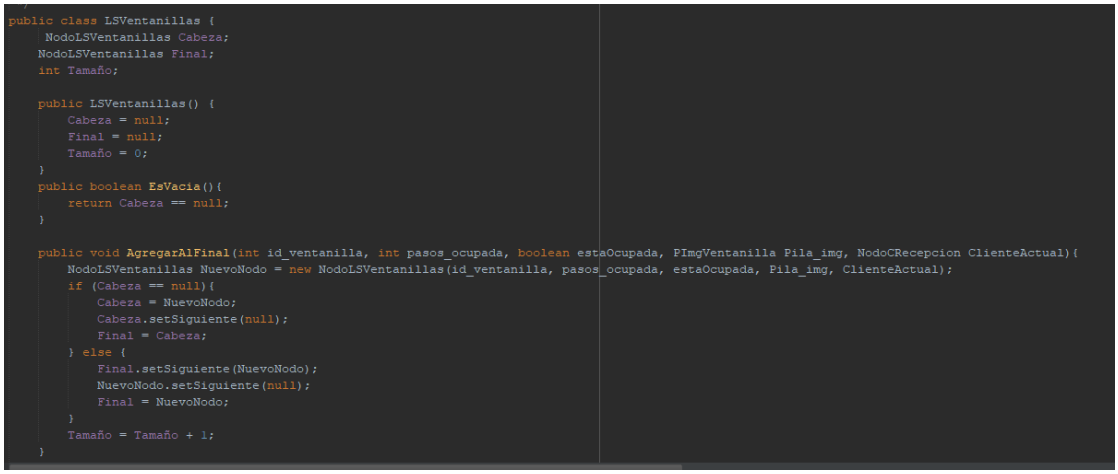
```java
public class CRecepcion {
    private NodoCRecepcion Cabeza;
    int Tamaño;

    public CRecepcion() {
        this.Cabeza = null;
        this.Tamaño = 0;
    }

    public boolean estaVacia() {
        return Cabeza == null;
    }

    public void push(int id_cliente, String nombre_cliente, int img_color, int img_bw){
        NodoCRecepcion NuevoNodo = new NodoCRecepcion(id_cliente, nombre_cliente, img_color, img_bw);
        if(this.getCabeza() == null){
            this.setCabeza(NuevoNodo);
        }else{
            NodoCRecepcion Auxiliar = this.getCabeza();
            while(Auxiliar.getSiguiente() != null){
               Auxiliar = Auxiliar.getSiguiente();
            }
            Auxiliar.setSiguiente(NuevoNodo);
        }
        Tamaño++;
    }
```

```java
public void LimpiarCola() {
    while (!estaVacia()) {
        pop();
    }
}

public NodoCRecepcion pop(){
    NodoCRecepcion Auxiliar = null;
    NodoCRecepcion NodoRetornado = null;

    if(this.getCabeza() != null){

        Auxiliar = this.getCabeza();
        NodoRetornado = this.getCabeza();

        if(this.getCabeza().getSiguiente() != null){
            this.setCabeza(Auxiliar.getSiguiente());
        }else{
            this.setCabeza(null);
        }
        Tamaño--;
    }
    System.out.println("No hay datos");
    return NodoRetornado;
}
```

Cada Estructura tiene una función String como esta que recorre la lista, genera del código del Subgrafo de GraphViz y devuelve un String con ese código a donde la hayan llamdo, es gracias a estos métodos que fue más sencillo graifcar todas las estructuras en el método de EstadoEstructuras donde se grafican todas las Estructuras del programa.

```
public String GenerarCodigoGraphViz() {
    if (Cabeza == null) {
        return "La Cola está vacia.";
    } else {
        String CInicial, CIntermedio, CFinal;
        CInicial = "subgraph cluster_Recepcion {" + "\n" + "label=\"Recepción\"\n" + "R0[label=\"INICIO\"];"+"\n"+"R0->R"+Cabeza.getId_cliente()+";"+"\n
        CIntermedio = CInicial;
        NodoCRecepcion Auxiliar = Cabeza;
        int IDAnterior;
        CIntermedio = CIntermedio + "R" + Auxiliar.getId_cliente() + "[shape=\"box\" label=\"ID_Cliente: " + Auxiliar.getId_cliente() + "\nNombre_Client
        while (Auxiliar.getSiguiente() != null) {
            IDAnterior = Auxiliar.getId_cliente();
            Auxiliar = Auxiliar.getSiguiente();
            CIntermedio = CIntermedio + "R" + Auxiliar.getId_cliente()+ "[shape=\"box\" label=\"ID_Cliente: " + Auxiliar.getId_cliente() + "\nNombre_Cli
            CIntermedio = CIntermedio + "R" + IDAnterior + " -> R" + Auxiliar.getId_cliente() + ";" + "\n\n";
        }
        CFinal = CIntermedio + "}";
        return CFinal;
    }
}

/**
```

11. **Lista Simple Ventanillas:** Esta es la clase de la Lista Simple Ventanillas:



```
public class LSVentanillas {
    NodoLSVentanillas Cabeza;
    NodoLSVentanillas Final;
    int Tamaño;

    public LSVentanillas() {
        Cabeza = null;
        Final = null;
        Tamaño = 0;
    }
    public boolean EsVacia(){
        return Cabeza == null;
    }

    public void AgregarAlFinal(int id_ventanilla, int pasos_ocupada, boolean estaOcupada, PImgVentanilla Pila_img, NodoCRecepcion ClienteActual){
        NodoLSVentanillas NuevoNodo = new NodoLSVentanillas(id_ventanilla, pasos_ocupada, estaOcupada, Pila_img, ClienteActual);
        if (Cabeza == null){
            Cabeza = NuevoNodo;
            Cabeza.setSiguiente(null);
            Final = Cabeza;
        } else {
            Final.setSiguiente(NuevoNodo);
            NuevoNodo.setSiguiente(null);
            Final = NuevoNodo;
        }
        Tamaño = Tamaño + 1;
    }
```

```java
public boolean Buscar (int ID){
    NodoLSVentanillas Auxiliar = Cabeza;
    boolean Encontrado = false;
    while (Auxiliar != null && Encontrado != true){
        if(ID == Auxiliar.getId_ventanilla()){
            Encontrado = true;
        } else {
            Auxiliar.getSiguiente();
        }
    }
    return Encontrado;
}
public void EliminarPorPosicion(int ID){
    NodoLSVentanillas Auxiliar = Cabeza;
    NodoLSVentanillas Anterior = null;
    while(Auxiliar != null){
        if (Auxiliar.getId_ventanilla() == ID){
            if(Auxiliar == Cabeza){
               Cabeza = Cabeza.getSiguiente();
            } else {
                Anterior.setSiguiente(Auxiliar.getSiguiente());
            }
        }
        Anterior = Auxiliar;
        Auxiliar = Auxiliar.getSiguiente();
    }
}
```

```java
public void LimpiarLista(){
    Cabeza = null;
    Final = null;
    Tamaño = 0;
}

public String GenerarCodigoGraphViz() {
    if (Cabeza == null){
        return "La Lista Simple está vacia.";
    } else {
        String CInicial, CIntermedio, CFinal;
        CInicial = "subgraph cluster_Ventanillas {" + "\n" +"label=\"Ventanillas\"\n";
        CIntermedio = CInicial;
        NodoLSVentanillas Auxiliar = Cabeza;

        int IDAnterior;
        CIntermedio = CIntermedio + "V" + Auxiliar.getId_ventanilla() + "[shape=\"box\" label=\"ID_Ventnailla: " + Auxiliar.getId_ventanilla() + "\nP
        while (Auxiliar.getSiguiente() != null) {
            IDAnterior = Auxiliar.getId_ventanilla();
            Auxiliar = Auxiliar.getSiguiente();
            CIntermedio = CIntermedio + "V" + Auxiliar.getId_ventanilla() + "[shape=\"box\" label=\"ID_Ventnailla: " + Auxiliar.getId_ventanilla() +
            CIntermedio = CIntermedio + "V" + IDAnterior + " -> V" + Auxiliar.getId_ventanilla() + ";" + "\n\n";
        }
        CFinal = CIntermedio + "}";

        return CFinal;
    }
}
```

12. **Pila Imágenes:** Esta es la clase de la Pila de Imágenes que está contenida en cada Ventanilla:

```
public class PImgVentanilla {
     NodoPImgVentanilla Cima;
    int Tamaño;

    public PImgVentanilla() {
        Cima = null;
        Tamaño = 0;
    }

    public boolean estaVacia(){
        return Cima == null;
    }

    public void push(int id_img, int id_clienteDueño, String tipo_de_img){//Agrega Valores a la Pila
        NodoPImgVentanilla NuevoNodo = new NodoPImgVentanilla(id_img, id_clienteDueño, tipo_de_img);
        NuevoNodo.setSiguiente(Cima);
        Cima = NuevoNodo;
        Tamaño++;
    }

    public NodoPImgVentanilla peek(){//Muestra el Ultimo Valor y ya
        return Cima;
    }

    public int TamañoPila(){
        return Tamaño;
    }
```

```
    public void LimpiarPila(){
        while(!estaVacia()){
            pop();
        }
    }

    public void pop(){//Muestra y elimina el Ultimo valor, en este caso solo Elimina
        Cima = Cima.getSiguiente();
        Tamaño--;
    }

    public String GenerarCodigoGraphViz() {
        if (Cima == null){
            return "La Pila está vacia.";
        } else {
            String CInicial, CIntermedio, CFinal;
            int IDCima = Tamaño + 1;
            CInicial = "subgraph PilaImgenes {" + "label=\"Pila Imagenes\"\n" + "\nPImg" + IDCima + "[label=\"CIMA\"];"+"\n\n";
            CIntermedio = CInicial;
            NodoPImgVentanilla Auxiliar = Cima;
            int IDAnterior;
            CIntermedio = CIntermedio + "PImg" + Auxiliar.getId_img() + "[shape=\"box\" label=\"ID Imagen: " + Auxiliar.getId_img() + "\nID Cliente: " + Aux
            while (Auxiliar.getSiguiente() != null) {
                IDAnterior = Auxiliar.getId_img();
                Auxiliar = Auxiliar.getSiguiente();
                CIntermedio = CIntermedio + "PImg" + Auxiliar.getId_img() + "[shape=\"box\" label=\"ID Imagen: " + Auxiliar.getId_img() + "\nID Cliente: " +
                CIntermedio = CIntermedio + "PImg" + IDAnterior + "-> PImg" + Auxiliar.getId_img() + ";" + "\n\n";
            }
            CFinal = CIntermedio +"PImg1 -> PImg0;"+"\n"+ "0[label=\"TOPE\"];"+"\n"+"}";
```