Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructura de Datos

MANUAL TÉCNICO

Henry Ronely Mendoza Aguilar
Carné: 202004810
PROYECTO FASE 1

Objetivos y Alcances del Sistema

Objetivo Principal

- Brindar la gestión eficaz de los procesos realizaos dentro de la empresa Drawing Paper

Objetivos Específicos

- Diseñar una plataforma agradable e intuitiva por medio del lenguaje de programación Java.
- Aprovechar de mejor maneral los recursos brindados por el entorno de trabajo NetBeans
- Programar un sistema funcional para cualquier usuario y hardware en el que se utilice.

Requisitos del Hardware

- 512 MB de RAM
- 20MB de disco duro para almacenamiento del software (ya que puede aumentar con la generación de imágenes).
- Procesador Pentium 2 a 266Mhz

Requisitos del Software

- Java
- Graphviz
- Windows 10 (64-bit), Linux o Mac OS

Librerías Empleadas

Graphviz:

Graphviz (abreviatura de Graph Visualization Software) es un paquete de herramientas de código abierto iniciado por AT&T Labs Research para dibujar gráficos especificados en scripts de lenguaje DOT que tienen la extensión de nombre de archivo "gv". También proporciona bibliotecas para que las aplicaciones de software utilicen las herramientas. Graphviz es un software gratuito con licencia de Eclipse Public License.

Gson:

(también conocido como Google Gson) es una biblioteca Java de código abierto para serializar y deserializar objetos Java a (y desde) JSON. La biblioteca Gson se desarrolló originalmente para fines internos de Google, y la versión 1.0 se lanzó más tarde el 22 de mayo de 2008 bajo los términos de la Licencia Apache 2.0. La última versión, 2.8.8, se lanzó el 20 de agosto de 2021.

Formato JSON:

JSON (JavaScript Object Notation) es un formato de archivo estándar abierto y un formato de intercambio de datos que utiliza texto legible por humanos para almacenar y transmitir objetos de datos que consisten en pares atributo-valor y arreglos (u otros valores serializables). Es un formato de datos común con diversos usos en el intercambio electrónico de datos, incluido el de aplicaciones web con servidores.

JSON es un formato de datos independiente del idioma. Se derivó de JavaScript, pero muchos lenguajes de programación modernos incluyen código para generar y analizar datos en formato JSON. Los nombres de archivo JSON usan la extensión. json.

Descripción de Métodos

Para el desarrollo de la aplicación se emplearon los métodos de lectura de archivos de texto plano que posee Java las cuales son las pertenecientes al paquete IO estos junto a la librería Gson se utilizaron para la función “read ()” el cual es el encargado de abrir el archivo en formato json y crear una Cola con objetos de tipo Cliente.

```
File file;
FileReader fr;
BufferedReader br;
String content = "";
try{
    file = new File(root);
    fr = new FileReader(file);
    br = new BufferedReader(fr);
    String line;
    while ((line = br.readLine()) != null) {
        content += line;
    }
    String json = "{"+content;

    json = json.replaceAll("\\{\\{", "[");
    json = json.replaceAll("}}", "}]");
    json = json.replaceAll("\"Cliente[0-9]*\":", "");

    JsonParser parser = new JsonParser();
    JsonArray clientes = parser.parse(json).getAsJsonArray();

    System.out.println("Cantidad de Clientes en la cola: " + clientes.size());
    index = clientes.size()+1;

    Cola filac = new Cola();

    for(int i=0;i<clientes.size();i++){
        JsonObject c = clientes.get(i).getAsJsonObject();
        Cliente aux = new Cliente(c.get("id_cliente").getAsString(),
                                  c.get("nombre_cliente").getAsString(),
                                  c.get("img_color").getAsString(),
                                  c.get("img_bw").getAsString(),"");
        aux.total=Integer.parseInt(aux.img_bw)+Integer.parseInt(aux.img_c);
        filac.append(aux);
    }

    return filac;
```

Función creat_v(): en esta función se realiza la creación de una “Lista de Pilas” las cuales serán las ventanillas del sistema.

```
public static ventanilla create_v(){

    System.out.println("Ingrese el número de ventanilas: ");
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();

    try{
        ventanilla v = new ventanilla();
        for(int i=0;i<n;i++){

            Pila p = new Pila();
            v.append(p, (i+1));
        }
        return v;

    }catch(Exception e){System.out.println(e);}

    return null;
}
```

Función paso: Dentro de esta función se realizan los pasos de ingreso a la ventanilla y la entrega de las imágenes a la ventanilla de parte de los clientes.

```
String paso="----------Paso "+contador+"----------\n";
Nodo aux = Ventanas.first;

 while(aux!=null){
    if(aux.clt==null && clientes.first!=null){
       aux.clt = (Cliente) clientes.first.data;
       aux.clt.setVentanilla(String.valueOf(aux.id));
       clientes.remove();
       paso+="EL CLIENTE "+aux.clt.id+" INGRESA A LA VENTANILLA "+aux.id+"\n";
       break;
    }else{
        if(aux.clt != null){
            int color = Integer.parseInt(aux.clt.getImg_c());
            int bw = Integer.parseInt(aux.clt.getImg_bw());
            if(bw>0){
                aux.pila.push("img_bw");
                paso+="LA VENTANILLA "+aux.id+" RECIBE UNA IMAGEN DEL CLIENTE "+aux.clt.id+"\n";
                bw--;
                aux.clt.setImg_bw(String.valueOf(bw));
            }else if(bw==0 && color>0){
                aux.pila.push("img_c");
                paso+="LA VENTANILLA "+aux.id+" RECIBE UNA IMAGEN DEL CLIENTE "+aux.clt.id+"\n";
                color--;
                aux.clt.setImg_c(String.valueOf(color));
            }else if(color==0 && bw == 0){
                Lista l_aux = new Lista();
                c_espera.append(l_aux, aux.clt);

                while(aux.pila.first != null){
                    if(aux.pila.first.data == "img_bw"){
                        print img = new print("bw",aux.clt,1);
                        img_bw.append(img);
                    }else if(aux.pila.first.data == "img_c"){
                        print img = new print("c",aux.clt,2);
                        img_color.append(img);
                    }
                    aux.pila.pop();
                }
                paso+="LA VENTANILLA "+aux.id+" ENVIA A COLA DE IMPRESION LAS IMAGENES DEL CLIENTE "+aux.clt.id+"\n";
                aux.clt = null;
```

Función Imprimir: Dentro de esta función se realizan los pasos de la impresión de imágenes y la entrega a los clientes.

```
public static void imprimir(){
    String paso = "";

    if(c_espera.first!=null){
        Espera.Nodo c = c_espera.first; //lista de clientes
        if(img_bw.first!=null){
            print p_bw = (print) img_bw.first.data; //impresora blanco y negro
            int tbw = p_bw.getPasos()-1;
            p_bw.setPasos(tbw);
            if(p_bw.getPasos()==0){
                while(c!=null){
                    if(p_bw.c.id.equals(c.cliente.id)){
                        c.lista.append("bw");
                        paso+="SE REALIZÓ LA IMPRESION A BLANCO Y NEGRO PARA EL CLIENTE "+c.cliente.id+"\n";
                    }
                    c=c.next;
                }
                img_bw.remove();
            }
        }
        if(img_color.first!=null){
            print p_c = (print) img_color.first.data; //impresora color
            int tc = p_c.getPasos()-1;
            p_c.setPasos(tc);
            c=c_espera.first;
            if(p_c.getPasos()==0){
                while(c!=null){
                    if(p_c.c.id.equals(c.cliente.id)){
                        c.lista.append("c");
                        paso+="SE REALIZÓ LA IMPRESION A COLOR PARA EL CLIENTE "+c.cliente.id+"\n";
                    }
                    c=c.next;
                }
                img_color.remove();
            }
        }
        System.out.println(paso);
    }
}
```

Función Espera: Dentro de esta función se realiza la ejecución de los pasos para los clientes en espera, en este se añaden las imágenes y a una Lista de Listas y al cumplir la condición se elimina al cliente de la lista y se añade al historial de atendidos.

```
public static void espera(){
    String paso="";
    if(c_espera.first!=null){
        Espera.Nodo c = c_espera.first; //lista de clientes
        int lista=0;
        while(c!=null){
            Lista.Nodo l = c.lista.first;
            while(l!=null){
                lista++;
                l=l.next;
            }

            if(c.cliente.total==lista){
                Atendidos a = new Atendidos(c.cliente.name,
                                            c.cliente.ventanilla,
                                            String.valueOf(c.cliente.tbw),
                                            String.valueOf(c.cliente.tc),
                                            String.valueOf(contador),
                                            c.cliente.id);
                paso+="EL CLIENTE "+c.cliente.id+" SALE DEL SISTEMA, SE HA REGISTRADO EN EL HISTORIAL\n";
                c_atendidos.append(a);
                c_espera.delete(c.cliente.id);
            }
            c=c.next;
            lista=0;
        }
        System.out.println(paso);
    }
}
```

Generación de Reportes: Los reportes se basan en ordenamientos de tipo burbuja y buscadores además del uso de graphviz.

Ordenamiento Burbuja:

```
if(aux!=null){
    Lista.Nodo actual = aux;
    boolean sw;
    Object temp;
    do{
        actual = L.first;
        Lista.Nodo siguiente = actual.next;
        sw=false;
        while(actual.next!=null){
            Atendidos x = (Atendidos) actual.data;
            Atendidos y = (Atendidos) siguiente.data;
            if (Integer.parseInt(x.n_imgc)<Integer.parseInt(y.n_imgc)) {
                sw=true;
                temp= actual.data;
                actual.data = siguiente.data;
                siguiente.data = temp;
                actual = actual.next;
                siguiente = siguiente.next;
            }else{
                actual = actual.next;
                siguiente = siguiente.next;
            }
        }
    }while(sw);
}
```

Uso de Graphviz: Para el uso de graphviz se utilizó la escritura de archivos y posterior mente la ejecución de consola,

```
FileWriter reportel = null;
    PrintWriter pw = null;
    try{

    reportel = new FileWriter("estructuras.dot");
    pw = new PrintWriter(reportel);

    pw.println("digraph G {");
    pw.println("label = \"Estado de Memoria de las Estructuras\"");
    pw.println("fontsize=\"20\"");


    pw.println("subgraph cluster_0 {");
    pw.println("node [style=filled];");
    Cola.Nodo aux_1 = clientes.first;
    while(aux_1!=null){
       Cliente c = (Cliente) aux_1.data;
       pw.println("C"+c.id+"[label = \"Cliente "+c.id+"\\n"+c.name+"\"];");
       aux_1=aux_1.next;
    }
    aux_1=clientes.first;
```

```
aux_5=c_espera.first;
while(aux_5!=null){
    Cliente c = aux_5.cliente;
    if(aux_5.next!=null){
        Cliente c2=aux_5.next.cliente;
        pw.println("rank=same{Es"+c.id+"->Es"+c2.id+"};");
    }
    aux_5=aux_5.next;
}

pw.println("label = \"Lista Clientes en Espera\";");
pw.println("color=blue}");

pw.println("}");

}catch(Exception e){
    e.printStackTrace();
}finally{
    try{
        if(null != reportel){
            reportel.close();
            ProcessBuilder buil = new ProcessBuilder("dot","-Tpng","-o","estructuras.png","estructuras.dot");
            buil.redirectErrorStream(true);
            buil.start();
        }
}catch(Exception e2){
e2.printStackTrace();
}
}
```

Almacenamiento de datos: Para el almacenamiento de datos se utilizó la programación orientada a objetos en el cual se creó una clase llamada Clientes para la información de clientes, Atendidos para la información de los clientes que salen del sistema, print para guardar información en la cola de impresión.

```
public class Cliente {

    String id, name, img_c, img_bw,ventanilla;
    int total=0;
    int tbw,tc;
    public Cliente(String id, String name, String img_c, String img_bw,String ventanilla) {
        this.id = id;
        this.name = name;
        this.img_c = img_c;
        this.img_bw = img_bw;
        this.ventanilla = ventanilla;
        this.tc=Integer.parseInt(img_c);
        this.tbw=Integer.parseInt(img_bw);
    }
```

```
public class print {

    String tipo;
    Cliente c;
    int pasos;

    public print(String tipo, Cliente c,int pasos) {
        this.tipo = tipo;
        this.c = c;
        this.pasos = pasos;
    }
```

```
public class Atendidos {

    String cliente, ventanilla, n_img, n_imgc, n_pasos,id;

    public Atendidos(String cliente, String ventanilla, String n_img,String n_imgc, String n_pasos,String id) {
        this.cliente = cliente;
        this.ventanilla = ventanilla;
        this.n_img = n_img;
        this.n_pasos = n_pasos;
        this.n_imgc = n_imgc;
        this.id = id;
    }

    public String getN_imgc() {
```

Estructuras de datos lineales: Para almacenar los objetos mencionados anteriormente se emplearon 3 estructuras: Pilas, Listas y Colas, derivadas de ellas se utilizaron la Lista de Lista y la Lista de Pilas.

```java
package edd.proyecto1_fase1;

public class Cola {

    Nodo first;

    public class Nodo{
        public Object data;
        public Nodo next = null;

        public Nodo(Object data){
            this.data = data;
        }
    }

    public void append(Object data){...13 lines }

    public void remove(){...7 lines }


}
```

```java
package edd.proyecto1_fase1;

public class Pila {
    Nodo first;

    public class Nodo{
        public Object data;
        public Nodo next = null;

        public Nodo(Object data){
            this.data = data;
        }
    }

    public void push(Object data){...6 lines }

    public void peek(){...9 lines }


    public void pop(){...8 lines }



}
```

```java
package edd.proyecto1_fase1;

public class Lista {

    Nodo first;

    public class Nodo{
        public Object data;
        public Nodo next = null;

        public Nodo(Object data){
            this.data = data;
        }
    }

    public void append(Object data){...13 lines }

}
```