Manual Técnico

EDD_Proyecto1_Fase2

Kevin Martinez
202004816 USAC

Índice

I. Introducción

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Objetivo

Instruir el uso adecuado del Sistema de Información, para el acceso oportuno y adecuado en la modificación de este, mostrando la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración y soporte de este.

Requerimientos

- Tener Java (versión 8 o superiores) instalado.
- Se recomienda tener un editor de código (NetBeans 8.2 u otro) para ver y editar el código de este programa.

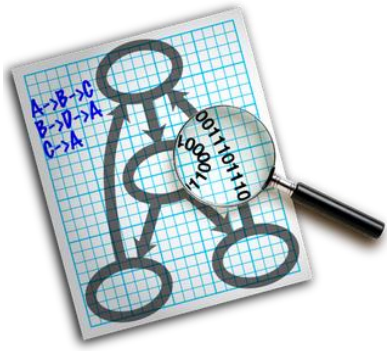
II. Descripción Del Sistema

1. Descripción del Contenido del Sistema

Este es un sistema en el cual el usuario interactúa directamente con una interfaz gráfica permitiendo registrar nuevos usuarios que puedan utilizar esta aplicación para crear imágenes, cargando capas al sistema, asimismo crear diferentes reportes sobre su información guardada; y teniendo un administrador que pueda ingresar, actualizar, buscar y eliminar los usuarios de la aplicación.

2. Lenguajes Utilizados

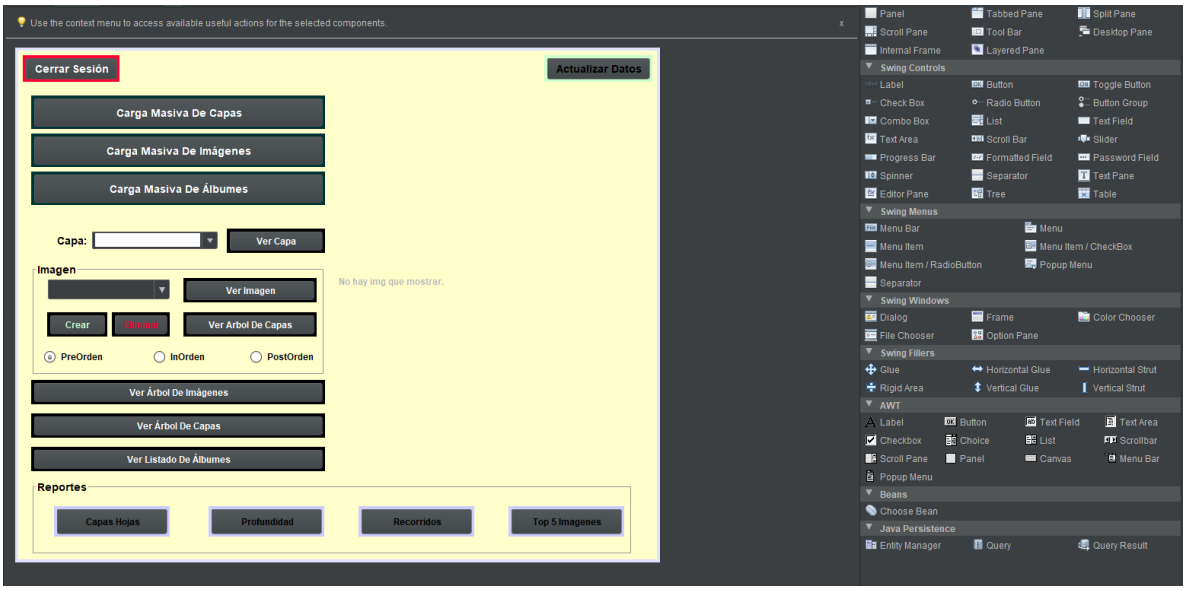
Para la creación de este sistema todas las funcionalidades se realizaron en lenguaje de JAVA y también se utilizó Graphviz para la creación de reportes. Utilizando un enfoque al paradigma de programación POO (Programación Orientada a Objetos). Creando las diferentes estructuras de datos para almacenar la información.



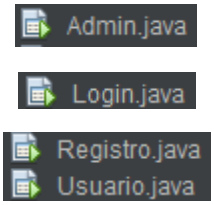
Todo realizado en el IDE NetBeans 8.2:

3. Interfaz Gráfica

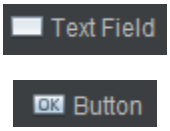
Para la creación de la interfaz gráfica se utilizó la herramienta de “Drag and Drop” que nos proporciona el entorno de NetBeans:



Se crearon 4 ventanas principales, estas corresponden al Log In (apartado donde se inicia sesión), Admin (apartado del administrador), Registro (apartado para registrarse en la aplicación) y Usuario (ventana donde el usuario interactúa).



Las principales herramientas utilizadas fueron TextField (para ingresar texto) y botones para hacer las instrucciones.



4. Lectura de Archivo

Para la lectura de archivo JSON se utilizó la librería de gson-2.8.2:

La lectura del archivo se inicia pidiendo la dirección de la ubicación del archivo que se desea cargar, este archivo se abre y se lee con la ayuda de:

```
import java.io.File;
```

```
import java.io.BufferedReader;
```

Utilizados de la siguiente manera:

```
File doc = new File(ruta);
Scanner obj1 = new Scanner(doc);
String t = "";
int cont = 0;
boolean exist = true;

JsonParser parser = new JsonParser();

while (obj1.hasNextLine()){
    t += obj1.nextLine();
    t += "\n";
}
```

En donde el archivo queda guardado como un String dentro de la variable t, y JsonParser pertenece a la librería gson el cual nos ayuda a parsear el String a JSON, utilizado en el siguiente fragmento de código que se utilizó para generar los objetos de clientes:

```java
public static void generarClientes(String text){
    JsonParser parser = new JsonParser();

    // Obtain Array
    JsonArray gsonArr = parser.parse(text).getAsJsonArray();
    // for each element of array
    for (JsonElement obj : gsonArr) {
        // Object of array
        JsonObject gsonObj = obj.getAsJsonObject();
        long dpi = Long.parseLong(gsonObj.get("dpi").getAsString());
        String nombre = gsonObj.get("nombre_cliente").getAsString();
        String contra = gsonObj.get("password").getAsString();

        Cliente n = new Cliente(dpi,nombre,contra);

        EDDProyectoF2.clientes.insertar(n);
    }

    JOptionPane.showMessageDialog(null, "Clientes Generados Correctamente.","Administrador",JOptionPane.INFORMATION_MESSAGE);
}
```

Mientras los usuarios de clientes se van creando estos se van guardando en un Árbol B implementado dentro de la aplicación. De esta manera se leyeron las imágenes, capas y álbumes en la carga masiva dentro del módulo de cliente, guardándolos en su respectiva estructura.

5. Objetos Utilizados

Las clases utilizadas para los objetos son las siguientes: de Cliente con los siguientes atributos:

```java
public class Cliente {

    //Datos Del Cliente
    long dpi;
    String nombre;
    String contra;

    // Estructuras
    Lista albumes;
    ABB capas;
    AVL imgs;

    public Cliente(long dpi, String nombre, String contra) {

        this.dpi = dpi;
        this.nombre = nombre;
        this.contra = contra;

        albumes = new Lista();
        capas = new ABB();
        imgs = new AVL();
    }
```

La clase Imagen, con id para identificarla y para tener un numero entero para organizarla dentro de un AVL, con un ABB el cual guarda los id de las capas y con un atributo “cant” que ayuda a saber cuántas capas tiene para hacer un reporte más adelante.

```
public class Imagen {
    int id;
    ABB capas;
    int cant;

    public Imagen(int id, ABB capas) {
        this.id = id;
        this.capas = capas;
    }
```

La clase Capa, con id para identificarla y tener un numero entero para guardarla dentro de un ABB, y una Matriz que guarde los pixeles necesarios para graficarla.

```
public class Capa {
    int id;
    Matriz pixeles;

    public Capa(int id, Matriz pixeles) {
        this.id = id;
        this.pixeles = pixeles;
    }
```

Por último, la clase álbum con un String para el nombre y una Lista de imágenes que este almacena.

```
public class Album {
    String nombre;
    Lista imgs;

    public Album(String nombre, Lista imgs) {
        this.nombre = nombre;
        this.imgs = imgs;
    }
```

6. Estructuras de datos utilizadas

Las estructuras utilizadas son las siguientes: ABB, AVL, Árbol B, Lista y Matriz; cada una de ellas tiene una clase Nodo y diferentes métodos necesarios para su funcionamiento. El ABB se utilizó para guardar las capas de los clientes, así como las capas que cada imagen necesita para graficarse. Esta clase tiene los métodos de agregar, graficar, buscar, crear imagen y la función de realizar los diferentes recorridos posibles.

```
public class ABB {
    public Nodo raiz;

    public static class Nodo{

        Object valor;
        Nodo izquierda;
        Nodo derecha;

        public Nodo(Object valor){
            this.valor = valor;
            this.izquierda = this.derecha = null;
        }
    }

    public ABB() {
        this.raiz = null;
    }

    public void agregar(Object valor){
        raiz = agregar_recursivo(valor,raiz);
    }

    public static Nodo agregar_recursivo(Object valor, Nodo raiz){

        if(raiz == null){
            return new Nodo(valor);
        }else{
            int id = ((Capa)valor).getId();
            int act = ((Capa)raiz.valor).getId();
            if(id < act){
                raiz.izquierda = agregar_recursivo(valor,raiz.izquierda);
            }else if(act < id){
                raiz.derecha = agregar_recursivo(valor,raiz.derecha);
            }/*else{
                System.out.println("ya existe");
            }*/
            return raiz;
        }
    }
```

La estructura de AVL se utilizó para guardar las imágenes que el cliente va generando, este es similar al ABB, con la diferencia que este puede hacer rotaciones al agregar la información. Además de tener método de eliminar y contar imágenes.

```
public class AVL {
    public Nodo raiz;

    public static class Nodo{

        Object valor;
        int equilibrio;
        Nodo izquierda;
        Nodo derecha;

        public Nodo(Object valor){
            this.valor = valor;
            this.izquierda = this.derecha = null;
            this.equilibrio = 0;
        }
    }

    public AVL() {
        this.raiz = null;
    }
```

El Árbol B se utilizó para guardar los Clientes que se van registrando en la aplicación, este árbol es de grado 5.

```
public class ArbolB {

    Nodo raiz;

    public class Nodo{
        Cliente info1,info2,info3,info4,info5;
        Nodo n0,n1,n2,n3,n4,n5;
        boolean esHoja;

        public Nodo(){
            info1=null;
            info2=info3=info4=info5=null;
            n0=null;
            n1=null;
            n2=null;
            n3=null;
            n4=null;
            this.esHoja = true;
        }
    }

    public ArbolB(){
        raiz = new Nodo();
    }
```

La estructura de Lista se utilizo para guardar los Álbumes que el cliente va creando, así como las imágenes que este guarda. Esta tiene el método de eliminar, agregar y devolver el tamaño de la lista.

```
public class Lista {
    Nodo raiz;

    public class Nodo{
        Object info;
        Nodo next;
        Nodo anterior;

        public Nodo(Object info) {
            this.info = info;
            this.next = null;
            this.anterior = null;
        }
    }

    public void add(Object info){
            Nodo nodonuevo = new Nodo(info);
            if(raiz == null){
                raiz = nodonuevo;
            }
            else{
                Nodo aux = raiz;
                while(aux.next != null){
                    aux=aux.next;
                }
                aux.next = nodonuevo;
            }
    }
```

Por último, la estructura de Matriz sus atributos son dos listas las cuales son las cabeceras para guardar los nodos necesarios para la creación de las capas e imágenes. Tiene los métodos de agregar nodo y capa; y graficar

```
public class Matriz {

    Lista horizontal = new Lista();
    Lista vertical = new Lista();


    public class Nodo{
        Object info;
        Nodo next, anterior;
        Nodo izquierda, derecha, arriba, abajo;
        int x,y;

        public Nodo(Object info) {
            this.info = info;
            x = y = 0;
            this.next= this.anterior = this.izquierda = this.derecha = this.abajo = this.arriba = null;
        }

        public Nodo(Object info,int x, int y) {
            this.info = info;
            this.x = x;
            this.y = y;
            this.next = this.anterior = this.izquierda = this.derecha = this.abajo = this.arriba = null;
        }
    }
```

7. Reportes

La generación de reportes se realizó con la herramienta de graphviz, el proceso es principalmente escribir la sintaxis deseada en un .txt, ya que cada estructura necesita un gráfico diferente, y ejecutándolo con el siguiente código:

```
try {

    String dotPath = "C:\\Program Files\\Graphviz\\bin\\dot.exe";
    String fileInputPath = System.getProperty("user.dir") + "\\"+title+".txt";
    String fileOutputPath = System.getProperty("user.dir") + "\\"+title+".jpg";

    String tParam = "-Tjpg";
    String tOParam = "-o";

    String[] cmd = new String[5];
    cmd[0] = dotPath;
    cmd[1] = tParam;
    cmd[2] = fileInputPath;
    cmd[3] = tOParam;
    cmd[4] = fileOutputPath;

    Runtime rt = Runtime.getRuntime();

    rt.exec( cmd );

    //System.out.println("Graficado");

} catch (Exception e) {
    e.printStackTrace();
}
```

Esto generará una imagen jpg con la estructura del reporte deseado.