UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS

ESTRUCTURAS DE DATOS

MANUAL TECNICO

Nombre: Joseph Jeferson Marroquín Monroy
Carné: 202010316

Manual Técnico

Lugar, fecha y responsables de la elaboración

El Proyecto Fase 2 fue desarrollado entre el 1 y el 28 de marzo del 2022, elaborado por Joseph Jeferson Marroquin Monroy.

Objetivos

El presente manual tiene como finalidad describir la funcionalidad del algoritmo aplicado para generar un imagen especial por medio de capas.

Configuración del sistema

El proyecto 2 está desarrollado en una Laptop Lenovo Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz con 8GB de RAM, Sistema operativo de 64 bits, procesador x64.

Login

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //cliente.mostrarArbolB(cliente.raiz.primero);
    if (jTextField1.getText().equals("admin") && jPasswordField1.getText().equals("EDD2022")) {
        moduloADMIN ma = new moduloADMIN();
        ma.setVisible(true);
        this.dispose();
    } else if (cliente.verificarLogin(Long.parseLong(jTextField1.getText()), jPasswordField1.getText()) == true) {
        moduloUsuario mu = new moduloUsuario();
        mu.setId_cliente(Long.parseLong(jTextField1.getText()));
        mu.setVisible(true);
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(null, "Usuario o contraseña incorrecto");
        jTextField1.setText("");
        jPasswordField1.setText("");
    }
}
```

Buscamos los datos del usuario y contraseña de cada usuario para darle acceso a su determinado modulo.

```
try {
    Gson json = new Gson();
    JFileChooser selector = new JFileChooser();
    File file;
    selector.setMultiSelectionEnabled(false);
    FileNameExtensionFilter filtro = new FileNameExtensionFilter(null, "json");
    selector.setFileFilter(filtro);

    if (selector.showDialog(null, null) == JFileChooser.APPROVE_OPTION) {
        file = selector.getSelectedFile();

        Scanner sc = new Scanner(file);
        String data = "";
        while (sc.hasNextLine()) {
            data += sc.nextLine() + "\n";
        }
        JSONParser parser = new JSONParser();
        Object obj = parser.parse(data);
        JSONArray array = (JSONArray) obj;
        JSONObject jobj;
        for (int i = 0; i < array.size(); i++) {
            jobj = (JSONObject) array.get(i);
            System.out.println("-----------------------------------");

            String dpi = String.valueOf(jobj.get("dpi"));
            String nombre_cliente = String.valueOf(jobj.get("nombre_cliente"));
            String password = String.valueOf(jobj.get("password"));

            Clientes clienteAdmin = new Clientes(Long.parseLong(dpi), nombre_cliente, password);
            login.cliente.insertar(clienteAdmin);

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

Usamos la librería Gson para leer los archivos con extensión json y así realizar las cargas masivas de usuarios, capas, imágenes, álbumes.

```
public void preorder(Node tmp, MatrizDispersa matriz, long id_cliente) {
    if (tmp != null) {
        if (tmp._capas.id_cliente == id_cliente) {
            matriz.insertar(tmp._capas.columna, tmp._capas.fila, tmp._capas.color);
        }
        //System.out.print(tmp._capas.id_capa + " ");
        preorder(tmp.left, matriz, id_cliente);
        preorder(tmp.right, matriz, id_cliente);
    }
}
```

Creamos un método donde accedemos a nuestro árbol binario de búsqueda y lo recorremos para graficar una imagen y mostrar el recorrido en un jTextField, ya sea preorder, inorder o postorder.

```java
public void graficar(Node tmp) {
    FileWriter fichero = null;
    PrintWriter escritor;
    String dot = "Estructuras\\ABB\\abb_" + tmp._capas.id_cliente + ".dot";
    String jpg = "Estructuras\\ABB\\abb_" + tmp._capas.id_cliente + ".jpg";
    try {
        fichero = new FileWriter(dot);
        escritor = new PrintWriter(fichero);
        escritor.print(getCodigoGraphviz(tmp));
    } catch (Exception e) {
        System.err.println("Error al escribir el archivo .dot");
    } finally {
        try {
            if (null != fichero) {
                fichero.close();
            }
        } catch (Exception e2) {
            System.err.println("Error al cerrar el archivo .dot");
        }
    }
    try {
        String dotPath = "dot";
        String tParam = "-Tjpg";
        String tOParam = "-o";
        String[] cmd = new String[5];
        cmd[0] = dotPath;
        cmd[1] = tParam;
        cmd[2] = dot;
        cmd[3] = tOParam;
        cmd[4] = jpg;

        Runtime.getRuntime().exec(cmd);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

Usamos graphviz para graficar las estructuras y así después mostrarlo en la aplicación durante la ejecución.

```java
//GRAIFCAR POR ID DE IMAGEN
public void bstIngresoCapa(Node tmp, MatrizDispersa matriz, long id_cliente, int id_capa) {
    if (tmp != null) {
        if (id_capa == tmp._capas.id_capa && tmp._capas.id_cliente == id_cliente) {
            matriz.insertar(tmp._capas.columna, tmp._capas.fila, tmp._capas.color);
            //System.out.println(tmp._capas.columna + " " + tmp._capas.fila + " " + tmp._capas.color);
        }
        bstIngresoCapa(tmp.left, matriz, id_cliente, id_capa);
        bstIngresoCapa(tmp.right, matriz, id_cliente, id_capa);
    }
}
```

Recorremos nuestro árbol binario de búsqueda y recibimos una o mas capas para graficar, las cuales buscamos y agregamos a nuestra matriz dispersa para después graficarlas.

```
public void imprimirNiveles(Node tmp, long id_cliente, JTextField jtextl) {
    if (tmp.left == null && tmp.right == null) {
        if (tmp._capas.id_cliente == id_cliente) {
            nivel = nivel + 1;
            //System.out.println("nivel " + nivel + " id: " + tmp._capas.id_capa);
            info inf = new info(id_cliente, tmp._capas.id_capa, nivel);
            ll.insert(inf);
        }

    } else {
        if (tmp._capas.id_cliente == id_cliente) {
            nivel = nivel + 1;
            //System.out.println("nivel " + nivel + " id: " + tmp._capas.id_capa);
            info inf = new info(id_cliente, tmp._capas.id_capa, nivel);
            ll.insert(inf);
        }
    }
    if (tmp.left != null) {
        imprimirNiveles(tmp.left, id_cliente, jtextl);
        if (tmp._capas.id_cliente == id_cliente) {
            nivel = 0;
        }
    }
    if (tmp.right != null) {
        imprimirNiveles(tmp.right, id_cliente, jtextl);
        if (tmp._capas.id_cliente == id_cliente) {
            nivel = 0;
        }
    }
}
```

Recorremos nuestro árbol binario de búsqueda donde indicamos el nivel en que se encuentra determinada capa y así mostrarlo en el apartado de reportes.

```
//CAPAS QUE SON HOJAS
public void capasHojas(Node tmp, long id_cliente, JTextField jtextl) {
    if (tmp.left == null && tmp.right == null) {
        if (tmp._capas.id_cliente == id_cliente) {
            capH = capH + tmp._capas.id_capa + " ";
            jtextl.setText(capH);
            nivel = nivel + 1;
            //System.out.println("nivel " + nivel + " id: " + tmp._capas.id_capa);
        }

    } else {
        if (tmp._capas.id_cliente == id_cliente) {
            nivel = nivel + 1;
            //System.out.println("nivel " + nivel + " id: " + tmp._capas.id_capa);
        }
    }
    if (tmp.left != null) {
        capasHojas(tmp.left, id_cliente, jtextl);
    }
    if (tmp.right != null) {
        capasHojas(tmp.right, id_cliente, jtextl);
    }
}
```

Recorremos nuestro árbol mandando a imprimir en un jTextField las capas que son hojas.