Manual Técnico

EDD_Proyecto1_Fase3

Kevin Steve Martinez Lemus
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.

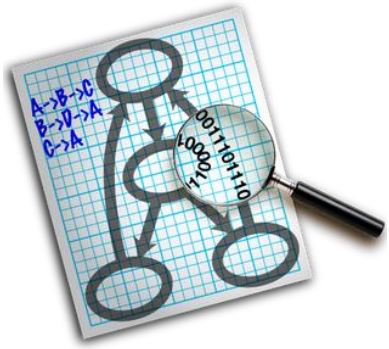
II. Descripción Del Sistema

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 solicitar envíos de imágenes, y con opción a que el administrador dirija las opciones de blockchain implementadas al mismo.

2. Lenguajes Utilizados

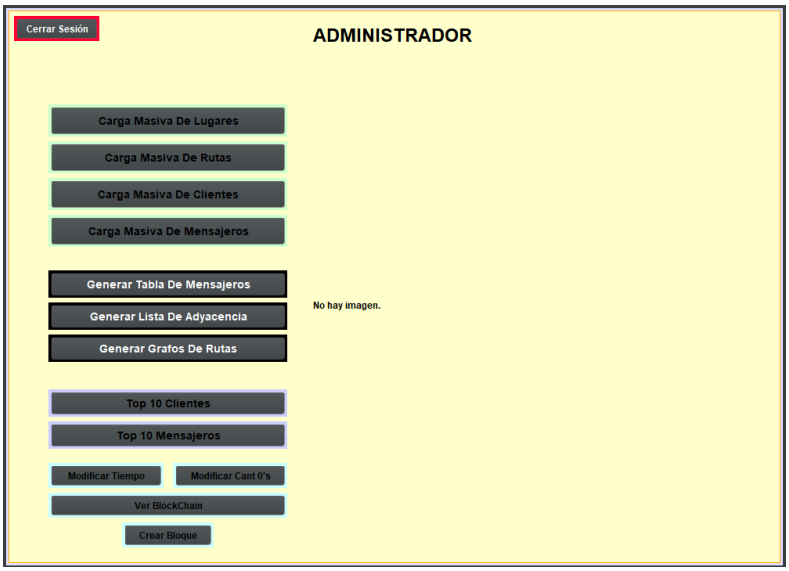
Para la creación de este sistema todas las funcionalidades se realizaron en lenguaje de JAVA y también se utilizó Graphviz para la creación de reportes. Utilizando un enfoque al paradigma de programación POO (Programación Orientada a Objetos). Creando las diferentes estructuras de datos para almacenar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



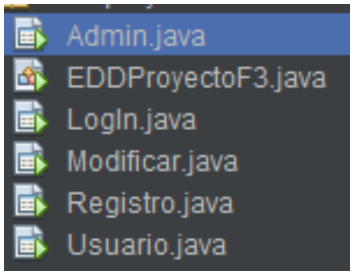
Todo realizado en el IDE NetBeans 8.2:

3. Interfaz Gráfica

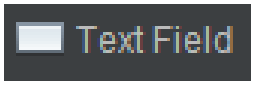
Para la creación de la interfaz gráfica se utilizó la herramienta de “Drag and Drop” que nos proporciona el entorno de NetBeans:



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 void generarClientes(String text){
    JsonParser parser = new JsonParser();

    JsonArray gsonArr = parser.parse(text).getAsJsonArray();

    for (JsonElement obj : gsonArr) {
        JsonObject gsonObj = obj.getAsJsonObject();
        long dpi = Long.parseLong(gsonObj.get("dpi").getAsString());
        String nombre = gsonObj.get("nombre_completo").getAsString();
        String usuario = gsonObj.get("nombre_usuario").getAsString();
        String correo = gsonObj.get("correo").getAsString();
        String contra = gsonObj.get("contrasenia").getAsString();
        String tel = gsonObj.get("telefono").getAsString();
        String dir = gsonObj.get("direccion").getAsString();
        int id_mun = gsonObj.get("id_municipio").getAsInt();

        String password = BCrypt.withDefaults().hashToString(12, contra.toCharArray());

        Cliente nc = new Cliente(dpi,nombre,usuario,correo, password, tel, dir, id_mun);

        EDDProyectoF3.clientes.add(nc);
    }
}
```

5. Objetos Utilizados

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

```java
long dpi;
String nombre;
String usuario;
String correo;
String contra;
String telefono;
String direccion;
int id_municipio;
int solicitudes = 0;

public Cliente(long dpi, String nombre, String usuario, String correo, String contra, String telefono, String direccion, int id_municipio) {
    this.dpi = dpi;
    this.nombre = nombre;
    this.usuario = usuario;
    this.correo = correo;
    this.contra = contra;
    this.telefono = telefono;
    this.direccion = direccion;
    this.id_municipio = id_municipio;
}
```

La clase Entregas para llevar el control de las solicitudes y entregas que el cliente requiere:

```
String sede;
String destino;
String datetime;
String cliente;
String mensajero;

public Entregas(String sede, String destino, String datetime, String cliente, String mensajero) {
    this.sede = sede;
    this.destino = destino;
    this.datetime = datetime;
    this.cliente = cliente;
    this.mensajero = mensajero;
}
```

Para guardar los Lugares se utilizó la siguiente estructura:

```
int id;
String departamento;
String nombre;
String sn_sucursal;

public Lugar(int id, String departamento, String nombre, String sn_sucursal) {
    this.id = id;
    this.departamento = departamento;
    this.nombre = nombre;
    this.sn_sucursal = sn_sucursal;
}
```

La clase Mensajeros se utilizo para llevar el control de los mensajeros de la empresa guardándolos en una tabla hash:

```
long dpi;
String nombre;
String apellido;
String tipo_licencia;
String genero;
String telefono;
String direccion;
int entregas = 0;

public Mensajero(long dpi, String nombre, String apellido, String tipo_licencia, String genero, String telefono, String direccion) {
    this.dpi = dpi;
    this.nombre = nombre;
    this.apellido = apellido;
    this.tipo_licencia = tipo_licencia;
    this.genero = genero;
    this.telefono = telefono;
    this.direccion = direccion;
}
```

Y, por último, la clase Ruta se utilizó para generar el grafo de rutas con la siguiente estructura:

```
int d1;
int d2;
int peso;

public Ruta(int d1, int d2, int peso) {
    this.d1 = d1;
    this.d2 = d2;
    this.peso = peso;
}
```

6. Estructuras de datos utilizadas

Se utilizo la clase Lista para la guardar los datos de los lugares así como las rutas en conjunto con la clase ListaAdyacencia con la siguiente estructura:

```
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 class ListaAdyacencia {
    Lista vertices;

    public class Vertice{
        int vert;
        Lista destinos;

        public Vertice(int vert){
            this.vert = vert;
            destinos = new Lista();
        }

        public void agregarDestino(Destino des){
            destinos.add(des);
        }
    }
```

Los bloques de la blockchain se guardan en una lista similar a la anterior solo que guardando bloques de información de la siguiente manera:

```java
public class BlockChein {

    Nodo genesis;
    int index;

    public BlockChein() {
        genesis = null;
        index = 0;
    }

    public class Bloque{
        int index;
        String timestamp;
        int nonce;
        ArrayList<Entregas> entregas;
        String previusHash;
        String rootMerkle;
        String hash;

        public Bloque(int index, String timestamp,String previusH, int nonce, String rootMerkle, String hash ,ArrayList<Entregas> entregas ) {
            this.index = index;
            this.timestamp = timestamp;
            this.nonce = nonce;
            this.previusHash = previusH;
            this.rootMerkle = rootMerkle;
            this.hash = hash;
            this.entregas = entregas;
        }
```

Por último, la estructura de AMerckle el cual representa un árbol de Merkle en donde se guardan las solicitudes de entrega y posee una raíz la cual se guarda en los bloques de la bolckchain:

```java
public class AMerckle {
    String merkleRoot = "";
    ArrayList<String> txLst = new ArrayList<String>();
    Nodo raiz;


    public class Nodo{
        String info;
        Nodo izq;
        Nodo der;

        public Nodo(String info) {
            this.info = info;
            this.izq = null;
            this.der = null;
        }
    }
```

7. Reportes

La generación de reportes se realizó con la herramienta de graphviz, el proceso es principalmente escribir la sintaxis (dot) 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.