Universidad de San Carlos de Guatemala
Escuela de Ciencias y Sistemas
Estructuras De Datos

Manual Técnico
Proyecto 1 Fase 3
UDrawing Paper

Arnoldo Luis Antonio González Camey
Carné: 201701548

INTRODUCCIÓN

Este manual va dirigido a todo público que cuente con conocimientos técnicos sobre la programación y que quiera ver la interacción entre clases, objetos para el manejo de la información y los datos, así como listas simples por medio de una simulación de una empresa denominada "Drawing Paper" la cual requiere registrar clientes. Para poder hacer uso de la aplicación el cliente debe registrarse.

La principal funcionalidad de la aplicación consiste en registrar clientes tanto por medio de un archivo de texto como dentro de la aplicación a su vez modificar la información de cada uno de ellos, eliminarlos todo esto realizado por medio de un administrador, los clientes a su vez pueden iniciar sesión en el sistema.

REQUERIMIENTOS MÍNIMOS

Para poder operar el programa se necesita: Una computadora con un procesador de 32 o 64 bits, sistema operativo Windows 7 en adelante, Linux o cualquier otro sistema que permita la instalación de JAVA, tener instalado JAVA en la versión más reciente, tener la capacidad de acceder a los comandos de Windows o Linux, contar con los archivos en formato json para la utilización de este.

Manual Técnico

- Método cargaMasivaCliente

Este método es utilizado para realizar la carga masiva del archivo json, para ello se hace la instancia del método leerArchivoJson, luego se hace uso de la librería gson y se hace un parser de la información obtenido, luego se convierte el Json de tipo elemento a tipo objeto. Luego esta variable se realiza un parse con un JsonArray, seguido de esto se recorre el JsonArray y se obtiene los atributos y se hace la instancia del método insertar.

```
public boolean cargaMasivaCliente(String Jsontxt, Registro registro) {
    JsonParser parser = new JsonParser();
    JsonArray gsonArr = parser.parse(Jsontxt).getAsJsonArray();
    for (JsonElement objt : gsonArr) {
        JsonObject gsonObj = objt.getAsJsonObject();
        String dpi = gsonObj.get("dpi").getAsString().trim();
        String nombre = gsonObj.get("nombre_cliente").getAsString().trim();

        String usuario = gsonObj.get("username").getAsString().trim();
        String correo = gsonObj.get("email").getAsString().trim();
        String contrasena = gsonObj.get("password").getAsString().trim();
        String telefono = gsonObj.get("phone").getAsString().trim();
        String direccion = gsonObj.get("address").getAsString().trim();
        int idMunicipio = gsonObj.get("department_id").getAsInt();
        registro.insertarUsuario(dpi, nombre, usuario, correo, contrasena, telefono, direccion, idMunicipio);
    }
    return true;
}
```

- Método validarUsuario

Este método es utilizado para validar si un usuario existe, para ello se envían como parámetros los valores correspondientes y se crean unas variables booleanas auxiliares, a cada una de este se le igual al método iquals para saber si son iguales o no, y al finalizar se retorna el valor verdadero o falso según sea el caso.

```
public boolean validarUsuario(String usuario, String password) {
    boolean userUs = this.getUsuario().equalsIgnoreCase(usuario);
    boolean contrasenaUs = this.getContrasena().equals(password);
    boolean correcto = userUs && contrasenaUs; //los dos true correcto es true
    return correcto;
}
```

- Método modificarUsuario

Este método es utilizado para modificar la información de un usuario, para ello se envían como parámetros los valores correspondientes y se recorre la lista de usuarios luego se valida si el valor del usuario en esa posición es igual al valor del usuario que se desea modificar, si lo es se modifican los valores.

```
public void modificarUsuario(String dpi, String usuario, String nombre, String contrasena,
    Cliente actual = this.getCabezaUsuario();
    while (actual != null) {
        if (String.valueOf(actual.getUsuario()).equalsIgnoreCase(usuario)) {
            actual.setDpi(dpi);
            actual.setNombre(nombre);
            actual.setContrasena(contrasena);
            actual.setTelefono(telefono);
            actual.setDireccion(direccion);
            actual.setCorreo(correo);
            actual.setIdMunicipio(idMunicipio);
        }
        actual = actual.getSiguiente();
    }
}
```

- Método insertarUsuario

Este método es utilizado para ingresar un usuario en el sistema, para ello se envían como parámetros los valores correspondientes, primero se valida si el nombre del usuario existe si no existe se procede a recorrer la lista de usuarios, luego se valida la posición que ocupara el usuario por medio de su número de dpi y se retorna el usuario ingresado.

```
public Cliente insertarUsuario(String dpi, String nombre, String usuario, String correo, String contrasena, Str
    if (existeUsuario(usuario)) {
        System.out.println("El nombre de usuario " + usuario + " ya existe");
        return null;
    }
    this.size++;
    System.out.println(dpi + " " + nombre + " " + usuario + " " + correo + " " + contrasena + " " + telefono + "
    Cliente nuevo = new Cliente(dpi, nombre, usuario, correo, contrasena, telefono, direccion, idMunicipio);
    if (this.getCabezaUsuario() == null) {
        this.setCabezaUsuario(nuevo);
    } else {
        if (Long.parseLong(dpi) < Long.parseLong(this.getCabezaUsuario().getDpi())) {
            nuevo.setSiguiente(this.getCabezaUsuario());
            this.setCabezaUsuario(nuevo);
        } else {
            Cliente anterior = this.getCabezaUsuario();
            Cliente aux = this.getCabezaUsuario();
            while (Long.parseLong(dpi) >= Long.parseLong(aux.getDpi()) && aux.getSiguiente() != null) {
                anterior = aux;
                aux = aux.getSiguiente();
            }
            if (Long.parseLong(dpi) >= Long.parseLong(aux.getDpi())) {
                aux.setSiguiente(nuevo);
                nuevo.setAnterior(aux);
            } else {
                nuevo.setSiguiente(aux);
                anterior.setSiguiente(nuevo);
                nuevo.setAnterior(anterior);
            }
        }
    }
    return nuevo;
}
```

- Método eliminarUsuario

Este método es utilizado para eliminar un usuario, para ello se envían como parámetro el nombre del usuario y se recorre la lista de usuarios luego se valida si el valor del usuario en esa posición es igual al valor del usuario que se desea eliminar, si lo es se elimina el usuario y se retorna verdadero, si nunca entra se retorna falso.

```
public boolean eliminarUsuario(String usuario) {
    Cliente user = this.getCabezaUsuario();
    while (user != null) {
        if (user.getUsuario().equalsIgnoreCase(usuario)) {
            Cliente aux = user.getSiguiente();
            if (user != this.getCabezaUsuario()) {
                user.getAnterior().setSiguiente(aux);
                if (aux != null) {
                    aux.setAnterior(user.getAnterior());
                }
            } else {
                aux.setAnterior(null);
                this.setCabezaUsuario(aux);
            }
            user.setSiguiente(null);
            user.setAnterior(null);
            this.size--;
            return true;
        }
        user = user.getSiguiente();
    }
    return false;
}
```

- Método existeUsuario

Este método es utilizado para saber si existe un usuario, para ello se envían como parámetro el nombre del usuario, primero se valida si el usuario es el administrador si no lo es se recorre la lista de usuarios luego se valida si el valor del usuario en esa posición es igual al valor del usuario que se desea encontrar, si lo es se retorna verdadero, si nunca entra se retorna falso.

```
public boolean existeUsuario(String usuario) {
    if (getAdministrador().getUsuario().equalsIgnoreCase(usuario)) {
        return true;
    }
    Cliente user = this.getCabezaUsuario();
    while (user != null) {
        if (user.getUsuario().equalsIgnoreCase(usuario)) {
            return true;
        }
        user = user.getSiguiente();
    }
    return false;
}
```