Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Ingeniería en Ciencias y Sistemas

Introducción a la programación y computación 1

MANUAL TÉCNICO

Integrantes:

Alberto Josue Hernández Armas 201903553

Leonel Antonio González García 201709088Tulio

Jafeth Pirir Schuman 201700698

Kevin Mark Hernández Chicol 202001053

Guatemala 2 de julio de 2021

Manual de técnico:

Introducción

En el siguiente manual vamos a dar a entender acerca del proceso que se utilizo para crear el programa y como pudimos crear unos métodos para hacer las estructura del mismo.

Clases

Primero comenzamos creando nuestras clases con los cuales nos apoyaremos para poder darle a cada uno sus atributos y realizar las respectivas llamadas. También se instanciaron los scanner para poder leer las entradas de datos que vamos a tener por cada uno de los documentos. En esta fase empezamos a utilizar lo que es gson para la serialización de la información por lo mismo que sea agregro a la librería y también el código se agrego la dependencia.

```
import java.util.ArrayList;
import java.util.Scanner;
import com.google.gson.*;

public class Main {

    static Scanner readStr = new Scanner(System.in);
    static Scanner readNum = new Scanner(System.in);
    static ArrayList<Configuraciones>configs = new ArrayList<>();
    static ArrayList<Usuarios>users = new ArrayList<>();
    static ArrayList<Productos>products = new ArrayList<>();
    static ArrayList<Clientes>clients = new ArrayList<>();
    static ArrayList<Facturas>invoices = new ArrayList<>();
    static int ContadorUsers = 0;
    static int ContadorClients = 0;
    static int ContadorProducts = 0;
    static int ContadorInvoices = 0;

    public static void main(String[] args) {
        Error.vericador();
        CargaConfig();
        Login();
    }
```

Usuario y carga de informacion

Empezamos con lo que es el ingreso de informacion y principalmente el usuario primario para poder acceder al menu. El usuario ya estara guardado en un archivo .json en el cual vamos comparar los datos y si resultan ser los mismo vamos a poder acceder de lo contrario se regresara el menu de usuario de nuevo. Enl el codigo se puede ver que se muestra un if dentro de un for para hacer la comparacion.

```
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == btningresar) {
        usuario = txtusuario.getText();
       contraseña = contra.getText();
        for (int i = 0; i < Main.users.size(); i++) {
            if (usuario.equals(Main.users.get(i).getUsername()) && contraseña.equals(Main.users.get(i).getPassword())){
                Main.logAcciones+=   Main.HoraFecha()+"\t"+usuario+": Inicio de Sesión Exitoso"+"\n";
                menu = new Menu();
                login.setVisible(false);

            } else if (i==Main.users.size()-1){
                Main.logAcciones+=   Main.HoraFecha()+"\t"+usuario+": Inicio de Sesión Fallido "+"\n";
                JOptionPane.showMessageDialog( parentComponent: null, message: "Datos Incorrectos");
            }

        }
```

También se agrego la clase cargaconfig en el cual vamos a tener el archivo que vamos a leer en json. El cual contiente también los datos preguardados y alemacenados para cuando el usuario acceda al menú ya se tenga una estructura previa pero también se puede ingresar nuevos datos por medio del menú que vamos a explicar mas adelante.

```
public static void CargaConfig(){
String ContenidoConfig = Archivos.getContentOfFile( pathname: "./JPruebas/config.json");
Gson gson = new Gson();
Configuraciones configuraciones = gson.fromJson(ContenidoConfig, Configuraciones.class);
configs.add(configuraciones);

if (configuraciones.getLoad().equals("bin")){
    System.out.println("BIN");
    users = (ArrayList<Usuarios>) Archivos.deserialize( pathname: "./Serealizables/users.ipcrm");
    products = (ArrayList<Productos>) Archivos.deserialize( pathname: "./Serealizables/products.ipcrm");
    clients = (ArrayList<Clientes>) Archivos.deserialize( pathname: "./Serealizables/clients.ipcrm");
    invoices = (ArrayList<Facturas>) Archivos.deserialize( pathname: "./Serealizables/invoices.ipcrm");
} else if (configuraciones.getLoad().equals("json")){
    System.out.println("JSON");
    CargaUsuarios();
    CargaProductos();
    CargaClientes();
    CargaFacturas();
}
```

```java
public static void CargaUsuarios(){
 String ContenidoUsuarios = Archivos.getContentOfFile( pathname: "./JPruebas/users.json");
 Gson gson = new Gson();
 Usuarios[] usuarios = gson.fromJson(ContenidoUsuarios, Usuarios[].class);

 for (Usuarios user: usuarios){
     users.add(user);
     ContadorUsers++;
    }
}

public static void CargaClientes(){
String ContenidoClientes = Archivos.getContentOfFile( pathname: "./JPruebas/clients.json");
Gson gson = new Gson();
Clientes[] clientes = gson.fromJson(ContenidoClientes, Clientes[].class);

    for (Clientes client: clientes){
        clients.add(client);
        ContadorClients++;
    }

    //Archivos.serialize("./Serealizables/clients.ipcrm", clients);
    //Object UserS = Archivos.deserialize("./Serealizables/users.ipcrm");
}
```

```java
public static void CargaProductos(){
String ContenidoProductos = Archivos.getContentOfFile( pathname: "./JPruebas/products.json");
Gson gson = new Gson();
Productos[]productos = gson.fromJson(ContenidoProductos, Productos[].class);

    for (Productos product: productos){
        products.add(product);
        ContadorProducts++;
    }

    //Archivos.serialize("./Serealizables/products.ipcrm", products);
}

public static void CargaFacturas(){
String ContenidoFacturas = Archivos.getContentOfFile( pathname: "./JPruebas/invoices.json");
Gson gson = new Gson();
Facturas[]facturas = gson.fromJson(ContenidoFacturas, Facturas[].class);

for (Facturas invoice: facturas){
    invoices.add(invoice);
    ContadorInvoices++;
    }
```

En esta parte se muestra como tambien se estaran cargando los usuarios, clientes, facturas y productoso, como se menciono anteriormente se utilizan estos datos para poder mostrar los datos mas adelante al usuario que va tener accesso al programa y despues podra acceder a ellos por medio de la selección de opciones. Todos los datos estan cargados en archivos

Clase Main

En esta parte se agrega el diseño a partir de una librería y además se ejecuta el login como el Frame principal
Y se llamana las funciones de log de errores.

```
JFrame.setDefaultLookAndFeelDecorated(true);

try {
    // COMANDO PARA AGREGAR UN LOOK AND FEEL

    UIManager.setLookAndFeel(new NoireLookAndFeel());

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

CargaConfig();
Login login = new Login();
RevisionProducts();
RevisionInvoices();
RevisionClients();
RevisionUsers();
```

Funciones para Editar

Con estas funciones se puede editar los datos de un cliente, producto, ingrediente o usuario especifica por medio de una comparación mediante su Id o su username y agrega su respectiva acción al log de acciones.

```
    public static void EditarCliente(int id, String name, String address, String phone, String nit){
    for (int i=0; i<clients.size(); i++){
        if(clients.get(i).getId() == id){
            clients.get(i).setName(name);
            clients.get(i).setAddress(address);
            clients.get(i).setPhone(phone);
            clients.get(i).setNit(nit);
            logAcciones+=HoraFecha()+"\t Se Edito el Cliente "+name+", id :"+id+"\n";
            logAcciones();
        }
    }
}
```

```
public static void EditarIngrediente(String index, String name, int quantity, String units, int cont){
    for (int i=0; i<Main.products.size(); i++){
        if (cont == Main.products.get(i).getId()) {
            for (int j = 0; j < Main.products.get(i).getIngredients().size(); j++) {
                if (Main.products.get(i).getIngredients().get(j).getName().equals(index)){
                    Main.products.get(i).getIngredients().get(j).setName(name);
                    Main.products.get(i).getIngredients().get(j).setQuantity(quantity);
                    Main.products.get(i).getIngredients().get(j).setUnits(units);
                }
            }
        }
    }
}
```

```
    public static void EditarProducto(int id, String name, String description, double cost, double price){
    for (int i=0; i< products.size(); i++){
        if(products.get(i).getId() == id) {
            products.get(i).setName(name);
            products.get(i).setDescription(description);
            products.get(i).setCost(cost);
            products.get(i).setPrice(price);
            logAcciones+=HoraFecha()+"\t Se Edito el Producto  "+name+", id :"+id+"\n";
            logAcciones();
        }
    }
}
```

```
  public static void EditarUsuario(String user, String username, String password){
      for (int i=0; i<users.size(); i++){
          if(users.get(i).getUsername().equals(user)){
              users.get(i).setUsername(username);
              users.get(i).setPassword(password);
              logAcciones+=HoraFecha()+"\t Se Edito el Usuario "+user+", id :"+username+"\n";
              logAcciones();
          }
      }
  }
```

Funciones para Agregar

Con estas funciones se puede agregar los datos de un cliente, producto, ingrediente o usuario especifica por medio de una comparación mediante su Id o su username y agrega su respectiva acción al log de acciones.

```
public static void AgregaUsuario(String username, String password){
    Usuarios UsuriosAgregados = new Usuarios(username, password);
    users.add(UsuriosAgregados);
    logAcciones+=HoraFecha()+" Se Agrego al usuario  :"+username+"\n";

    logAcciones();
}
```

```
public static void AgregaCliente(int id, String name, String address,String nit, String phone ){
Clientes ClientesAgregados = new Clientes(id, name, address, phone, nit);
    logAcciones+=HoraFecha()+"\t Se Agrego el Cliente "+name+", id :"+id+"\n";
    logAcciones();
clients.add(ClientesAgregados);
}
```

```java
public static void AgregarIngredienteI(int Index, String name, int quantity, String units){
Ingredientes IngredientesP = new Ingredientes(name, quantity, units);
for (int i=0; i<products.size();i++) {
    if (products.get(i).getId() == Index) {
        logAcciones+=HoraFecha()+"\t Se Agrego el ingrediente "+name+", id :"+Index+"\n";
        logAcciones();
        products.get(i).ingredients.add(IngredientesP);
     }
    }
}
```

```java
public static void AgregaProducto(int id, String name, String description, int cost, int price, ArrayList<Ingredientes>ig){
Productos ProductosAgregados = new Productos(id,name,description,cost,price, ig);
    logAcciones+=HoraFecha()+"\t SeAgrego El Producto  "+name+", id :"+id+"\n";
    logAcciones();
products.add(ProductosAgregados);
}
```

```java
public static void AgregaFactura(int id, int user, String date, ArrayList<ProductoF>PF){
Facturas FacturasAgregadas = new Facturas(id, user,date,PF);
logAcciones+=HoraFecha()+"\t  Se Agrego la Factura "+", id :"+id+"\n";
logAcciones();
invoices.add(FacturasAgregadas);
}
```

Funciones para poder llenar las tablas

Para poder llenar la tablas hacemos uso de funciones de tipo objeto que reciben como parámetro una matriz de Objetos al igual que le agrega el botón de eliminar y editar con su respectivo Index a cada botón así almacenando Cada uno en su respectiva columna.

```java
public static Object[][] DatosUsuarios(){
    Object[][] arreglo= new Object[users.size()][4];

    for (int i =0 ; i< users.size(); i++){
        arreglo [i][0]=users.get(i).getUsername();
        arreglo[i][1]=users.get(i).getPassword();
        JButton edit = new JButton( text: "Edit");
        edit.setName(users.get(i).getUsername());
        arreglo[i][2]=edit;
        JButton delete = new JButton( text: "Delete");
        delete.setName(users.get(i).getUsername());
        arreglo[i][3]=delete;


    }
    return arreglo;
}
```

```java
public static Object [][]DatosClientes(){
    Object [][] arreglo = new Object[clients.size()][7];
    for (int i =0 ; i<clients.size();i++){
        arreglo[i][0]=clients.get(i).getId();
        arreglo[i][1]=clients.get(i).getName();
        arreglo[i][2]=clients.get(i).getAddress();
        arreglo[i][3]=clients.get(i).getNit();
        arreglo [i][4]=clients.get(i).getPhone();
        JButton edit = new JButton( text: "Edit");
        edit.setName(clients.get(i).getId()+"");
        arreglo[i][5]=edit;
        JButton delete = new JButton( text: "Delete");
        delete.setName(clients.get(i).getId()+"");
        arreglo[i][6]=delete;


    }
    return  arreglo;
}
```

```java
public static Object [][]DatosProductos(){

    Object  [][] arreglo = new Object [products.size()][8];
    for (int i =0 ; i<products.size();i++){
        arreglo[i][0]=products.get(i).getId();
        arreglo[i][1]=products.get(i).getName();
        arreglo[i][2]=products.get(i).getCost();
        arreglo[i][3]=products.get(i).getPrice();
        arreglo[i][4]=products.get(i).getDescription();
        arreglo[i][5]=products.get(i).Ingredietes();
        JButton edit = new JButton( text: "Edit");
        edit.setName(""+products.get(i).getId());
        arreglo[i][6]=edit;
        JButton delete = new JButton( text: "Delete");
        delete.setName(""+products.get(i).getId());
        arreglo[i][7]=delete;

    }
 return  arreglo;
}
```

```java
public static Object [][]DatosFacturas(){
    Object  [][] arreglo = new Object [invoices.size()][7];
    for (int i =0 ; i<invoices.size();i++){
        arreglo[i][0]=invoices.get(i).getId();
        //
        arreglo[i][1]=ClienteId(invoices.get(i).getClient());

        arreglo[i][2]=invoices.get(i).getDate();
        arreglo[i][3]=invoices.get(i).NombreProducto();
        arreglo[i][4]=invoices.get(i).Precio();
        JButton edit = new JButton( text: "Edit");
        edit.setName(invoices.get(i).getId()+"");
        arreglo[i][5]=edit;
        JButton delete = new JButton( text: "Delete");
        delete.setName(invoices.get(i).getId()+"");
        arreglo[i][6]=delete;

    }
    return arreglo;

}
```

```
    for (int i=0; i<products.size(); i++) {
        if (products.get(i).getId() == IdPr){
            for (int j = 0; j < products.get(i).getIngredients().size(); j++) {
                arreglo[j][0] = products.get(i).getIngredients().get(j).getName();
                arreglo[j][1] = products.get(i).getIngredients().get(j).getQuantity();
                arreglo[j][2] = products.get(i).getIngredients().get(j).getUnits();
                JButton edit = new JButton( text: "Edit");
                edit.setName("" + products.get(i).getIngredients().get(j).getName());
                arreglo[j][3] = edit;
                JButton delete = new JButton( text: "Delete");
                delete.setName("" + products.get(i).getIngredients().get(j).getName());
                arreglo[j][4] = delete;
            }

        }
    }
    return arreglo;
}
```

Función del botón eliminar

Con estas funciones podemos vincular el botón que aparecerá en las tablas para poder eliminar la posición Del arreglo que esta en dicha fila en nuestra tabla del CRUD

```
public static void BotonEliminarUsuario(String username) {

    for (int i = 0; i < users.size(); i++) {
        if (username.equals(users.get(i).getUsername())) {
            logAcciones+=HoraFecha()+"\t Se Eliminoel Usuario "+users.get(i).getUsername()+"\n";
            logAcciones();
            users.remove(i);

        }
    }
}
public static void BotonEliminarClientes(int idcliente) {

    for (int i = 0; i < clients.size(); i++) {
        if (idcliente == clients.get(i).getId()) {
            logAcciones+= HoraFecha()+ "\tSe Elimino el Cliente "+clients.get(i).getName()+", id:" +clients.get(i).getId()+"\n";
            logAcciones();
            clients.remove(i);
        }
    }
}
```

```java
public static void BotonEliminarIngrediente(int IdProd, String Nombre){
    for (int i=0; i<products.size();i++){
        if (IdProd == products.get(i).getId()){
            for (int j=0; j<products.get(i).getIngredients().size();j++){
                if (Nombre.equals(products.get(i).getIngredients().get(j).getName())){
                    logAcciones+=HoraFecha()+"\tSe Elimino el Ingrediente "+products.get(i).getIngredients().get(j).getName() +", id:"+products.get(i).getId()+"\n";
                    logAcciones();
                    products.get(i).getIngredients().remove(j);
                }
            }
        }
    }
}

public static void BotonEliminarFacturas(int idFactura) {

    for (int i = 0; i < invoices.size(); i++) {
        if (idFactura == invoices.get(i).getId()) {
            logAcciones+=HoraFecha()+"\t Se Elimino la Factura  "+invoices.get(i).getId()+", id :"+invoices.get(i).getId()+"\n";
            logAcciones();
            invoices.remove(i);

        }
    }
}
```

Funcion detección de ID repetido en la creación: Este método aplica un algoritmo general que detecta un objeto de tipo ID, si ya esta repetido, para agregarlo o no.

```
    public static  boolean idCliente(int id ){
        for (int i =0 ; i<clients.size();i++){
            if (id == clients.get(i).getId()){
                return false;
            }
        }
        return true;
    }
    public static  boolean idusuario(String username){
        for (int i =0 ; i<users.size();i++){
            if (username.equals(users.get(i).getUsername()) ){
                return false;
            }
        }
        return true;
    }
    public static  boolean idfacturas(int  id ){
        for (int i =0 ; i<invoices.size();i++){
            if (id == invoices.get(i).getId()){
                return false;
            }
        }
        return true;
    }
    public static boolean idProducts(int  id ){
        for (int i =0 ; i<products.size();i++){
            if (id == products.get(i).getId()){
                return false;
            }
```

Creación del archivo de log de errores:

```
public static String logErrores ="";
public static void logErrores(){

    try {
        FileWriter archivo = new FileWriter("Errores.log");

        archivo.write(logErrores + "\n");


        archivo.close();

    } catch (Exception e) {

    }
}
```

Funciones que recorren lo arreglos para la detección de objetos duplicados, y los eliminan dentro del arreglo con el que se trabaja:

```
public static void  RevisionUsers(){

    int cont=0;
    for (int i = 0;i<users.size();i++)
    {
        for(int j=0;j<users.size();j++)
        {
            if(users.get(i).getUsername().equals(users.get(j).getUsername())) {
                cont++;
                if (cont > 1) {
                    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

                    String HoraFecha=dtf.format(LocalDateTime.now());
                    logErrores+="\n"+HoraFecha+" \t"+" :"+ "Clase Usuarios:  "+"Se repitió el nombre de usuario:  " + users.ge
                    logErrores();

                    users.remove(j);
                }
            }
        }
        cont=0;
    }


}
```

```java
public static void RevisionProducts(){
    int cont=0;
    for (int i = 0;i<products.size();i++)
    {
        for(int j=0;j<products.size();j++)
        {
            if(products.get(i).getId()==(products.get(j).getId())) {
                cont++;
                if (cont > 1) {
                    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

                    String HoraFecha=dtf.format(LocalDateTime.now());
                    logErrores+="\n"+HoraFecha+" \t"+" :"+ "Clase Productos:  "+"Se repitió el ID:  " + products.get(i).getId(
                    logErrores();
                    System.out.println();
                    products.remove(j);
                }
            }
        }
        cont=0;
    }

}
```

```java
public static void RevisionInvoices(){
    int cont=0;
    for (int i = 0;i<invoices.size();i++)
    {
        for(int j=0;j<invoices.size();j++)
        {
            if(invoices.get(i).getId()==(invoices.get(j).getId())) {
                cont++;
                if (cont > 1) {
                    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

                    String HoraFecha=dtf.format(LocalDateTime.now());
                    logErrores+="\n"+HoraFecha+" \t"+" :"+ "Clase Facturas:  "+"Se repitió el ID:  " + invoices.get(i).getId()
                    logErrores();

                    invoices.remove(j);
                }
            }
        }
        cont=0;
    }

}
```

```
    public static void RevisionClients(){
        int cont=0;
        for (int i = 0;i<clients.size();i++)
        {
            for(int j=0;j<clients.size();j++)
            {
                if(clients.get(i).getId()==(clients.get(j).getId())) {
                    cont++;
                    if (cont > 1) {
                        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

                        String HoraFecha=dtf.format(LocalDateTime.now());
                        logErrores+="\n"+HoraFecha+" \t"+" :"+ "Clase Clientes:  "+"Se repitió el ID:  " + clients.get(i).getId()+
                        logErrores();
                        clients.remove(j);
                    }
                }
            }
            cont=0;
        }

    }

}
```