



MANUAL TECNICO



JUNE 7, 2021
GRUPO 7

Tabla de contenido

Introducción.....	2
Objetivos.....	2
1. Vocabulario técnico.....	3
2. Requerimientos.....	4
3. Instalación y configuración	5
4. Configuración del sistema	6
Estructura del proyecto.....	6
Directorio <i>src</i>	7
5. Lógica del programa.....	9

Introducción

La aplicación *RESTAURANT MANAGER* implementa el uso el paradigma de la programación orientada a objetos, abstrayendo y conceptualizando ideas o entornos pertenecientes a la realidad, así como también se incorpora la serialización de los objetos. El documento introduce al usuario en los aspectos y detalles que componen a la aplicación, guiando al usuario de una manera sencilla y específica, mostrando así las características del programa y su funcionamiento.

Objetivos

- ✓ Familiarizar al usuario con la implementación de un sistema grafico compuesto por varias ventanas.
- ✓ Familiarizar al usuario con el manejo de archivos del tipo “*json*”.
- ✓ Brindar al usuario las herramientas e información necesaria para el correcto uso y configuración de la aplicación.
- ✓ Ser una guía para el usuario permitiendo orientarlo a través del entorno que compone a la aplicación.

Vocabulario técnico

- Variable global: variable a la cual se puede acceder desde cualquier ámbito o bloque de código.
- IDE de programación: entorno en el cual se desarrolla el código, es decir, es un editor, compilador, depurador y constructor de código.
- Aplicación de consola: aplicación que se ejecuta dentro de una ventana de línea de comandos.
- CMD: CoMmanD, es un intérprete de comandos incluido en los sistemas operativos de las computadoras
- Java class: se le conocen como plantillas para la creación de objetos dentro de un entorno de programación.
- Librerías: se entiende como un conjunto de clases que facilitan operaciones y tareas ofreciendo al programador funcionalidad ya implementada y lista para ser usada través de una Interfaz de Programación de Aplicaciones
- Interfaz gráfica: es el entorno de objetos gráficos disponibles para un usuario en el marco de una aplicación o sistema operativo.
- Herencia: mecanismo que permite la definición de una clase a partir de la definición de otra ya existente
- Encapsulamiento: hace referencia a limitar el acceso a las variables de nuestras clases Java de tal forma que podamos tener un mayor control sobre ellas.
- Archivos Json: es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de Java
- Serializacion: consiste en generar una secuencia de bytes lista para su almacenamiento o transmisión.

1. Requerimientos

La aplicación puede ser ejecutada en cualquier sistema operativo, sin embargo, al ser una aplicación de consola requiere de los siguientes requerimientos:

- Software de Java versión 1.08
- IDE de Programación (IntelliJ, eclipse, Netbeans, etc)
- Java virtual machine

I. Windows

- ✓ Windows 10 (8u51 y superiores)
- ✓ Windows 8.x (escritorio)
- ✓ Windows 7 SP1
- ✓ Windows Vista SP2
- ✓ Windows Server 2008 R2 SP1 (64 bits)
- ✓ Windows Server 2012 y 2012 R2 (64 bits)
- ✓ RAM: 128 MB
- ✓ Espacio en disco: 124 MB para JRE; 2 MB para Java Update
- ✓ Procesador: Mínimo Pentium 2 a 266 MHz
- ✓ Exploradores: Internet Explorer 9 y superior, Firefox
- ✓ Resolución de pantalla: 1024×768 resolución mínima de pantalla

II. Mac OS X

- ✓ Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+
- ✓ Privilegios de administrador para la instalación
- ✓ Explorador de 64 bits
- ✓ Se requiere un explorador de 64 bits (Safari, Firefox, por ejemplo) para ejecutar Oracle Java en Mac OS X.

III. Linux

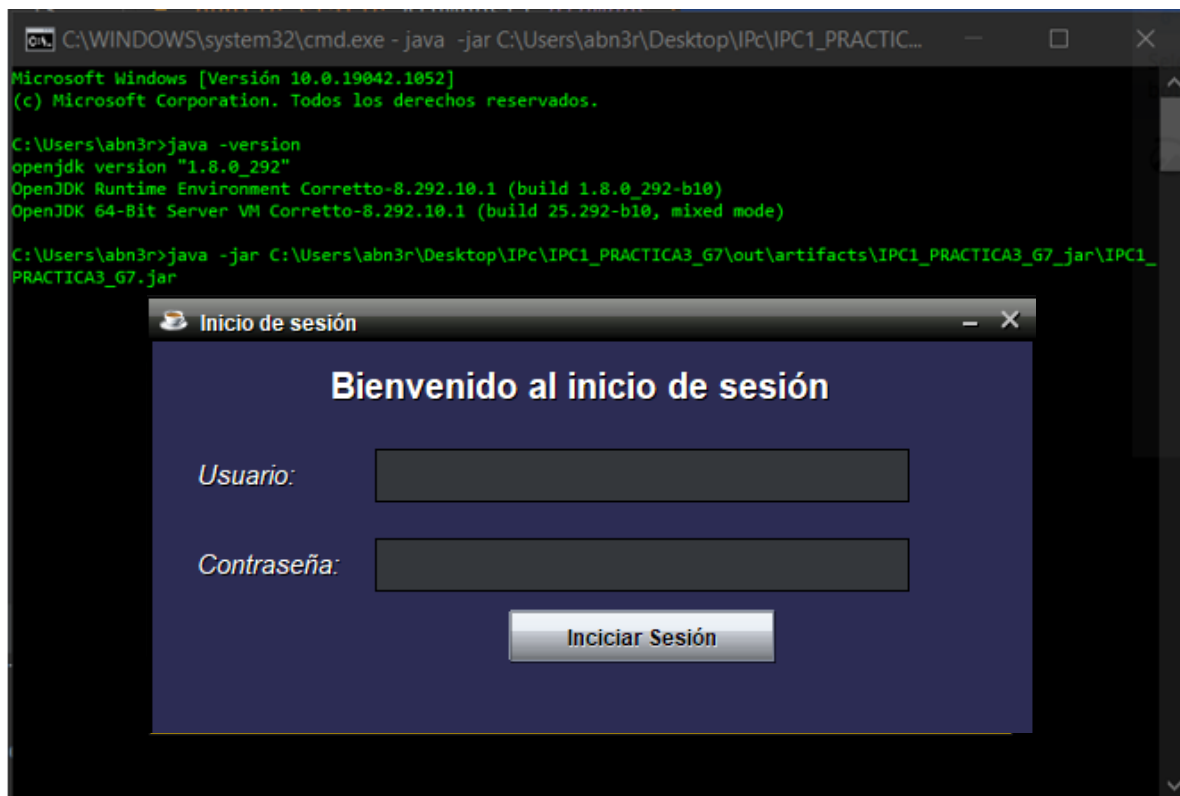
- ✓ Oracle Linux 5.5+1
- ✓ Oracle Linux 6.x (32 bits), 6.x (64 bits)2
- ✓ Oracle Linux 7.x (64 bits)2 (8u20 y superiores)
- ✓ Red Hat Enterprise Linux 5.5+1, 6.x (32 bits), 6.x (64 bits)2
- ✓ Red Hat Enterprise Linux 7.x (64 bits)2 (8u20 y superiores)
- ✓ Suse Linux Enterprise Server 10 SP2+, 11.x
- ✓ Suse Linux Enterprise Server 12.x (64 bits)2 (8u31 y superiores)
- ✓ Ubuntu Linux 12.04 LTS, 13.x
- ✓ Ubuntu Linux 14.x (8u25 y superiores)
- ✓ Exploradores: Firefox

2. Instalación y configuración

Nos encontramos con una aplicación del tipo de escritorio, es decir, cuenta con una interfaz gráfica, la cual la hace más amigable al usuario. Por lo tanto, para la ejecución de la aplicación de la aplicación solo necesitaremos ejecutar el “JAR” para que posteriormente nuestra interfaz gráfica sea mostrada.

Para ello, se deberán de seguir los siguientes pasos:

1. Descargue la carpeta que contiene la aplicación y guarde la ruta del archivo (Ejemplo: C:\Users\user\files)
2. Posteriormente, ejecutar el comando *CMD* de la computadora:
 - 2.1 Utilizar la combinación de teclas *windows+enter*, en el cuadro de texto escribir “cmd” y presionar enter.
3. Una vez ejecutado el *CMD* de la computadora, escribir lo siguiente, “*java -jar + ruta del archivo*”.
4. A continuación, la aplicación será ejecutada.



Nota: para el correcto funcionamiento de la aplicación, es necesario que se cuente con la versión indicada de java (véase especificaciones técnicas y requerimientos).

3. Configuración del sistema

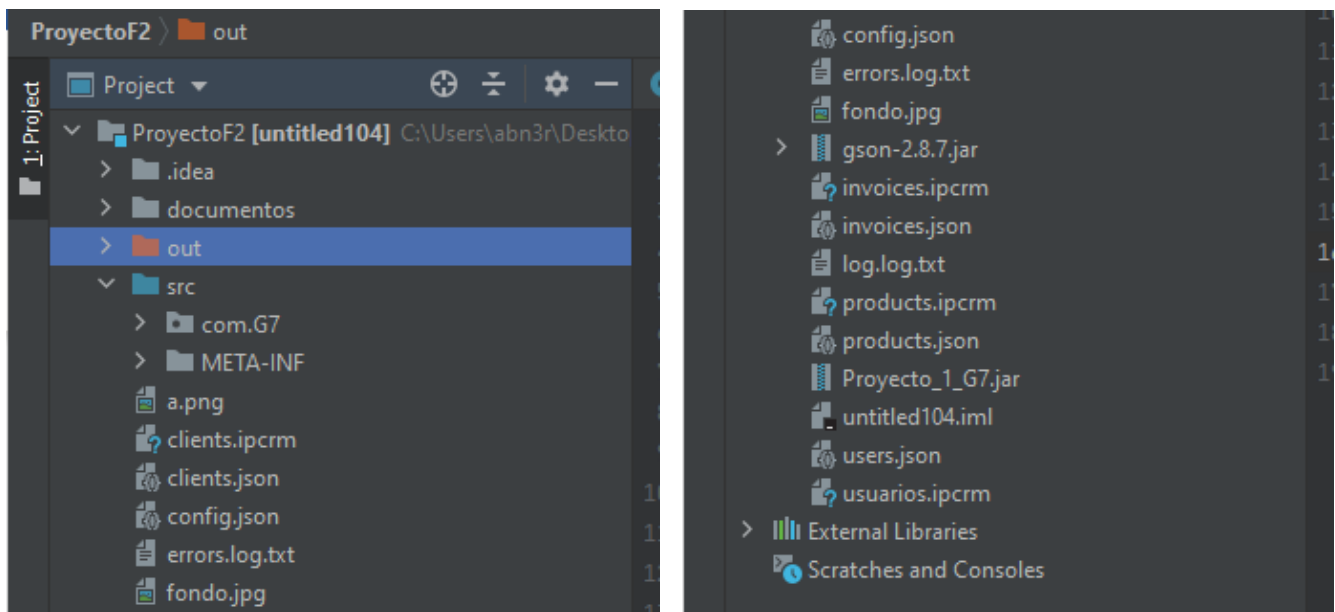
La aplicación *RESTAURANT MANAGER* está desarrollada bajo el lenguaje de programación JAVA, lo cual permite un desarrollo y ejecución en casi cualquier tipo de dispositivo que cuente con la Java virtual machine. A través del uso de propiedades entre clases, como lo es la herencia, se desarrolló el apartado gráfico de la aplicación, implementación que será descrita más adelante.

A continuación, se describe como *RESTAURANT MANAGER* está confirmado.

Estructura del proyecto

Con el uso de la programación orientada a objetos se logra la abstracción de los elementos a utilizar para el desarrollo de la aplicación. Dichos objetos determinan a forma en la que la aplicación funciona. Con base en esto se desarrollaron distintas clases para identificar los objetos, así como también procesos designados para cada uno de los mismos.

además del uso de clases, también se emplea el uso de librerías que contienen métodos que facilitan el desarrollo de la aplicación.

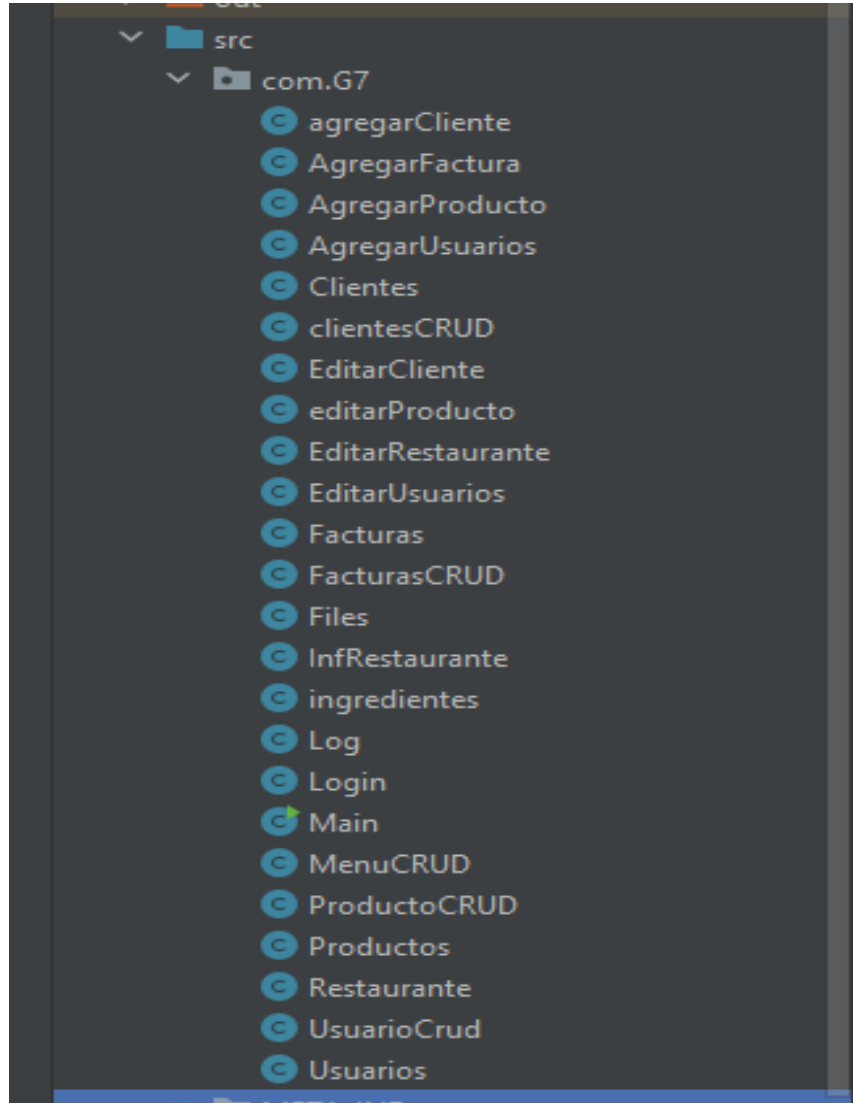


Nota: estructura general de RESTAURANT MANAGER.

Directorio *src*

Dentro de este directorio se encuentran los recursos del programa, o bien, las clases que representan a los objetos del entorno real.

Es en este directorio donde se encuentra el código fuente de la aplicación



Nota: paquete de recursos de RESTAURANT MANAGER.

Vistazo al desarrollo de la interfaz gráfica:

```
7 import java.util.ArrayList;
8 import java.util.Date;
9
10 public class Login extends JFrame implements ActionListener{
11
12     public static String user;
13     JFrame Ventana;
14     String psw, hola, adios;
15     JLabel l1, l2, l3, l4;
16     JTextField t1;
17     JPasswordField t2;
18     JButton b1;
19     public Login(){
20         Ventana = new JFrame( title: "Inicio de sesión");
21         Ventana.setBounds( x: 700, y: 400, width: 500, height: 250);
22         Ventana.setLayout(null);
23         Ventana.getContentPane().setBackground(Color.orange);
24         Ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         Ventana.setResizable(false);
26         Ventana.setVisible(true);
27
28         l1 = new JLabel( text: "Bienvenido al inicio de sesión.");
29         l1.setFont(new Font( name: "arial", Font.BOLD, size: 20));
30         l1.setBounds( x: 100, y: 10, width: 400, height: 30);
31         l1.setVisible(true);
32         Ventana.add(l1);
33
34         l2 = new JLabel( text: "Usuario:");
35         l2.setFont(new Font( name: "arial", Font.ITALIC, size: 15));
36         l2.setBounds( x: 25, y: 50, width: 400, height: 50);
37         l2.setVisible(true);
38         Ventana.add(l2);
39
40         l3 = new JLabel( text: "Contraseña:");
41         l3.setFont(new Font( name: "arial", Font.ITALIC, size: 15));
42         l3.setBounds( x: 25, y: 100, width: 400, height: 50);
43         l3.setVisible(true);
```

Es la clase “*Jframe*” la que permite el desarrollo de la aplicación a través de una ventana de escritorio.

Nuestra clase *Login* extiende de la clase *Jframe*, en consecuencia, tiene acceso a los métodos de dicha clase, tal como agregar paneles, botones y ventanas.

4. Lógica del programa

En cuanto a el desarrollo de la aplicación, la aplicación fue desarrollada tomando el paradigma de la programación orientada a objetos, con el desarrollo de distintas clases y objetos que permitieron la abstracción del entorno y ambiente real, además que la aplicación también cuenta con el uso de un paquete de clases propia del lenguaje Java, para la generación de ventanas, dentro de las cuales el usuario interactúa.

```
package com.G7;

import java.io.Serializable;

public class Clientes implements Serializable {
    public int id;
    public String name;
    public String address;
    public int phone;
    public String nit;

    public Clientes (int id, String name,String address, int phone, String nit){
        this.id = id;
        this.name = name;
        this.address = address;
        this.phone = phone;
        this.nit = nit;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public int getPhone() {
        return phone;
    }

    public void setPhone(int phone) {
        this.phone = phone;
    }

    public String getNit() {
        return nit;
    }

    public void setNit(String nit) {
        this.nit = nit;
    }

    public Object[] toArreglo() {
        String retorno[] = {String.valueOf(this.id), this.name,
this.address,String.valueOf(this.phone), this.nit};
        return retorno;
    }
}
```

Ejemplo de una clase u objeto, “Clientes”.

Cada clase que representa a un objeto se compone de diversos elementos, entre los cuales se encuentran los constructores, *setters* and *getters*. Son parámetros que posteriormente se utilizan en la asignación de actividades.

La aplicación cuenta con el uso de una clase “propia” del sistema. Por lo tanto y de la misma forma, al ser una clase, cuenta con las mismas propiedades que cualquier otra, como lo es la herencia o el encapsulamiento:

```
public class MenuCRUD extends JFrame implements ActionListener{
    JFrame menuc;
    JButton b1,b2,b3,b4,b5,b6,b7,b8;
    JLabel l1;

    public MenuCRUD(){
        menuc =new JFrame( title: "Menu");
        menuc.setBounds( x: 700, y: 400, width: 600, height: 300);
        menuc.setLayout(null);
        menuc.getContentPane().setBackground(Color.orange);
        menuc.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        menuc.setResizable(false);
        menuc.setVisible(true);

        l1 = new JLabel( text: "Bienvenido al menu");
        l1.setFont(new Font( name: "arial", Font.BOLD, size: 20));
        l1.setBounds( x: 200, y: 10, width: 400, height: 30);
        l1.setVisible(true);
        menuc.add(l1);

        b1 = new JButton( text: "Informacion del restaurante");
        b1.setBounds( x: 25, y: 50, width: 250, height: 30);
        b1.setVisible(true);
        b1.addActionListener( this);
        menuc.add(b1);

        b2 = new JButton( text: "CRUD Usuarios");
        b2.setBounds( x: 300, y: 50, width: 250, height: 30);
        b2.setVisible(true);
        b2.addActionListener( this);
        menuc.add(b2);
    }
}
```

Código correspondiente a una ventana de RESTAURANT MANAGER

La aplicación contará con archivos previamente cargados, este tipo de archivos son del tipo Json, el cual es un tipo de archivo que se utiliza para representar datos estructurados en la sintaxis de objetos.

```
[
  {
    "id": 1,
    "name": "Darwin Arevalo",
    "address": "Ciudad de Guatemala",
    "phone": 42562365,
    "nit": "5283282-3"
  },
  {
    "id": 2,
    "name": "Chus",
    "address": "Ciudad de Guatemala",
    "phone": 42562365,
    "nit": "5283282-3"
  },
]
```

Ejemplo de archivos Json, correspondiente a los clientes de RESTAURANT MANAGER.

Para asignar las diversas actividades y funciones con las que la aplicación contará, se desarrollaron distintos métodos, como los que se muestran a continuación, dichos métodos se encuentran en nuestro código fuente, en la clase principal.

```
public static void menuUsuarios() {
    System.out.println("01. Listar Usuarios.");
    System.out.println("02. Eliminar Usuario.");
    System.out.println("03. Ver Usuarios.");
    System.out.println("04. Regresar al menu principal.");
    String menu_opciones = "";
    menu_opciones = consola.nextLine();

    switch (menu_opciones) {
        case "1":
            mostrarUsuario();
            break;
        case "2":
            eliminarUsuario();
            break;
        case "3":
            verUsuario();
            break;
        case "4":
            System.out.println("\n");
            break;
        default:
            System.out.println("Opción no valida\n");
            break;
    }
}

public static void mostrarUsuario() {
    System.out.println("ID\t\t Username\t\t\t password*");
    for (int i = 0; i < usuariosArr.size(); i++) {
        System.out.println("|" + i + " \t\t" + usuariosArr.get(i).getUsername() + "\t\t" + usuariosArr.get(i).getPassword() + "|");
    }
}
```

```

public static void eliminarUsuario() {
    try {
        System.out.println("Escriba el id del usuario que desea eliminar.");
        int eliminar = consola.nextInt();
        String nombreEliminado= usuariosArr.get(eliminar).getUsername();
        usuariosArr.remove(eliminar);
        Log.addToEndFile( pathname: "log.log.txt", data: " " + new Date().toString() + "\t---" + usuarioLog+" "+" Elimino el usuario "+" \u0022"+nombreEliminado +"\u0022"+" co
        System.out.println(usuarioLog+" "+" Elimino el usuario "+" \u0022"+nombreEliminado +"\u0022"+" con id "+" eliminar);
    } catch (Exception e) {
        Log.addToEndFile( pathname: "errors.log.txt", data: " " + new Date().toString() + "\t---" + "Usuarios: Error en la eliminación de usuario, el id no existe." + "\n")
    }
}

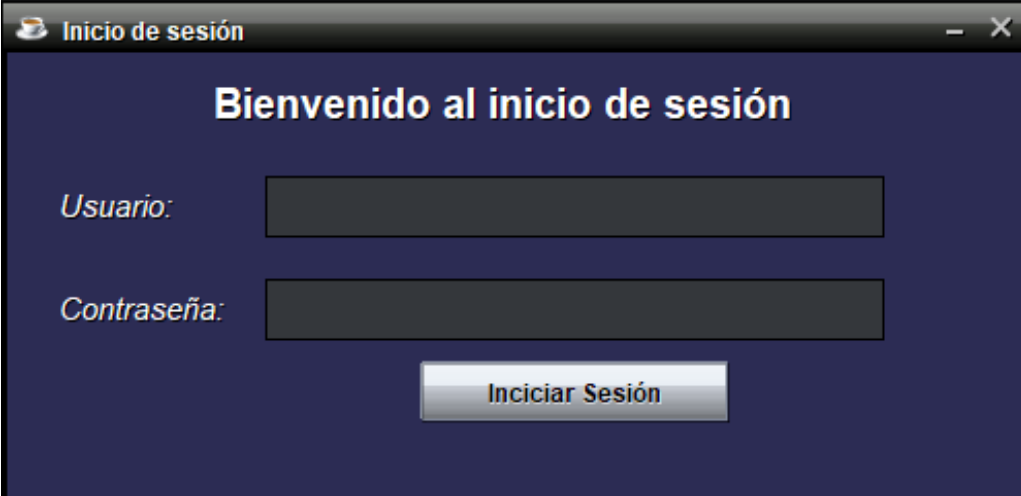
public static void verUsuario() {
    try {
        System.out.println("Escriba el id del usuario que desea ver:");
        int ver = consola.nextInt();
        System.out.println("Username: " + usuariosArr.get(ver).getUsername());
        System.out.println("Password: " + usuariosArr.get(ver).getPassword());
        System.out.println();
    } catch (Exception e){
        Log.addToEndFile( pathname: "errors.log.txt", data: " " + new Date().toString() + "\t---Usuarios: Error de lectura de mostrar usuario, el usuario con id ingresado no
    }
}

public static void menuProductos() {
    System.out.println("01. Listar Productos.");
    System.out.println("02. Eliminar Producto.");
    System.out.println("03. Ver Producto.");
    System.out.println("04. Regresar al menu principal.");
    String menu_opciones = "";
    menu_opciones = consola.nextLine();
}

```

Es de esta forma y a través del uso de distintas herramientas como lo son las librerías o clases que la aplicación fue desarrollada, integrando y aplicando diversos métodos, funciones y conocimientos

Obteniendo como resultado final lo siguiente:



The image shows a Java Swing window titled "Inicio de sesión". The window has a dark blue background and a title bar with standard Windows window controls. The main content area displays the text "Bienvenido al inicio de sesión" in a large, white, serif font. Below this text, there are two input fields for user credentials. The first field is labeled "Usuario:" and the second is labeled "Contraseña:". Both labels are in a white, italicized serif font. At the bottom center of the window, there is a button with a light blue gradient and the text "Iniciar Sesión" in a bold, black, sans-serif font.

