Interfaz

Peluquerías Paquita

DAVANTE | MEDAC
Carlos Piernas Jiménez
Desarrollo de interfaces
José Ferrer Grau
30-11-2025

Índice

Introducción.

Como proyecto final de trimestre se nos ha solicitado el desarrollo de una aplicación de escritorio completa en Java con la librería Swing. El objetivo principal es gestionar y representar visualmente el funcionamiento de la "Peluquería Paquita" en tiempo real. Esta interfaz debe estar dividida en dos apartados:

1. Un panel de gestión de datos donde se podrán consultar los datos de la base de datos creada anteriormente para la asignatura de “Acceso a Datos” y llamada “peluquería” y sincronizar una simulación a la base de datos.
2. Un panel de simulación, el cual servirá para controlar y visualizar los procesos de concurrencia realizados en la asignatura de “Programación de Servicios y Procesos”

Arquitectura de la aplicación.

Esta aplicación se basa en la estructura modular MVC (Modelo - Vista - Controlador).

Modelo: Este módulo se encarga de la lógica de la aplicación, usando las clases de conexión a la base de datos y las clases necesarias para que funcione la simulación.

Vista: Este módulo se encarga de la parte visual de la aplicación, así como el uso de los diferentes botones y funcionalidades generales.

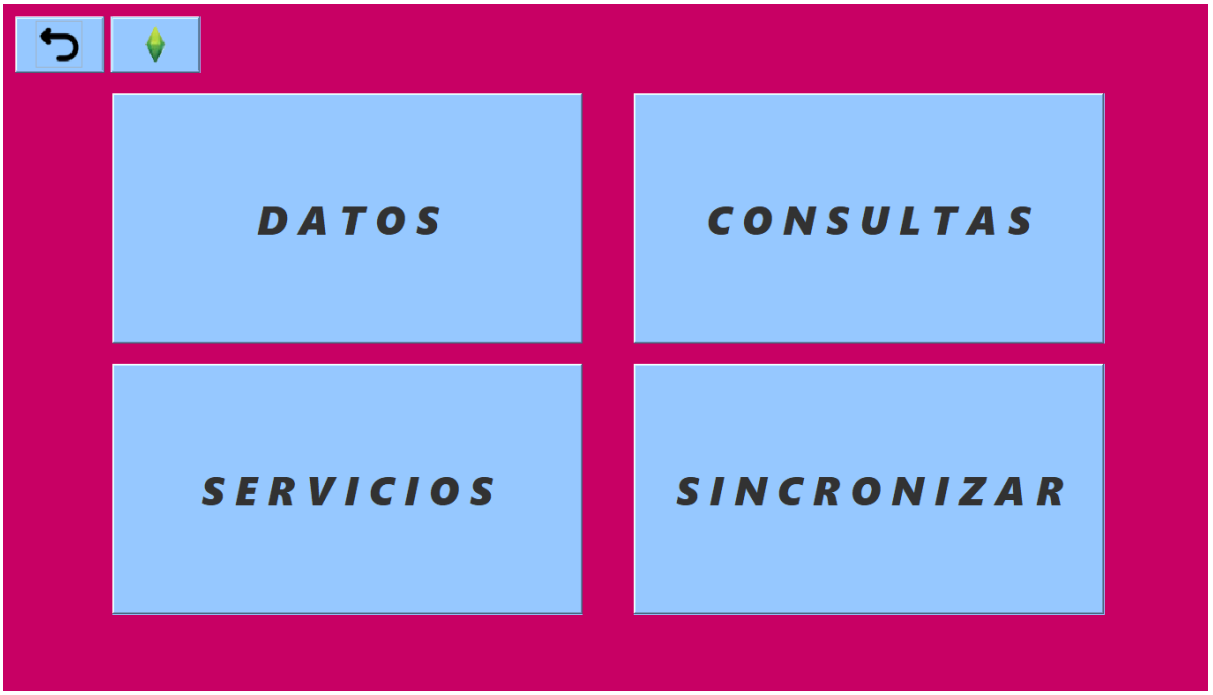
Controlador: Este módulo se encarga de iniciar el programa, manteniendo solo la clase main dentro.

Para moverse entre las diferentes vistas creadas se utiliza el método abrir de cada clase, asociándolo al botón correspondiente.

Panel de gestión de datos.

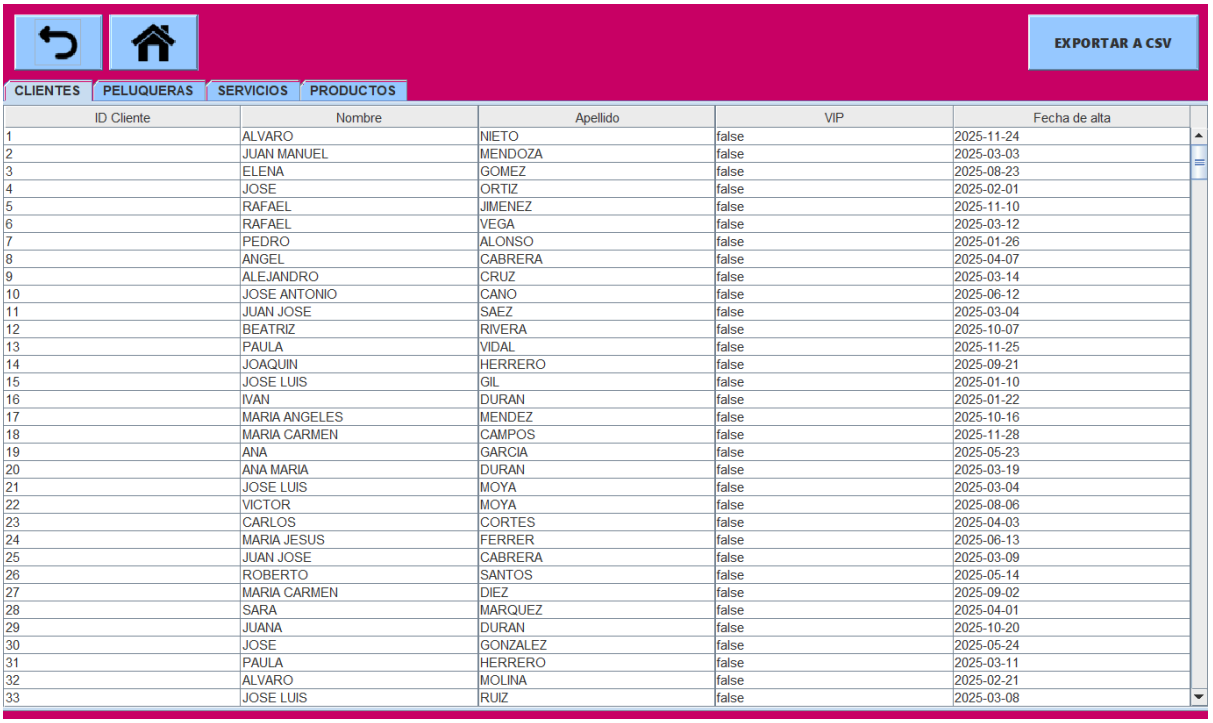
Al ingresar en este panel tenemos cuatro botones, los cuales nos llevan a las diferentes opciones de consultas a la base de datos conectada por el driver JDBC.

Al igual que dos botones en la parte superior que nos permiten tanto volver a la ventana principal como a la ventana de simulación.



Estos cuatro botones serían los siguientes:

1. Datos.



ID Cliente	Nombre	Apellido	VIP	Fecha de alta
1	ALVARO	NIETO	false	2025-11-24
2	JUAN MANUEL	MENDOZA	false	2025-03-03
3	ELENA	GOMEZ	false	2025-08-23
4	JOSE	ORTIZ	false	2025-02-01
5	RAFAEL	JIMENEZ	false	2025-11-10
6	RAFAEL	VEGA	false	2025-03-12
7	PEDRO	ALONSO	false	2025-01-26
8	ANGEL	CABRERA	false	2025-04-07
9	ALEJANDRO	CRUZ	false	2025-03-14
10	JOSE ANTONIO	CANO	false	2025-06-12
11	JUAN JOSE	SAEZ	false	2025-03-04
12	BEATRIZ	RIVERA	false	2025-10-07
13	PAULA	VIDAL	false	2025-11-25
14	JOAQUIN	HERRERO	false	2025-09-21
15	JOSE LUIS	GIL	false	2025-01-10
16	IVAN	DURAN	false	2025-01-22
17	MARIA ANGELES	MENDEZ	false	2025-10-16
18	MARIA CARMEN	CAMPOS	false	2025-11-28
19	ANA	GARCIA	false	2025-05-23
20	ANA MARIA	DURAN	false	2025-03-19
21	JOSE LUIS	MOYA	false	2025-03-04
22	VICTOR	MOYA	false	2025-08-06
23	CARLOS	CORTES	false	2025-04-03
24	MARIA JESUS	FERRER	false	2025-06-13
25	JUAN JOSE	CABRERA	false	2025-03-09
26	ROBERTO	SANTOS	false	2025-05-14
27	MARIA CARMEN	DIEZ	false	2025-09-02
28	SARA	MARQUEZ	false	2025-04-01
29	JUANA	DURAN	false	2025-10-20
30	JOSE	GONZALEZ	false	2025-05-24
31	PAULA	HERRERO	false	2025-03-11
32	ALVARO	MOLINA	false	2025-02-21
33	JOSE LUIS	RUIZ	false	2025-03-08

Al seleccionar este botón se nos llevará a una pantalla con cuatro pestañas. Estas cuatro pestañas muestran las tablas completas de clientes, peluqueras, servicios y productos en ese orden. También tenemos en la parte superior los botones para

volver atrás y al inicio. Además un botón que exporta los datos de estas cuatro tablas en formato CSV y que está relacionado con un punto extra.

2. Consultas.

Nombre	Apellido	Visitas totales
MARIA JOSE	TORRES	12
PATRICIA	LOZANO	11
ROBERTO	SANTOS	11
SANTIAGO	SANTIAGO	11
MARIA CARMEN	LOZANO	11
JOSE MARIA	CABALLERO	11
MONICA	IGLESIAS	11
MARIA DOLORES	VAZQUEZ	10
PABLO	SUAREZ	10
DOLORES	MENDOZA	10

En esta pestaña se hacen consultas directas personalizadas a la base de datos. Estas consultas serían las siguientes:

1. Ranking de clientes con más visitas:

```
String query = "SELECT nombre, apellido, \"Visitas totales\" FROM clientesVisitas ";
```

```
CREATE VIEW clientesVisitas AS
SELECT cl.nombre, cl.apellido,COUNT(c.id_cita) AS "Visitas totales"
FROM clientes cl
JOIN citas c ON cl.id_cliente = c.id_cliente_clientes
GROUP BY cl.id_cliente, cl.nombre, cl.apellido
ORDER BY "Visitas totales" DESC
LIMIT 10;
```

2. Ranking de servicios más solicitados:

```
String query = "SELECT nombre, \"Solicitudes totales\" FROM serviciosSolicitados ";
```

```
CREATE VIEW serviciosSolicitados AS
SELECT s.nombre, COUNT(cs.id_cita_citas) AS "Solicitudes totales"
FROM servicios s
JOIN citas_servicios cs ON s.id_servicio = cs.id_servicio_servicios
GROUP BY s.nombre
ORDER BY "Solicitudes totales" DESC
LIMIT 5;
```

3. Ranking de días con más citas:

```
String query = "SELECT fecha, \"Citas totales\" FROM diasCitas";
```

```
CREATE VIEW diasCitas AS
SELECT c.fecha, COUNT(c.id_cita) AS "Citas totales"
FROM citas c
GROUP BY c.fecha
ORDER BY "Citas totales" DESC
LIMIT 10;
```

4. Ranking de citas más caras:

```
String query = "SELECT id_cita, CAST(precio as decimal (10,2)), nombre, apellido FROM citasCaras";
```

```
CREATE VIEW citasCaras AS
SELECT c.id_cita, CAST(c.precio as decimal (10,2)), cl.nombre, cl.apellido
FROM citas c
JOIN clientes cl ON c.id_cliente_clientes = cl.id_cliente
ORDER BY c.precio DESC
LIMIT 5;
```

5. Clientes que han pagado con tarjeta:

```
String query = "SELECT distinct nombre, apellido, metodo_pago FROM clientesTarjeta";
```

```
CREATE VIEW clientesTarjeta AS
SELECT distinct cl.nombre, cl.apellido, f.metodo_pago
FROM clientes cl
JOIN citas c ON c.id_cliente_clientes = cl.id_cliente
JOIN facturas f ON c.id_factura_facturas = f.id_factura
WHERE f.metodo_pago='tarjeta';
```

6. Clientes que han pagado con efectivo:

```
String query = "SELECT distinct nombre, apellido, metodo_pago FROM clientesEfectivo";
```

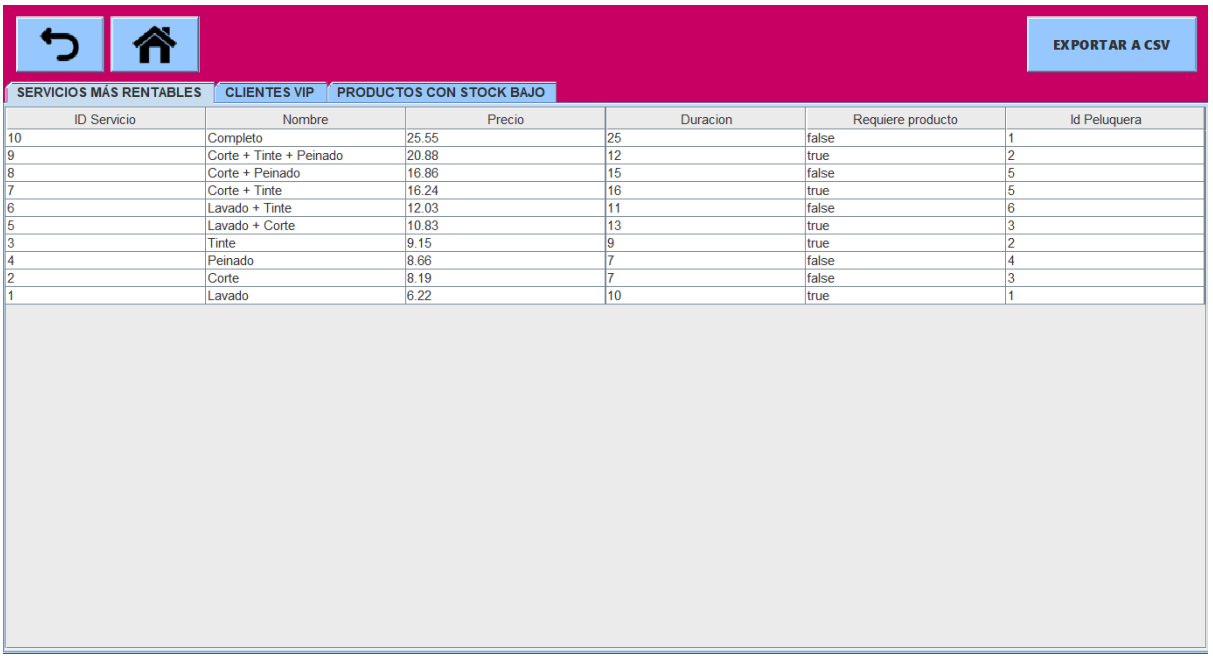
```
CREATE VIEW clientesEfectivo AS
SELECT distinct cl.nombre, cl.apellido, f.metodo_pago
FROM clientes cl
JOIN citas c ON c.id_cliente_clientes = cl.id_cliente
JOIN facturas f ON c.id_factura_facturas = f.id_factura
WHERE f.metodo_pago='efectivo';
```

7. Clientes que han pagado con tarjeta de compra del corte inglés:

```
String query = "SELECT distinct nombre, apellido, metodo_pago FROM clientesCorteIngles";
```

```
CREATE VIEW clientesCorteIngles AS
SELECT distinct cl.nombre, cl.apellido, f.metodo_pago
FROM clientes cl
JOIN citas c ON c.id_cliente_clientes = cl.id_cliente
JOIN facturas f ON c.id_factura_facturas = f.id_factura
WHERE f.metodo_pago='tarjeta de compra del Corte Inglés';
```

3. Servicios.



ID Servicio	Nombre	Precio	Duracion	Requiere producto	Id Peluquera
10	Completo	25.55	25	false	1
9	Corte + Tinte + Peinado	20.88	12	true	2
8	Corte + Peinado	16.86	15	false	5
7	Corte + Tinte	16.24	16	true	5
6	Lavado + Tinte	12.03	11	false	6
5	Lavado + Corte	10.83	13	true	3
3	Tinte	9.15	9	true	2
4	Peinado	8.66	7	false	4
2	Corte	8.19	7	false	3
1	Lavado	6.22	10	true	1

Esta ventana muestra tres pestañas donde se consultan los servicios más rentables, los clientes VIP y los productos con stock bajo.

4. Sincronizar.

Este botón no hace nada, pues no se ha logrado sincronizar la simulación de la asignatura “Programación de servicios y procesos” a esta práctica.

Conexión JDBC.

Las consultas realizadas en las ventanas anteriores han sido realizadas por la clase conexión del proyecto. En esta clase cada método se conecta a la base de datos mediante las clases importadas “Connection” y “DriverManager”. Esto se ejecuta de la siguiente manera:

```
try (Connection connection = DriverManager.getConnection(url, "postgres", "3434");
     Statement stmt = connection.createStatement();
     ResultSet rs = stmt.executeQuery(query)) {
```

En este caso la variable “url” que pasa por parámetro es un String donde se determina la ubicación de la base de datos.

```
String url = "jdbc:postgresql://localhost:5432/peluqueria";
```

Luego se crea el objeto “stmt” de la clase Statement y el objeto “rs” de la clase “ResultSet”, esto hará que el objeto “rs” ejecute el “query” (la consulta) gracias a las propiedades del objeto “stmt”.

Una vez realizada la consulta se guardan los datos en un array mediante un bucle.

```
//bucle para obtener los datos de la consulta
while (rs.next()) {
    //Crea un arrays de objetos llamado fila
    Object[] fila = new Object[5];
    //Cada objeto del arrays se llena con los atributos de las tablas: id_cliente, nombre, apellido, vip, fecha_alta
    fila[0] = rs.getString("id_cliente");
    fila[1] = rs.getString("nombre");
    fila[2] = rs.getString("apellido");
    fila[3] = rs.getBoolean("vip");
    fila[4] = rs.getString("fecha_alta");
    //añade estos datos en orden al arrays datosClientes
    datosClientes.add(fila);
```

Todo este código está dentro de un método “List<Object[]>” que devuelve un array cuando es llamado.

Este método se llama en la clase donde se quieren mostrar los datos, y se copia el array que devuelve a un nuevo array

```
//Copia el arrays que devuelve la funcion clientes de la clase cx en un arrays
List<Object[]> datosClientes = cx.Clientes();
```

Luego se crea un array multidimensional que tiene como parámetros el array recientemente creado y uno donde se determinan cuántas columnas debe tener la tabla y sus nombres, esto determina el tamaño debe tener.

```
//Crea un arrays con los nombres de las columnas de la tabla
String[] nombresColumnas = {"ID Cliente", "Nombre", "Apellido", "VIP", "Fecha de alta"};

//Crea un arrays multidimensional con las filas del array copiado y las columnas dadas en nombresColumnas
Object[][] datosArray = new Object[datosClientes.size()][nombresColumnas.length];
```

Este último array se llena de datos mediante un bucle y pasa a mostrarse en la ventana.

```
//Bucle para rellenar el arrays datosArray
for (int i = 0; i < datosClientes.size(); i++) {
    datosArray[i] = datosClientes.get(i);
}
```

```
//Crea el modelo de tabla cogiendo los datos de los arrays
DefaultTableModel modelo = new DefaultTableModel(datosArray, nombresColumnas);

//Hace que la tablaClientes tenga el modelo propuesto antes
tablaClientes.setModel(modelo);

//Hace que la tabla haga un autoresize dependiendo del tamaño del arrays
tablaClientes.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_ALL_COLUMNS);
```

Panel de simulación.

Este panel mostrará la simulación desarrollada en la práctica de “Programación de servicios y procesos”. Se ha creado sobre un JScrollPane para que entre toda la información que debe aparecer.

En la parte superior aparece un botón que nos permite volver al inicio y un panel donde aparece la información de los clientes atendidos y pendientes, al igual que el de las peluqueras activas y descansando.

De seguido tendríamos las cuatro zonas en las que se basa la simulación: lavado, corte, tinte y peinado, con los nombres de las peluqueras trabajando en estos servicios y los clientes que están siendo atendidos. Debajo aparecen barras de progreso que determinarán lo avanzado que va el servicio que se está llevando a cabo.

Para el funcionamiento de la simulación se han creado los botones de “Iniciar”, “Pausar/Reanudar” y “Finalizar”, estos botones hacen lo que su nombre indica.

Después aparecen las estadísticas de la simulación, para los que se ha creado un panel independiente para cada requisito.

El primer panel muestra las peluqueras, el estado en el que se encuentran en ese momento (disponible / descansando), y una barra de progreso que determina el tiempo de descanso de cada peluquera.

El segundo panel muestra con una tabla los clientes que han sido atendidos y los que están siendo atendidos.

El tercer panel muestra con una tabla las estadísticas de la simulación, determinando que simulación es, la hora de inicio, la cantidad de siestas y cambios de zonas realizadas en dicha simulación y la hora a la que se ha finalizado la simulación.

El cuarto y último panel está dedicado a mostrar los contadores de servicios completados, el tiempo medio por servicio y las ganancias estimadas de la simulación.

Extras.

Para este proyecto también se ofrecen ciertos extras para subir nota, de los cuales se han realizado:

2. Sistema de usuarios con login.

Para llevar a cabo este extra se ha creado una vista de login, en la cual mediante un ComboBox se selecciona el tipo de usuario que va a ingresar (administrador / empleado) y la contraseña de cada usuario. Para facilitar el acceso se ha configurado ambas contraseñas para que sean “1234”.

3. Exportación de datos a CSV.

Esta función se puede realizar desde el botón de las vistas del panel de gestión de datos mencionado anteriormente.

Este botón llama al método de exportación de datos creado en la clase conexión. Este método sigue una lógica similar a los métodos para mostrar datos en las tablas de las consultas, pero cambiando las consultas por un código SQL que copia los datos a un archivo CSV que se crea en el escritorio.