

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from the bar, containing the text 'DI'.

DI

Práctica 4

GENERACIÓN DE INTERFACES
GRÁFICAS DE USUARIO

Álvaro Manuel Navarro Cruz

2º DAM

04/12/2024

Índice

Introducción (↑).....	4
Enlace a repositorio en GitHub	4
1.- Estructura del proyecto Maven (↑).....	5
2.- Clase GestorConexiones (↑)	6
3.- Clase Principal (↑).....	11
4.- Clase NuevoArticulo (↑)	12
4.1.- WindowBuilder	12
4.2.- Código	12
5.- Clase EditarArticulo (↑)	14
5.1.- WindowBuilder	14
5.2.- Código	14
6.- Clase ConsultarArticulo (↑)	17
6.1.- WindowBuilder	17
6.2.- Código	17
7.- Clase BajaArticulo (↑).....	18
7.1.- WindowBuilder	18
7.2.- Código	18
8.- Clase NuevoTicket (↑)	20
8.1.- WindowBuilder	20
8.2.- Código	20
9.- Clase ConsultaTicket (↑).....	23
9.1.- WindowBuilder	23
9.2.- Código	23
10.- Ejecución (↑).....	25
10.1.- Menú Principal.....	25
10.2.- Nuevo Artículo	26
10.3.- Editar Artículo	29
10.4.- Consultar Artículo	32
10.5.- Baja Artículo.....	33
10.6.- Nuevo Ticket	36

10.7.- Consulta Ticket	42
11.- Valoración Personal (↑).....	43
12.-Bibliografía	44

Introducción

En esta práctica se nos pide completar la funcionalidad del programa diseñado en la práctica 2 de la asignatura “Desarrollo de Interfaces”. En la que tuvimos que realizar el boceto inicial del programa en WindowBuilder.

La funcionalidad a completar será la de realizar operaciones CRUD (CREATE, READ, UPDATE, DELETE) en la base de datos también diseñada en la práctica 2, llamada, en este caso, “TiendecitaANC”.

Se debe aportar funcionalidad CRUD completa a la tabla “Artículos” y funcionalidad de creación y consulta para la tabla “Tickets”.

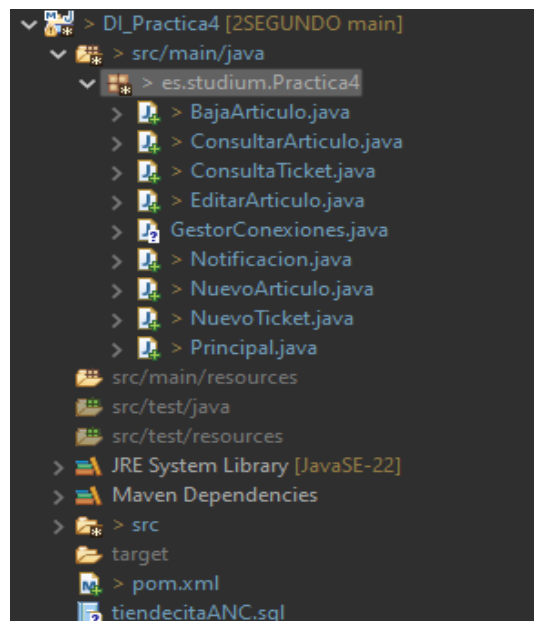
Se debe proporcionar capturas de ejecución, así como del diseño en WindowBuilder y del propio código.

Enlace a repositorio en GitHub

https://github.com/AlvaroMfco/DI_Practica4.git

1.- Estructura del proyecto Maven

Desde el explorador de paquetes, encontraremos la siguiente estructura:



El archivo pom.xml, encontraremos las dependencias de MySQL, además de las propiedades del compilador.

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<?xml version="1.0" encoding="UTF-8"?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4  <groupId>es.studium.Practica4</groupId>
5  <artifactId>DI_Practica4</artifactId>
6  <version>0.0.1-SNAPSHOT</version>
7  <name>DI_Practica4</name>
8  <dependencies>
9    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
10    <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
11    <dependency>
12      <groupId>com.mysql</groupId>
13      <artifactId>mysql-connector-j</artifactId>
14      <version>9.1.0</version>
15    </dependency>
16  </dependencies>
17
18  <properties>
19    <maven.compiler.source>22</maven.compiler.source>
20    <maven.compiler.target>22</maven.compiler.target>
21  </properties>
22
23</project>
```

2.- Clase GestorConexiones

```

1 package es.studium.Practica4;
2
3 import java.sql.Connection;
13
14 public class GestorConexiones {
15     private final static String MySQL_DB_USUARIO = "root";
16     private final static String MySQL_DB_PASSWORD = "Stodium2023";
17
18     private final static String MySQL_DB_DRIVER = "com.mysql.cj.jdbc.Driver";
19     private final static String MySQL_DB_URL = "jdbc:mysql://localhost/";
20
21     public static Connection getMySQL_Connection(String database) {
22
23         Connection connMySQL = null;
24         try {
25             Class.forName(MySQL_DB_DRIVER);
26             connMySQL = DriverManager.getConnection(MySQL_DB_URL + database, MySQL_DB_USUARIO,
27                 MySQL_DB_PASSWORD);
28
29         } catch (ClassNotFoundException e) {
30             e.printStackTrace();
31         } catch (SQLException eq) {
32             eq.printStackTrace();
33         }
34         return connMySQL;
35     }
36     //Método para Insertar un artículo en la BBDD
37     public static void insertarArticulo(String descripcion, String precio, String stock) {
38         Connection conexion = getMySQL_Connection("tiendecitaanc");
39         //Sentencia SQL para el alta.
40         String sentencia = "INSERT INTO articulos VALUES (null, '" + descripcion + "', " + precio
41             + ", " + stock + ")";
42         Statement st;
43         try {
44             //Creamos el statement y lanzamos la sentencia SQL
45             st = conexion.createStatement();
46             st.executeUpdate(sentencia);
47             //Si se ejecuta correctamente, mostramos el diálogo de éxito
48             new Notificacion("NuevoArticulo", "creado").setVisible(true);
49         } catch (SQLException e) {
50             //Si no se ejecuta correctamente, mostramos el diálogo de error
51             new Notificacion("NuevoArticulo", "error").setVisible(true);
52             e.printStackTrace();
53         }
54     }
55     //Método para rellenar el JComboBox que contiene los artículos
56     public static void rellenarChoiceArticulos(JComboBox<String> comboBox) {
57         //Primero eliminamos el contenido, para agregarlos en el bucle y así mantenerlos actualizados.
58         comboBox.removeAllItems();
59         Connection conexion = getMySQL_Connection("tiendecitaanc");
60         //Sentencia SQL de consulta.
61         String sentencia = "SELECT * FROM tiendecitaanc.articulos";
62         try {
63             Statement st = conexion.createStatement();
64             //Ejecutamos la sentencia SQL.
65             ResultSet rs = st.executeQuery(sentencia);
66             comboBox.addItem("Seleccionar Artículo");
67             //Mientras haya datos, los añadimos al JComboBox mostrando su id y descripción.
68             while (rs.next()) {
69                 comboBox.addItem(rs.getString("idArticulo") + " - " + rs.getString("descripcionArticulo"));
70             }
71         } catch (SQLException e) {
72             //Si no se puede rellenar el JComboBox, mostramos el diálogo de error.
73             new Notificacion("NuevoArticulo", "errorChoice").setVisible(true);
74             e.printStackTrace();
75         }
76     }

```

```

77 //Método para rellenar la tabla de artículos de un artículo seleccionado en "Editar Artículo".
78 public static String[] rellenarArticulo(String id) {
79     Connection conexion = getMySQL_Connection("tiendecitaanc");
80     //Obtenemos los campos que necesitamos filtrando por el id pasado como parámetro.
81     String sentencia = "SELECT descripcionArticulo, precioArticulo, stockArticulo FROM "
82         + "tiendecitaanc.articulos WHERE idArticulo = " + id + ";";
83     try {
84         Statement st = conexion.createStatement();
85         //Lanzamos la sentencia SQL.
86         ResultSet rs = st.executeQuery(sentencia);
87         String[] datos = { "", "", "" };
88         while (rs.next()) {
89             //Almacenamos en el array los datos obtenidos.
90             datos[0] = rs.getString("descripcionArticulo");
91             datos[1] = rs.getString("precioArticulo");
92             datos[2] = rs.getString("stockArticulo");
93         }
94         //Devolvemos los datos.
95         return datos;
96     } catch (SQLException e) {
97         //Si no se puede leer los datos, mostramos el diálogo de error.
98         new Notificacion("EditarArticulo", "errorLectura").setVisible(true);
99         e.printStackTrace();
100     }
101     return new String[] { };
102 }
103 //Método para actualizar artículos en la BBDD.
104 public static void modificarArticulo(String id, String descripcion, String precio, String stock) {
105     Connection conexion = getMySQL_Connection("tiendecitaanc");
106     String sentencia = "UPDATE tiendecitaanc.articulos SET descripcionArticulo = " + descripcion
107         + ", precioArticulo = " + precio + ", stockArticulo = " + stock
108         + " WHERE idArticulo = " + id + ";";
109     try {
110         Statement st = conexion.createStatement();
111         //Lanzamos la sentencia SQL de Update.
112         st.executeUpdate(sentencia);
113     } catch (SQLException e) {
114         //Si hay algún problema en la modificación, se muestra el diálogo de error.
115         new Notificacion("EditarArticulo", "error").setVisible(true);
116         e.printStackTrace();
117     }
118 }
119 //Método para rellenar la tabla de "Consulta Artículos".
120 public static void rellenarTablaArticulos(DefaultTableModel modelo) {
121     String idArticulo = "";
122     String descripcion = "";
123     String precio = "";
124     String stock = "";
125     Connection conexion = getMySQL_Connection("tiendecitaanc");
126     String sentencia = "SELECT * FROM tiendecitaanc.articulos";
127     try {
128         Statement st = conexion.createStatement();
129         ResultSet rs = st.executeQuery(sentencia);
130         while (rs.next()) {
131             idArticulo = rs.getString("idArticulo");
132             descripcion = rs.getString("descripcionArticulo");
133             precio = rs.getString("precioArticulo");
134             stock = rs.getString("stockArticulo");
135             //Añadimos los datos obtenidos a la tabla.
136             modelo.addRow(new Object[] { idArticulo, descripcion, precio, stock });
137         }
138     } catch (SQLException e) {
139         //Si no se pueden obtener los datos, mostramos el diálogo de error.
140         new Notificacion("Consultar", "error").setVisible(true);
141         e.printStackTrace();
142     }
143 }
144 }

```

```

145 //Método para obtener el id mediante el JComboBox.
146 public static String[] obtenerDatos(JComboBox<String> comboBox) {
147     String seleccion = (String) comboBox.getSelectedItem();
148     if (seleccion != null) {
149         //Separamos los resultados del JComboBox por " - ".
150         String[] partes = seleccion.split(" - ");
151         if (partes.length > 0) {
152             return partes;
153         }
154     }
155     return new String[] {};
156 }
157 //Método para Eliminar Artículos
158 public static void eliminarArticulo(String id) {
159     Connection conexion = getMySQL_Connection("tiendecitaanc");
160     String sentencia = "DELETE FROM tiendecitaanc.articulos WHERE idArticulo = " + id + ";";
161     try {
162         Statement st = conexion.createStatement();
163         //Lanzamos la sentencia SQL de DELETE mediante el id pasado como parámetro.
164         st.executeUpdate(sentencia);
165     } catch (SQLException e) {
166         //Si no se puede eliminar, mostramos el diálogo de error.
167         new Notificacion("BajaArticulo", "error").setVisible(true);
168         e.printStackTrace();
169     }
170 }
171 //Método para Insertar Tickets.
172 /*Esta función llamará al método insertarHistorico, ya que se deben realizar ambas altas a la vez.
173 A su vez, se llamará a la función actualizarStock para restar la cantidad añadida al ticket.*/
174 public static void insertarTicket(String fecha, String total, String articulos,
175     JComboBox<String> comboBox) {
176     String sentenciaId = "SELECT idTicket FROM tickets ORDER BY 1 DESC LIMIT 1";
177     String idTicket;
178     Connection conexion = getMySQL_Connection("tiendecitaanc");
179     String sentencia = "INSERT INTO tiendecitaanc.tickets VALUES (null, '" + fecha
180         + "', '" + total + "')";
181     try {
182         Statement st = conexion.createStatement();
183         if (st.executeUpdate(sentencia) > 0) {
184             //Si la sentencia se ejecuta correctamente, mostramos el diálogo de éxito.
185             new Notificacion("NuevoTicket", "creado").setVisible(true);
186             //Si la sentencia se ejecuta correctamente, consultamos el id del ticket recién creado.
187             ResultSet rs = st.executeQuery(sentenciaId);
188             if (rs.next()) {
189                 idTicket = rs.getString("idTicket");
190                 //Por cada articulo en el txtArticulos, separados por salto de línea.
191                 for (String articulo : articulos.split("\n")) {
192                     //Almacenamos el id y la cantidad separándolas por ", ".
193                     String[] partes = articulo.split(", ");
194                     //Obtenemos el id del artículo mediante obtenerIdArticulo();
195                     String idArticulo = obtenerIdArticulo(partes[0]);
196                     //Realizamos el insert en la tabla "historico".
197                     insertarHistorico(idTicket, idArticulo, partes[1]);
198                     //Actualizamos el stock.
199                     actualizarStock(partes[1], idArticulo);
200                 }
201             }
202         }
203     } catch (SQLException e) {
204         //Si no se puede crear el ticket, mostramos el diálogo de error.
205         new Notificacion("NuevoTicket", "error").setVisible(true);
206         e.printStackTrace();
207     }
208 }

```



```

209 //Método para insertar datos en la tabla "historico".
210● public static void insertarHistorico(String idTicket, String idArticulo, String cantidad) {
211     Connection conexion = getMySQL_Connection("tiendecitaanc");
212     String sentencia = "INSERT INTO tiendecitaanc.historico VALUES (null, "
213         + idTicket + "," + idArticulo + "," + cantidad + "));";
214     try {
215         Statement st = conexion.createStatement();
216         //Lanzamos la sentencia SQL.
217         st.executeUpdate(sentencia);
218     } catch (SQLException e) {
219         e.printStackTrace();
220     }
221 }
222 //Método para obtener el id del artículo en base a su descripción.
223● public static String obtenerIdArticulo(String descripcion) {
224     String id = "";
225     Connection conexion = getMySQL_Connection("tiendecitaanc");
226     String sentencia = "SELECT idArticulo FROM tiendecitaanc.articulos WHERE descripcionArticulo = '"
227         + descripcion + "'";
228     try {
229         Statement st = conexion.createStatement();
230         //Lanzamos la sentencia SQL y obtenemos el campo "idArticulo".
231         ResultSet rs = st.executeQuery(sentencia);
232         while (rs.next()) {
233             id = rs.getString("idArticulo");
234         }
235         return id;
236     } catch (SQLException e) {
237         e.printStackTrace();
238         return "-1";
239     }
240 }
241 //Método para comprobar si la cantidad introducida en el ticket es inferior al stock.
242● public static boolean comprobarStock(String descripcion, String cantidad) {
243     String stock = "";
244     Connection conexion = getMySQL_Connection("tiendecitaanc");
245     String sentencia = "SELECT stockArticulo FROM tiendecitaanc.articulos WHERE descripcionArticulo = '"
246         + descripcion + "'";
247     try {
248         Statement st = conexion.createStatement();
249         //Lanzamos la sentencia SQL almacenando el campo "stockArticulo".
250         ResultSet rs = st.executeQuery(sentencia);
251         if (rs.next()) {
252             stock = rs.getString("stockArticulo");
253         }
254         //Realizamos la comprobación.
255         if (Integer.valueOf(cantidad) <= Integer.valueOf(stock)) {
256             return true;
257         }
258         return false;
259     } catch (SQLException e) {
260         e.printStackTrace();
261         return false;
262     }
263 }
264 //Método para actualizar el stock después de insertar un ticket.
265● public static void actualizarStock(String unidades, String idArticulo) {
266     Connection conexion = getMySQL_Connection("tiendecitaanc");
267     String sentenciaUpdate = "UPDATE tiendecitaanc.articulos SET stockArticulo = stockArticulo - " + unidades
268         + " WHERE idArticulo = '" + idArticulo + "'";
269     try {
270         Statement st = conexion.createStatement();
271         st.executeUpdate(sentenciaUpdate);
272     } catch (SQLException e) {
273         e.printStackTrace();
274     }
275 }

```

```

276 //Método para rellenar la tabla "ConsultaTicket".
277 public static void rellenarTablaTickets(DefaultTableModel modelo) {
278     Connection conexion = getMySQL_Connection("tiendecitaanc");
279     //Como debemos mostrar campos ubicados en varias tablas, realizamos una sentencia con JOIN.
280     String sentencia = "SELECT t.fechaTicket, a.descripcionArticulo, h.cantidadArticulo, t.precioTotal "
281         + "FROM tiendecitaanc.tickets t " + "JOIN tiendecitaanc.historico h ON "
282         + "t.idTicket = h.idTicketFK "
283         + "JOIN tiendecitaanc.articulos a ON h.idArticuloFK = a.idArticulo "
284         + "ORDER BY t.fechaTicket";
285
286     try (Statement st = conexion.createStatement();
287         ResultSet rs = st.executeQuery(sentencia)) {
288
289         // Lista para almacenar temporalmente los datos
290         List<Object[]> filas = new ArrayList<>();
291
292         String fechaActual = "";
293         StringBuilder articulosActuales = new StringBuilder();
294         String precioTotal = "";
295         //Almacenamos los datos obtenidos de las tablas.
296         while (rs.next()) {
297             String fechaTicket = rs.getString("fechaTicket");
298             String descripcion = rs.getString("descripcionArticulo");
299             int cantidad = rs.getInt("cantidadArticulo");
300             String precio = rs.getString("precioTotal");
301
302             //Si cambia la fecha, guarda los datos acumulados y comienza un grupo nuevo
303             if (!fechaTicket.equals(fechaActual) && !fechaActual.isEmpty()) {
304                 filas.add(new Object[] { formatearFechaAEU(fechaActual),
305                     articulosActuales.toString(), precioTotal });
306                 //Limpia los artículos acumulados
307                 articulosActuales.setLength(0);
308             }
309
310             //Actualiza los valores actuales
311             fechaActual = fechaTicket;
312             precioTotal = precio;
313
314             //Añade el artículo actual con su cantidad
315             if (articulosActuales.length() > 0) {
316                 articulosActuales.append("\n");
317             }
318             articulosActuales.append(descripcion).append(" (").append(cantidad).append(")");
319         }
320
321         //Añade la última fila acumulada
322         if (!fechaActual.isEmpty()) {
323             filas.add(new Object[] { formatearFechaAEU(fechaActual), articulosActuales.toString(),
324                 precioTotal });
325         }
326
327         //Añade todas las filas al modelo de la tabla
328         for (Object[] fila : filas) {
329             modelo.addRow(fila);
330         }
331     } catch (SQLException e) {
332         //Si hay error en la consulta, mostramos el diálogo de error.
333         new Notificacion("Consultar", "error").setVisible(true);
334         e.printStackTrace();
335     }
336 }
337
338 //Método para formatear fecha desde SQL a Europeo.
339 public static String formatearFechaAEU(String fecha) {
340     String[] partes = fecha.split("-");
341     return partes[2] + "/" + partes[1] + "/" + partes[0];
342 }
343 }

```

3.- Clase Principal

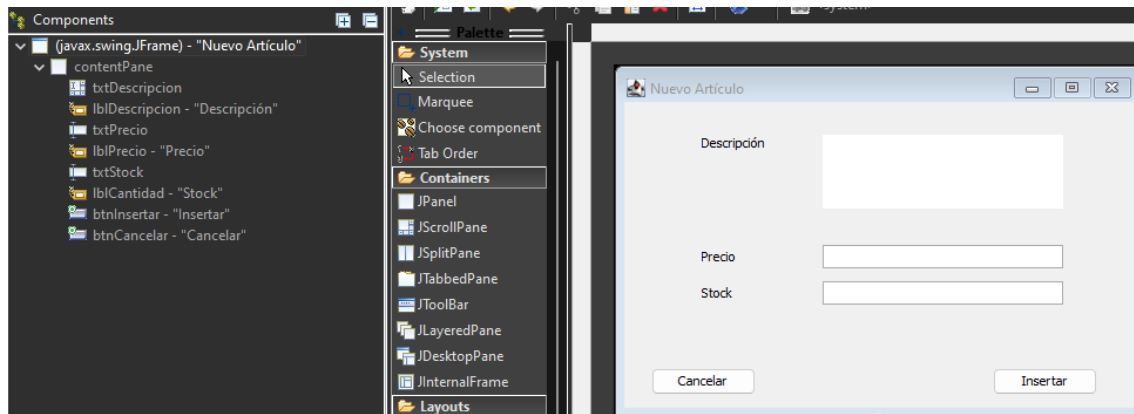
```

1 package es.studium.Practica4;
2
3 import java.awt.EventQueue;
4
14 public class Principal extends JFrame implements ActionListener {
15
16     private static final long serialVersionUID = 1L;
17     private JPanel contentPane;
18     JMenuBar menuBar = new JMenuBar();
19     JMenu mnArticulos = new JMenu("Articulos");
20     JMenuItem mniNuevoArticulo = new JMenuItem("Nuevo Articulo");
21     JMenuItem mniEditarArticulo = new JMenuItem("Editar Articulo");
22     JMenuItem mniConsultaArticulo = new JMenuItem("Consultar Articulos");
23     JMenuItem mniBajaArticulo = new JMenuItem("Baja Articulos");
24     JMenu mnTickets = new JMenu("Tickets");
25     JMenuItem mniNuevoTicket = new JMenuItem("Nuevo Ticket");
26     JMenuItem mniConsultaTicket = new JMenuItem("Consultar Tickets");
27
28     public static void main(String[] args) {
29         EventQueue.invokeLater(new Runnable() {
30             public void run() {
31                 try {
32                     Principal frame = new Principal();
33                     frame.setTitle("TiendecitaANC");
34                     frame.setVisible(true);
35                     frame.setLocationRelativeTo(null);
36                     frame.setResizable(false);
37                 } catch (Exception e) {
38                     e.printStackTrace();
39                 }
40             }
41         });
42     }
43
44     public Principal() {
45         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46         setBounds(100, 100, 450, 300);
47         contentPane = new JPanel();
48         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
49
50         setContentPane(contentPane);
51         contentPane.setLayout(null);
52
53         menuBar.setBounds(0, 0, 434, 22);
54         contentPane.add(menuBar);
55
56         menuBar.add(mnArticulos);
57         mnArticulos.add(mniNuevoArticulo);
58         mnArticulos.add(mniEditarArticulo);
59         mnArticulos.add(mniConsultaArticulo);
60         mnArticulos.add(mniBajaArticulo);
61         menuBar.add(mnTickets);
62         mnTickets.add(mniNuevoTicket);
63         mnTickets.add(mniConsultaTicket);
64
65         //Listeners
66         mniNuevoArticulo.addActionListener(this);
67         mniEditarArticulo.addActionListener(this);
68         mniConsultaArticulo.addActionListener(this);
69         mniBajaArticulo.addActionListener(this);
70         mniNuevoTicket.addActionListener(this);
71         mniConsultaTicket.addActionListener(this);
72     }
73
74     @Override
75     public void actionPerformed(ActionEvent e) {
76         if(e.getSource().equals(mniNuevoArticulo)) new NuevoArticulo().setVisible(true);
77
78         if(e.getSource().equals(mniEditarArticulo)) new EditarArticulo().setVisible(true);
79
80         if(e.getSource().equals(mniConsultaArticulo)) new ConsultarArticulo().setVisible(true);
81
82         if(e.getSource().equals(mniBajaArticulo)) new BajaArticulo().setVisible(true);
83
84         if(e.getSource().equals(mniNuevoTicket)) new NuevoTicket().setVisible(true);
85
86         if(e.getSource().equals(mniConsultaTicket)) new ConsultaTicket().setVisible(true);
87     }
88 }
89
90

```

4.- Clase NuevoArticulo

4.1.- WindowBuilder



4.2.- Código

```

1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
5
6
7
8 public class NuevoArticulo extends JFrame implements ActionListener {
9
10     private static final long serialVersionUID = 1L;
11     private JPanel contentPane;
12     private JTextArea txtDescripcion;
13     private JTextField txtPrecio;
14     private JTextField txtStock;
15     JButton btnInsertar = new JButton("Insertar");
16     JButton btnCancelar = new JButton("Cancelar");
17
18     public NuevoArticulo() {
19         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20         setBounds(100, 100, 450, 300);
21         setTitle("Nuevo Artículo");
22         setLocationRelativeTo(null);
23         setResizable(false);
24         contentPane = new JPanel();
25         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
26         setContentPane(contentPane);
27         contentPane.setLayout(null);
28
29         txtDescripcion = new JTextArea();
30         txtDescripcion.setBounds(169, 28, 205, 63);
31         contentPane.add(txtDescripcion);
32
33         JLabel lblDescripcion = new JLabel("Descripción");
34         lblDescripcion.setBounds(66, 28, 75, 14);
35         contentPane.add(lblDescripcion);
36

```

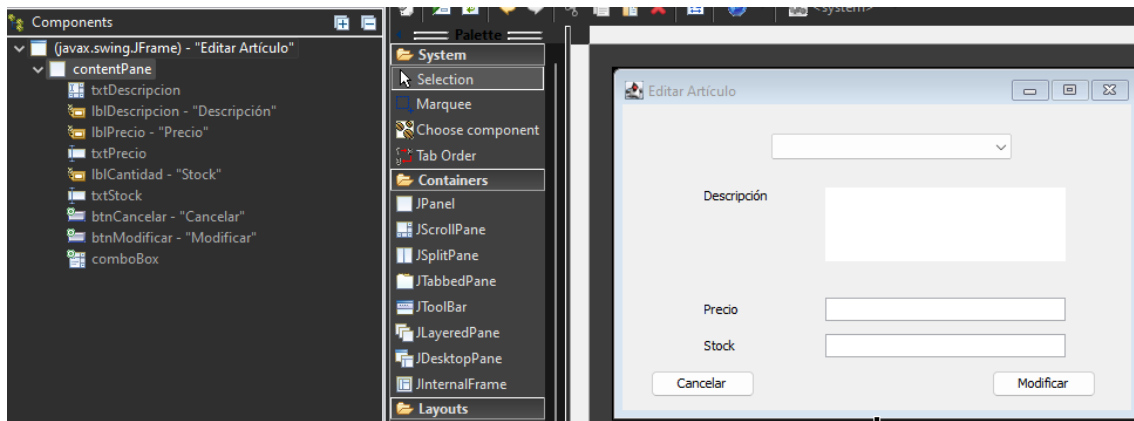
```

37     txtPrecio = new JTextField();
38     txtPrecio.setBounds(169, 122, 205, 20);
39     contentPane.add(txtPrecio);
40
41     JLabel lblPrecio = new JLabel("Precio");
42     lblPrecio.setBounds(66, 125, 46, 14);
43     contentPane.add(lblPrecio);
44
45     txtStock = new JTextField();
46     txtStock.setBounds(169, 153, 205, 20);
47     contentPane.add(txtStock);
48
49     JLabel lblCantidad = new JLabel("Stock");
50     lblCantidad.setBounds(66, 156, 75, 14);
51     contentPane.add(lblCantidad);
52
53     btnInsertar.setBounds(314, 227, 89, 23);
54     contentPane.add(btnInsertar);
55
56     btnCancelar.setBounds(23, 227, 89, 23);
57     contentPane.add(btnCancelar);
58
59     //Listeners
60     btnCancelar.addActionListener(this);
61     btnInsertar.addActionListener(this);
62 }
63
64 //Validaciones
65 public boolean hayError() {
66     // Verificar campos vacíos
67     if (txtDescripcion.getText().trim().isEmpty() || txtPrecio.getText().trim().isEmpty()
68         || txtStock.getText().trim().isEmpty()) {
69         new Notificacion("NuevoArticulo", "camposVacios").setVisible(true);
70         return true;
71     }
72     //Verificar si el precio es numérico
73     try {
74         Double.parseDouble(txtPrecio.getText().replace(",", "."));
75     } catch (NumberFormatException e) {
76         new Notificacion("NuevoArticulo", "precioNoNumerico").setVisible(true);
77         return true;
78     }
79     //Verificar si el stock es un número entero
80     try {
81         Integer.parseInt(txtStock.getText());
82     } catch (NumberFormatException e) {
83         new Notificacion("NuevoArticulo", "stockNoEntero").setVisible(true);
84         return true;
85     }
86     return false; //No hay errores
87 }
88
89 @Override
90 public void actionPerformed(ActionEvent e) {
91     if (e.getSource().equals(btnInsertar)) {
92         //Si todo está correcto, realizar la inserción
93         if (!hayError()) {
94             GestorConexiones.insertarArticulo(txtDescripcion.getText(),
95                 txtPrecio.getText().replace(",", "."), txtStock.getText());
96             dispose();
97         }
98     }
99     if (e.getSource().equals(btnCancelar)) {
100         dispose();
101     }
102 }
103 }

```

5.- Clase EditarArticulo

5.1.- WindowBuilder



5.2.- Código

```

1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
5
6
7
8 public class EditarArticulo extends JFrame implements ActionListener {
9
10     private static final long serialVersionUID = 1L;
11     private JPanel contentPane;
12     private JTextArea txtDescripcion;
13     private JTextField txtPrecio;
14     private JTextField txtStock;
15     JComboBox<String> comboBox = new JComboBox<>();
16     JButton btnCancelar = new JButton("Cancelar");
17     JButton btnModificar = new JButton("Modificar");
18
19     public EditarArticulo() {
20         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21         setBounds(100, 100, 450, 300);
22         setLocationRelativeTo(null);
23         setTitle("Editar Articulo");
24         setResizable(false);
25         contentPane = new JPanel();
26         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
27         setContentPane(contentPane);
28         contentPane.setLayout(null);
29
30         txtDescripcion = new JTextArea();
31         txtDescripcion.setBounds(173, 71, 205, 63);
32         contentPane.add(txtDescripcion);
33
34         JLabel lblDescripcion = new JLabel("Descripción");
35         lblDescripcion.setBounds(70, 71, 75, 14);
36         contentPane.add(lblDescripcion);
37
38         JLabel lblPrecio = new JLabel("Precio");

```

```

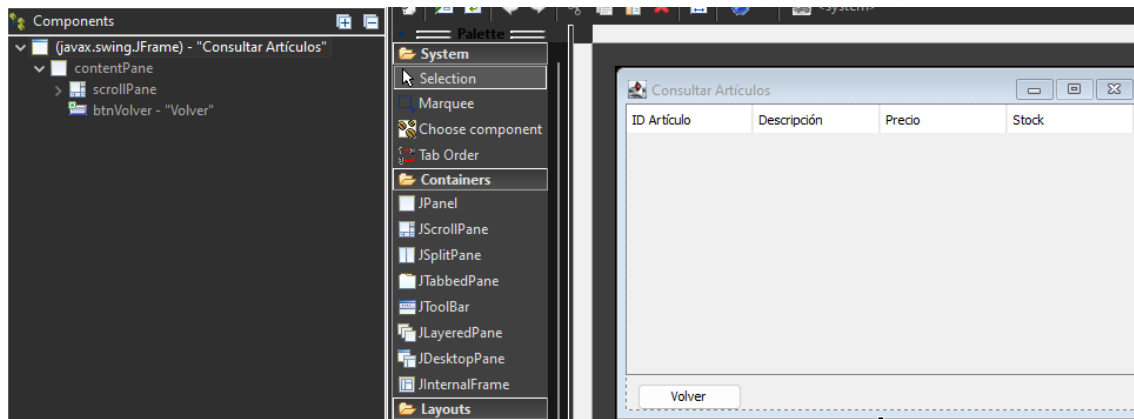
39     lblPrecio.setBounds(70, 168, 46, 14);
40     contentPane.add(lblPrecio);
41
42     txtPrecio = new JTextField();
43     txtPrecio.setBounds(173, 165, 205, 20);
44     contentPane.add(txtPrecio);
45
46     JLabel lblCantidad = new JLabel("Stock");
47     lblCantidad.setBounds(70, 199, 75, 14);
48     contentPane.add(lblCantidad);
49
50     txtStock = new JTextField();
51     txtStock.setBounds(173, 196, 205, 20);
52     contentPane.add(txtStock);
53
54     btnCancelar.setBounds(24, 227, 89, 23);
55     contentPane.add(btnCancelar);
56
57     btnModificar.setBounds(315, 227, 89, 23);
58     contentPane.add(btnModificar);
59
60     comboBox.setBounds(127, 25, 205, 22);
61     contentPane.add(comboBox);
62     //Llamamos al método para rellenar el Choice de Artículos
63     GestorConexiones.rellenarChoiceArticulos(comboBox);
64
65     // Listeners
66     comboBox.addActionListener(this);
67     btnModificar.addActionListener(this);
68     btnCancelar.addActionListener(this);
69 }
70 //Validaciones
71 public boolean hayError() {
72     //Verificar campos vacíos
73     if (txtDescripcion.getText().trim().isEmpty() || txtPrecio.getText().trim().isEmpty()
74         || txtStock.getText().trim().isEmpty()) {
75         new Notificacion("EditarArticulo", "camposVacios").setVisible(true);
76         return true;
77     }
78     //Verificar si el precio es numérico
79     try {
80         Double.parseDouble(txtPrecio.getText().replace(",", "."));
81     } catch (NumberFormatException e) {
82         new Notificacion("EditarArticulo", "precioNoNumerico").setVisible(true);
83         return true;
84     }
85     //Verificar si el stock es un número entero
86     try {
87         Integer.parseInt(txtStock.getText());
88     } catch (NumberFormatException e) {
89         new Notificacion("EditarArticulo", "stockNoEntero").setVisible(true);
90         return true;
91     }
92     return false; //No hay errores
93 }
94
95 @Override
96 public void actionPerformed(ActionEvent e) {
97     String[] datos = GestorConexiones.obtenerDatos(comboBox);
98     String id = datos[0];
99
100     if (e.getSource().equals(comboBox)) {
101         // Funcionalidad del JComboBox
102         if (comboBox.getSelectedIndex() != 0) {
103             String[] datosArticulo = GestorConexiones.rellenarArticulo(id);
104             txtDescripcion.setText(datosArticulo[0]);
105             txtPrecio.setText(datosArticulo[1]);
106             txtStock.setText(datosArticulo[2]);
107         } else {
108             txtDescripcion.setText("");
109             txtPrecio.setText("");
110             txtStock.setText("");
111         }
112     }

```

```
113         if (e.getSource().equals(btnModificar)) {
114             if (comboBox.getSelectedIndex() == 0) {
115                 new Notificacion("EditarArticulo", "noSeleccionado").setVisible(true);
116             } else if (!hayError()) {
117                 dispose();
118                 GestorConexiones.modificarArticulo(id, txtDescripcion.getText(),
119                     txtPrecio.getText().replace(",", "."),
120                     txtStock.getText());
121                 new Notificacion("EditarArticulo", "modificado").setVisible(true);
122             }
123         }
124         if (e.getSource().equals(btnCancelar)) {
125             dispose();
126         }
127     }
128 }
```


6.- Clase ConsultarArticulo

6.1.- WindowBuilder



6.2.- Código

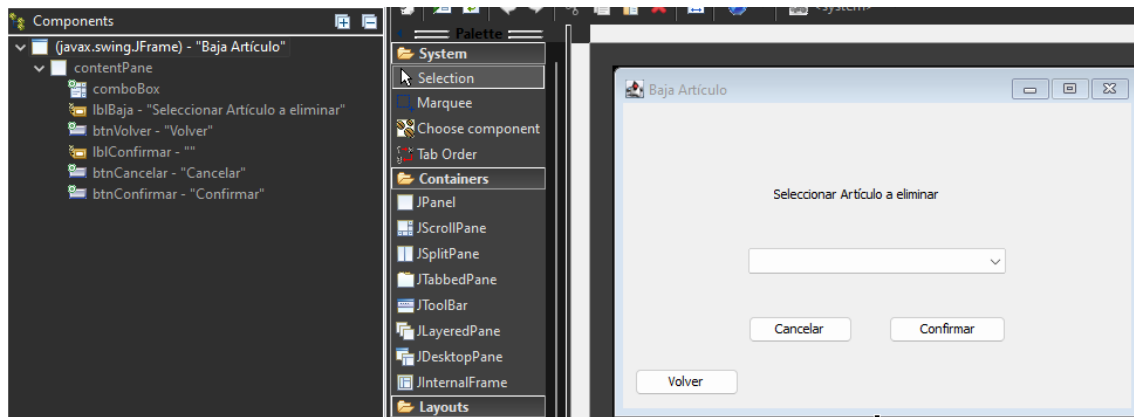
```

1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
5 public class ConsultarArticulo extends JFrame implements ActionListener {
6
7     private static final long serialVersionUID = 1L;
8     private JPanel contentPane;
9     private JTable tabla;
10    private JButton btnVolver;
11    private DefaultTableModel modelo;
12
13    public ConsultarArticulo() {
14        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
15        setBounds(100, 100, 450, 300);
16        setLocationRelativeTo(null);
17        setTitle("Consultar Artículos");
18        setResizable(true);
19        contentPane = new JPanel();
20
21        setContentPane(contentPane);
22        contentPane.setLayout(null);
23
24        modelo = new DefaultTableModel(null, obtenerNombresColumnas());
25        tabla = new JTable(modelo);
26        //Desactivar la selección de celdas y filas
27        tabla.setEnabled(false);
28
29        JScrollPane scrollPane = new JScrollPane(tabla);
30        scrollPane.setBounds(0, 0, 434, 231);
31        contentPane.add(scrollPane);
32        //Llamamos al método rellenarTablaArticulos();
33        GestorConexiones.rellenarTablaArticulos(modelo);
34
35        btnVolver = new JButton("Volver");
36        btnVolver.setBounds(10, 238, 89, 23);
37        contentPane.add(btnVolver);
38        btnVolver.addActionListener(this);
39    }
40
41    @Override
42    public void actionPerformed(ActionEvent e) {
43        if (e.getSource().equals(btnVolver))
44            dispose();
45    }
46    //Dar nombre a las columnas
47    private String[] obtenerNombresColumnas() {
48        return new String[] { "ID Artículo", "Descripción", "Precio", "Stock" };
49    }
50 }

```

7.- Clase BajaArticulo

7.1.- WindowBuilder



7.2.- Código

```

1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
13 public class BajaArticulo extends JFrame implements ActionListener{
14
15     private static final long serialVersionUID = 1L;
16     private JPanel contentPane;
17
18     JComboBox<String> comboBox = new JComboBox<String>();
19
20     JLabel lblConfirmar = new JLabel("");
21     JButton btnVolver = new JButton("Volver");
22     JButton btnCancelar = new JButton("Cancelar");
23     JButton btnConfirmar = new JButton("Confirmar");
24     String[] partes;
25
26     public void devolverLabel(String articulo) {
27         comboBox.setEnabled(false);
28         String mensaje="¿Desea eliminar " + articulo + "?";
29         lblConfirmar.setText(mensaje);
30         btnConfirmar.setVisible(true);
31         btnCancelar.setVisible(true);
32     }
33
34     public BajaArticulo() {
35         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
36         setBounds(100, 100, 450, 300);
37         setLocationRelativeTo(null);
38         setTitle("Baja Artículo");
39         setResizable(false);
40         contentPane = new JPanel();
41         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
42

```

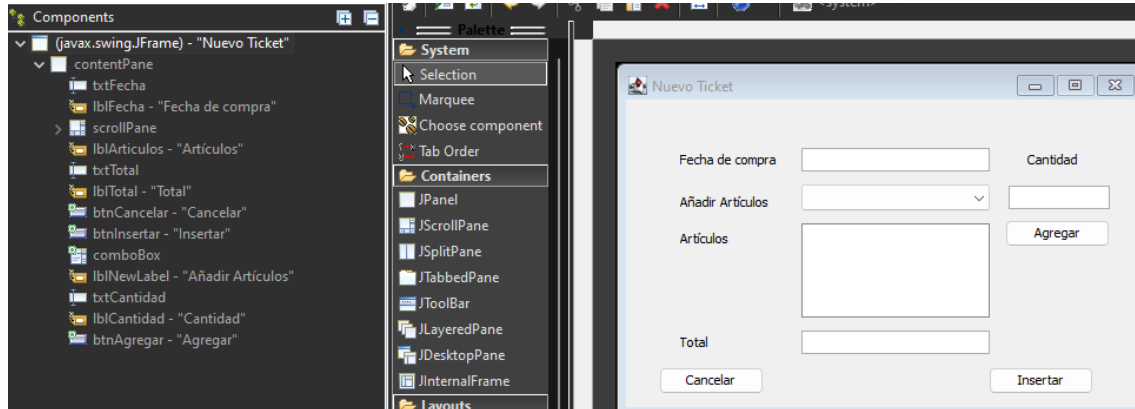
```

43     setContentPane(contentPane);
44     contentPane.setLayout(null);
45
46     comboBox.setBounds(107, 124, 220, 22);
47     //Rellenar el choice
48     GestorConexiones.rellenarChoiceArticulos(comboBox);
49     contentPane.add(comboBox);
50
51     JLabel lblBaja = new JLabel("Seleccionar Artículo a eliminar");
52     lblBaja.setBounds(129, 71, 209, 14);
53     contentPane.add(lblBaja);
54
55     btnVolver.setBounds(10, 227, 89, 23);
56     contentPane.add(btnVolver);
57
58     //Confirmación
59     lblConfirmar.setBounds(117, 157, 290, 14);
60     contentPane.add(lblConfirmar);
61
62     btnCancelar.setBounds(107, 182, 89, 23);
63     contentPane.add(btnCancelar);
64     btnCancelar.setVisible(false);
65
66     btnConfirmar.setBounds(227, 182, 100, 23);
67     contentPane.add(btnConfirmar);
68     btnConfirmar.setVisible(false);
69
70     //Listeners
71     comboBox.addActionListener(this);
72     btnVolver.addActionListener(this);
73     btnCancelar.addActionListener(this);
74     btnConfirmar.addActionListener(this);
75 }
76
77 @Override
78 public void actionPerformed(ActionEvent e) {
79     if(e.getSource().equals(comboBox) && comboBox.getSelectedIndex() != -1) {
80         if(comboBox.getSelectedIndex() == 0) {
81             lblConfirmar.setText("Debes seleccionar un Artículo");
82         }
83         else {
84             partes = GestorConexiones.obtenerDatos(comboBox);
85             devolverLabel(partes[1]);
86         }
87     }
88     if(e.getSource().equals(btnCancelar)) {
89         comboBox.setEnabled(true);
90         lblConfirmar.setText("");
91         btnConfirmar.setVisible(false);
92         btnCancelar.setVisible(false);
93     }
94     if(e.getSource().equals(btnConfirmar)) {
95         GestorConexiones.eliminarArticulo(partes[0]);
96         new Notificacion("BajaArticulo", "eliminado").setVisible(true);
97         GestorConexiones.rellenarChoiceArticulos(comboBox);
98         comboBox.setSelectedIndex(0);
99         lblConfirmar.setText("");
100         btnConfirmar.setVisible(false);
101         btnCancelar.setVisible(false);
102
103         comboBox.setEnabled(true);
104     }
105     if(e.getSource().equals(btnVolver)) dispose();
106 }
107 }

```

8.- Clase NuevoTicket

8.1.- WindowBuilder



8.2.- Código

```

1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
15
16 public class NuevoTicket extends JFrame implements ActionListener{
17
18     private static final long serialVersionUID = 1L;
19     private JPanel contentPane;
20     private JTextField txtFecha;
21     private JTextField txtTotal;
22     JLabel lblCantidad = new JLabel("Cantidad");
23     JComboBox<String> comboBox = new JComboBox<String>();
24     JButton btnCancelar = new JButton("Cancelar");
25     JButton btnInsertar = new JButton("Insertar");
26     JButton btnAgregar = new JButton("Agregar");
27     JTextArea txtArticulos = new JTextArea();
28     private JTextField txtCantidad;
29
30     //Validaciones
31     public boolean hayError() {
32         //Validar si hay campos vacíos
33         if (txtFecha.getText().isEmpty() || txtTotal.getText().isEmpty() || txtArticulos.getText().isEmpty()) {
34             new Notificacion("NuevoTicket", "camposVacios").setVisible(true);
35             return true;
36         }
37
38         //Validar que la fecha no contenga caracteres alfabéticos
39         if (!txtFecha.getText().matches("\\d{1,2}/\\d{1,2}/\\d{4}")) {
40             new Notificacion("NuevoTicket", "errorFecha").setVisible(true);
41             return true;
42         }
43     }

```

```

44 //Validar que el total sea un número válido con decimales
45 try {
46     Double.parseDouble(txtTotal.getText().replace(",", "."));
47 } catch (NumberFormatException e) {
48     new Notificacion("NuevoTicket", "totalNoNumerico").setVisible(true);
49     return true;
50 }
51 return false;
52 }
53
54 public NuevoTicket() {
55     setTitle("Nuevo Ticket");
56     setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
57     setBounds(100, 100, 450, 300);
58     setLocationRelativeTo(null);
59     setResizable(false);
60     contentPane = new JPanel();
61     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
62
63     setContentPane(contentPane);
64     contentPane.setLayout(null);
65
66     txtFecha = new JTextField();
67     txtFecha.setBounds(149, 41, 160, 20);
68     contentPane.add(txtFecha);
69     txtFecha.setColumns(10);
70
71     JLabel lblFecha = new JLabel("Fecha de compra");
72     lblFecha.setBounds(46, 44, 106, 14);
73     contentPane.add(lblFecha);
74
75     JScrollPane scrollPane = new JScrollPane();
76     scrollPane.setBounds(149, 105, 160, 80);
77     contentPane.add(scrollPane);
78     scrollPane.setViewportView(txtArticulos);
79
80     JLabel lblArticulos = new JLabel("Articulos");
81     lblArticulos.setBounds(46, 111, 106, 14);
82     contentPane.add(lblArticulos);
83
84     txtTotal = new JTextField();
85     txtTotal.setBounds(149, 196, 160, 20);
86     contentPane.add(txtTotal);
87     txtTotal.setColumns(10);
88
89     JLabel lblTotal = new JLabel("Total");
90     lblTotal.setBounds(46, 199, 46, 14);
91     contentPane.add(lblTotal);
92
93     btnCancelar.setBounds(28, 227, 89, 23);
94     contentPane.add(btnCancelar);
95
96     btnInsertar.setBounds(308, 227, 89, 23);
97     contentPane.add(btnInsertar);
98
99     comboBox.setBounds(149, 72, 160, 22);
100    contentPane.add(comboBox);
101    GestorConexiones.rellenarChoiceArticulos(comboBox);
102
103    JLabel lblNewLabel = new JLabel("Añadir Articulos");
104    lblNewLabel.setBounds(46, 80, 118, 14);
105    contentPane.add(lblNewLabel);
106
107    txtCantidad = new JTextField();
108    txtCantidad.setBounds(325, 73, 86, 20);
109    contentPane.add(txtCantidad);
110    txtCantidad.setColumns(10);
111    txtCantidad.setVisible(false);
112
113    lblCantidad.setBounds(341, 44, 70, 14);
114    contentPane.add(lblCantidad);
115    lblCantidad.setVisible(false);
116
117    btnAgregar.setBounds(322, 102, 89, 23);
118    contentPane.add(btnAgregar);
119    btnAgregar.setVisible(false);
120
121    //Listeners
122    btnCancelar.addActionListener(this);
123    btnInsertar.addActionListener(this);
124    comboBox.addActionListener(this);
125    btnAgregar.addActionListener(this);
126 }

```

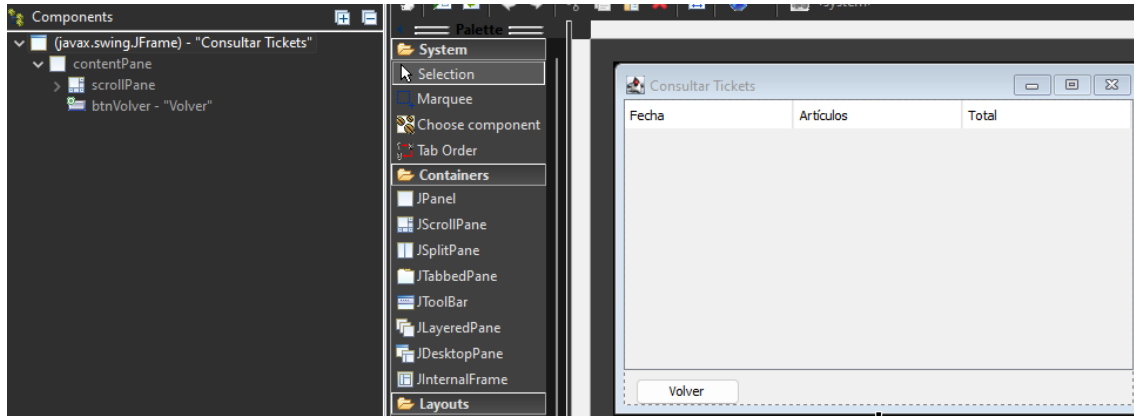
```

127
128 • @Override
129 public void actionPerformed(ActionEvent e) {
130     if (e.getSource().equals(comboBox)) {
131         int selectedIndex = comboBox.getSelectedIndex(); // Obtener el índice seleccionado
132         //Mostrar los campos solo si se selecciona un artículo cuyo índice sea diferente de 0
133         if (selectedIndex != 0) {
134             txtCantidad.setVisible(true);
135             lblCantidad.setVisible(true);
136             btnAgregar.setVisible(true);
137         } else {
138             //Ocultar si el índice es 0 ("Seleccionar Artículo")
139             txtCantidad.setVisible(false);
140             lblCantidad.setVisible(false);
141             btnAgregar.setVisible(false);
142         }
143     }
144     //Acción al presionar el botón Agregar
145     if (e.getSource().equals(btnAgregar)) {
146         String articulo = comboBox.getSelectedItem().toString().split(" - ")[1];
147         String cantidad = txtCantidad.getText();
148
149         if (!articulo.equals("Seleccionar Artículo") && !cantidad.isEmpty()) {
150             // Añadir el artículo y cantidad al área de texto con un salto de línea
151             if (GestorConexiones.comprobarStock(articulo, cantidad) == true) {
152                 String textoActual = txtArticulos.getText();
153                 txtArticulos.setText(textoActual + articulo + ", " + cantidad + "\n");
154                 // Limpiar el campo de cantidad para una nueva entrada
155                 txtCantidad.setText("");
156             }
157             else new Notificacion("NuevoTicket", "errorStock").setVisible(true);
158         }
159         lblCantidad.setVisible(false);
160         txtCantidad.setVisible(false);
161         btnAgregar.setVisible(false);
162     }
163     // Manejo de botón Insertar y Cancelar
164     if (e.getSource().equals(btnInsertar)) {
165         if (!hayError()) {
166             GestorConexiones.insertarTicket(formatearFechaASQL(txtFecha.getText()),
167                 txtTotal.getText(), txtArticulos.getText(), comboBox);
168             dispose();
169         }
170     }
171     if (e.getSource().equals(btnCancelar)) {
172         dispose();
173     }
174 }
175 • private static String formatearFechaASQL(String fecha) {
176     String[] partes = fecha.split("/");
177     return partes[2] + "-" + partes[1] + "-" + partes[0];
178 }
179 }

```

9.- Clase ConsultaTicket

9.1.- WindowBuilder



9.2.- Código

```

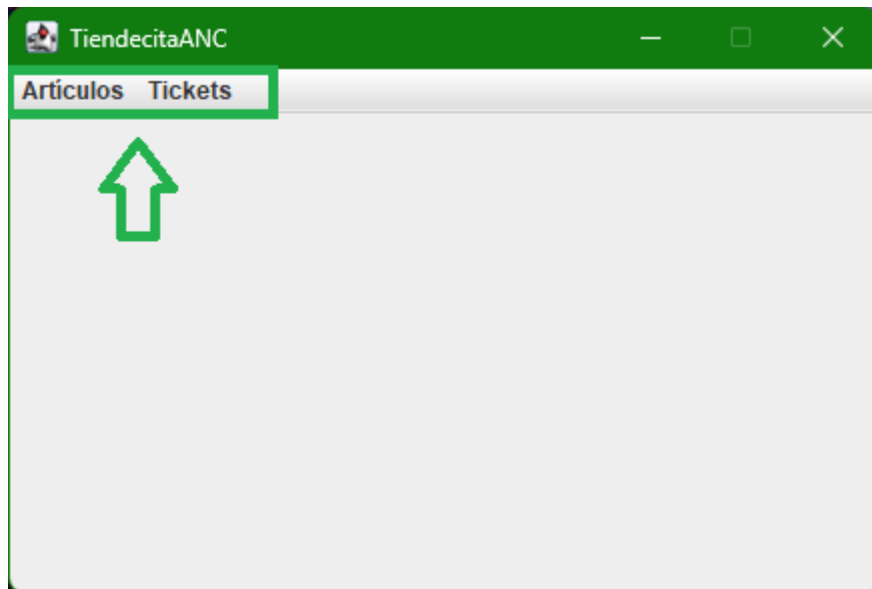
1 package es.studium.Practica4;
2
3 import java.awt.event.ActionEvent;
4
15 public class ConsultaTicket extends JFrame implements ActionListener {
16
17     private static final long serialVersionUID = 1L;
18     private JPanel contentPane;
19     private JTable tabla;
20     private JButton btnVolver;
21     private DefaultTableModel modelo;
22
23     public ConsultaTicket() {
24         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
25         setBounds(100, 100, 450, 300);
26         setLocationRelativeTo(null);
27         setTitle("Consultar Tickets");
28         setResizable(true);
29         contentPane = new JPanel();
30
31         setContentPane(contentPane);
32         contentPane.setLayout(null);
33
34         modelo = new DefaultTableModel(null, obtenerNombresColumnas());
35         tabla = new JTable(modelo);
36
37         tabla.setEnabled(false);
38
39         //Establecer el renderizador para la columna Articulos
40         tabla.getColumnModel().getColumn(1).setCellRenderer(new DefaultTableCellRenderer() {
41             private static final long serialVersionUID = 1L;
42             @Override
43             public java.awt.Component getTableCellRendererComponent(JTable table, Object value,
44                 boolean isSelected, boolean hasFocus, int row, int column) {

```

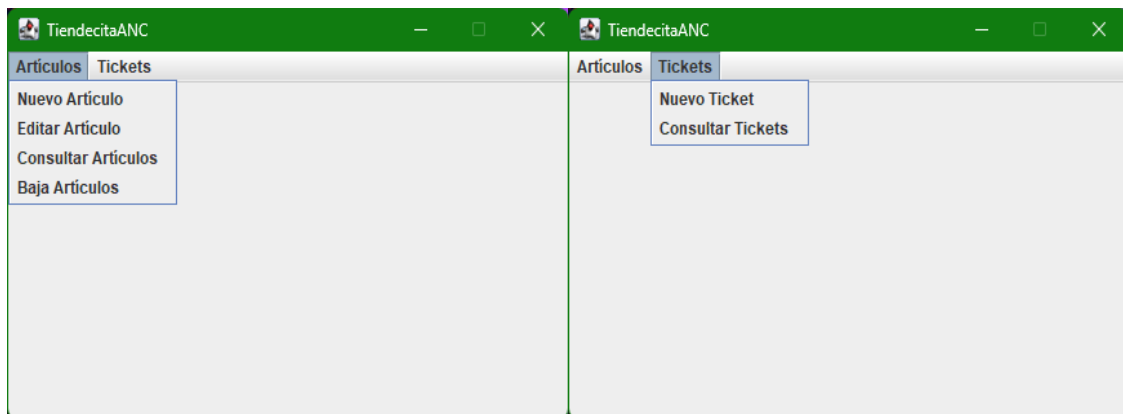
```
45         JTextArea textArea = new JTextArea(value.toString());
46         int numberOfLines = textArea.getLineCount();
47         // Ajustar la altura de la fila según el número de líneas
48         table.setRowHeight(row, numberOfLines * 20);
49         return textArea;
50     }
51     });
52
53     //Ajustar automáticamente el tamaño de las columnas
54     tabla.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
55
56     JScrollPane scrollPane = new JScrollPane(tabla);
57     scrollPane.setBounds(0, 0, 434, 231);
58     contentPane.add(scrollPane);
59
60     //Rellenar la tabla con los datos de los tickets
61     GestorConexiones.rellenarTablaTickets(modelo);
62
63     btnVolver = new JButton("Volver");
64     btnVolver.setBounds(10, 238, 89, 23);
65     contentPane.add(btnVolver);
66     btnVolver.addActionListener(this);
67 }
68
69 @Override
70 public void actionPerformed(ActionEvent e) {
71     if (e.getSource().equals(btnVolver)) {
72         dispose();
73     }
74 }
75
76 //Dar nombre a las columnas
77 private String[] obtenerNombresColumnas() {
78     return new String[] { "Fecha", "Artículos", "Total" };
79 }
```


10.- Ejecución (↑)

10.1.- Menú Principal



La vista de nuestro menú principal constará de una barra de menús, que incluirá “Artículos” y “Tickets”.



Al hacer clic en uno de los dos menús, se desplegarán las posibles opciones de las que dispondrá cada uno.

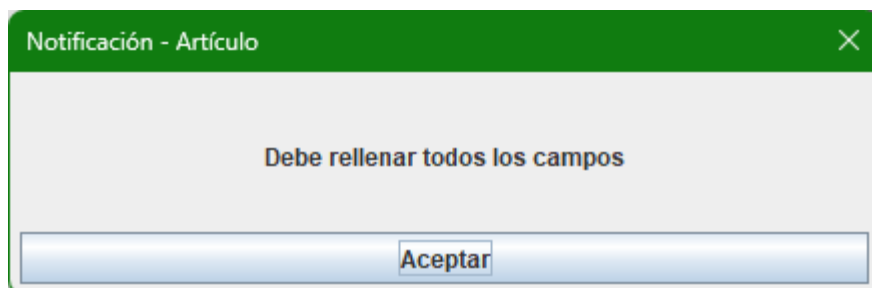
10.2.- Nuevo Artículo



The image shows a Windows-style dialog box titled "Nuevo Artículo". It has a green title bar with a close button (X) on the right. The main area is light gray and contains three labels on the left: "Descripción", "Precio", and "Stock". To the right of "Descripción" is a large white text area. To the right of "Precio" and "Stock" are two single-line white text input fields. At the bottom left is a blue button labeled "Cancelar", and at the bottom right is a blue button labeled "Insertar".

En la ventana de “Nuevo Artículo”, encontraremos 3 áreas de texto, a saber, “Descripción, Precio y Stock”.

Si pulsamos “Insertar” con algún campo vacío, se nos mostrará un diálogo con el siguiente mensaje de error:



The image shows a smaller Windows-style dialog box titled "Notificación - Artículo". It has a green title bar with a close button (X) on the right. The main area is light gray and contains the text "Debe rellenar todos los campos" in the center. At the bottom is a blue button labeled "Aceptar".

Two screenshots of the 'Nuevo Artículo' dialog box illustrating validation errors:

- Left Screenshot:** The 'Precio' field contains the value 'a'. A green border highlights the field. Below the dialog, a notification box displays the message: 'El precio debe ser numérico'.
- Right Screenshot:** The 'Stock' field contains the value 'a'. A green border highlights the field. Below the dialog, a notification box displays the message: 'El Stock debe ser un número entero'.

Los campos “Precio” y “Stock”, solo aceptarán valores numéricos, siendo un valor entero para “Stock” y posibilidad de decimales para “Precio”.

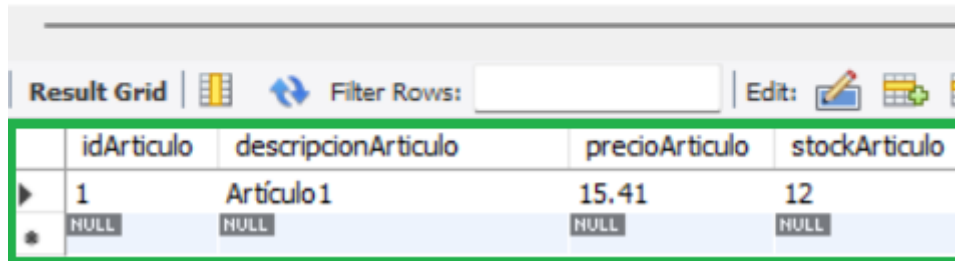
Si no introducimos un valor correcto en alguno de estos campos, nos aparecerán los errores mostrados en la captura anterior.

Two screenshots of the 'Nuevo Artículo' dialog box showing successful data entry:

- Top Screenshot:** The 'Precio' field contains the value 15.41 and the 'Stock' field contains the value 12. Both fields are highlighted with green borders.
- Bottom Screenshot:** A notification box displays the message: 'Artículo creado con éxito'.

Si los valores son correctos, nos aparecerá el diálogo de éxito.

```
1 • SELECT * FROM tiendecitaanc.articulos;  
2
```



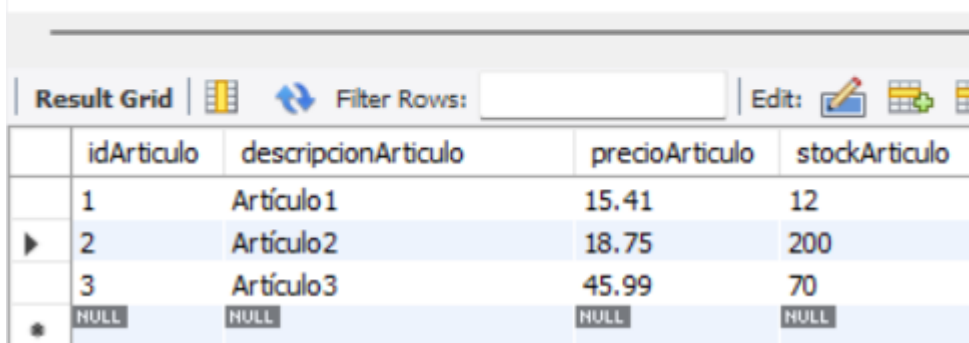
The screenshot shows a database query result grid. The grid has a toolbar at the top with 'Result Grid', 'Filter Rows', and 'Edit' buttons. The grid contains one row of data with the following values:

	idArticulo	descripcionArticulo	precioArticulo	stockArticulo
▶	1	Artículo1	15.41	12
★	NULL	NULL	NULL	NULL

Como podemos apreciar, al realizar un SELECT en la BBDD tiendecitaanc.articulos, aparecerá el recién creado.

Vamos a realizar 2 altas más para poder trabajar con estos artículos posteriormente, quedando la BBDD de la siguiente manera:

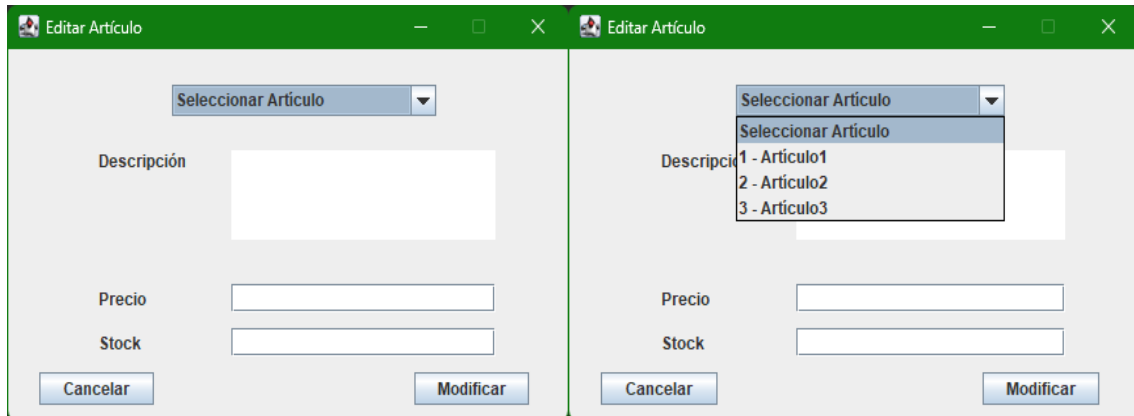
```
1 • SELECT * FROM tiendecitaanc.articulos;  
2
```



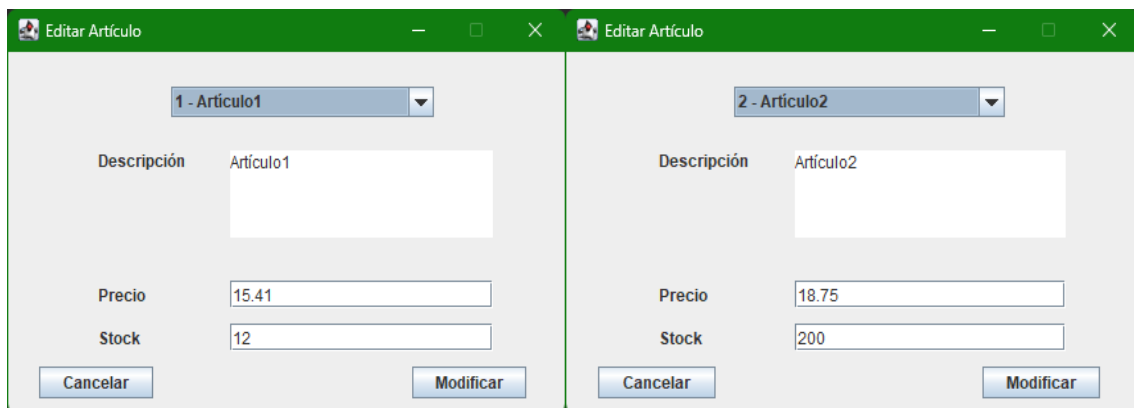
The screenshot shows a database query result grid. The grid has a toolbar at the top with 'Result Grid', 'Filter Rows', and 'Edit' buttons. The grid contains three rows of data with the following values:

	idArticulo	descripcionArticulo	precioArticulo	stockArticulo
	1	Artículo1	15.41	12
▶	2	Artículo2	18.75	200
	3	Artículo3	45.99	70
★	NULL	NULL	NULL	NULL

10.3.- Editar Artículo



En la ventana de “Editar Artículo”, encontraremos un desplegable que contendrá los artículos insertados en la base de datos. También encontraremos los mismos campos que en “Nuevo Artículo”, salvo que esta vez, se rellenarán automáticamente con los valores asignados a los mismos una vez creados al seleccionar un artículo del desplegable.



Si dejamos algún campo vacío, al pulsar “Modificar”, se mostrará un diálogo de error indicando el problema:

Editar Artículo

1 - Artículo1

Descripción: ArtículoPrueba

Precio:

Stock:

Cancelar Modificar

Notificación - Artículo

Debe rellenar todos los campos

Aceptar

Editar Artículo

1 - Artículo1

Descripción: ArtículoPrueba

Precio:

Stock:

Cancelar Modificar

Notificación - Artículo

El precio debe ser numérico

Aceptar

The screenshot shows a dialog box titled "Editar Artículo". At the top, there is a dropdown menu set to "1 - Artículo1". Below it, the "Descripción" field contains the text "ArtículoPrueba". The "Precio" field contains "10.99". The "Stock" field contains "200.1" and is highlighted with a green border. At the bottom of the dialog are two buttons: "Cancelar" and "Modificar". Below the main dialog, a green notification bar with a close button (X) contains the text "Notificación - Artículo". Below this bar, a message box states "El Stock debe ser un número entero". At the bottom of the message box is an "Aceptar" button.

Si, por el contrario, todos los campos están correctamente cumplimentados, al pulsar "Modificar", se mostrará un diálogo de éxito para indicar que la modificación se ha realizado.

The screenshot shows the same "Editar Artículo" dialog box, but now the "Stock" field contains the integer value "100". The "Modificar" button is highlighted. Below the dialog, the green notification bar now displays the message "Artículo modificado con éxito". The "Aceptar" button is visible at the bottom of the notification area.

```
1 • SELECT * FROM tiendecitaanc.articulos;
2
```

	idArticulo	descripcionArticulo	precioArticulo	stockArticulo
▶	1	ArtículoPrueba	10.99	100
	2	Artículo2	18.75	200
	3	Artículo3	45.99	70
★	NULL	NULL	NULL	NULL

Si volvemos a realizar una consulta en la tabla artículos desde la base de datos, podemos ver como el Artículo1 se ha modificado correctamente.

10.4.- Consultar Artículo

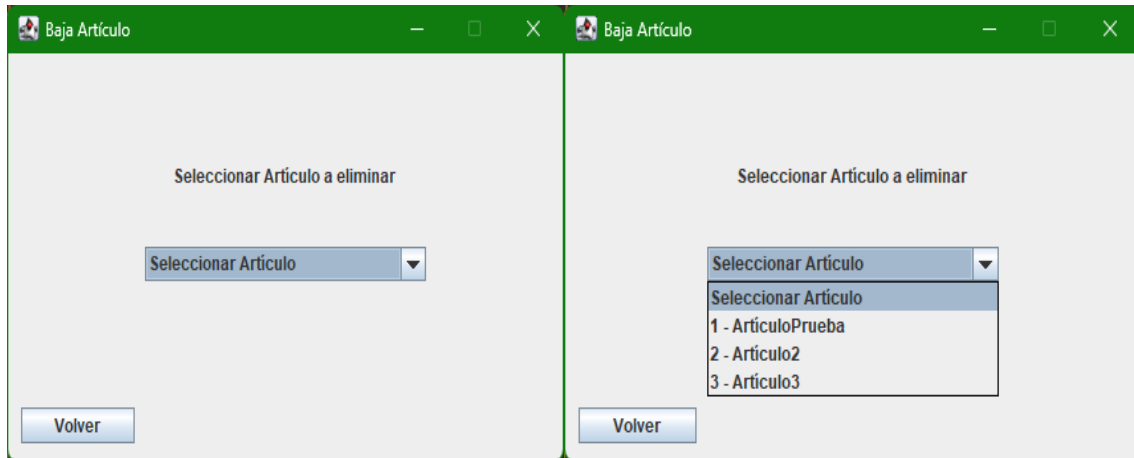
ID Artículo	Descripción	Precio	Stock
1	ArtículoPrueba	10.99	100
2	Artículo2	18.75	200
3	Artículo3	45.99	70

Volver

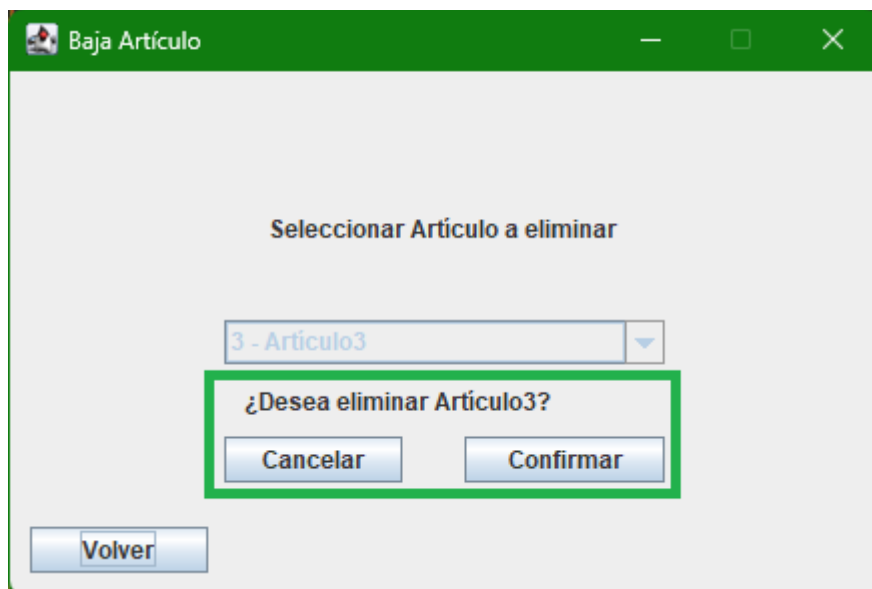
En la ventana de “Consultar Artículos”, encontraremos una tabla con todos los artículos creados en nuestra base de datos, mostrando su ID, Descripción, Precio y Stock.

Como debe ser, concuerda con la consulta realizada anteriormente.

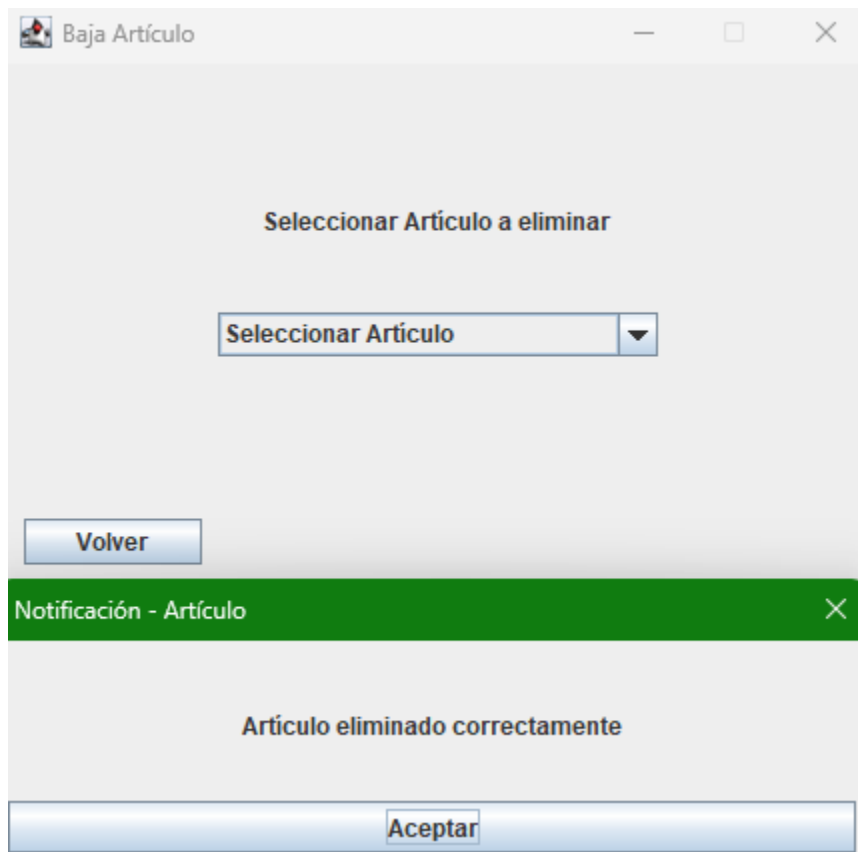
10.5.- Baja Artículo



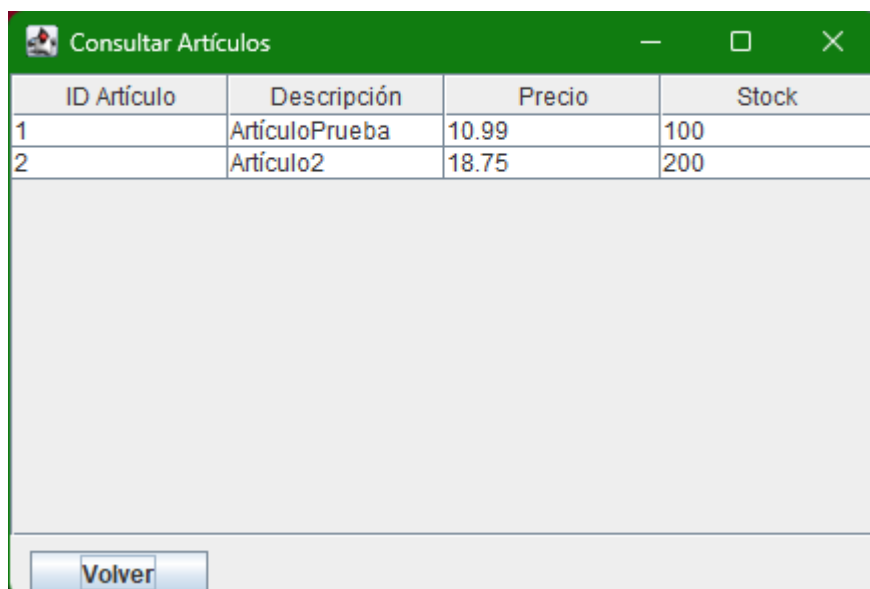
En la ventana de Baja, encontraremos un desplegable que contendrá los artículos insertados en nuestra base de datos.



Al seleccionar un artículo, el desplegable dejará de estar habilitado hasta que seleccionemos “Cancelar” o “Confirmar”, respondiendo a la confirmación.

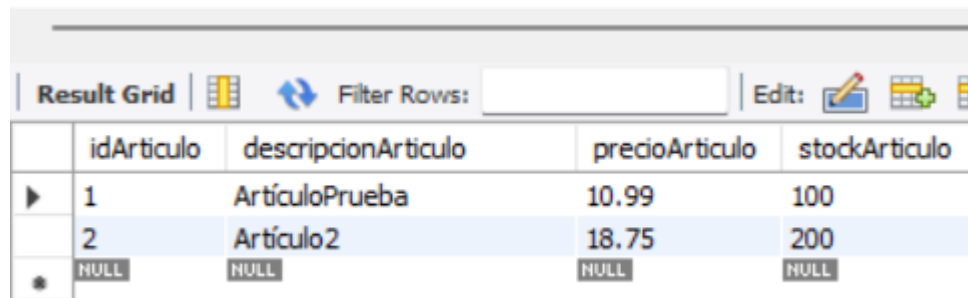


Si pulsamos “Confirmar”, se mostrará un diálogo de éxito y el desplegable volverá a estar operativo por si deseamos continuar eliminando artículos.



Si volvemos a la ventana de Consultar Artículos, vemos que ha desaparecido el artículo recién eliminado.

```
1 • SELECT * FROM tiendecitaanc.articulos;  
2
```

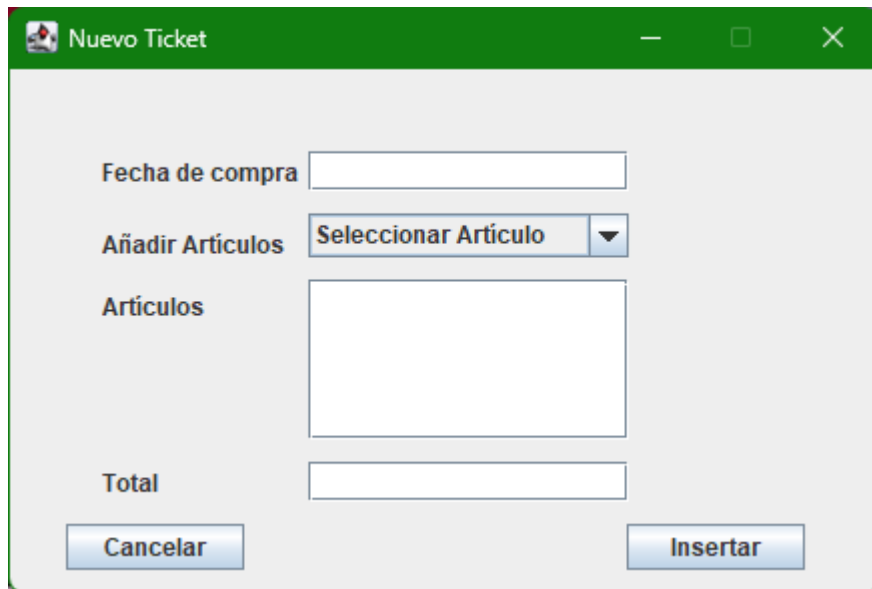


The screenshot shows a 'Result Grid' interface with a toolbar containing icons for grid view, refresh, filter, and edit. Below the toolbar is a table with four columns: 'idArticulo', 'descripcionArticulo', 'precioArticulo', and 'stockArticulo'. The table contains three rows: a header row, a row with values 1, ArtículoPrueba, 10.99, and 100; a row with values 2, Artículo2, 18.75, and 200; and a row with NULL values for all columns. A small asterisk icon is visible in the first column of the NULL row.

	idArticulo	descripcionArticulo	precioArticulo	stockArticulo
▶	1	ArtículoPrueba	10.99	100
	2	Artículo2	18.75	200
*	NULL	NULL	NULL	NULL

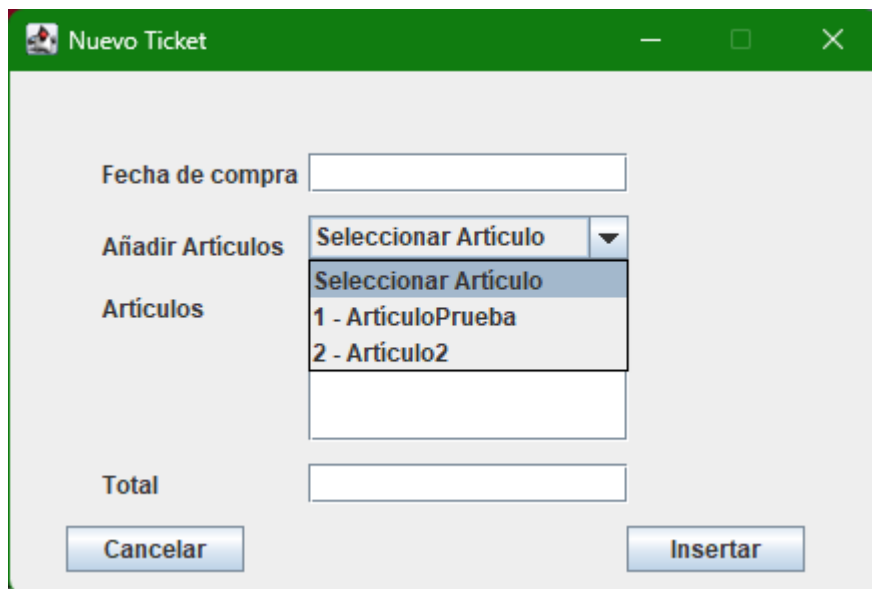
Cambio que también aparece al realizar la consulta SELECT en la base de datos.

10.6.- Nuevo Ticket



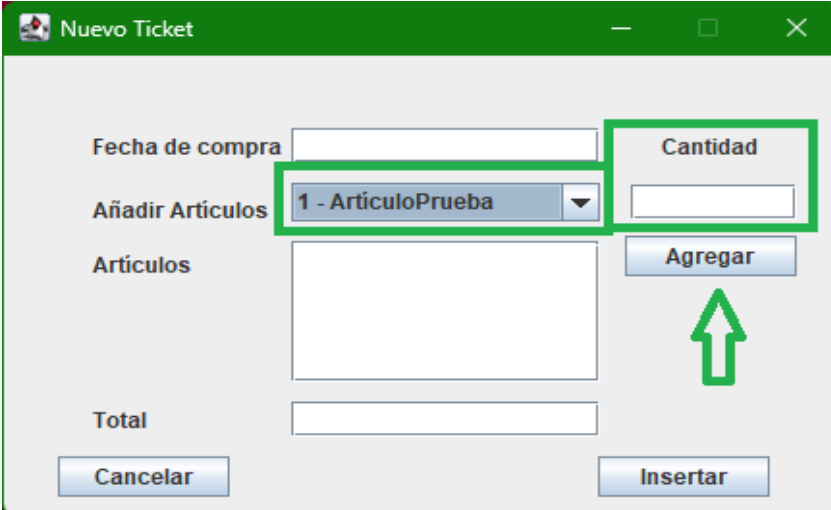
The screenshot shows a window titled "Nuevo Ticket" with a green header bar. Inside, there are four input fields: "Fecha de compra", "Añadir Artículos" (with a dropdown menu showing "Seleccionar Artículo"), "Artículos" (a large empty text area), and "Total". At the bottom, there are two buttons: "Cancelar" and "Insertar".

En la ventana de “Nuevo Ticket”, encontraremos un campo para la fecha, un desplegable para seleccionar artículos que se añadirán al campo “Artículos”, y el campo total para indicar el precio.



This screenshot shows the same "Nuevo Ticket" window, but the dropdown menu for "Añadir Artículos" is now open. It displays two items: "1 - ArtículoPrueba" and "2 - Artículo2". The "Artículos" text area remains empty.

Al eliminar el Artículo3, ahora tenemos 2 artículos con los que trabajar, y así se muestra en el desplegable.



Nuevo Ticket

Fecha de compra

Añadir Artículos **1 - ArtículoPrueba**

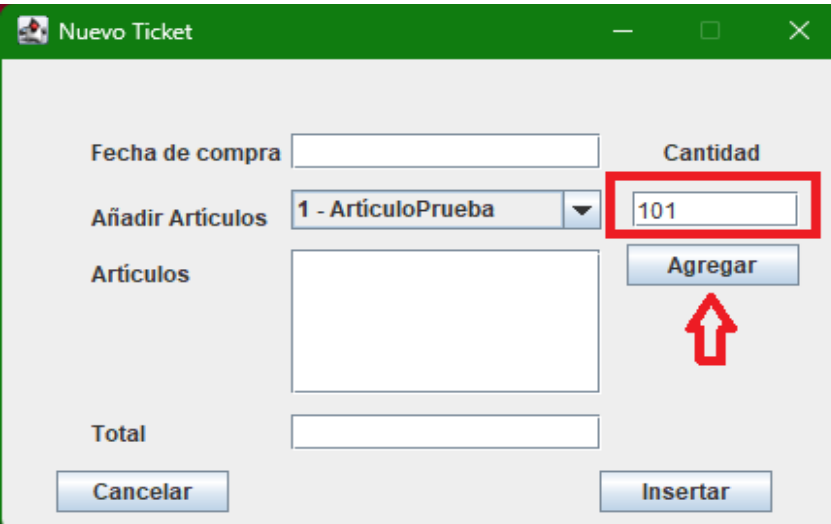
Artículos

Total

Cancelar Insertar

Agregar

Al seleccionar un artículo, aparecerá un campo adicional en el que debemos indicar la cantidad de artículos que se ha comprado.



Nuevo Ticket

Fecha de compra

Añadir Artículos **1 - ArtículoPrueba**


Artículos

Total

Cancelar Insertar

Agregar

Si la cantidad introducida es mayor al stock almacenado en la base de datos, nos saldrá un mensaje de error al pulsar en “Agregar”:



Nuevo Ticket

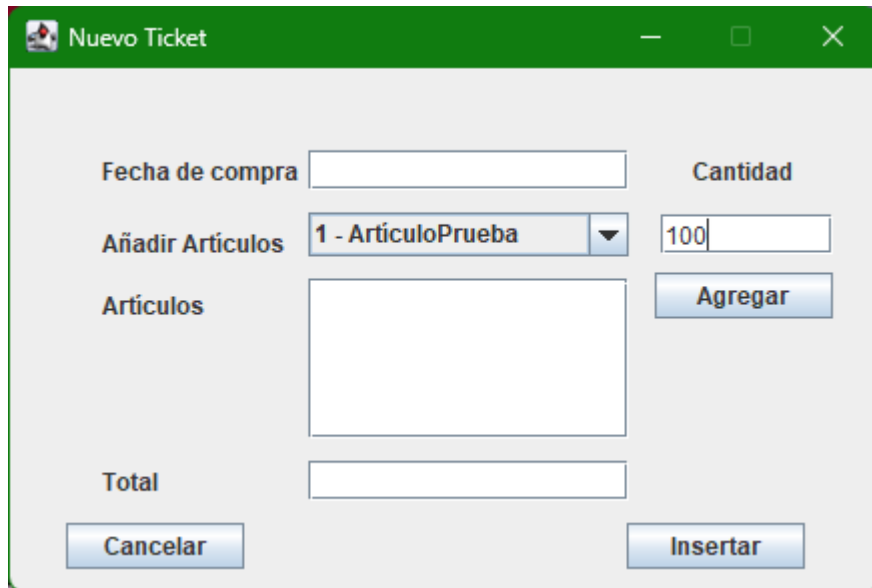
Notificación - Ticket

No hay stock suficiente

Aceptar

Total

Cancelar Insertar



Nuevo Ticket

Fecha de compra

Añadir Artículos 1 - ArtículoPrueba ▼

Artículos

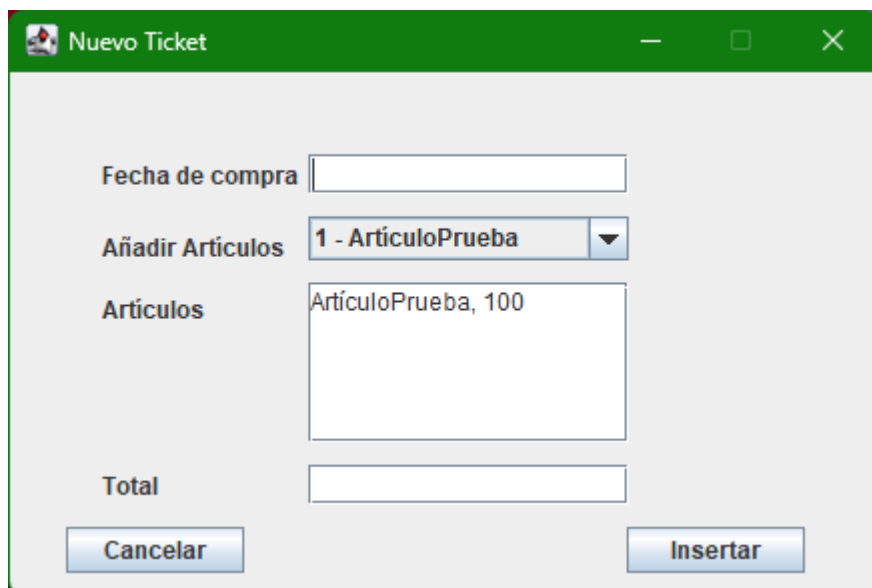
Cantidad 100

Agregar

Total

Cancelar Insertar

El stock almacenado en la base de datos para “ArtículoPrueba” es de 100 unidades, por lo que, si introducimos esta cantidad o una inferior, debe agregar el artículo al campo que le corresponde.



Nuevo Ticket

Fecha de compra

Añadir Artículos 1 - ArtículoPrueba ▼

Artículos ArtículoPrueba, 100

Total

Cancelar Insertar

Al pulsar “Agregar” con un stock correcto, el campo “Cantidad” desaparecerá junto con el botón “Agregar” y se añadirá el producto al campo “Artículos”, separado por una coma de su cantidad.

Al igual que en las ventanas anteriores, debemos cumplimentar correctamente todos los campos, ya que, de no hacerlo, se nos mostrarán los siguientes mensajes de error:

The image displays two screenshots of a 'Nuevo Ticket' (New Ticket) window, illustrating validation errors.

Top Screenshot: The window shows the 'Fecha de compra' (Purchase Date) field highlighted with a green box. Below it, the 'Añadir Artículos' (Add Items) dropdown is set to '2 - Artículo2'. The 'Artículos' (Items) list shows 'ArtículoPrueba, 100' and 'Artículo2, 80'. The 'Total' field is also highlighted with a green box. The 'Cancelar' (Cancel) and 'Insertar' (Insert) buttons are visible. A green notification bar at the bottom reads 'Notificación - Ticket' and 'Debe rellenar todos los campos' (All fields must be filled).

Bottom Screenshot: The window shows the 'Fecha de compra' field with the value '2' entered. The 'Añadir Artículos' dropdown is still '2 - Artículo2'. The 'Artículos' list remains the same. The 'Total' field now shows '100.60'. The 'Cancelar' and 'Insertar' buttons are visible. A green notification bar at the bottom reads 'Notificación - Ticket' and 'La fecha debe ser en formato dd/mm/aaaa' (The date must be in dd/mm/aaaa format).

The screenshot shows a window titled "Nuevo Ticket". It contains the following fields and controls:

- Fecha de compra:** A text box containing "25/11/2024".
- Añadir Artículos:** A dropdown menu showing "2 - Artículo2".
- Artículos:** A text box containing "ArtículoPrueba, 100" and "Artículo2, 80".
- Total:** A text box containing "a", which is highlighted with a green border.
- Buttons:** "Cancelar" and "Insertar" at the bottom.

Below the main form, a green notification bar with a close button (X) displays the message "Notificación - Ticket". Below this, a larger gray area contains the error message "El total debe ser numérico". At the bottom of this area is an "Aceptar" button.

Si, por el contrario, rellenamos todos los campos correctamente, se nos mostrará el mensaje de éxito:

The screenshot shows a window titled "TiendecitaANC" with two tabs: "Artículos" and "Tickets". The "Tickets" tab is active. It contains the following elements:

- Notificación - Ticket:** A green notification bar with a close button (X).
- Message:** The text "Ticket creado con éxito" is displayed in the center.
- Button:** An "Aceptar" button is located at the bottom.

ID Artículo	Descripción	Precio	Stock
1	ArtículoPrueba	10.99	0
2	Artículo2	18.75	120

Volver

Si volvemos a la ventana “Consultar Artículos”, vemos que el Stock se ha actualizado según la cantidad de los mismos incluida en el ticket.

Al realizar la inserción, hemos trabajado en dos tablas diferentes de nuestra base de datos, a saber: Tabla “tickets”.

```
1 • SELECT * FROM tiendecitaanc.tickets;
2
```

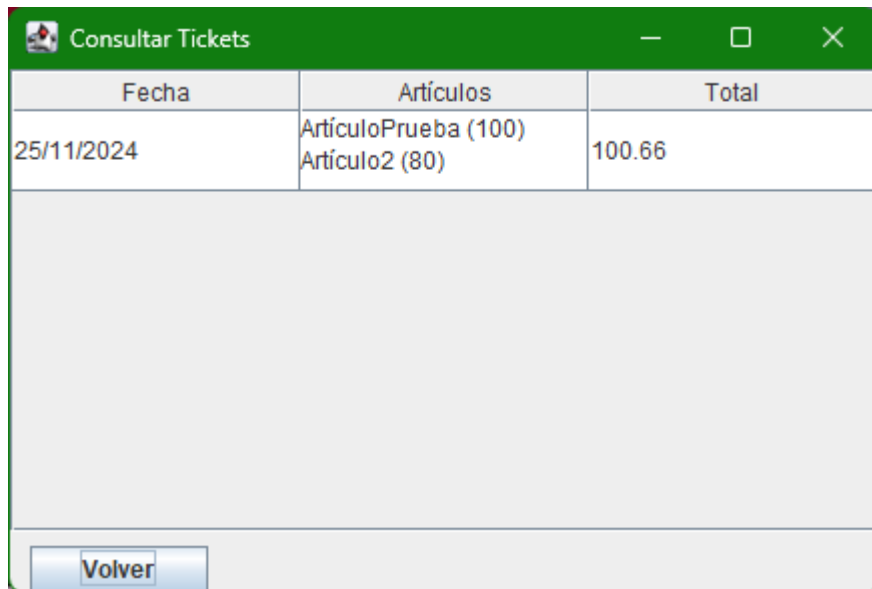
	idTicket	fechaTicket	precioTotal
▶	1	2024-11-25	100.66
*	NULL	NULL	NULL

Y Tabla “historico”.

```
1 • SELECT * FROM tiendecitaanc.historico;
```

	idHistorico	idTicketFK	idArticuloFK	cantidadArticulo
▶	1	1	1	100
	2	1	2	80
*	NULL	NULL	NULL	NULL

10.7.- Consulta Ticket



The screenshot shows a window titled 'Consultar Tickets' with a green header bar. Inside the window is a table with three columns: 'Fecha', 'Artículos', and 'Total'. The table contains one row of data for the date 25/11/2024, listing 'ArtículoPrueba (100)' and 'Artículo2 (80)' with a total of 100.66. Below the table is a large empty grey area and a 'Volver' button at the bottom left.

Fecha	Artículos	Total
25/11/2024	ArtículoPrueba (100) Artículo2 (80)	100.66

En la ventana de “Consultar Tickets”, encontraremos una tabla que mostrará:

- 1- La fecha del ticket.
- 2- Los artículos añadidos al ticket, con su cantidad correspondiente entre paréntesis.
- 3- El precio total del ticket.

11.- Valoración Personal

El desarrollo de esta práctica me ha servido para afianzar los conocimientos aprendidos tanto en la asignatura “Acceso a Datos” como en la asignatura “Desarrollo de Interfaces”. Permitiéndome realizar labores de operaciones CRUD en BBDD de una manera más eficiente y útil que en el año anterior del grado.

En lo personal, siento especial atracción por este tipo de prácticas, ya que me resulta de lo más interesante desarrollar programas con esta dinámica.

Estoy deseando profundizar en Hibernate para poder dar un paso más en la optimización de recursos y facilidad a la hora de realizar operaciones CRUD en bases de datos relacionales.

12.-Bibliografía

Grupo Studium ([↑](#))

1. *María José Martínez Navas, Desarrollo de Interfaces (Práctica Tema 4)*. Publicado en [Grupo Studium](#).

Recuperado:

https://campustudium.com/pluginfile.php/2069/mod_resource/content/9/Tema%204-Generaci%C3%B3n%20de%20Interfaces%20Gr%C3%A1ficas%20de%20Usuario%20Basadas%20en%20XML%20-%20Pr%C3%A1ctica_v1.pdf

Último acceso (25/11/2024).