

PROYECTO JAVA ALEMESAS3DPLACE



Proyecto y documentación hecho por: Alejandro Meneses Sánchez de la clase 1ºDAW
Logo hecho por: @Sanhehiart en instagram

1. Introducción	1
1.1 Justificación del proyecto	1
1.2 Objetivo o finalidad del proyecto	1
2. Ficha del proyecto	2
3. Manual del usuario/Administrador	2
3.1 Administración	4
3.2 Usuario	8
4. Servidor Xampp	11
5. Diseño JSwing o JOption	14
6. Cosas a mejorar	24

1. Introducción

Bienvenido a la documentación del proyecto de Alemesas3Dplace desarrollado por Alejandro Meneses Sánchez. Aquí encontrarás una guía rápida para su uso, así como información básica sobre su estructura y las instrucciones para contribuir al desarrollo.

1.1 Justificación del proyecto

Este proyecto es una aplicación totalmente funcional para la gestión de una empresa o tienda, con un enfoque principal en la administración de pedidos. La aplicación se divide en dos partes: una para la administración y otra para los usuarios. En este caso particular, se ha desarrollado en torno a un proyecto personal relacionado con pedidos de impresión 3D. Por ello, incluye especificaciones detalladas como material, tamaño y opciones de pintura. El objetivo del proyecto es proporcionar a tanto los usuarios como a los administradores una forma sencilla y eficiente de trabajar y realizar pedidos.

1.2 Objetivo o finalidad del proyecto

El objetivo principal de este proyecto ha sido aprender y aprovechar al máximo la interfaz visual de Java. Además, he trabajado en profundizar mis conocimientos sobre Java y su utilización junto con bases de datos. De esta manera, he desarrollado un programa completamente funcional que guarda todo el progreso realizado, permitiéndome su uso para gestionar encargos o pedidos de manera administrativa.

2. Ficha del proyecto

En este proyecto he utilizado

IDE: Eclipse.

Lenguaje: Java y MariaDB(SQL).

Herramientas externas: Chatgpt.

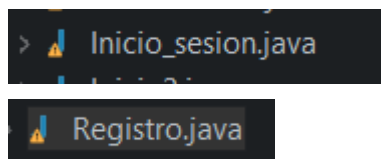
Información y aprendizaje: Algún video de YT

como(https://www.youtube.com/watch?v=LdBI0th_U_Q) y muchas cosas de ayuda de la pagina de Stack Overflow.

Imágenes: Todas sacadas de stock de internet, algunas de ellas por ejemplo sacadas de Google Icons.

3. Manual del usuario/Administrador

Para empezar tenemos una seccion entera de inicio se de sesión y registro conectada a la base de datos como podemos ver lo tenemos en diferentes clases y es lo que nos va a ejecutar todo el programa



Además de ello tendremos una conexión hecha para la base de datos en cada clase para facilitar algunos procesos

```
public static void main(String[] args) {  
    try {  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/alemesas3dplace", "root", "");  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Pediremos de forma gráfica con un Jswing que veremos más tarde el usuario y contraseña para entrar con este código en el que vemos que esta usando consultas

sql para comprobar que ese usuario existe en la base de datos

```
public void comprobacion() {
    try {
        String sql = "SELECT * FROM usuarios WHERE Usuario = ? AND Contraseña = ?";
        PreparedStatement statement = this.cn.prepareStatement(sql);
        statement.setString(1, usuario);
        statement.setString(2, contraseña);
        ResultSet rs = statement.executeQuery();
        if (rs.next()) {
            JOptionPane.showMessageDialog(null, "Inicio de sesion correcto");
            inicio = true;
            dispose();
        } else {
            JOptionPane.showMessageDialog(null, "Inicio de sesion incorrecto", "Error",
                JOptionPane.OK_CANCEL_OPTION);
            JOptionPane.showMessageDialog(null, "Vuelve a intentarlo", "Error",
                JOptionPane.OK_CANCEL_OPTION);
            inicio = false;
        }
    } catch (SQLException e2) {
        System.out.println("No se ha podido conectar con la base de datos");
        System.out.println("O ha habido algun fallo en la consulta");
    }
}
```

en el propio main con una array con todos los usuarios administradores veremos si ejecutamos el programa como usuario o como administrador

```
public static boolean admin(String usuario, String contrasena) {
    String[][] Admins = {
        {"Alemesa", "1234"},
        {"Vargues", "4321"},
        {"Sanhehi", "pipopi"}
    };

    for (int i = 0; i < Admins.length; i++) {
        if (Admins[i][0].equals(usuario) && Admins[i][1].equals(contrasena)) {
            return true;
        }
    }
    return false;
}
```

También tenemos la opción de registro para tener un usuario en la base de datos y así poder guardar toda la información para los envíos y pedidos

```

String direccion = sc.next();
String sql = "SELECT * FROM usuarios WHERE Usuario = ? OR Correo_electronico = ?";
PreparedStatement statement = this.cn.prepareStatement(sql);
statement.setString(1, usuario);
statement.setString(2, contraseña);
ResultSet rs = statement.executeQuery();

if (rs.next()) {
    JOptionPane.showMessageDialog(null, "Ya existe este usuario o correo, deberia cambiarlo",
        "Error", JOptionPane.OK_CANCEL_OPTION);
    registra = false;
} else {
    String sql2 = "INSERT INTO usuarios(Usuario,Contraseña,Correo_electronico,Direccion,Pais,Ciudad,Fecha_creacion) VALUES(?,?,?,?,?,?,?)";

    PreparedStatement insertstatement = this.cn.prepareStatement(sql2);
    insertstatement.setString(1, usuario);
    insertstatement.setString(2, contraseña);
    insertstatement.setString(3, correo);
    insertstatement.setString(4, direccion);
    insertstatement.setString(5, pais);
    insertstatement.setString(6, ciudad);

    int filasInsertadas = insertstatement.executeUpdate();
    registra = true;
}
} else {
    JOptionPane.showMessageDialog(null, "El usuario tiene mas de 25 caracteres", "Error",
        JOptionPane.OK_CANCEL_OPTION);
}
}

```

Primero se comprobará si el usuario o correo electrónico existe en la base de datos y si es así le pide al usuario que lo cambie y si no se guardará en la base de datos todos esos datos.

A partir de ahí veremos dos main diferentes el de usuarios y el de administración.

3.1 Administración

En el frame principal de administración veremos varias partes en la que a parte de decoración, podemos ver dos partes importantes a la izquierda vemos botones reactivos con este código

```

Panel_inicio.add(lblNewLabel, gbc);

JPanel Panel_Tabla_Pedidos = new JPanel();
Panel_Tabla_Pedidos.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
Panel_Tabla_Pedidos.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseExited(MouseEvent e) {
        Panel_Tabla_Pedidos.setBackground(new Color(1, 22, 49));
    }

    public void mouseEntered(MouseEvent e) {
        Panel_Tabla_Pedidos.setBackground(new Color(41, 94, 150));
    }

    public void mouseClicked(MouseEvent e) {
        Tabla_Pedidos t1 = new Tabla_Pedidos(cn);
        content.removeAll();
        content.add(t1, BorderLayout.CENTER);
        content.revalidate();
        content.repaint();
        t1.Cargartablau(cn);
    }
});

```

Podemos ver que al meter el ratón y sacarlo cambia de color y podemos ver que al clickarlo ocurre lo que vemos que es que se inicia una instancia de la clase de tabla pedidos, cogemos el frame de content lo limpiamos añadimos el frame de tabla pedidos y lo recargamos y repintamos como ultimo usamos el metodo de cargar la tabla para meter los datos

```

tableModel.setRowCount(0);
tableModel.setColumnIdentifiers(new String[] {
    "ID_pedido", "Nombre_pedido", "Fecha_Envio", "Estado", "Precio", "Dirección_del_pedido", "Metodo_Pago", "id_Cliente", "Cantidad", "Notas"
});

Statement statement = cn.createStatement();
String sql = "SELECT * FROM pedidos";
ResultSet rs = statement.executeQuery(sql);

while (rs.next()) {
    Object[] row = new Object[10];
    row[0] = rs.getInt("ID_pedido");
    row[1] = rs.getString("Nombre_Pedido");
    row[2] = rs.getDate("Fecha_envio");
    row[3] = rs.getString("Estado");
    row[4] = rs.getInt("Precio");
    row[5] = rs.getString("Direccion_del_pedido");
    row[6] = rs.getString("Metodo_pago");
    row[7] = rs.getInt("id_cliente");
    row[8] = rs.getInt("Cantidad");
    row[9] = rs.getString("Notas");
    tableModel.addRow(row);
}
} catch (SQLException e) {
    e.printStackTrace();
} else {
    try {

```

Vemos que carga las columnas le ponen los datos y luego rellena con la sentencia de sql con un while todos los datos de la base de datos, en el mismo frame también hemos cargado un botón en el que podemos ver los detalles de los pedidos

```

tableModel.setRowCount(0);
tableModel.setColumnIdentifiers(new String[]{"ID_pedido", "Color", "Material", "Pintado", "Terminado"});

try {
    String sql = "SELECT * FROM detalle_pedido";
    Statement statement = cn.createStatement();
    ResultSet rs = statement.executeQuery(sql);

    while (rs.next()) {
        Object[] row = new Object[5];
        row[0] = rs.getInt("ID_pedido");
        row[1] = rs.getString("Color");
        row[2] = rs.getString("Material");
        row[3] = rs.getString("Pintado");
        row[4] = rs.getString("Terminado");
        tableModel.addRow(row);
    }
} catch (SQLException e) {
    e.printStackTrace();
}

```

Al volver a clicar lo quitamos y cargamos la tabla anterior con este método

```

JLabel lblVerMasDetalles = new JLabel("Ver Mas detalles");
lblVerMasDetalles.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (!orden) {
            Cargartablad();
        } else {
            Cargartabla(cn);
        }
        orden = !orden;
    }
}

```

Todo esto funciona con una autenticación de si el que carga la tabla es el usuario o el administrador ya que cargamos la misma base de datos y comprobamos quién es al igual que el inicio de sesión con el array de administradores.

Hacemos lo mismo con el botón de la tabla de usuarios en la que mostramos todos los usuarios registrados y vemos todos sus datos para manejar y monitorizar las cuentas.

```

public void loadData(Connection cn) {
    try {
        Statement statement = cn.createStatement();
        String sql = "SELECT * FROM usuarios";
        ResultSet rs = statement.executeQuery(sql);

        while (rs.next()) {
            Object[] row = new Object[8];
            row[0] = rs.getInt("ID");
            row[1] = rs.getString("Usuario");
            row[2] = rs.getString("Contraseña");
            row[3] = rs.getString("Correo_electronico");
            row[4] = rs.getString("País");
            row[5] = rs.getString("Ciudad");
            row[6] = rs.getString("Direccion");
            row[7] = rs.getDate("Fecha_Creacion");
            tableModel.addRow(row);
        }
        rs.close();
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

En el el mainframe de las dos partes podemos ver una página de inicio de decoración para que veamos un poco las opiniones y una manera de tener más presencia.

Tenemos también como es necesario una forma de editar los pedidos y los usuarios funciona de la misma manera por ello vamos a ver por ejemplo el de los pedidos primero para ello en los dos casos vamos a pedir el id con un JOptionPane.

```

while (true) {
    String input = JOptionPane.showInputDialog(null, "Ingrese el número de pedido:");

    try {
        return Integer.parseInt(input);
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Entrada no válida. Por favor, ingrese un número entero.");
    }
}

```

Y aquí la verificación de la id esto se aplica también para el usuario de la misma manera

```

id_pedido = Mod_Pedido.solicitarNumeroDePedido();

try {
    String sql = "SELECT * FROM pedidos WHERE id_pedido = ?";
    PreparedStatement statement = cn.prepareStatement(sql);
    statement.setInt(1, id_pedido);
    live...ResultSet rs = statement.executeQuery();

    if (rs.next()) {
        JOptionPane.showMessageDialog(null, "La id Existe");

        Mod_Pedido M2 = new Mod_Pedido(cn, id_pedido);
        content.removeAll();
        content.add(M2, BorderLayout.CENTER);
        content.revalidate();
        content.repaint();
    } else {
        JOptionPane.showMessageDialog(null, "La id no existe, vuelve a intentarlo", "Error",
            JOptionPane.ERROR_MESSAGE);
    }
} catch (Exception ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error al verificar el pedido", "Error",
        JOptionPane.ERROR_MESSAGE);
}

```

Para recoger los datos en el frame pasaremos a cada uno de ellos un filtro para saber si esta vacío o no, ya que si no nos dará error al insertarlo en la base de

datos, tendremos un contador y haremos que al menos rellene algún cambio para que pulsar el botón no de totalmente error.

Por ejemplo con las partes de los datos que son texto normal sera

```
if (dateChooser.getDate() != null) {
    String sqlFechaEnvio = "UPDATE pedidos SET Fecha_envio = ? WHERE Id_pedido = ?";
    PreparedStatement stFechaEnvio = cn.prepareStatement(sqlFechaEnvio);
    stFechaEnvio.setDate(1, new java.sql.Date(dateChooser.getDate().getTime()));
    stFechaEnvio.setInt(2, id_pedido);
    mpedido = mpedido + stFechaEnvio.executeUpdate();
}
```

Con los datos que funcionan en una combobox lo haremos transformándolo antes

```
}
if ((String) metodo_pago.getSelectedItem() != ("")) {
    String sqlMetodoPago = "UPDATE pedidos SET Metodo_Pago = ? WHERE Id_pedido = ?";
    PreparedStatement stMetodoPago = cn.prepareStatement(sqlMetodoPago);
    stMetodoPago.setString(1, (String) metodo_pago.getSelectedItem());
    stMetodoPago.setInt(2, id_pedido);
    mpedido = mpedido + stMetodoPago.executeUpdate();
}
```

mientras que las que son de cantidades, lo haremos de una manera más controlado y controlaremos los negativos además de si está vacío

```
if (Precio_1.getText() != null && !Precio_1.getText().isEmpty()) {
    try {
        int precio = Integer.parseInt(Precio_1.getText());
        if (precio < 0) {
            JOptionPane.showMessageDialog(null, "El precio no puede ser negativo.", "Error",
                JOptionPane.ERROR_MESSAGE);
        } else {
            String sqlPrecio = "UPDATE pedidos SET Precio = ? WHERE Id_pedido = ?";
            PreparedStatement stPrecio = cn.prepareStatement(sqlPrecio);
            stPrecio.setInt(1, precio);
            stPrecio.setInt(2, id_pedido);
            mpedido = mpedido + stPrecio.executeUpdate();
        }
    } catch (NumberFormatException e1) {}
    JOptionPane.showMessageDialog(null, "Ingrese un valor numérico válido para el precio.",
        "Error", JOptionPane.ERROR_MESSAGE);
}
```

vamos cambiando los datos de a poco para así poder cambiar no todos los datos y poder hacerlo de manera más personalizada es decir que no tendras que cambiar todo si no solo algun dato

En esta sección también tenemos la opción de borrar el pedido o usuario


```

    }

    public void mouseClicked(MouseEvent e) {
        try {
            String sql = "Delete from pedidos where id_pedido= ?; ";
            PreparedStatement st = cn.prepareStatement(sql);
            st.setInt(1, id_pedido);
            String sql2 = "Delete from detalle_pedido where id_pedido = ?;";
            PreparedStatement st2 = cn.prepareStatement(sql2);
            st2.setInt(1, id_pedido);
            funciona = st2.executeUpdate();
            funciona = funciona + st.executeUpdate();
            if (funciona > 1) {
                JOptionPane.showMessageDialog(null, "El pedido y sus detalles han sido borrados");
            } else {
                JOptionPane.showMessageDialog(null, "El pedido no se ha podido borrar", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
}

```

En el de pedido también podemos ver cómo borramos sus detalles de la otra tabla para así no haya residuos por ninguna de los dos lados, vemos que tenemos un contador para ver las líneas ejecutadas y así saber si se ha borrado bien el pedido o ha habido algún tipo de error.

En la otra parte al modificar el usuario sigue siendo el mismo esquema pero con diferentes valores que coinciden con la tabla de usuario.

3.2 Usuario

En el main de usuario tenemos también el frame con el inicio estético además de 3 opciones que son poder ver tus pedidos y aquí entra lo que comentamos anteriormente de la verificación de si es un usuario o administrador.

```

} else {
    try {
        tableModel.setRowCount(0);
        tableModel.setColumnIdentifiers(new String[] {
            "ID_pedido", "Nombre_pedido", "Fecha_Envio", "Estado", "Precio", "Dirección_del_pedido", "Metodo_Pago", "id_cliente", "Cantidad", "Notas"
        });
        String Usuario = Inicio_sesion.getUsuario();
        String sql = "SELECT * FROM pedidos WHERE id_cliente = (SELECT id FROM usuarios WHERE Usuario = ?);";
        PreparedStatement statement = cn.prepareStatement(sql);
        statement.setString(1, Usuario);

        ResultSet rs = statement.executeQuery();

        while (rs.next()) {
            Object[] row = new Object[10];
            row[0] = rs.getInt("ID_pedido");
            row[1] = rs.getString("Nombre_pedido");
            row[2] = rs.getDate("Fecha_envio");
            row[3] = rs.getString("Estado");
            row[4] = rs.getInt("Precio");
            row[5] = rs.getString("Direccion_del_pedido");
            row[6] = rs.getString("Metodo_pago");
            row[7] = rs.getInt("id_cliente");
            row[8] = rs.getInt("Cantidad");
            row[9] = rs.getString("Notas");
            tableModel.addRow(row);
        }
        rs.close();
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

podemos ver que en el else entra la tabla del pedido pero con una subconsulta en la que solo se muestra los pedidos con el id del usuario que se ha registrado así no tenemos que pedir id de ningún tipo al usuario.

Otra de las opciones que tiene el usuario es hacer un pedido en la que rellenará unos datos algunos opcionales y otro obligatorios para rellenar las dos tablas, mientras que otras cosas como el precio el estado y algunas variables más inician todas de la misma manera e irá modificando el administrador conforme avance el tiempo, aquí podemos ver cómo lo hacemos en un if para que el usuario no introduzca ningún valor nulo o vacío

```
public void comprobacion1() {
    int cantidad = Integer.parseInt(Cantidad.getText());
    if (Nombre_Pedido_1.getText() != null && !Nombre_Pedido_1.getText().isEmpty()
        && (String) metodo_pago.getSelectedItems() != ("") && Direccion_del_pedido.getText() != null
        && !Direccion_del_pedido.getText().isEmpty() && Cantidad.getText() != null
        && !Cantidad.getText().isEmpty() && cantidad > 0
        && (String) Material.getSelectedItems() != ("")) {
    }
    if (cantidad < 0) {
        JOptionPane.showMessageDialog(null, "La cantidad no puede ser negativa.", "Error",
            JOptionPane.ERROR_MESSAGE);
    } else {
        try {
            String sql = "Insert Into Pedidos(Nombre_Pedido,Precio,Fecha_envio,Estado,Fecha_Llegada,Notas,Direccion_del_pedido,Metodo_pago,cantidad,id_cliente) VALUES (?,?,DATE_";
            double precio = 0;
            double altura = Integer.parseInt(Altura.getText());
            int cantidad = Integer.parseInt(Cantidad.getText());
            if ((String) Material.getSelectedItems() == ("Resina")) {
                precio = ((altura / 5) * 5) * cantidad;
            } else if ((String) Material.getSelectedItems() == ("PLA")) {
                precio = ((altura / 5) * 3) * cantidad;
            }
            if (Notas.getText() == null && Notas.getText().isEmpty()) {
                notas = "";
            } else {
                notas = Notas.getText();
            }
            PreparedStatement stPedido = cn.prepareStatement(sql);
            stPedido.setString(1, Nombre_Pedido_1.getText());
            stPedido.setDouble(2, precio);
            stPedido.setString(3, notas);
            stPedido.setString(4, Direccion_del_pedido.getText());
            stPedido.setString(5, (String) metodo_pago.getSelectedItems());
            stPedido.setInt(6, cantidad);
            stPedido.setString(7, Inicio_sesion.getUsuario());
            int ejecutado = stPedido.executeUpdate();
            if (ejecutado > 0) {
                int response = JOptionPane.showMessageDialog(null, "El coste sera "+precio, "Confirmar Pedido", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
                if (response == JOptionPane.YES_OPTION) {
                    JOptionPane.showMessageDialog(null,
                        "El pedido Pasara por revision le enviaremos un correo con la informacion de si ha sido aceptado");
                    String sql2 = "Insert into detalle_pedido(id_pedido,color,Material,Pintado,Terminado) VALUES((Select MAX(id_pedido) from pedidos),?,?, 'No', 'Sin empezar')";
                    PreparedStatement stPedido2 = cn.prepareStatement(sql2);
                    stPedido2.setString(1, Colorr.getText());
                    stPedido2.setString(2, (String) Material.getSelectedItems());
                    stPedido2.executeUpdate();
                    GuardarArchivo(Archivo3d);
                }
            }
        } catch (IOException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "El pedido no se ha podido mandar por un error ");
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "El pedido no se ha podido mandar por un error");
        }
    }
}
```

Ademas como extra el usuario podra aportar un archivo de referencia o de modelo 3d para imprimir con la librería files y un jButtonon como veremos ahora:

```
private void GuardarArchivo(File file) throws IOException {
    String Direccion = "/alemesas3Dplace/src/archivos3d";
    File targetDirectory = new File(Direccion);

    if (!targetDirectory.exists()) {
        targetDirectory.mkdirs();
    }

    Path targetPath = new File(targetDirectory, file.getName()).toPath();

    Files.copy(file.toPath(), targetPath, StandardCopyOption.REPLACE_EXISTING);
}
```

```

JButton Archivo = new JButton("Selecciona");
Archivo.setBounds(10, 29, 161, 33);
Archivo.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showOpenDialog(null);
        if (result == JFileChooser.APPROVE_OPTION) {
            Archivo3d = fileChooser.getSelectedFile();
        }
    }
});

// ...
GuardarArchivo(Archivo3d);

```

También podemos ver cómo se calcula el precio en base a lo introducido por el usuario y se le permite al usuario aceptar o no el precio dado y cambiar si es necesario algún detalle para abaratar o cobrar más.

Como usuario también tienes la opción de ver una tabla de precio sencilla y explicativa en la que saldrá en un frame diferente con una imagen y una política de devoluciones sencilla.

```

JTextArea policyText = new JTextArea();
policyText.setText("Política de Devolución:\n" +
    "1. Si el producto llega con desperfectos, por favor contactanos en un plazo de 7 días para gestionar la devolución.\n" +
    "2. Los productos personalizados no tienen derecho a devolución, salvo que presenten desperfectos.\n" +
    "3. Los gastos de envío de la devolución correrán a cargo del comprador, excepto en casos de productos defectuosos.");
policyText.setFont(new Font("Roboto", Font.PLAIN, 14));
policyText.setEditable(false);
JScrollPane policyScrollPane = new JScrollPane(policyText);
policyScrollPane.setBounds(10, 410, 777, 87);
bg.add(policyScrollPane, BorderLayout.CENTER);

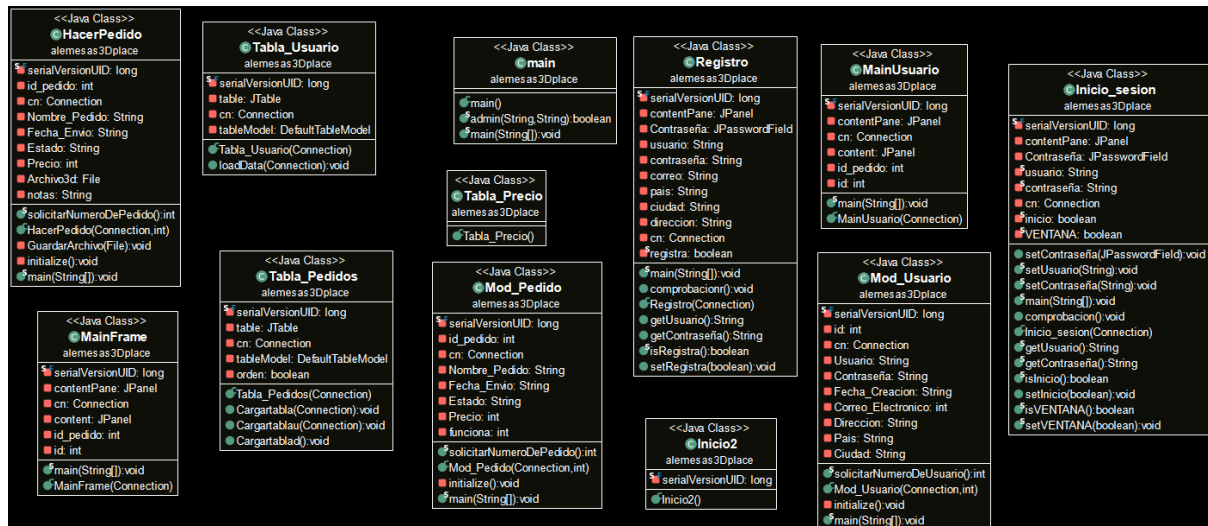
JLabel lblNewLabel = new JLabel("");
lblNewLabel.setIcon(new ImageIcon(Tabla_Precio.class.getResource("/imagenes/Tabla (1).png")));
lblNewLabel.setBounds(196, 10, 400, 380);
bg.add(lblNewLabel);

JLabel lblNewLabel_6 = new JLabel("");
lblNewLabel_6.setIcon(new ImageIcon(Tabla_Precio.class.getResource("/imagenes/imagen_2024-05-24_024020329.png")));
lblNewLabel_6.setBounds(0, 0, 797, 507);
bg.add(lblNewLabel_6);

// Crear panel de política de devoluciones
JPanel policyPanel = new JPanel();
policyPanel.setLayout(new BorderLayout());
policyPanel.setBorder(BorderFactory.createTitledBorder("Derecho de Devolución y Desperfectos"));

```

Aquí estaría el uml del proyecto:



4. Servidor Xampp

En el servidor de xampp hemos trabajado con una base de datos simple pero efectiva con algunas restricciones que comentaremos después y sobre todo totalmente útil y reactiva al programa

Antes de todo nos conectaremos a la base de datos de esta manera en el programa de java, para ello tendremos que insertar las librerías indicadas yo he utilizado estas:

```

> mysql-connector-java-8.0.11.jar

try {
    Connection cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/alemesas3dplace", "root", "");
    System.out.println("Conexion completada");
}

```

Donde vemos la dirección del servidor local de xampp y el usuario root con contraseña vacía.

Tenemos 3 tablas una de Usuario, Pedidos y detalle pedido
A continuación veremos el código de creación de cada una:

Usuarios:

```

CREATE TABLE `usuarios` (
  `Id` int(11) NOT NULL AUTO_INCREMENT,
  `Usuario` varchar(25) DEFAULT NULL,
  `Contraseña` varchar(40) DEFAULT NULL,
  `Fecha_Creacion` date DEFAULT NULL,

```

```

`correo_electronico` varchar(60) NOT NULL,
`Direccion` varchar(255) NOT NULL,
`Pais` varchar(20) NOT NULL,
`Ciudad` varchar(20) NOT NULL,
PRIMARY KEY (`Id`),
UNIQUE KEY `Usuario` (`Usuario`)
);

```

Pedidos:

```

CREATE TABLE `pedidos` (
  `id_pedido` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre_Pedido` varchar(30) DEFAULT NULL,
  `Precio` int(11) DEFAULT NULL,
  `Fecha_envio` date DEFAULT NULL,
  `Estado` varchar(20) DEFAULT NULL,
  `Fecha_llegada` date DEFAULT NULL,
  `Notas` varchar(255) DEFAULT NULL,
  `Direccion_del_pedido` varchar(255) DEFAULT NULL,
  `Metodo_pago` varchar(50) DEFAULT NULL,
  `id_cliente` int(11) DEFAULT NULL,
  `Cantidad` int(11) NOT NULL,
  PRIMARY KEY (`id_pedido`),
  KEY `id_cliente` (`id_cliente`),
  CONSTRAINT `pedidos_ibfk_1` FOREIGN KEY (`id_cliente`) REFERENCES
`usuarios` (`Id`),
  CONSTRAINT `estado_valido` CHECK (`Estado` in ('Preparando','En
camino','Entregado','Problema'))
);

```

Detalle_Pedido:

```

CREATE TABLE `detalle_pedido` (
  `id_detalle` int(11) NOT NULL AUTO_INCREMENT,
  `id_pedido` int(11) DEFAULT NULL,
  `Color` varchar(20) DEFAULT NULL,
  `Material` varchar(20) DEFAULT NULL,
  `Pintado` varchar(2) DEFAULT NULL,
  `Terminado` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id_detalle`),
  KEY `id_pedido` (`id_pedido`),

```

```

CONSTRAINT `detalle_pedido_ibfk_1` FOREIGN KEY (`id_pedido`)
REFERENCES `pedidos` (`id_pedido`),
CONSTRAINT `chk_pintado` CHECK (`Pintado` in ('Si','No')),
CONSTRAINT `chk_terminado` CHECK (`Terminado` in ('Si','En proceso','Sin
empezar'))
);

```

Esto es por parte de la creación de bases de datos mientras que tenemos muchas consultas hechas en el programa ya vistas anteriormente, pero ahora pondre algunas de las más complejas para explicar su funcionamiento.

Updates:

Hay muchos updates en el programa ya que lo utilizo para poder modificar las tablas y así poder mantenerla útil y viva con paso del tiempo.

Ej: `"UPDATE usuarios SET Fecha_Creacion = ? WHERE id = ?"`

Select:

Tenemos también varios select entre ellos lo utilizamos para la comprobación del inicio de sesión o de cosas más complejas como:

```

"SELECT * FROM pedidos WHERE id_cliente = (SELECT id FROM usuarios WHERE Usuario =
?);"

```

En este caso utilizamos el select con una subconsulta para poder buscar los pedidos del id con el nombre de usuario que nos ha indicado el programa.

```

"SELECT detalle_pedido.* FROM detalle_pedido INNER JOIN pedidos ON
detalle_pedido.id_pedido = pedidos.id_pedido INNER JOIN usuarios ON
pedidos.id_cliente = usuarios.id WHERE usuarios.usuario = ?"

```

En este otro caso podemos ver como utilizamos los inner join para combinar las tres tablas ya que queremos hacer un puente para buscar los detalles de los pedidos de los pedidos realizados por el usuario dado

Insert:

Tenemos unos insert que sirven para la introducción de los pedidos hechos por los usuarios y los registros hechos en la aplicación.

```

"INSERT INTO
usuarios(Usuario,Contraseña,Correo_electronico,Direccion,País,Ciudad,Fecha_creacion)

```

```
VALUES(?,?,?,?,?,?,Current_date)
```

Aquí podemos ver como introducimos todos los datos del usuario y la fecha de creación del usuario con un CURRENT_DATE haciendo que sea automático.

```
"Insert Into  
Pedidos(Nombre_Pedido,Precio,Fecha_envio,Estado,Fecha_Llegada,Notas,Direccion_del_pedido,Metodo_pago,cantidad,id_cliente) VALUES (?,?,DATE_ADD(CURDATE(), INTERVAL 7 DAY),'Preparando',DATE_ADD(CURDATE(), INTERVAL 14 DAY),?,?,?,(SELECT id from usuarios where usuario=?))"
```

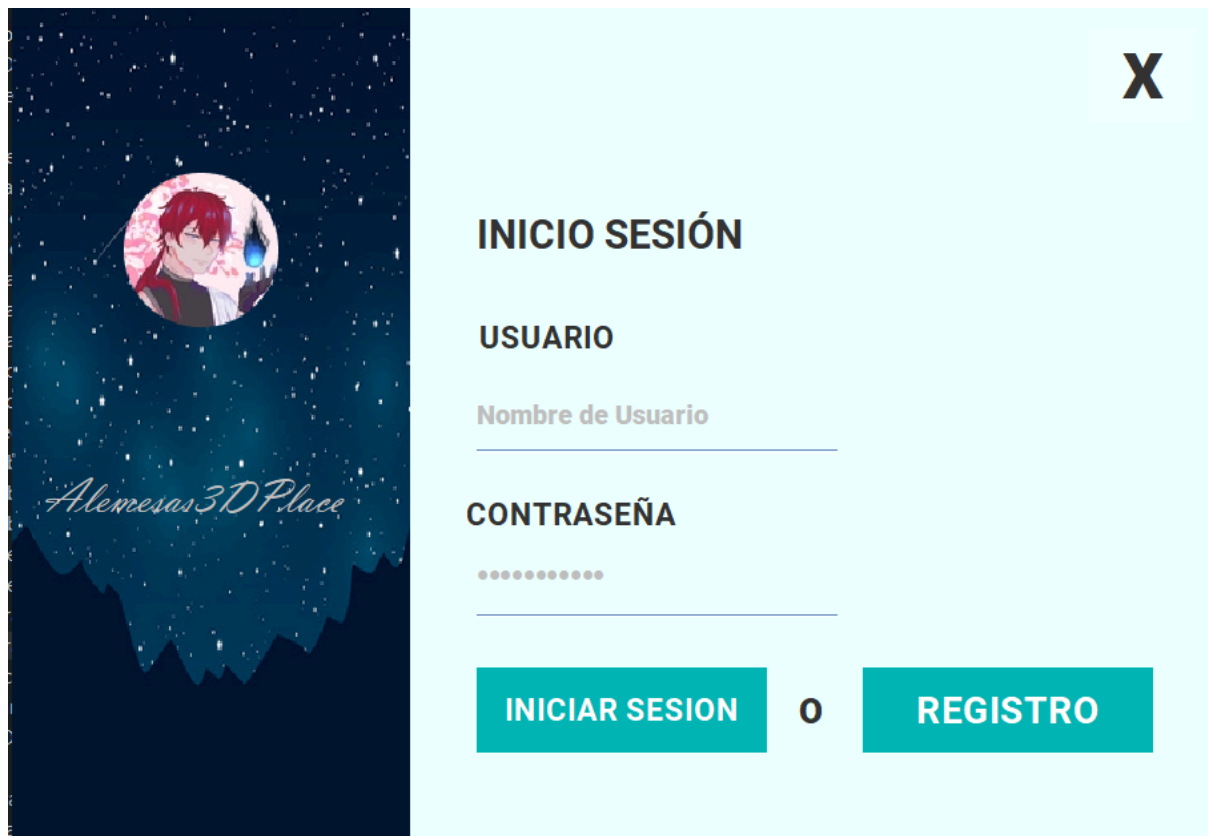
Aquí podemos ver mas funciones utilizadas para la inserción de datos como un Date_add lo que hace que a una fecha dada que en este caso es curdate es decir la fecha actual le añadimos 7 días para la fecha del envío, mientras que le añadimos 15 para la fecha de llegada, también insertamos algunos datos predeterminados y como final insertamos la id del usuario que ha hecho el pedido para así poder tener la id del usuario bien colocado y así que el mismo pueda ver sus pedidos más tarde.

```
"Insert Into detalle_pedido(id_pedido,Color,Material,Pintado,Terminado)  
VALUES((Select MAX(id_pedido) from pedidos),?,?, 'No', 'Sin empezar')"
```

También tenemos el caso de detalle_pedido donde le vamos a insertar la id del máximo número de id_pedido ya que al haberlo insertado anteriormente será la id correspondiente del pedido haciendo que este totalmente conectadas.

5. Diseño JSwing o JOption

Como podemos ver he usado sobre todo Jswing para el diseño y he intentado exprimir todo lo que tiene java en tema de diseño para que quede lo mas bonito posible, empezamos por el Inicio de sesión que se veria asi:



Viendo esto podemos ver varios botones jlabels y un poco de todo, constantemente va a estar la x en la zona de arriba derecha ya que he quitado el decorado que da windows a estas ventanas y creado el mio propio codigo de la x:

```
JLabel Cierre = new JLabel("X");
Cierre.setFont(new Font("Roboto Black", Font.PLAIN, 39));
Cierre.setBackground(new Color(238, 255, 255));
panelcierre.add(Cierre);
Cierre.addMouseListener(new MouseAdapter() {

    @Override
    public void mouseEntered(MouseEvent e) {
        panelcierre.setBackground(new Color(215, 255, 255));
    }

    public void mouseExited(MouseEvent e) {
        panelcierre.setBackground(new Color(243, 255, 255));
    }

    public void mouseClicked(MouseEvent e) {
        System.exit(0);
    }
});
Cierre.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
```

Vemos cosas como que al pasar el ratón por encima o quitarlo cambia el color y que tiene de manera predeterminada el ratón en modo pulsado, cuando se clicka se cierra todo el programa

INICIAR SESION

O

REGISTRO

Podemos ver algunos detalles como estos en los que el raton hace cambiar de color a los botones al pasar por encima es el mismo tipo de codigo que en la X

```
Contraseña = new JPasswordField();
Contraseña.setFont(new Font("Tahoma", Font.PLAIN, 16));
Contraseña.addFocusListener(new FocusAdapter() {
    @Override
    public void focusGained(FocusEvent e) {
        if (String.valueOf(Contraseña.getPassword()).equals("Aqui tu con")) {
            Contraseña.setText("");
            Contraseña.setForeground(Color.black);
        }
        if (Usuario.getText().isEmpty()) {
            Usuario.setText("Nombre de Usuario");
            Usuario.setForeground(Color.gray);
        }
    }
});
Contraseña.setBorder(null);
```

Para poder hacer que las letras en los JPasswordField y JTextField se quite y cambien al ganar el focus este seria el codigo haciendo que si clickas o le das al tabulador se borra lo que esta predeterminado y cambia al color negro para escribir tu contraseña y usuario

En la parte de registro tenemos algo muy parecido pero más extendido para rellenar los datos además de un botón de vuelta atrás en la que hace que vuelvas a la zona de inicio de sesión así es como funcionaria

```
panel.add(lblNewLabel, gbc);

JPanel Atras = new JPanel();
Atras.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
Atras.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        Inicio_sesion inicio_sesion = new Inicio_sesion(cn);
        inicio_sesion.setVisible(true);

        dispose();
    }
    public void mouseEntered(MouseEvent e) {
        Atras.setBackground(new Color(215, 255, 255));
    }

    public void mouseExited(MouseEvent e) {
        Atras.setBackground(new Color(243, 255, 255));
    }
});
```

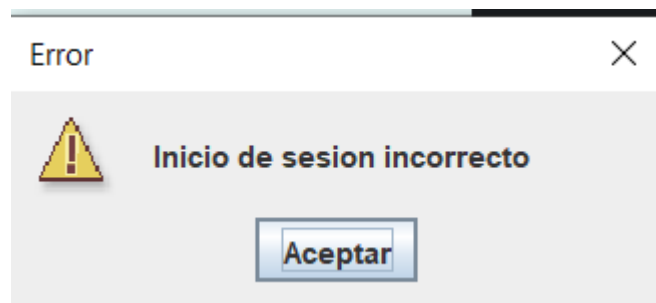
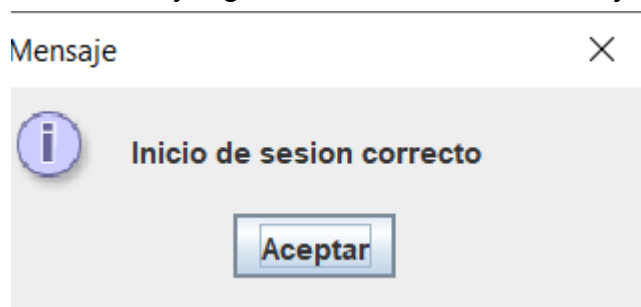
y un ejemplo del textfield en esta sección sería así en todos los casos menos contraseña

```

JFormattedTextField Ciudad_1 = new JFormattedTextField();
Ciudad_1.addFocusListener(new FocusAdapter() {
    public void focusGained(FocusEvent e) {
        if (Ciudad_1.getText().equals("Ciudad")) {
            Ciudad_1.setText("");
            Ciudad_1.setForeground(Color.black);
        }
    }
});

```

Tambien hay algunas cosas indicadas con JOptionPane como esta al iniciar sesion



Al avanzar veremos esta parte que será el mainframe ya sea de usuario o admin



Podemos distinguir fondo colores y fotos puestas con JLabels JPanel, etc, también a la izquierda el nombre del usuario que ha iniciado sesión y debajo cada una de los Jpanels que se van a ir actualizando al pulsarlo en la zona de inicio


```
content = new JPanel();
content.setBounds(289, 117, 797, 520);
content.setBorder(null);
content.setBackground(new Color(173, 189, 225));
content.setLayout(new BorderLayout());
bg.add(content);

Inicio2 I1 = new Inicio2();
content.removeAll();
content.add(I1, BorderLayout.CENTER);
content.revalidate();
content.repaint();
```

Creamos el content y lo inicializamos en el inicio

Luego dependiendo de lo pulsado se ira cambiando de la misma manera

```
public void mouseClicked(MouseEvent e) {
    Tabla_Pedidos t1 = new Tabla_Pedidos(cn);
    content.removeAll();
    content.add(t1, BorderLayout.CENTER);
    content.revalidate();
    content.repaint();
    t1.Cargartabla(cn);
}
```



Alemesa

Inicio

Tabla Pedidos

Modificar Pedido

Tabla Usuarios

Modificar Usuario

Alemesa3DPlace

X

ID_pedido	Nombre...	Fecha_E...	Estado	Precio	Direcci...	Metodo...	id_Clien...	Cantidad	Notas
1	añaña	1245-0...	Problema...	5	mi casa	A mano	11	4	ñi
7	ResinaC...	2024-0...	Prepara...	200	Mi casa	Paypal	1	0	
8	tarta	2024-0...	Prepara...	0	Blobi	Paypal	21	2	Hola
9	Psp	2024-0...	Prepara...	0	asdasd	Tarjeta ...	11	1	
11	Psp	2024-0...	Prepara...	0	asdasd	Tarjeta ...	11	1	
12	Psp	2024-0...	Prepara...	0	asdasd	Tarjeta ...	11	1	
13	Pspspsp	2024-0...	Prepara...	0	Asdasd	Tarjeta ...	11	12	
14	Pruebaa...	2024-0...	Prepara...	0	123123	Transfer...	11	12	
15	ASdasd	2024-0...	Prepara...	0	asdasd	Transfer...	11	1	
17	Goku	2024-0...	Prepara...	0	Mi casa	Tarjeta ...	11	2	
18	Vegeta	2024-0...	Prepara...	24	Casa	Tarjeta ...	11	2	
19	Mario	2024-0...	Prepara...	10	Mi casa	Tarjeta ...	11	2	
20	Mario	2024-0...	Prepara...	10	Mi casa	Tarjeta ...	11	2	
21	Mario	2024-0...	Prepara...	10	Mi casa	Tarjeta ...	11	2	
22	Kirby	2024-0...	Prepara...	39	Tu casa	Tarjeta ...	11	3	
23	Kirby	2024-0...	Prepara...	36	Tu casa	Tarjeta ...	11	3	

Ver Mas detalles

En tabla pedidos podremos ver una tabla llena de datos y un botón de ver mas de talles para ir cambiando entre dos diferentes tablas se ha hecho asi:

```

table.setFont(new Font("Roboto Black", Font.BOLD, 15));

JTableHeader header = table.getTableHeader();
header.setBackground(new Color(70, 130, 180));
header.setForeground(Color.WHITE);
header.setFont(new Font("Roboto Black", Font.BOLD, 16));

TableCellRenderer rendererFromHeader = header.getDefaultRenderer();
DefaultTableCellRenderer renderer = (DefaultTableCellRenderer) rendererFromHeader;
renderer.setHorizontalAlignment(JLabel.CENTER);
scrollPane.setViewportView(table);

JLabel lblNewLabel = new JLabel("New label");
lblNewLabel.setIcon(new ImageIcon(Tabla_Pedidos.class.getResource("/imagenes/imagen_2024-05-24_024020329.png")));
lblNewLabel.setBounds(0, 0, 797, 520);
gb.add(lblNewLabel);
}


```

Y en el botón como ya mostrado anteriormente cambiamos al otro estilo de tabla

Al modificar un pedido veremos un joption saltar y pedir una id

Entrada

X



Ingrese el número de pedido:

Aceptar

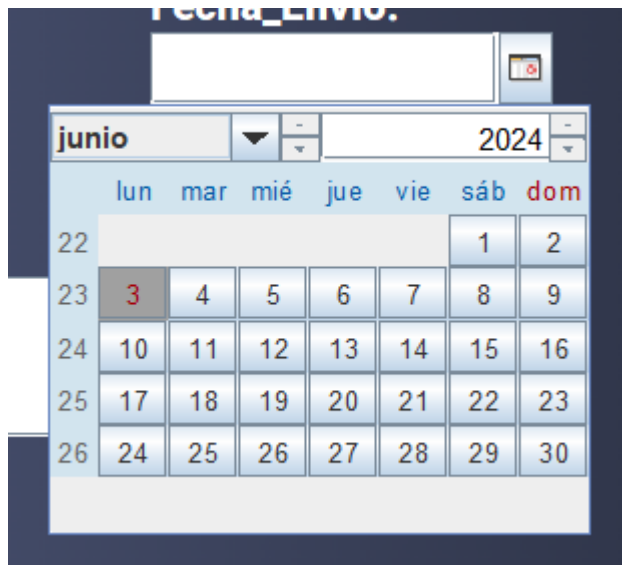
Cancelar

Además de tener control sobre lo que introduce el usuario tendremos al entrar esta sección

Con varios combobox, textfield y un Jcalendar

Los combobox funcionaran con un array creados para cada uno de ellos y el Jcalendar es una librería externa introducida para poder seleccionar una fecha y asi poder editar la fecha del envío

```
String[] Estadop = { "", "Preparando", "En Camino", "Entregado", "Problema" };
JComboBox<String> Estado_1 = new JComboBox<>(Estadop);
Estado_1.setFont(new Font("Roboto Black", Font.PLAIN, 17));
Estado_1.setBounds(10, 27, 161, 31);
panel_1_1.add(Estado_1);
```



```
panel_1_1_5_1.add(separator_1_5_1);

JDateChooser dateChooser = new JDateChooser();
dateChooser.setDateFormatString("yyyy-MM-dd");
dateChooser.setBounds(10, 31, 161, 29);
panel_1_1_5_1.add(dateChooser);
```

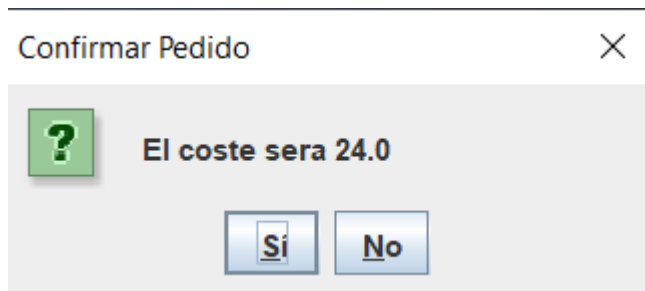
También vemos que el botón de modificar se ilumina y veremos que al pulsarlo se actualizará los datos mientras que si le damos a borrar borraremos todo el pedido. Lo mismo para tabla usuario y para modificar usuario

Pasamos al main usuario que se veria asi



Vemos el mismo inicio pero con menos opciones, entre ellas está mis pedidos la cual nos mostrará la tabla de pedidos ya mostrada anteriormente pero solo con los pedidos del usuario iniciado mientras que la opción de hacer pedido mostrará:

Cuando todo se haya rellenado y se haya subido o no un archivo podrá calcular el precio lo que hará que te salte una opción:



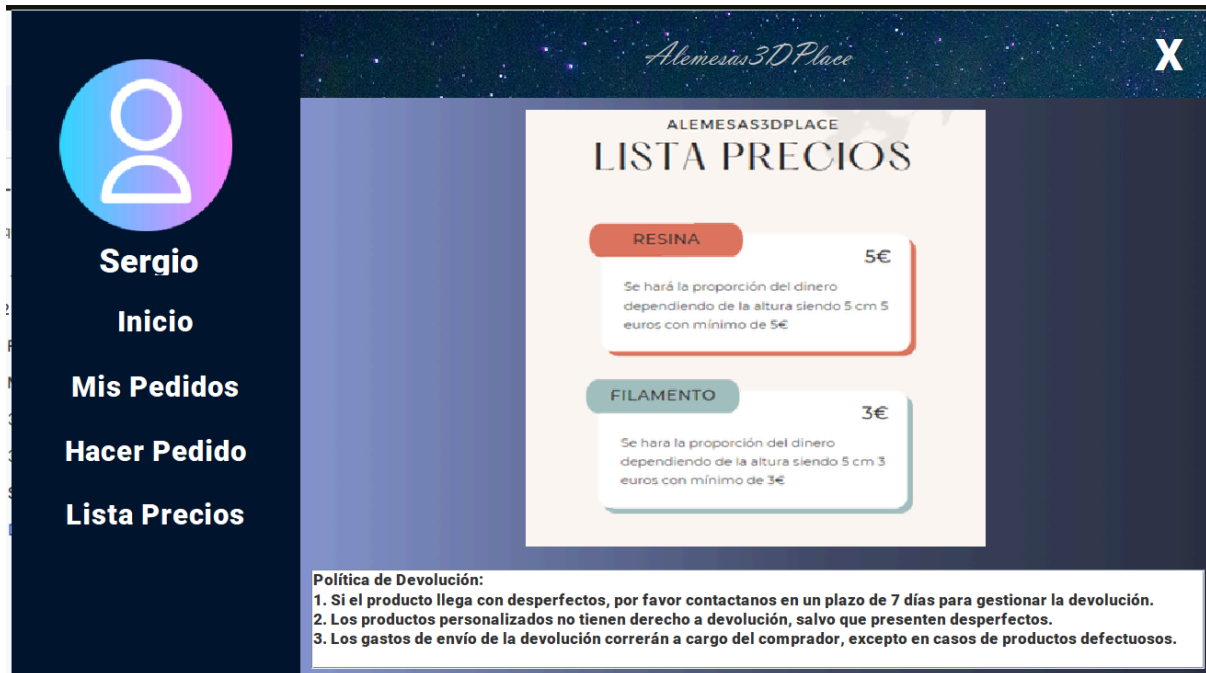
Si se acepta se mandará la información a la base de datos, si se rechaza se podrá modificar lo necesario

```
int response = JOptionPane.showConfirmDialog(null, "El coste sera " + precio, "Confirmar Pedido", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);

if (response == JOptionPane.YES_OPTION) {
    JOptionPane.showMessageDialog(null, "El pedido Pasara por revision le enviaremos un correo con la informacion de si ha sido aceptado");
    String sql2 = "Insert Into detalle_pedido(id_pedido,Color,Material,Pintado,Terminado) VALUES((Select MAX(id_pedido) from pedidos),?,?,'No','Sin empezar')";
    PreparedStatement stPedido = cn.prepareStatement(sql2);
    stPedido.setString(1, Colorr.getText());
    stPedido.setString(2, (String) Material.getSelectedItems());
    stPedido.executeUpdate();
    GuardarArchivo(Archivo3d);
}

catch (IOException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null, "El pedido no se ha podido mandar por un error");
}
catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, "El pedido no se ha podido mandar por un error");
}
```


Mientras tanto en lista de precios de manera sencilla se coloca un jlabel con la lista de precios actual y una política de devolución sencilla:



6. Cosas a mejorar

El programa como esta tiene muchas cosas a mejorar y creo que puede llegar a tener mucho potencial de uso ya que es una tienda o proyecto que en una empresa de encargos sería fácil y yo creo que intuitiva de usar obviamente corrigiendo algunas cosas::

1. Mayor control de errores y de introducción de datos como direcciones de correo o de países, ciudades, etc.
2. Una tienda con stock y recibos en la que podrá el usuario comprar material para impresiones de 3d propia o incluso impresoras 3d.
3. Una base de datos subida a la nube para poder conectarse de manera remota de cualquier manera.
4. Algún tipo de personalización del aspecto de la aplicación como foto de perfil guardada en una base de datos para cada usuario.
5. Más información sobre encargos hechos y opiniones de diferentes usuarios con imágenes.
6. Posibilidad de optimizar código.

1. Introducción 1.1. Justificación del proyecto(para qué sirve , qué quieres conseguir con este proyecto....) 1.2. Objetivos o finalidad de tu proyecto. 2. Ficha del proyecto: qué vas a utilizar: (programas o aplicaciones, lenguajes de programación utilizados, sistema operativo...)

3. Manual del usuario/Administrador: Desarrollo del proyecto. -Todo código y utilidad del programa. Desarrollo de [clases](#), conexión a base de datos, SQL, subida a Github, [ficheros](#). - Explicación de los posibles menús del programa. - Codificación de la aplicación del programa(cómo está estructurado)

4. Servidor Xampp. - Codificación de la base de datos - Script a utilizar - Desarrollo y aplicación en la app - Conexiones.

5. Diseño con Swim o JOption - Código - Creación del diseño(imagen)

6. Posibles mejoras que se pueden dar y adaptaciones .(vender el producto, cómo puede influir en una empresa y de qué manera se puede mejorar si es posible)

La documentación o memoria final se entregará en formato pdf. Los requisitos mínimos, en cuanto a formato, de esta documentación son:

- Debe contener una portada en la que aparezca el título del proyecto, vuestro nombre y apellidos, el ciclo y el curso.
- Debe incluir un índice.
- Fuente: Arial, tamaño 12
- Márgenes: 2 cm (sup. e inf.), 2,5 cm (izdo, dcho)
- Páginas numeradas
- Que no se olvide una de las cosas más importantes: NO PUEDE HABER FALTAS DE ORTOGRAFÍA