Aplicación de funcionalidad NUI

Alberto Yerpes Jimenez

1.- Extraer mediante cámara, la foto del usuario en un registro y/o desde la vista perfil de usuario.

Principalmente modificamos la base de datos para guardar la imagen junto a cada usuario:

```
CREATE TABLE `user` (
  `email` varchar(255) NOT NULL,
  `username` varchar(16) NOT NULL,
  `password` longblob NOT NULL,
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `active` tinyint(1) DEFAULT NULL,
  `foto` LONGBLOB DEFAULT NULL,
  PRIMARY KEY (`email`),
  UNIQUE KEY `username_UNIQUE` (`username`),
  UNIQUE KEY `email_UNIQUE` (`email`)
);

INSERT INTO `user` (`email`, `username`, `password`, `active`, `foto`) VALUES
('admin@empresa.com', 'admin', 0xc91d0f919f23f1d56157b94ba87db805, 1, NULL);
```

Y también modificamos los métodos SQL que tenemos en el código:

```
public boolean insertarUsuarios(String user, String email, String contrasenia, InputStream fotoStream) {
    Connection con = null;
    PreparedStatement ps;
    String sql = "INSERT INTO user (username, email, password, active, FOTO) VALUES (?, ?, AES_ENCRYPT(?, 'key'), 1, ?)";

    try {
        con = Conection.getConnection();
        if (con == null) {
            return false;
        }

        ps = con.prepareStatement(sql);
        ps.setString(parameterIndex: 1, x: user);
        ps.setString(parameterIndex: 2, x: email);
        ps.setString(parameterIndex: 3, x: contrasenia);
        ps.setBlob(parameterIndex: 4, inputStream:fotoStream);

        ps.executeUpdate();
        return true;

    } catch (SQLException ex) {
        App.showAlert(title: "Error SQL", "Error al guardar: " + ex.getMessage(), type:Alert.AlertType.ERROR);
        return false;
    } finally {
        try {
            if (con != null) {
                con.close();
            }
        } catch (SQLException ex) {
            System.out.println("Error al cerrar conexión: " + ex.getMessage());
        }
    }
}
```

Y creamos un método de SQL para seleccionar la foto para mostrarla:

```
public Image obtenerFoto(String username) {
    Connection con = null;
    PreparedStatement ps;
    ResultSet rs;

    try {
        con = Conection.getConnection();
        String sql = "SELECT foto FROM user WHERE username = ?";
        ps = con.prepareStatement(sql);
        ps.setString(parameterIndex: 1, x: username);
        rs = ps.executeQuery();

        if (rs.next()) {
            InputStream is = rs.getBinaryStream(columnLabel:"foto");
            if (is != null) {
                return new Image(in: is);
            }
        }
    } catch (Exception e) {
        System.out.println("Error al cargar foto: " + e.getMessage());
    } finally {
        try {
            if (con != null) {
                con.close();
            }
        } catch (SQLException ex) {
        }
    }
    return null; // Si no hay foto o falla
}
```

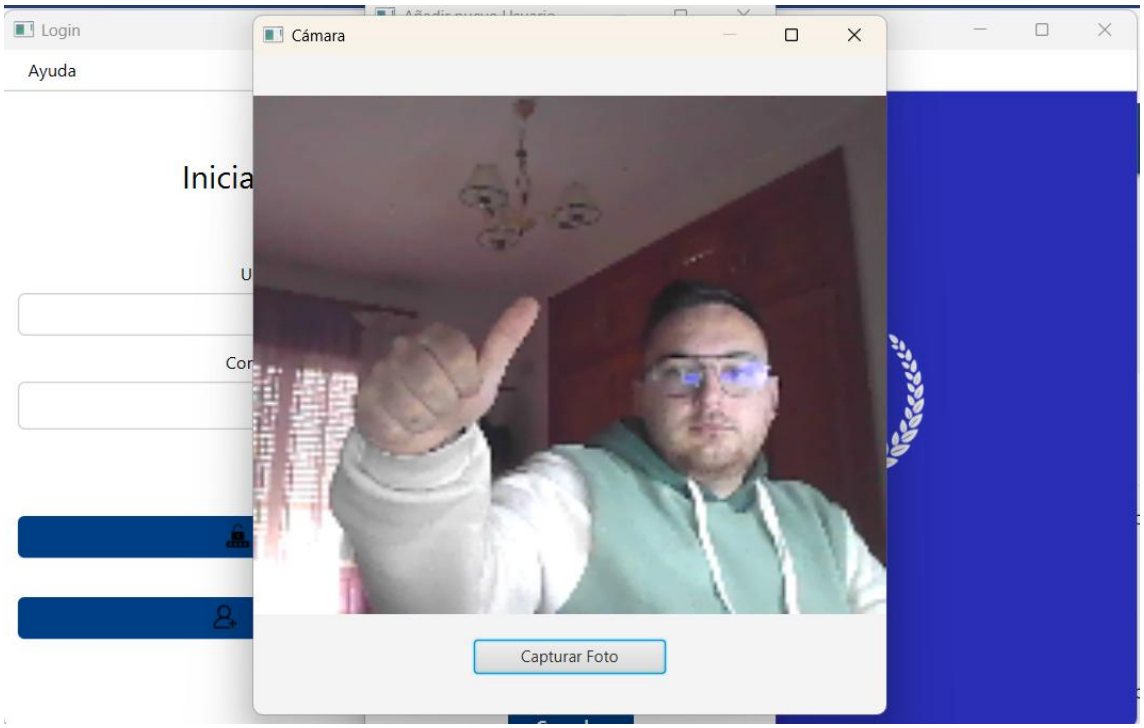
Creamos un gestor de imagen donde metemos los métodos que vamos a necesitar para echar nuestra foto:

```
public class GestorImagen {

    public static void abrirCamara(ImageView destino, Class<?> claseContexto) {
        try {
            // Buscamos el fxml relativo a la clase que nos llama
            FXMLLoader loader = new FXMLLoader(url:claseContexto.getResource(name:"camara.fxml"));
            Parent root = loader.load();

            CamaraController controller = loader.getController();

            Stage stage = new Stage();
            stage.initModality(modality:Modality.APPLICATION_MODAL);
            stage.setScene(new Scene(parent: root));
            stage.setTitle(value:"Cámara");

            stage.setOnCloseRequest(event -> controller.cerrarCamara());

            stage.showAndWait();

            if (controller.getImagenCapturada() != null && destino != null) {
                destino.setImage(value:controller.getImagenCapturada());
            }

        } catch (Exception e) {
            App.showAlert(title:"Error", message:"No se pudo abrir la cámara: ", type:Alert.AlertType.ERROR);
            e.printStackTrace();
        }
    }
```

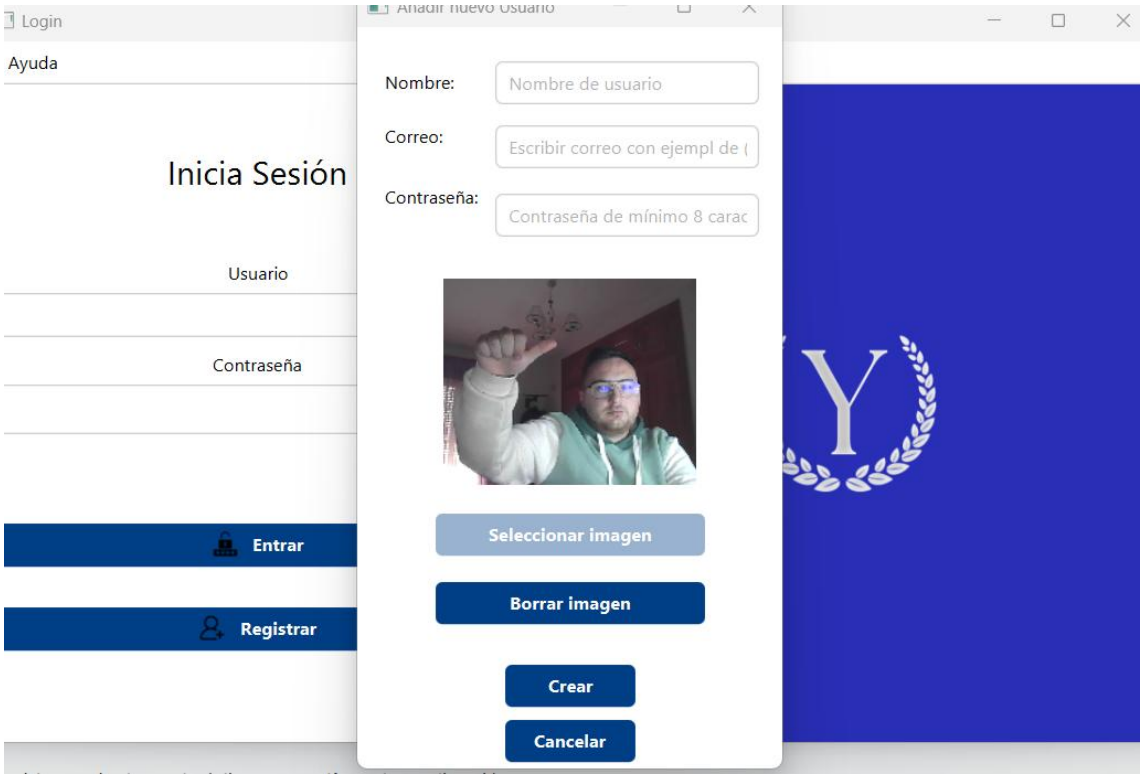
Dentro del controlador del nuevo usuario ponemos este método para cuando le demos a seleccionar foto:

```
//Metodo para abrir la camara y echar foto
@FXML
private void abrirCamara(ActionEvent event) {
    // Guardamos la imagen que estaba antes, que tiene que ser la de por defecto
    Image imagenAnterior = imagenPerfil.getImage();

    //Nos metemos en el gestor para echar la foto
    GestorImagen.abrirCamara(destino:imagenPerfil, claseContexto:getClass());

    /* Si la imagen no es la misma que la del anterior ponemos los botones
    para que no se puedan seleccionar ninguna otra y solo borrar*/
    if (imagenPerfil.getImage() != imagenAnterior) {
        btnImagen.setDisable(value:true);
        btnBorrarImagen.setDisable(value:false);

    }
}
```

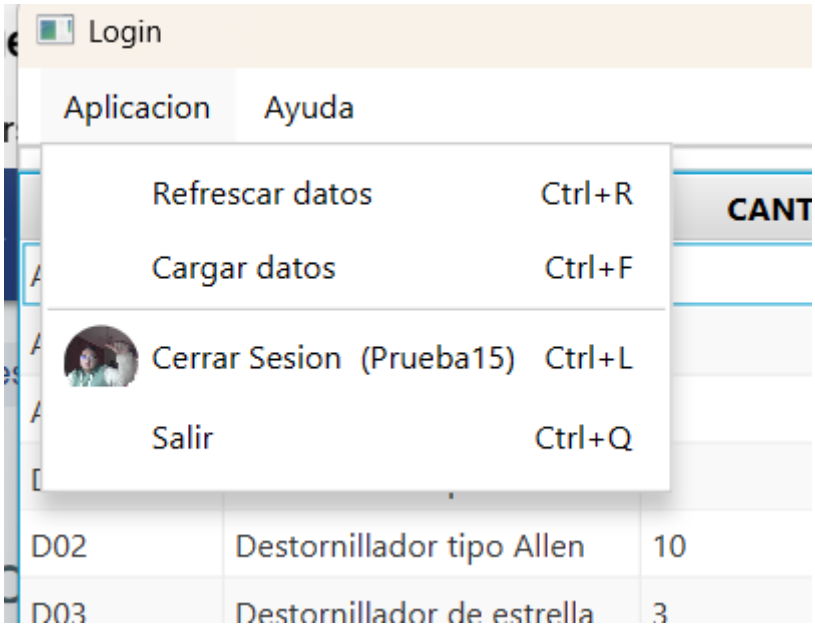
En la captura se ve el método en el cual nos dirige a la escena de la cámara y echamos la imagen dándole a seleccionar imagen en el registro:



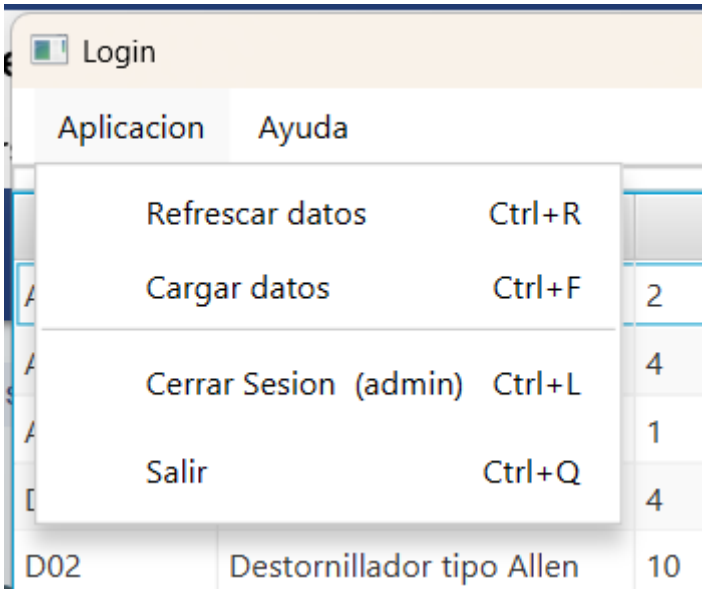
Le damos a capturar imagen y nos devuelve al registro poniéndose en la imagen view que tenemos en el registro ya que la enviamos con destino.setImage(controller.getImagenCapturada()); y quedaría así:



Ahora rellenamos todos los datos con usuario Prueba15 y le damos a crear, y entramos y vemos como en el menú bar donde pone nuestro nombre y nuestra foto para mostrarlo:



Si entramos con un usuario que no tenga foto como en el caso que es Admin, no pasa nada quedaría así:



2.- Reconocer voz mediante la cual se abra/cierre una aplicación o proporcione funcionalidad nueva

Creamos la clase funcion hablar que es donde le diremos que voz tiene que poner y de donde, en nuestro caso usaremos la de Windows que viene por defecto:

```
public class FuncionHablar {

    private static boolean vozActivada = false;

    private static Process procesoActual;
    private static final Object BLOQUEO = new Object();

    public static void setVozActivada(boolean activada) {
        vozActivada = activada;
    }

    public static boolean estaVozActivada() {
        return vozActivada;
    }

    public static void hablar(String texto) {
        if (!vozActivada || texto == null || texto.trim().isEmpty()) {
            return;
        }

        new Thread(() -> {
            synchronized (BLOQUEO) {
                if (procesoActual != null && procesoActual.isAlive()) {
                    procesoActual.destroyForcibly();
                    procesoActual = null;
                }

                try {
                    String comando = "PowerShell -Command \"Add-Type -AssemblyName System.Speech; "
                            + "(New-Object System.Speech.Synthesis.SpeechSynthesizer).Speak('" + texto + "');\"";
                    procesoActual = Runtime.getRuntime().exec(command:comando);

                } catch (Exception e) {
                    App.showAlert(title: "Error", message:"No se pudo procesar la voz", type:Alert.AlertType.WARNING);
                }
            }
        }).start();
    }

    /*  MÉTODOS PONER VOZ */

    public static void ponerVoz(Labeled elemento) {
        if (elemento != null) {
            elemento.setOnMouseEntered(e -> hablar(texto: elemento.getText()));
        }
    }

    public static void ponerVoz(TextInputControl campo) {
        if (campo != null) {
            campo.setOnMouseEntered(e -> {
                String txt = (campo.getText() != null && !campo.getText().isEmpty()) ? campo.getText() : campo.getPromptText()
                hablar(texto: txt);
            });
        }
    }

    public static void ponerVoz(ComboBox<?> combo) {
        if (combo != null) {
            combo.setOnMouseEntered(e -> {
                String txt = (combo.getValue() != null) ? combo.getValue().toString() : combo.getPromptText();
                hablar(texto: txt);
            });
        }
    }
```

```
private static void detenerSilencio() {
    synchronized (BLOQUEO) {
        if (procesoActual != null && procesoActual.isAlive()) {
            procesoActual.destroyForcibly();
        }
    }
}

private static synchronized void setProcesoActual(Process p) {
    procesoActual = p;
}
```

Para hacer que lea en voz alta por ejemplo creamos otro gestor para controlar los métodos de hablar que sería así:

```
public class GestorHablar {

    public static void adjudicarVoces(Node... elementos) {
        for (Node nodo : elementos) {
            // Si es un Botón, Label, Checkbox...
            if (nodo instanceof Labeled) {
                FuncionHablar.ponerVoz((Labeled) nodo);
            } // Si es un TextField, TextArea, PasswordField...
            else if (nodo instanceof TextInputControl) {
                FuncionHablar.ponerVoz((TextInputControl) nodo);
            } // Si es un ComboBox
            else if (nodo instanceof ComboBox) {
                FuncionHablar.ponerVoz((ComboBox) nodo);
            }
        }
    }
```

Este método principalmente para poder darle voces a todos los elementos de la escena.

En el método siguiente es para darle voz a la lista del combo box que devuelve el string de cada uno:

```
public static void ponerVozComboBox(ComboBox<Ubicacion> combo) {

    combo.setCellFactory(new Callback<ListView<Ubicacion>, ListCell<Ubicacion>>() {
        @Override
        public ListCell<Ubicacion> call(ListView<Ubicacion> param) {
            return new ListCell<Ubicacion>() {
                @Override
                protected void updateItem(Ubicacion item, boolean empty) {
                    super.updateItem(item, empty);

                    if (item == null || empty) {
                        setText(value: null);
                        setOnMouseEntered(eh: null);
                    } else {
                        setText(value: item.toString());

                        setOnMouseEntered(new EventHandler<MouseEvent>() {
                            @Override
                            public void handle(MouseEvent event) {
                                // Comprobamos si la voz esta activa antes de hablar
                                if (FuncionHablar.estaVozActivada()) {
                                    FuncionHablar.hablar(texto: item.toString());
                                }
                            }
                        });
                    }
                }
            };
        }
    });
}
```

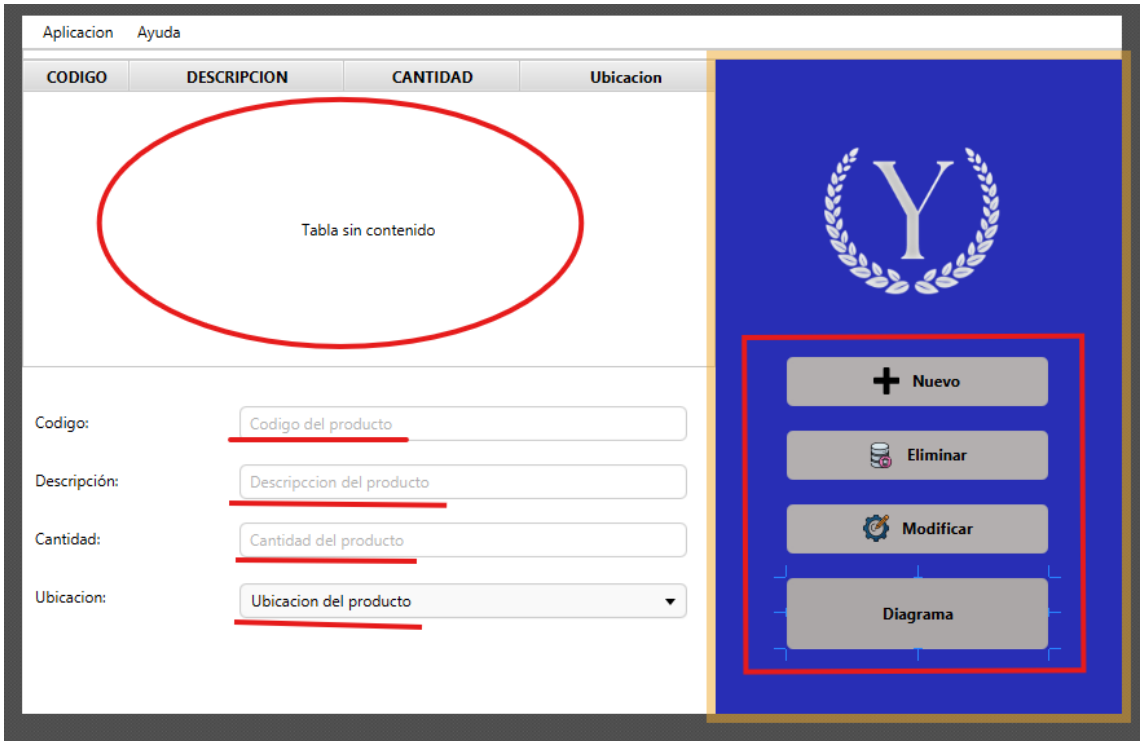
En los controladores que podemos activar la voz o no ponemos esto en initialize para activarlo a la paz que le ponemos nombre y le adjudicamos las voces:

```
cbVoz.selectedProperty().addListener(new ChangeListener<Boolean>() {
    @Override
    public void changed(ObservableValue<? extends Boolean> observable, Boolean oldValue, Boolean newValue) {
        FuncionHablar.setVozActivada(activada: newValue);

        if (newValue) {
            FuncionHablar.hablar(texto: "Modo voz activado");
        }
    }
});

FuncionHablar.hablar(texto: "Login de usuario");

GestorHablar.adjudicarVoces(elementos: jbLogin, elementos: jpfPassword, elementos: jtfUser, elementos: btnRegistrar);
```

En los demás controladores solo nos hace falta adjudicar las voces:

```
/*Mandamos a funcion hablar los elementos que queremos que hablen*/
    GestorHablar.adjudicarVoces(elementos: btnCrear, elementos: btnImagen, elementos: btnCancelar,
            elementos: tfContrasenia, elementos: tfCorreo, elementos: tfNombre, elementos: btnBorrarImagen);

    // Mensaje de bienvenida
    FuncionHablar.hablar(texto: "Bienvenido al registro de usuario");
```

En la imagen lo que hablaria seria todo lo que señalo en la siguiente imagen:



3.- Aplicar funcionalidad mediante elementos TouchPad o pantalla táctil.

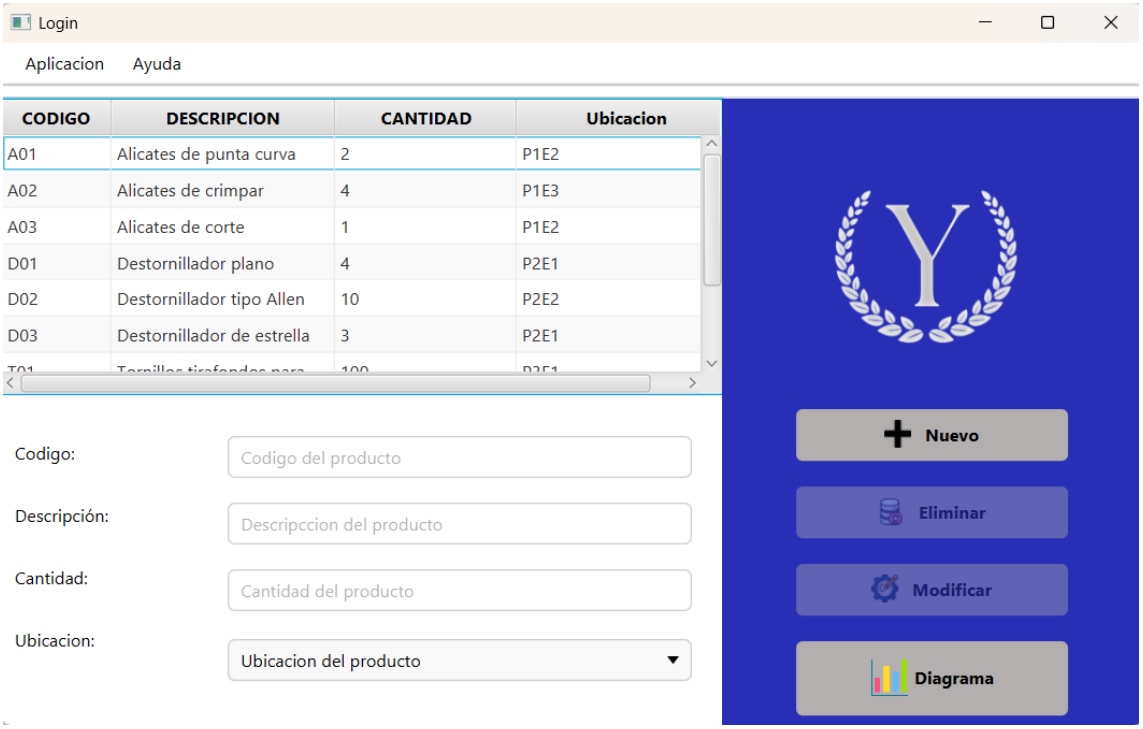
Para aplicar funciones táctiles es muy sencillo solo necesitaremos unos métodos para hacerlo para todas las escenas:

```
public class GestorTactil {
    private static double ratonX, ratonY;
    private static double nodoX, nodoY;
    private static double anguloInicial;

    public static void hacerInteractable(Node nodo) {

        /*Al pulsar el raton guardamos las coordinadas iniciales*/
        nodo.setOnMousePressed((MouseEvent event) -> {
            ratonX = event.getSceneX();
            ratonY = event.getSceneY();

            // Guardamos dónde estaba la imagen para moverla
            nodoX = nodo.getTranslateX();
            nodoY = nodo.getTranslateY();

            // Guardamos el ángulo actual para rotarla
            anguloInicial = nodo.getRotate();
        });

        // Al arrastrar el boton calculamos el movimiento manual
        nodo.setOnMouseDragged((MouseEvent event) -> {

            /*Click izquierdo*/
            if (event.isPrimaryButtonDown()) {
                // Nueva posición = Posición guardada + Distancia movida por el ratón
                nodo.setTranslateX(nodoX + (event.getSceneX() - ratonX));
                nodo.setTranslateY(nodoY + (event.getSceneY() - ratonY));

            } else if (event.isSecondaryButtonDown()) {
                /* Click Derecho*/
                // Calculamos cuánto se ha movido el ratón horizontalmente
                double deltaX = event.getSceneX() - ratonX;
                // Rotamos proporcionalmente al movimiento
                nodo.setRotate(anguloInicial + deltaX);
            }
        });

        // Zoom manual, es decir, la rueda del boton
        nodo.setOnScroll((ScrollEvent event) -> {
            double zoomFactor = 1.05;
            double deltaY = event.getDeltaY();

            if (deltaY < 0) {
                // Alejar
                zoomFactor = 0.95;
            }

            // Aplicamos la escala matemática manualmente
            nodo.setScaleX(nodo.getScaleX() * zoomFactor);
            nodo.setScaleY(nodo.getScaleY() * zoomFactor);

            event.consume();
        });
    }
```

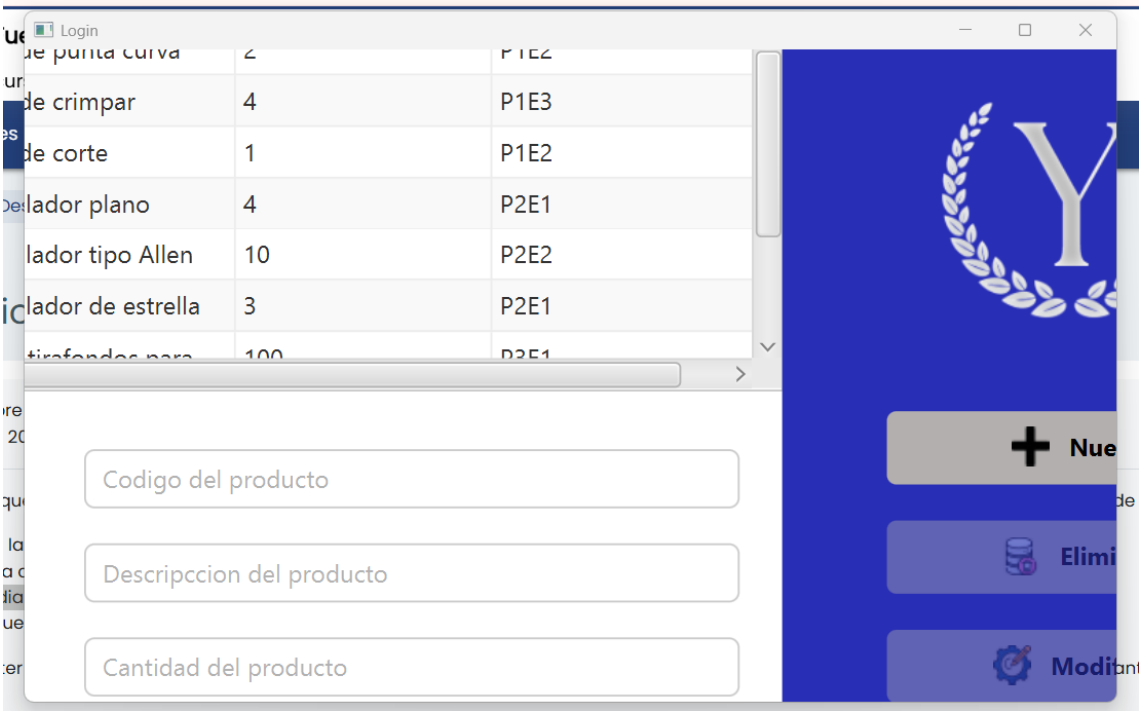
Y para poder usarlo en las escenas en los controladores solo tenemos que llamar al método hacer interactable y pasarle el nodo principal que esta todo metido:

```
GestorEstilos.cargarEstilos(nodo: anchorPane);
GestorTactil.hacerInteractable(nodo: anchorPane);
```

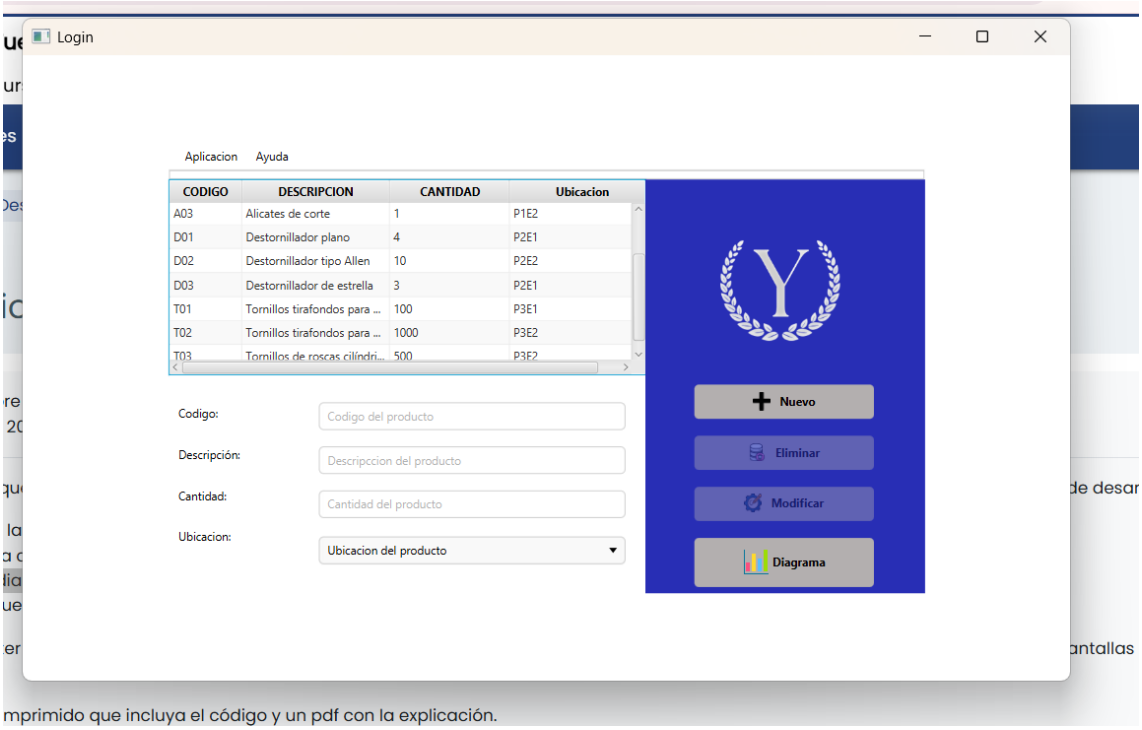
Y para probarlo en un portátil, cogemos y lo probamos aumentándolo:
Normal ->



Con zoom ->



Ahora alejándolo ->



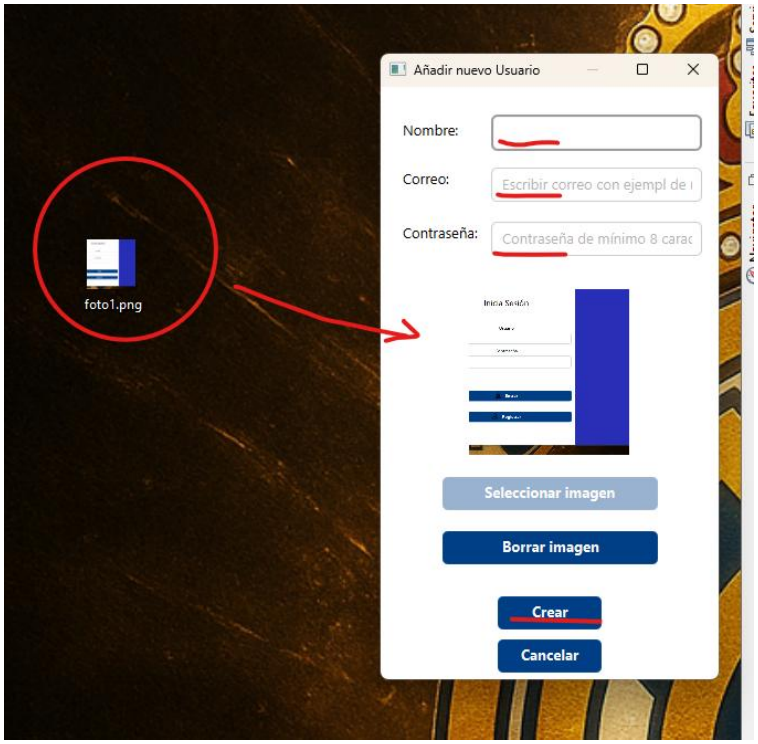
Como también puede servir para bajar o subir la pantalla

4.- Funcionalidad adicional que se ocurra para aplicaciones NUI de uso práctico

He añadido que pueda añadir a parte imágenes arrastrándolas no solo con la cámara. Para ello en el gestor de la imagen el metodo de drag and drop que necesitaremos para arrastrar y soltar:

```
public static void configurarDragAndDropImagen(ImageView imagenDestino, Button btnImagen, Button btnBorrar) {
    // Detectar arrastre
    imagenDestino.setOnDragOver((DragEvent event) -> {
        if (event.getDragboard().hasFiles()) {
            event.acceptTransferModes(transferModes: TransferMode.COPY_OR_MOVE);
        }
        event.consume();
    });

    // Soltar archivo
    imagenDestino.setOnDragDropped((DragEvent event) -> {
        boolean exito = false;
        if (event.getDragboard().hasFiles()) {
            List<File> archivos = event.getDragboard().getFiles();
            if (!archivos.isEmpty()) {
                File archivo = archivos.get(index: 0);
                try {
                    Image imagen = new Image(string: archivo.toURI().toString());
                    imagenDestino.setImage(value: imagen);

                    if (btnImagen != null) {
                        btnImagen.setDisable(value: true);   // Bloquear cámara
                    }
                    if (btnBorrar != null) {
                        btnBorrar.setDisable(value: false);  // Activar borrar
                    }
                    exito = true;
                } catch (Exception e) {
                    App.showAlert(title: "Error", message:"No se puede cargar la imagen ", type:Alert.AlertType.ERROR);
                }
            }
        }
        event.setDropCompleted(isTransferDone: exito);
        event.consume();
    });
}
```

Añadimos la imagen que vemos en el escritorio arrastrándola hacia donde pone no imagen:



Y como vemos se guarda perfectamente: