CÓDIGO. DECONTROLLER.JAVA.

```java
package sample;

import com.jfoenix.controls.*;
import javafx.application.Platform;
import javafx.beans.value.ChangeListener;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.embed.swing.SwingFXUtils;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.SnapshotParameters;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.stage.FileChooser;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Optional;
import java.util.ResourceBundle;


public class DE_Controller_from implements Initializable {


    //private String texto = "";
    Alert dialogoAlerta = new Alert(Alert.AlertType.INFORMATION);

    @FXML
    private Button siguiente;
    @FXML
    private Button btnBuscarImagen;
    @FXML
    private Button contYGuardar;

    @FXML
    private ImageView iviImage;
    @FXML
    private ImageView imagenDetalle;


    @FXML
    private AnchorPane helpPanel;
    @FXML
    private AnchorPane pedidoPanel;
    @FXML
    private AnchorPane infoPedidoPanel;
    @FXML
    private AnchorPane imagenPanel;
```

```
61      @FXML
62      private AnchorPane creditCardPanel;
63      @FXML
64      private AnchorPane efectivoPanel;
65      @FXML
66      private AnchorPane detallePanel;
67      @FXML
68      private AnchorPane confirmacionPanel;
69      @FXML
70      private AnchorPane graciasPanel;
71      @FXML
72      private AnchorPane detallePagoEfectivoPanel;
73      @FXML
74      private AnchorPane detallePagoTarjetaPanel;
75
76
77      @FXML
78      private JFXTextArea textPedido;
79      @FXML
80      private JFXTextArea textPedidoFinal;
81      @FXML
82      private JFXTextArea direccionOrigen;
83      @FXML
84      private JFXTextArea direccionDestino;
85      @FXML
86      private JFXTextArea textPedidoFinal1;
87      @FXML
88      private JFXTextArea direccionOrigen1;
89      @FXML
90      private JFXTextArea direccionDestino1;
```

```
92      @FXML
93      private TextField fechaVenc;
94      @FXML
95      private TextField nombreTitular;
96      @FXML
97      private TextField nroTarjeta;
98      @FXML
99      private TextField codSeguridad;
100     @FXML
101     private TextField cantEfective;
102     @FXML
103     private TextField vuelto;
104     @FXML
105     private TextField precioProducto;
106
107     @FXML
108     private TextField fechaVenc1;
109     @FXML
110     private TextField nombreTitular1;
111     @FXML
112     private TextField nroTarjeta1;
113     @FXML
114     private TextField codSeguridad1;
115     @FXML
116     private TextField cantEfective1;
117     @FXML
118     private TextField vuelto1;
119     @FXML
```

```
120     private TextField precioProducto1;
121     @FXML
        private JFXTextField metodoDePago1;
123
124     @FXML
125     private ComboBox<String> comboMetodoPago;
126
127     @FXML
128     private RadioButton cantidadEfectivo;
129     @FXML
130     private RadioButton noLoSe;
131     @FXML
132     private RadioButton cantidadEfectivo1;
133     @FXML
134     private RadioButton noLoSe1;
135
136     @FXML
        private JFXToggleButton antesPosible;
138     @FXML
        private JFXToggleButton antesPosible1;
140
141     @FXML
        private JFXTimePicker horaEntrega;
143     @FXML
        private JFXDatePicker fechaEntrega;
145     @FXML
        private JFXTimePicker horaEntrega1;
147     @FXML
        private JFXDatePicker fechaEntrega1;
149
```

```
150
151     ObservableList<String> comboId =
152     FXCollections.observableArrayList("Pago en efectivo", "Pago con tarjeta");
153
154     @Override
        public void initialize(URL location, ResourceBundle resources) {
156     pedidoPanel.setVisible(true);
157     infoPedidoPanel.setVisible(false);
158     creditCardPanel.setVisible(false);
159     efectivoPanel.setVisible(false);
160     helpPanel.setVisible(false);
161     comboMetodoPago.setItems(comboId);
162     ToggleGroup grupo = new ToggleGroup();
163     noLoSe.setToggleGroup(grupo);
164     cantidadEfectivo.setToggleGroup(grupo);
165     cantEfective.setText("0");
166     verificarTextField(codSeguridad);
167     verificarTextField(nroTarjeta);
168     verificarTextField(precioProducto);
169     verificarTextField(cantEfective);
170
171     }
172
173
174     public void showDate() {
175     Date d = new Date();
176     SimpleDateFormat sf = new SimpleDateFormat("MM/YY");
177     fechaVenc.setText(sf.format(d));
```

```
178     }
179     //Metodo para salir de la aplicacion
180     public void salirAplicacion(MouseEvent event) {
181     confirmarSalida("¿Realmente quiere salir?");
182
183
184     }
185     //Metodo para continuar al detalle del pedido
186     public void continuarDetallePedido(MouseEvent event) {
187     String texto = textPedido.getText();
188
189     if (texto.length() > 2) { //Valida que se ingrese algun pedido
190     //  siguiente.setDisable(true);
191     infoPedidoPanel.setVisible(true);
192     pedidoPanel.setVisible(false);
193     textPedidoFinal.setText(textPedido.getText());
194     } else mensajesDeError("Error", "Debe ingresar un pedido");
195     }
196     //Metodo para entrar al panel que carga imagenes
197     public void buscarImagen(MouseEvent event) {
198     imagenPanel.setVisible(true);
199
200     }
201     //Metodo para buscar y setear imagenes
202     public void setImagen() {
203
204     btnBuscarImagen.setOnAction(event -> {
205     FileChooser fileChooser = new FileChooser();
206     fileChooser.setTitle("Buscar Imagen");
207
208     // Agregar filtros para facilitar la busqueda
209     fileChooser.getExtensionFilters().addAll(
210     new FileChooser.ExtensionFilter("All Images", "*.*"),
211     new FileChooser.ExtensionFilter("JPG", "*.jpg"),
212     new FileChooser.ExtensionFilter("PNG", "*.png")
213     );
214
215     // Obtener la imagen seleccionada
216     File imgFile = fileChooser.showOpenDialog(null);
217
218     // Mostar la imagen
219     if (imgFile != null) {
220     Image image = new Image("file:" + imgFile.getAbsolutePath());
221     iviImage.setImage(image);
222     }
223     });
224
225     }
226     //Metodo para confirmar y volver al panel anterior
227     public void clickHecho(MouseEvent event) {
228     imagenPanel.setVisible(false);
229     }
230     //Metodo para ingresar al panel de ayuda
231     public void clickHelp(MouseEvent event) {
232     helpPanel.setVisible(true);
233     }
234     //Metodo para volver al panel de pedido
235     public void clickEntendido(MouseEvent event) {
236     helpPanel.setVisible(false);
237     pedidoPanel.setVisible(true);
238     }
```

```
240  public void clickConfirmar(MouseEvent event) {
241  if (textPedidoFinal.getLength() > 2 & direccionOrigen.getLength() > 2
242  & direccionDestino.getLength() > 2 & horaEntrega.getEditor().getLength() > 2
243  & fechaEntrega.getEditor().getLength() > 2 & precioProducto.getLength() > 2 & comboMetodoPago.getSelectionModel().getSelectedIndex() > -1)
244
245  confirmarPedido();
246  if (comboMetodoPago.getSelectionModel().getSelectedIndex() == 0) {
247  detallePagoEfectivoPanel.setVisible(true);
248  detallePagoTarjetaPanel.setVisible(false);
249  detallePanel.setVisible(true);
     }else  if (comboMetodoPago.getSelectionModel().getSelectedItem() == "Pago con tarjeta") {
251  detallePagoEfectivoPanel.setVisible(false);
252  detallePagoTarjetaPanel.setVisible(true);
253  detallePanel.setVisible(true);
254  }
255
256  } else if (textPedidoFinal.getLength() > 2 & direccionOrigen.getLength() > 2
257  & direccionDestino.getLength() > 2 & antesPosible.isSelected() == true & precioProducto.getLength() > 2 & comboMetodoPago.getSelectionModel
258  confirmarPedido();
259  if (comboMetodoPago.getSelectionModel().getSelectedIndex() == 0) {
260  detallePagoEfectivoPanel.setVisible(true);
261  detallePagoTarjetaPanel.setVisible(false);
262  detallePanel.setVisible(true);
     }else  if (comboMetodoPago.getSelectionModel().getSelectedItem() == "Pago con tarjeta") {
264  detallePagoEfectivoPanel.setVisible(false);
265  detallePagoTarjetaPanel.setVisible(true);
266  detallePanel.setVisible(true);
```

```
266      detallePanel.setVisible(true);
267      }
268
269      } else {
270
271      mensajesDeError("Error", "Asegurese de llenar todos los campos.");
272      }
273      }
274
275      //Metodo para salir del efectivo sin guardar
276      public void atrasEfectivo(MouseEvent event) {
277      confirmarSiguiente("¿Seguro de volver atras?",efectivoPanel,infoPedidoPanel);
278
279      //  efectivoPanel.setVisible(false);
280      if (efectivoPanel.isVisible()==false) {
281      comboMetodoPago.setValue(null);
282      vuelto.clear();
283      cantEfective.clear();
284      noLoSe.setSelected(false);
285      cantidadEfectivo.setSelected(false);
286      }
287
288      }
289
290      public void atrasTarjeta(MouseEvent event) {
291      confirmarSiguiente("¿Seguro de volver atras?",creditCardPanel,infoPedidoPanel);
292      // creditCardPanel.setVisible(false);
293      if (creditCardPanel.isVisible()== false) {
294
```

```
294
295     comboMetodoPago.getSelectionModel().clearSelection();
296     nombreTitular.clear();
297     nroTarjeta.clear();
298     codSeguridad.clear();
299     fechaVenc.clear();
300     }
301
302     }
303
304     public void atrasPedido(MouseEvent event) {
305     pedidoPanel.setVisible(true);
306     infoPedidoPanel.setVisible(false);
307
308     }
309
310     @FXML//Metodo para seleccionar elmetodo de pago y ingresar al respectivo panel (Tarjeta o efectivo)
311     public void ingresarMetodoPago(MouseEvent event) {
312
313     String tarjeta = "Pago con tarjeta";
314     String efectivo = "Pago en efectivo";
315     if (precioProducto.getLength() == 0) {
316     mensajesDeError("Error","Ingrese un precio aproximado");
317     comboMetodoPago.setValue(null);
318
319     precioProducto.requestFocus();
320     } else {comboMetodoPago.setOnAction(event1 -> {
321     String comboValue = comboMetodoPago.getSelectionModel().getSelectedItem();
322     if (precioProducto.getLength() > 1) {
323     if (comboValue.equals(efectivo)) {
```

```
324     efectivoPanel.setVisible(true);
325     cantEfective.setDisable(true);
326     }
327     if (comboValue.equals(tarjeta)) creditCardPanel.setVisible(true);
328     }
329     });
330     /**  String comboValue = comboMetodoPago.getSelectionModel().getSelectedItem();
331     if (precioProducto.getLength() > 1) {
332     if (comboValue.equals(efectivo)) {
333     efectivoPanel.setVisible(true);
334     cantEfective.setDisable(true);
335     }
336     if (comboValue.equals(tarjeta)) creditCardPanel.setVisible(true);
337     }*/
338     }/**else if (precioProducto.getLength() == 0) { //Valida que haya ingresado el precio del producto
339     mensajesDeError("Error","Ingrese un precio aproximado");
340     comboMetodoPago.setValue(null);
341
342     precioProducto.requestFocus();
343     }*/
344     }
345
346
347     public void continuarTarjeta(MouseEvent event) throws IOException {
348     int MIN_NOMBRE_TITULAR = 5;
349     int CANT_DIGITOS_TARJETA = 16;
350     int MIN_COD_SEGURIDAD = 3;
351     int MAX_COD_SEGURIDAD=4;
352     System.out.println(String.valueOf(fechaVenc.getLength()));
353     if (nombreTitular.getLength() < MIN_NOMBRE_TITULAR){
```

```
355   nombreTitular.requestFocus();
356   }else if (nroTarjeta.getLength()!=CANT_DIGITOS_TARJETA){
357   mensajesDeError("Error","El número de tarjeta esta mal");
358   nroTarjeta.requestFocus();
359   }else if (fechaVenc.getLength() != 5 ){
360   mensajesDeError("Error","La fecha de vencimiento esta mal");
361   fechaVenc.requestFocus();
362   }else if (codSeguridad.getLength() != MIN_COD_SEGURIDAD & codSeguridad.getLength() != MAX_COD_SEGURIDAD){
363   mensajesDeError("Error","El código de seguridad esta mal");
364   codSeguridad.requestFocus();
365   }else {
366   confirmarSiguiente("¿Esta seguro que los datos ingresados son correctos?",creditCardPanel,pedidoPanel);
367   //  creditCardPanel.setVisible(false);
368   //pedidoPanel.setVisible(true);
369   }
370
371
372   }
373
374   public void continuarEfectivoClick(MouseEvent event) {
375   int PRECIO_DELIVERY = 40;
376   int auxTotal = Integer.parseInt(precioProducto.getText()) + PRECIO_DELIVERY;
377   if (noLoSe.isSelected() == true) {
378   pedidoPanel.setVisible(true);
379   efectivoPanel.setVisible(false);
380   }
381
382   if (cantidadEfectivo.isSelected() == true & auxTotal <= Integer.parseInt(cantEfective.getText())) {
383   if (vuelto.getLength() == 0){
384   int auxVuelto = Integer.parseInt(cantEfective.getText()) - (PRECIO_DELIVERY + Integer.parseInt(precioProducto.getText()));
385   vuelto.setText(String.valueOf(auxVuelto));
386   }else {
387   pedidoPanel.setVisible(true);
388   efectivoPanel.setVisible(false);
389   }
390   } else if (cantidadEfectivo.isSelected() == true & Integer.parseInt(cantEfective.getText()) < auxTotal) {
391   mensajesDeError("Error", "El monto debe ser superior a: $" + auxTotal);
392   } else
393   if ((cantidadEfectivo.isSelected() == false) & noLoSe.isSelected() == false) {
394   mensajesDeError("Error", "Asegurese de seleccionar una opción.");
395   }
396   }
397
398   //Metodo que habilita el textfield cantidad de efectivo
399   public void habilitarCantidadEfectivo(MouseEvent event) {
400   if (cantidadEfectivo.isSelected() == true) {
401   cantEfective.setDisable(false);
402   }
403   }
404
405   @FXML//Metodo para calcular el vuelto
406   public void calcularVuelto(MouseEvent event) {
407   int PRECIO_DELIVERY = 40;
408   if (cantEfective.getLength()>2) {
409   int aux = Integer.parseInt(cantEfective.getText()) - (PRECIO_DELIVERY + Integer.parseInt(precioProducto.getText()));
410   vuelto.setText(String.valueOf(aux));
411   }
412
413   }
```

```
413     }
414     //Metodo que selecciona la opcion "No lo se" y limpia los demas textfield
415     public void seleccionarNoLoSe(MouseEvent event) {
416     if (noLoSe.isSelected() == true) {
417     cantEfective.clear();
418     vuelto.clear();
419     cantEfective.setDisable(true);
420     }
421
422     }
423     //Metodo que habilita la opcion lo antes posible, y bloquea fecha y hora de entrega
424     public void seleccionarLoAntesPosible(MouseEvent event) {
425     if (antesPosible.isSelected() == false) {
426     fechaEntrega.setDisable(false);
427     horaEntrega.setDisable(false);
428
429     }
430     if (antesPosible.isSelected() == true) {
431     horaEntrega.getEditor().setText("");
432     fechaEntrega.getEditor().setText("");
433     fechaEntrega.setDisable(true);
434     horaEntrega.setDisable(true);
435     }
436     }
437
438     public void mensajesDeError(String titulo, String contenido) {
439     dialogoAlerta.setTitle(titulo);
440     dialogoAlerta.setHeaderText(null);
441     dialogoAlerta.setContentText(contenido);
442     dialogoAlerta.showAndWait();
```

```
442     dialogoAlerta.showAndWait();
443
444     }
445
446
447     private void verificarTextField(TextField textField) {
448     textField.textProperty().addListener(agregarListenerAlTextField(textField));
449     }
450     //Metodo que asegura que solo se ingrese digitos
451     private ChangeListener<String> agregarListenerAlTextField(TextField aTextField) {
452     return (observable, oldValue, newValue) -> {
453     if (!newValue.matches("\\d*")/**&!newValue.matches("'\\b' ")*/){
454     aTextField.setText(newValue.replaceAll("[^\\d]", ""));
455     }
456     };
457     }
458
459     //
460     public void confirmarPedido() {
461     imagenDetalle.setImage(iviImage.getImage());
462     textPedidoFinal1.setText(textPedidoFinal.getText());
463     direccionOrigen1.setText(direccionOrigen.getText());
464     direccionDestino1.setText(direccionDestino.getText());
465     fechaVenc1.setText(fechaVenc.getText());
466     nombreTitular1.setText(nombreTitular.getText());
467     nroTarjeta1.setText(nroTarjeta.getText());
468     codSeguridad1.setText(codSeguridad.getText());
469     cantEfective1.setText(cantEfective.getText());
470     vuelto1.setText(vuelto.getText());
471     precioProducto1.setText(precioProducto.getText());
```

```
473   if (cantidadEfectivo.isSelected() == true) {
474   cantidadEfectivo1.setSelected(true);
475   }
476   if (noLoSe.isSelected() == true) {
477   noLoSe1.setSelected(true);
478   }
479   if (antesPosible.isSelected() == true) {
480   antesPosible1.setSelected(true);
481   } else if   (antesPosible.isSelected()==false) {
482
483   horaEntrega1.setValue(horaEntrega.getValue());
484   fechaEntrega1.setValue(fechaEntrega.getValue());
485
486   }
487   }
488   public void confirmarSalida(String contenido) {
489   Alert dialogo = new Alert(Alert.AlertType.CONFIRMATION);
490   dialogo.setTitle("Confirmación");
491   dialogo.setHeaderText(null);
492   dialogo.setContentText(contenido);
493   Optional<ButtonType>result=dialogo.showAndWait();
494   if (result.get() == ButtonType.OK){
495   Platform.exit();
496   System.exit(0);
497   }
498   }
499
500   public  void confirmarSiguiente(String contenido,AnchorPane origen, AnchorPane destino){
501   Alert dialogo = new Alert(Alert.AlertType.CONFIRMATION);
502   dialogo.setTitle("Confirmación");
```

```
500   public  void confirmarSiguiente(String contenido,AnchorPane origen, AnchorPane destino){
501   Alert dialogo = new Alert(Alert.AlertType.CONFIRMATION);
502   dialogo.setTitle("Confirmación");
503   dialogo.setHeaderText(null);
504   dialogo.setContentText(contenido);
505   Optional<ButtonType>result=dialogo.showAndWait();
506   if (result.get() == ButtonType.OK){
507   destino.setVisible(true);
508   origen.setVisible(false);
509   }
510   }
511
512
513   public void imprimirPedido() {
514   contYGuardar.setOnAction(event -> {
515   TextInputDialog dialog = new TextInputDialog("G:\\");
516   dialog.setTitle("Guardar tu pedido");
517
518   dialog.setContentText("Por favor, ingrese una ruta");
519   Optional<String> result = dialog.showAndWait();
520
521   WritableImage image = confirmacionPanel.snapshot(new SnapshotParameters(), null);
522   int aux1 = (new Date().getHours());
523   int aux2 = new Date().getMinutes();
524   int aux3 = new Date().getSeconds();
525   String date = String.valueOf(aux1);
526   date = date + String.valueOf(aux2);
527   date = date + String.valueOf(aux3);
528
529   System.out.println(date);
```

```
523     int aux2 = new Date().getMinutes();
524     int aux3 = new Date().getSeconds();
525     String date = String.valueOf(aux1);
526     date = date + String.valueOf(aux2);
527     date = date + String.valueOf(aux3);
528
529     System.out.println(date);
530
531     File file = new File(result.get() +  date + ".png");
532
533     try {
534     ImageIO.write(SwingFXUtils.fromFXImage(image, null), "png", file);
535     } catch (IOException e) {
          e.printStackTrace();
537     }
538     graciasPanel.setVisible(true);
539     detallePanel.setVisible(false);
540     });
541   }
542   public void atrasDetalle(){
543   confirmarSiguiente("Esta seguro de volver atras",detallePanel,infoPedidoPanel);
544   }
545   }
546
547
```