Primera Aplicación. Parte 3

En esta última parte vamos a implementar la lógica de la aplicación. Lo que se pretende conseguir con la aplicación es un medio para insertar/modificar/eliminar registros en la tabla que hemos creado en la parte anterior. Una tabla de usuarios con los campos:

- DNI
- Nombre
- Apellidos
- E-mail
- Password
- Notas (cualquier tipo de anotación que consideremos oportuna sobre esa cuenta)

Descripción y requisitos

El funcionamiento será el siguiente; disponemos de tres botones de control para interaccionar con la BD y uno más para limpiar el formulario. Al lado de este último botón disponemos de una zona para presentar mensajes relativos a la última operación realizada. Para estos mensajes utilizamos un control de tipo Label.

Todos los botones de BD estarán deshabilitados al comienzo y se habilitarán/deshabilitarán en función del estado del formulario.

Mientras no se introduzca un valor para el DNI (clave primaria) los botones seguirán inactivos. En cuanto se haya introducido un valor en DNI, se comprobará si este usuario ya existía en la BD. De existir, se rellenarán automáticamente el resto de los campos, y el de DNI quedará bloqueado, dándose la opción de Eliminar el registro o modificar uno o varios del resto de campos y actualizar el registro. El campo DNI quedará bloqueado hasta que se pulse alguno de los botones Actualizar/Eliminar/Borrar Datos (el de Insertar estará desactivado). En caso de que el DNI no se corresponda con ninguno de los existentes en la BD, se activará el botón de Insertar y quedarán desactivados los de Actualizar/Eliminar. En caso de que modifiquemos el valor de DNI, se volverá a comprobar su existencia en la BD, actualizando, si procede, la información visualizada.

Hay que tener en cuenta, que, aunque no introduzcamos valores en algún campo (salvo el DNI), las operaciones de Inserción/Actualización funcionarán, pese a que hayamos definido los campos en la BD como NOT NULL (salvo el de notas). Esto es debido a que, aunque en un campo de texto no aparezca ningún dato, su valor no es **null** sino una cadena vacía **“”**.

La pulsación de cualquiera de los Botones llevará asociado el borrado de los datos presentes en el formulario, además de la función que tengan asignada.

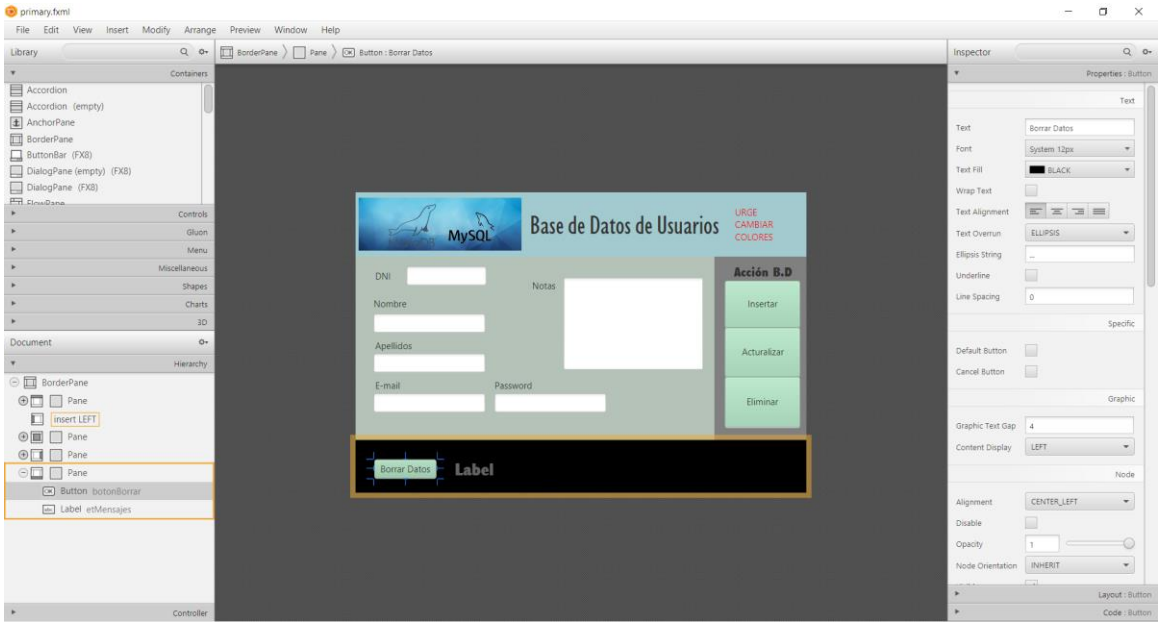
Event Listeners y fx:id

Para implementar este comportamiento nos basaremos en la programación orientada a eventos. Básicamente consiste en implementar métodos que “observan” o “escuchan” (se les suele denominar **Listeners**) en segundo plano las posibles acciones que se producen sobre los controles de la interfaz gráfica. Por ejemplo, que un botón has sido pulsado, que se ha chequeado una casilla, que un campo ha perdido el foco, etc.

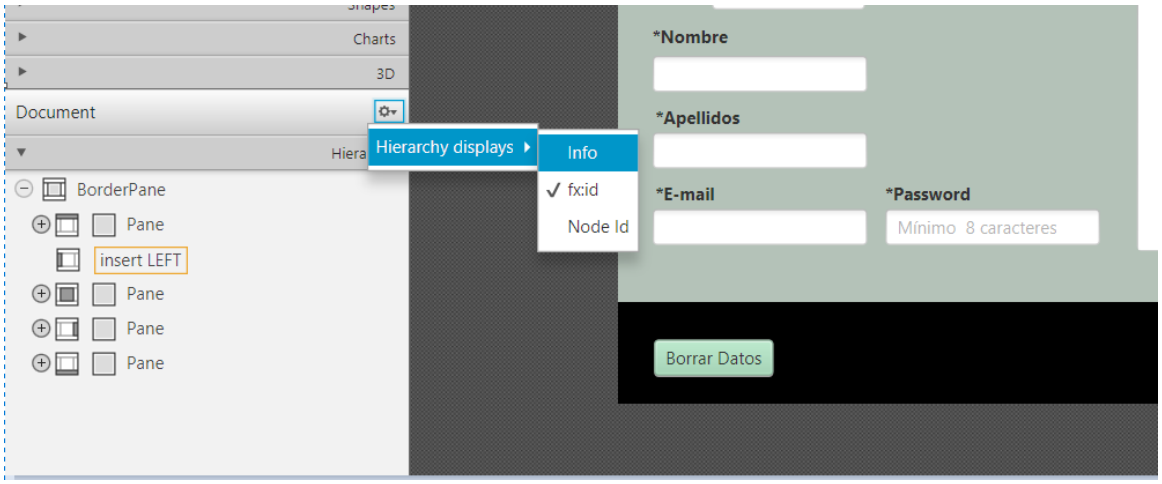
Para ello lo primero que debemos hacer es asignar un nombre a cada uno de los controles, para poder referirnos a ellos desde el código Java. Se suelen nombrar utilizando identificadores con un prefijo que indica el tipo de control, por ejemplo, para el botón de insertar podría ser: **buttonInsertar.** Estos nombres son lo que se denomina en JavaFX **fx:id**

Otro aspecto importante es organizarlos correctamente de modo que vayan obteniendo el foco de modo ordenado. En principio el orden es aquel en que los hemos creado, y este puede no ser el correcto. Pongamos que creamos primero el TextField para el DNI, después el de los Apellidos, y después el del Nombre. Aunque los movamos por la interfaz de modo que estén el DNI arriba, el nombre a continuación y los apellidos al final, cuando lancemos el programa y nos movamos entre campos con el teclado, el foco irá de DNI a Apellidos y después a Nombre.

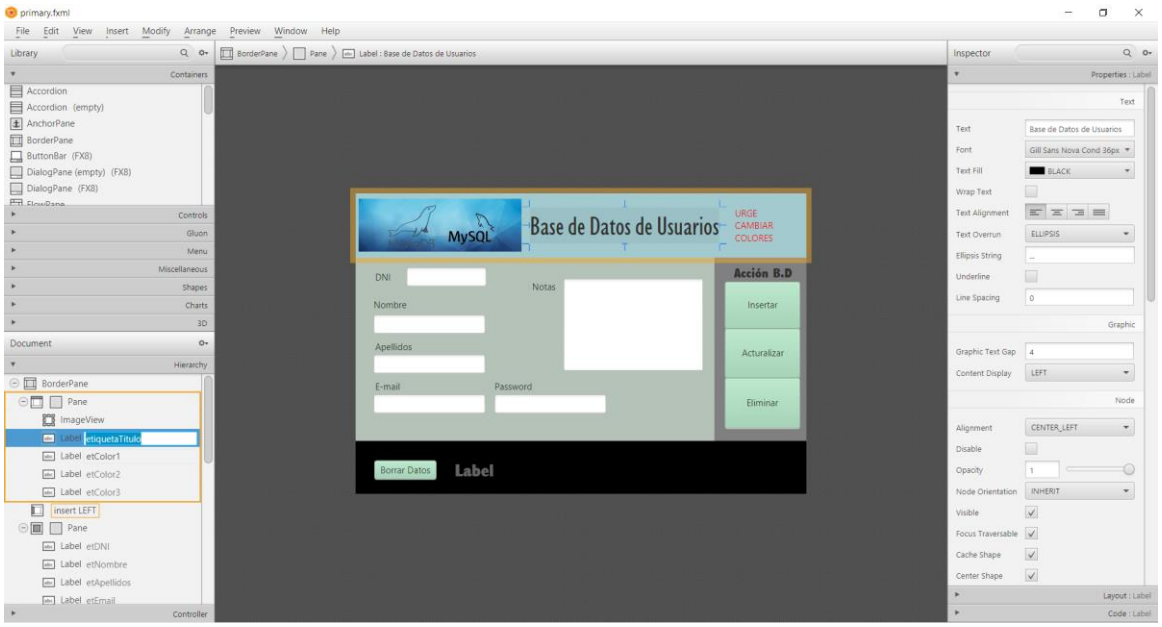
Vamos pues a darle nombre a los controles y simultáneamente comprobaremos si su orden es el correcto. Abrimos el Scene Builder



En la sección Document, pulsamos en el botón que está a su derecha, que nos permite elegir cual será la información a visualizar en Hirarchy, y escogemos fx:id



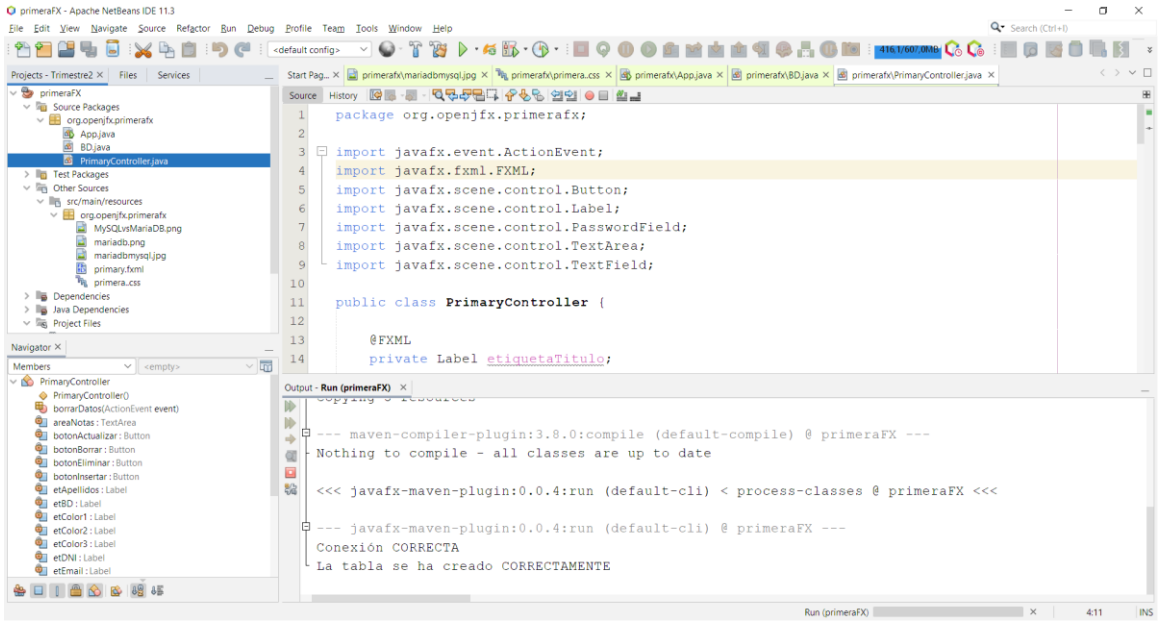
Desplegamos todos los paneles (parte inferior izquierda) pulsando sobre el signo + en cada uno de ellos. Para darle un nombre a un control, hacemos doble click a su lado, y ya nos permite escribirlo.



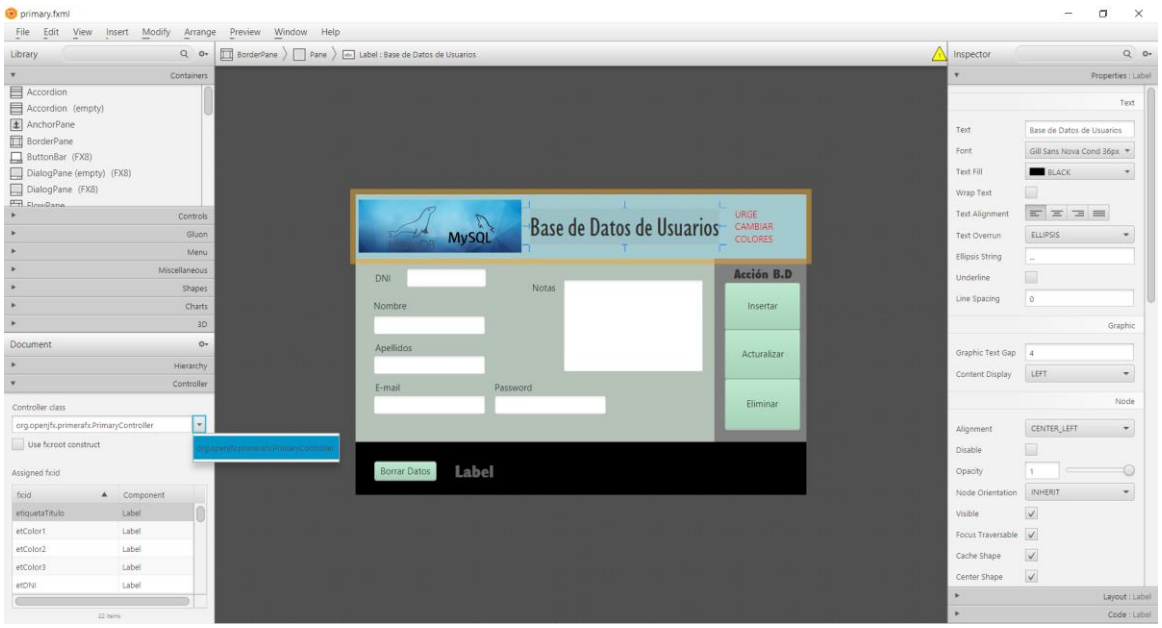
Vamos nombrado todos (las etiquetas también, así podremos controlarlas igualmente desde Java) y de paso nos fijamos en el orden vertical en que están. Este sería el orden en que van a coger el foco. Si alguno está donde no le corresponde pulsamos sobre el mismo y lo arrastramos a su posición correcta.

Controller

El fichero donde vamos a escribir el código a ejecutar por los eventos será el fichero **Controlador.java** (o el nombre que le hayamos dado)

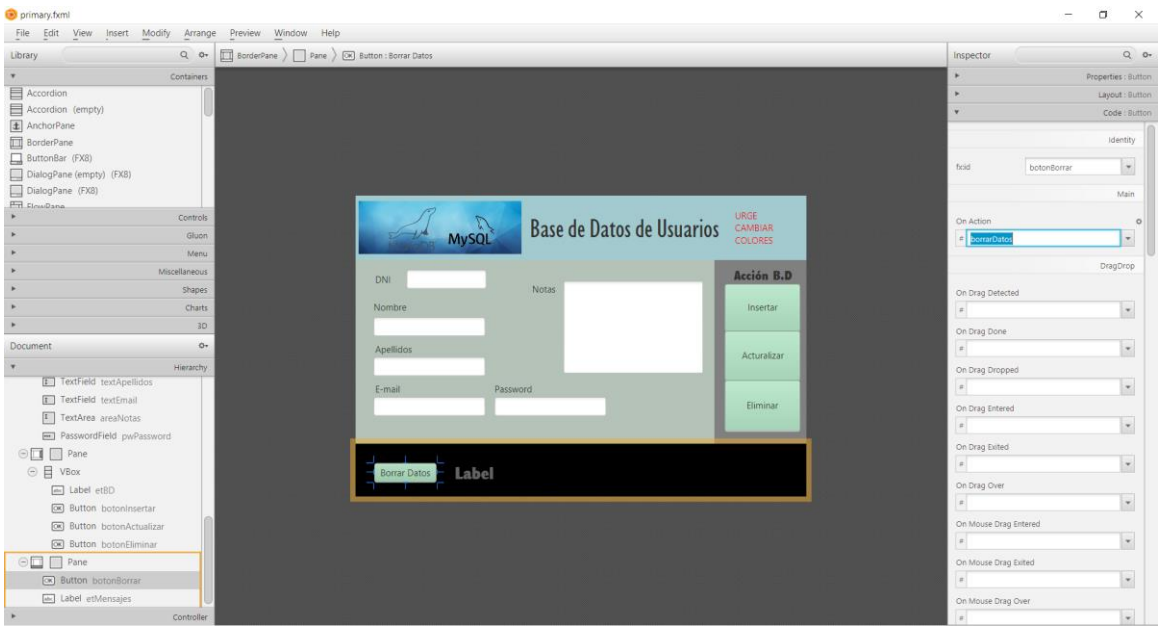


Para que el código generado por Scene Builder se ajuste correctamente a lo presente en NetBeans, abrimos la última sección de la parte izquierda, denominada **Controller**, y ahí, en la entrada Controller Class pulsamos la flecha hacia abajo y seleccionamos el fichero **Controlador.java**

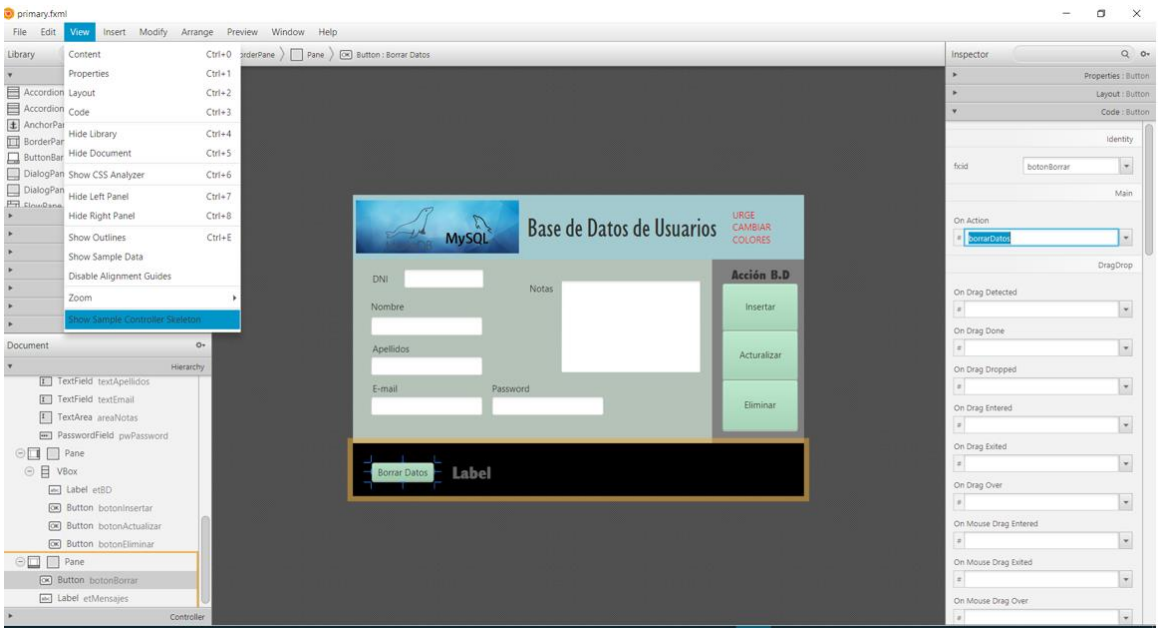


Elección de eventos y asignación de código

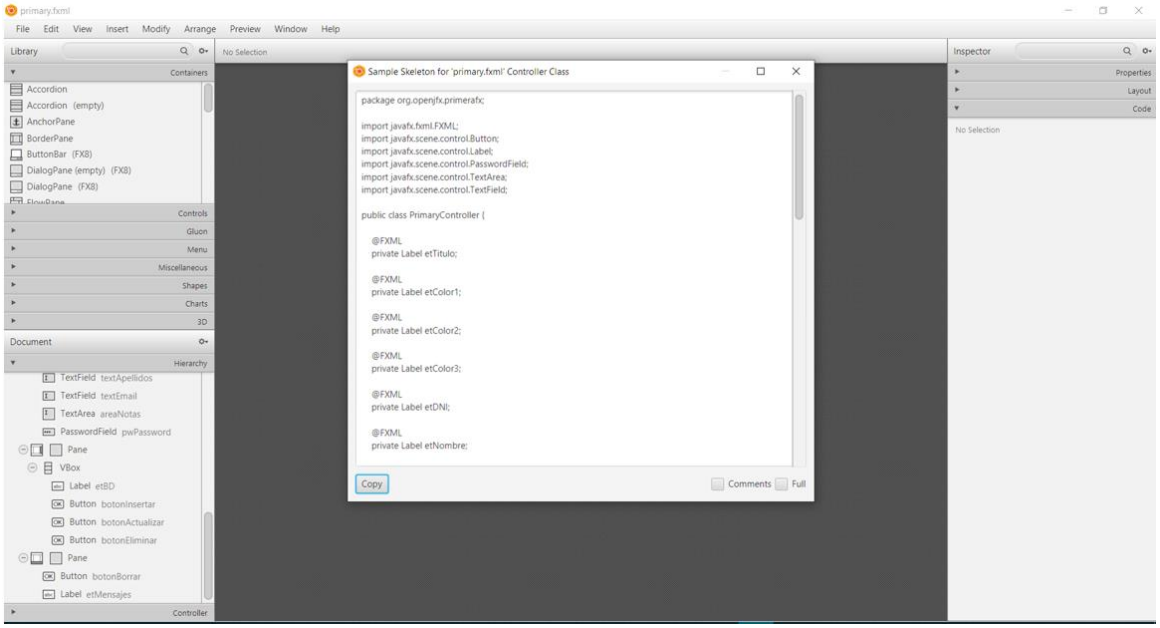
Vamos a comenzar por dotar de funcionalidad al botón Borrar Datos, que es el más sencillo. Seleccionamos el botón en el diseñador, y en la parte derecha abrimos la última sección, llamada **Code** rellenando el campo On Action con el nombre que queremos que tenga el método que se ejecutará al pulsar este botón.



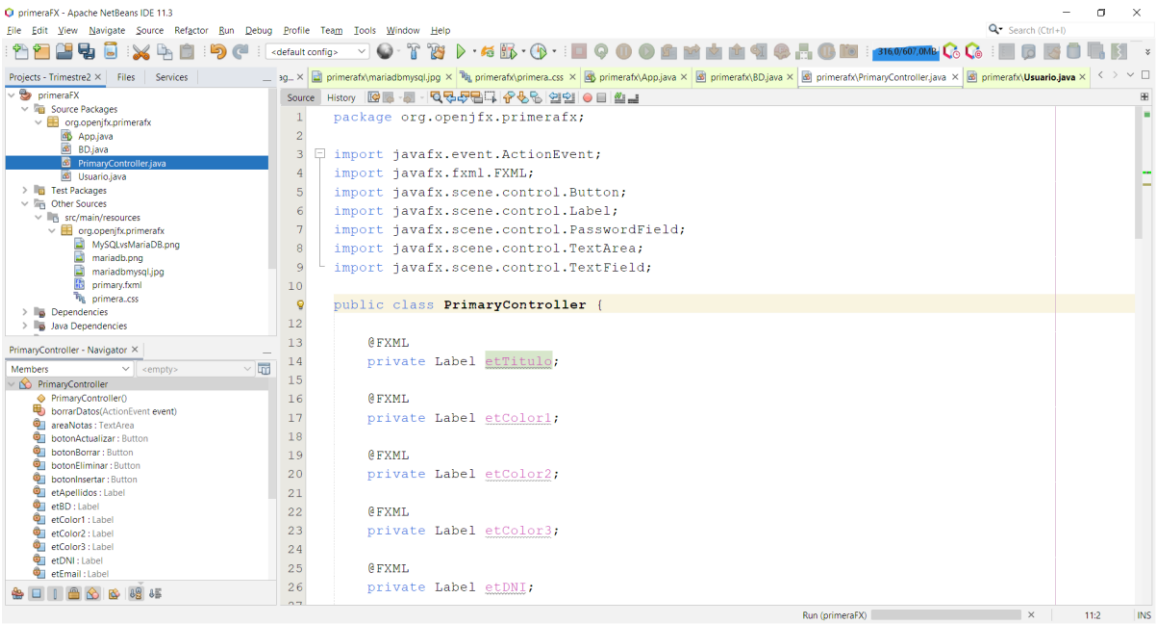
Pulsando en View->Show Sampler Controller Skeleton:



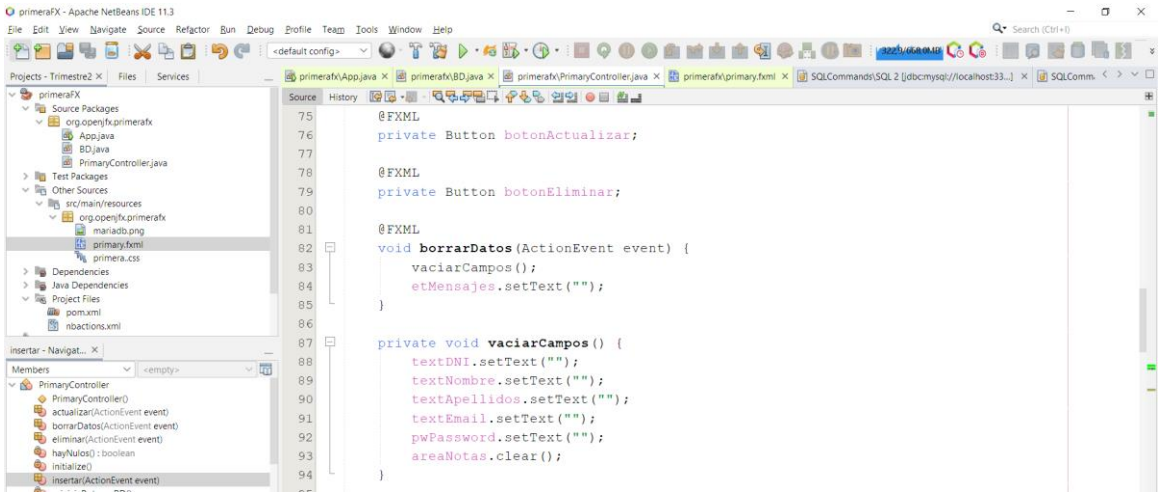
Se abre una ventana con el “esqueleto” del código necesario para nuestro controlador. Como aun no teníamos escrito ningún código específico en el mismo, podemos copiar todo el código presentado y sustituir el que teníamos en **Controlador.java** por este**.**



Vemos el resultado. Cada uno de los controles creados aparece como un atributo de la clase Controladorer (etiquetas, botones, campos de texto,) Es muy importante la anotación **@FXML** que precede tanto a las variables como a los métodos, ya que es la actúa como ligazón entre ellos y la interfaz gráfica definida en el fichero **principal.fxml** (que se actualiza automáticamente cada vez que grabamos lo diseñado en Scene Builder y viceversa).



Al final del código vemos que aparece un método vacío, denominado borrarDatos (o el nombre que hubiésemos introducido en la entrada On Action en Scene Builder). Este método se ejecutará cada vez que pulsemos el botón Borrar Datos. Su funcionalidad es borrar el contenido de todos los campos, para lo cual rellenaremos el método con el siguiente código:



```
@FXML
private Button botonActualizar;

@FXML
private Button botonEliminar;

@FXML
void borrarDatos(ActionEvent event) {
    vaciarCampos();
    etMensajes.setText("");
}

private void vaciarCampos() {
    textDNI.setText("");
    textNombre.setText("");
    textApellidos.setText("");
    textEmail.setText("");
    pwPassword.setText("");
    areaNotas.clear();
}
```

Que lo que hace es fijar el texto asociado a cada uno de los controles de introducción de información, como una cadena vacía: “”

Ejecutamos el programa y probamos su funcionamiento. Rellenamos campos y pulsamos el botón

Y este es el resultado:

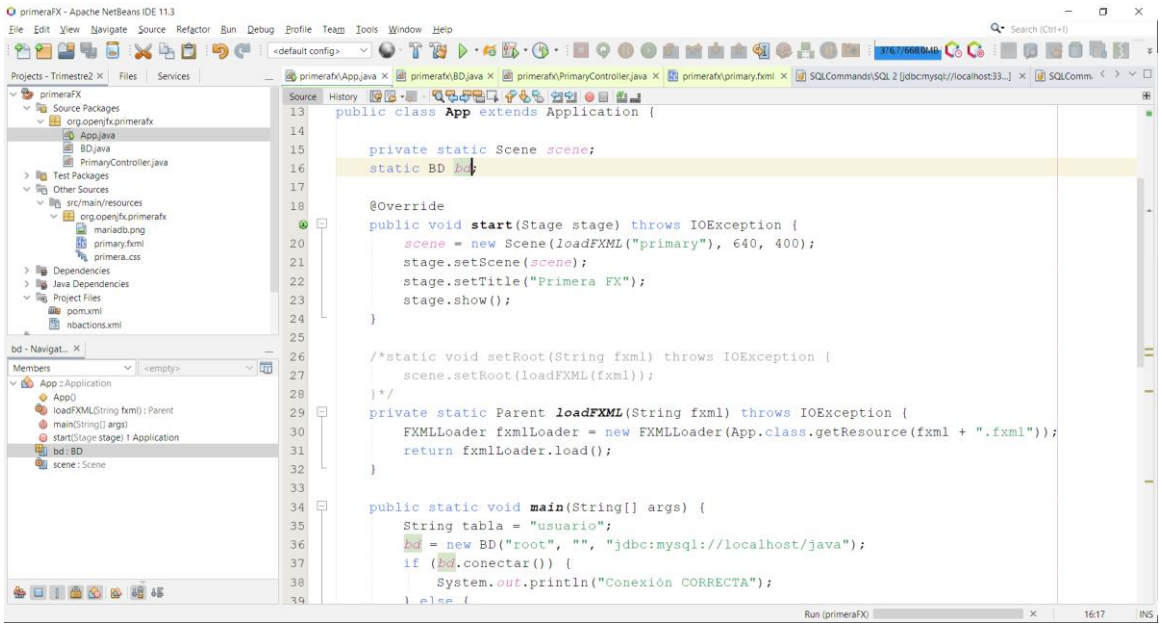
Botones de BD

Como ya sabemos cómo definir un método asociado a un evento On Action de un botón (cuando se pulsa sobre el mismo), vamos a crear los métodos que utilizaremos para la Inserción la Actualización y la Eliminación de registros. Realizamos el proceso inverso al caso anterior. Para el botón Borrar Datos, le asignamos un nombre al método en el FXML, asociado al evento On Action, y a partir de eso generamos el "esqueleto" de código enlazado en el controlador. Ahora vamos primero a crear los métodos en el controlador y enlazarlos después al diseño (FXML directamente o por medio del Scene Builder).

Creamos tres métodos con la misma firma que el de Borrar Datos, esto es, métodos void y que reciben un objeto de tipo ActionEvent. Introducimos en cada uno de ellos la llamada correspondiente al método necesario de BD, que tenemos creados en el fichero **BD.java**. En cada uno de ellos añadiremos además un borrado de los campos y una asignación a la etiqueta de mensajes de pie de nuestro formulario. Desactivamos los botones BD, y, en caso de estar desactivado, activamos de nuevo el campo textDNI.

Al arrancar la aplicación los tres botones tienen que estar deshabilidatos.76

Antes, debemos modificar una parte del código en el fichero **App.java**, el correspondiente con el programa principal, donde comienza la ejecución del código de la aplicación. Aquí habíamos establecido la conexión con la BD, pero con variables locales, con lo cual no va a ser conocida en el controlador. Así que sacaremos la declaración del objeto de BD del método en el que estaba declarada y la pondremos como atributo estático de la clase, para que pueda ser referenciada desde las otras clases que componen la aplicación:



Hemos sacado la declaración que estaba en la seguna línea del main() y la hemos puesto como variable global estática.

Y así nos quedarían los métodos para los botones BD:

```
102     @FXML
103     void insertar(ActionEvent event) {
104         boolean exito = App.bd.insertar("usuario", new String[][]{
105             {"dni", "'" + textDNI.getText() + "'"},
106             {"nombre", "'" + textNombre.getText() + "'"},
107             {"apellido", "'" + textApellidos.getText() + "'"},
108             {"email", "'" + textEmail.getText() + "'"},
109             {"password", "'" + pwPassword.getText() + "'"},
110             {"notas", "'" + areaNotas.getText() + "'"}
111         });
112         etMensajes.setText(
113                 exito ? "Registro " + textDNI.getText() + " insertado"
114                       : "Error en la inserción del registro " + textDNI.getText());
115         vaciarCampos();
116         reiniciaBotonesBD();
117     }
```

```
119     @FXML
120     void actualizar(ActionEvent event) {
121         boolean exito = App.bd.actualizar("usuario", new String[][]{
122             {"nombre", "'" + textNombre.getText() + "'"},
123             {"apellido", "'" + textApellidos.getText() + "'"},
124             {"email", "'" + textEmail.getText() + "'"},
125             {"password", "'" + pwPassword.getText() + "'"},
126             {"notas", "'" + areaNotas.getText() + "'"}
127         }, "dni = '" + textDNI.getText() + "'");
128         etMensajes.setText(
129                 exito ? "Registro " + textDNI.getText() + " actualizado"
130                       : "Error en la actualización del registro " + textDNI.getText());
131         vaciarCampos();
132         reiniciaBotonesBD();
133         textDNI.setDisable(false);
134     }
135
```

```
136     @FXML
137     void eliminar(ActionEvent event) {
138         boolean exito = App.bd.eliminar("usuario", "dni = '" + textDNI.getText() + "'");
139         etMensajes.setText(
140                 exito ? "Registro " + textDNI.getText() + " borrado"
141                       : "Error en la eliminación del registro " + textDNI.getText());
142         vaciarCampos();
143         textDNI.setDisable(false);
144     }
```

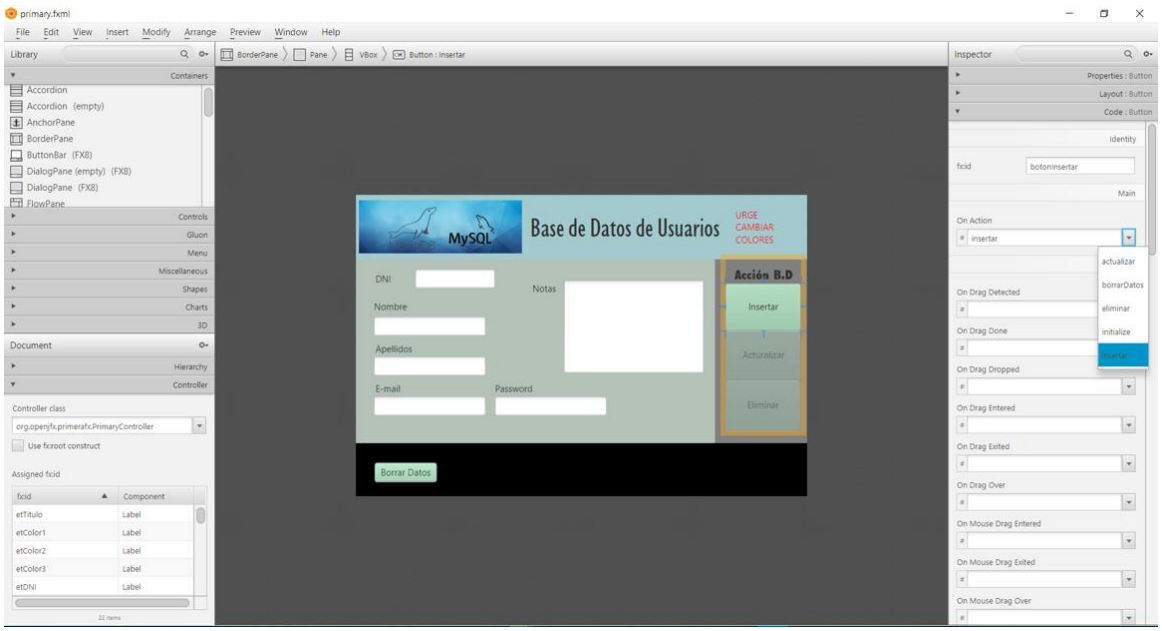
Y el método auxiliar **reiniciaBotonesBD**() que es llamado desde ellos:

```
96      private void reiniciaBotonesBD() {
97          botonActualizar.setDisable(true);
98          botonInsertar.setDisable(true);
99          botonEliminar.setDisable(true);
100     }
101
```

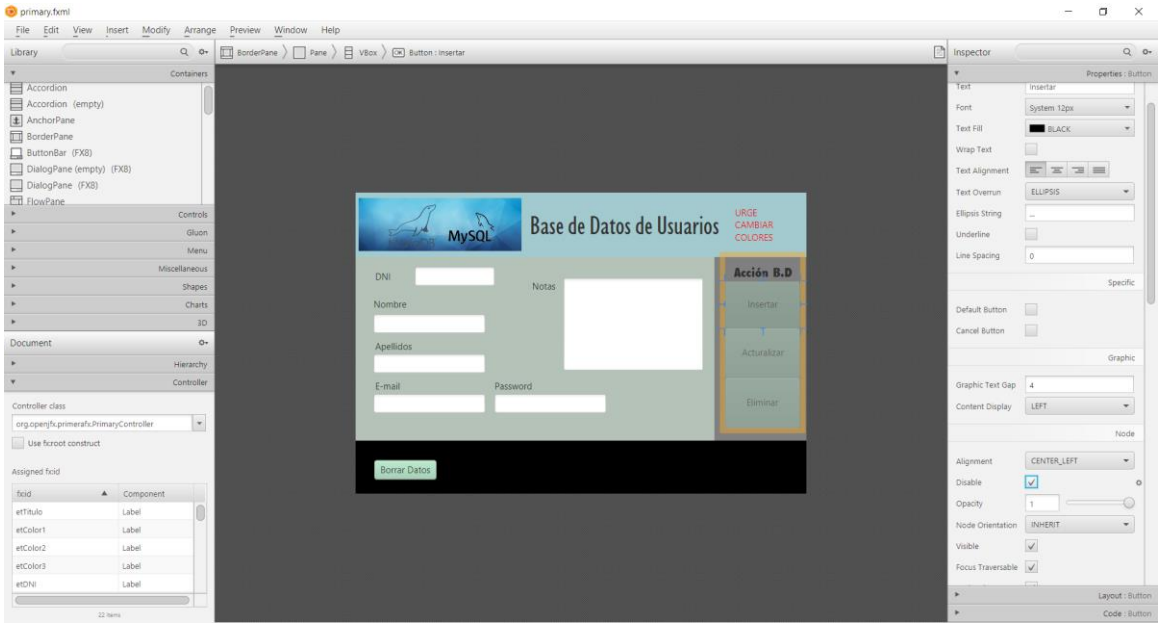
Falta ahora enlazar estos métodos con la interfaz gráfica. Podemos hacerlo escribiendo código en el fichero **principal.fxml**, pero más sencillo, lo haremos con Scene Builder. Después de grabar el código en NetBeans, abrimos (si no estaba ya abierto) Scene Builder haciendo doble click sobre el fichero **principal.fxml**

Y aquí vamos a hacer dos cosas. Primero asignaremos a los eventos On Action de cada botón, el método correspondiente (en la sección Code, la última del lado derecho). Como ya están creados los métodos, no

tenemos que escribir su nombre, sino que pulsamos sobre la flecha hacia abajo y nos aparecerán los métodos disponibles y escogemos el que corresponda:



En segundo lugar, ponemos los botones como deshabilitados. Para ello tenemos que seleccionar el botón de check disponible en la sección Properties (la primera del lado derecho):



Repetimos las operaciones para los otros dos botones, y guardamos: File->Save

Ejecutamos la aplicación para probar el funcionamieto. Creamos un registro y lo insertamos. A continuación intentamos insertar otro registro con el mismo DNI: (NOTA: aun falta código por escribir)


Primera FX
Base de Datos de Usuarios
URGE
CAMBIAR
COLORES
DNI
76706923z
Notas
Copying 3 resources
--- maven-compiler-plugin:3.8
Nothing to compile - all classe
<<< javafx-maven-plugin:0.0.4
Nombre
Amador
Apellidos
Abelleira Gómez
E-mail
abelleira@edu.xunta.es
Password
Acción B.D
Insertar
Acturalizar
Eliminar
Borrar Datos


Primera FX
Base de Datos de Usuarios
URGE
CAMBIAR
COLORES
DNI
Notas
Nombre
Apellidos
E-mail
Password
Acción B.D
Insertar
Acturalizar
Eliminar
Borrar Datos
Registro 76706923z insertado


Primera FX
Base de Datos de Usuarios
URGE
CAMBIAR
COLORES
DNI
76706923z
Notas
Usrupación de identidad
Nombre
Impostor
Apellidos
Sin Escrúpulos
E-mail
nosoyyo@edu.xunta.es
Password
Acción B.D
Insertar
Acturalizar
Eliminar
Borrar Datos

Para que lo anterior funcione, todavía nos falta introducir el código responsable de la activación/desactivación de los botones BD, así como el correspondiente a la búsqueda de la información de los registros ya existentes y el bloqueo del campo textDNI. Pero como podemos ver, la interacción con la BD se realiza correctamente.

Creación de Listeners

El código que nos falta estará en función del valor presente en el campo textDNI. Eso nos indica que debemos asociar un método a algún evento que se produzca en estos campo. El primer evento que se nos ocurre es el utilizado hasta ahora con los botones: On Action. Cuando este evento se relaciona con un campo de texto, se lanza al pulsar la tecla **INTRO**. Pero esto no es lo que buscamos ¿Qué pasa si se sale del campo pulsando la tecla de tabulación, o pulsando con el ratón sobre otro campo?, pues evidentemene, el evento no ser producirá y el método asociado no será llamado.

Como podéis observar en Scene Builder en la sección Code (última de la parte derecha) debajo de la entrada para el evento On Event, hay muchas otras entradas relacionadas con diversos tipos de eventos, agrupadas por el tipo. Eventos relacionados con el ratón, con el teclado, … Pero ninguna nos resulta adecuada para este caso. Lo que buscamos es un evento que se produzca cada vez que **salimos** (o **entramos**) en el campo textDNI. Este sería un evento **On Focus** que se lanza al obtener o perder el foco un determinado componente. Como no tenemos ninguna entrada para el mismo en Scene Builder, vamos a proceder a crearlo manualmente en la clase controlador (**Controlador.java**)

Vamos asociar un Listener (un método que está en segundo plano esperando a que se produzca un determinado evento) al campo textDNI. En una primera aproximación vamos a limitar su código a poner un mensaje en función de si el campo está vacío o no, y habilitar/deshabilitar el botón de Insertar en función de ello. Hay un método especial denominado **initialize**() que se ejecuta automáticamente al comenzar el controlador. Lo que pongáis aquí se ejecutará al inicializar la interfaz gráfica. No lo tenemos en nuestro código, así que vamos a escribirlo ahora (en en fichero **Controlador.java**)

```
146     @FXML
147     private void initialize() {
148         System.out.println("COMENZAMOS");
149         textDNI.focusedProperty().addListener(new DNIChangeListener());
150     }
151
```

La primera línea no es necesaria, es solo para comprobar que realmente este método se ejecuta automáticamento (no lo llamamos explícitamente desde ningún lugar del código).

La segunda linea crea el Listener y lo asocia a la propiedad cambio de foco del contro textDNI. Como podéis ver el Listener es un objeto de una clase que vamos a crear a continuación, y que denominamos DNIChangeListener (el nombre puede ser cualquiera que creáis conveniente). Aunque nosotos vamos a crear la clase aparte (dentro de la propia clase Controlador), lo normal es utilizar una clase anómina, y normalmente con utilización de expresiones lambda. Pondré después como quedaría, pero optamos por la clase interna con nombre, porque creo que en principio resulta más legible.

Creamos la clase. Esta clase tiene que implementar la interfaz ChangeListener:

```
165
166     private class DNIChangeListener implements ChangeListener<Boolean> {
167
168         @Override
            public void changed(ObservableValue<? extends Boolean> arg0,
170                 Boolean oldPropertyValue, Boolean newPropertyValue) {
171             if (!newPropertyValue) {
172                 if (!textDNI.getText().isBlank()) {
173                     etMensajes.setText("DNI con datos");
174                     botonInsertar.setDisable(false);
175                 } else {
176                     etMensajes.setText("DNI en blanco");
177                     botonInsertar.setDisable(true);
178                 }
179             }
180         }
181     }
182
```

Estamos indicando que cada vez que textDNI pierda el foco, se compruebe si su nuevo valor es una cadena vacía o no, y actuamos en función del resultado.

Como curiosidad, la forma de escribirlo con clases anóminas (sin nombre) y expresiones lambda sería:

```
146     @FXML
147     private void initialize() {
148         System.out.println("COMENZAMOS");
149         textDNI.focusedProperty().addListener((ObservableValue<? extends Boolean> arg0,
150                 Boolean oldPropertyValue, Boolean newPropertyValue) -> {
151             if (!newPropertyValue) {
152                 if (!textDNI.getText().isBlank()) {
153                     etMensajes.setText("DNI con datos");
154                     botonInsertar.setDisable(false);
155                 } else {
156                     etMensajes.setText("DNI en blanco");
157                     botonInsertar.setDisable(true);
158                 }
159             }
160         });
161
162     }
```

Ejecución: He salido del campo textDNI (y situado en textNombre) y el DNI está vacío. Botón Insertar deshabiliato y mensaje indicando que el DNI no tiene valor.

Salimos del campo textDNI (nos situamos en textNombre) y el DNI no está vacío. Lo indicamos con el mensaje correspondiente y se habilita el botón Insertar.

Una vez visto su funcionamiento, procedemos a escribir el resto del código. Todo lo que enseñamos a continuación estaría en la clase **Controlador.java**

Imports necesarios:

```
1  package org.openjfx.primerafx;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.util.logging.Level;
6  import javafx.beans.value.ChangeListener;
7  import javafx.beans.value.ObservableValue;
8  import javafx.event.ActionEvent;
9  import javafx.fxml.FXML;
10 import javafx.scene.control.Button;
11 import javafx.scene.control.Label;
12 import javafx.scene.control.PasswordField;
13 import javafx.scene.control.TextArea;
14 import javafx.scene.control.TextField;
15
```

Variables globales correspondientes con los controles de la interfaz gráfica. Los que están subrayados, no están siendo utilizados desde el código, con lo cual podríamos borrarlos, pero los dejamos por si en algún momento decidimos usarlos, así no nos olvidaremos de incluir su definición:

```
17 public class PrimaryController {
18
19     @FXML
20     private Label etTitulo;
21
22     @FXML
23     private Label etColor1;
24
25     @FXML
26     private Label etColor2;
27
28     @FXML
29     private Label etColor3;
30
31     @FXML
32     private Label etDNI;
33
34     @FXML
35     private Label etNombre;
36
37     @FXML
38     private Label etApellidos;
39
40     @FXML
41     private Label etEmail;
```

```
43      @FXML
44      private Label etPassword;
45
46      @FXML
47      private Label etNotas;
48
49      @FXML
50      private TextField textDNI;
51
52      @FXML
53      private TextField textNombre;
54
55      @FXML
56      private TextField textApellidos;
57
58      @FXML
59      private TextField textEmail;
60
61      @FXML
62      private PasswordField pwPassword;
63
64      @FXML
65      private TextArea areaNotas;
66
```

```
67      @FXML
68      private Button botonBorrar;
69
70      @FXML
71      private Label etMensajes;
72
73      @FXML
74      private Label etBD;
75
76      @FXML
77      private Button botonInsertar;
78
79      @FXML
80      private Button botonActualizar;
81
82      @FXML
83      private Button botonEliminar;
84
```

Código para ejecutar al pulsar el botón Borrar Datos. Vaciamos todos los campos, y también la etiqueta para mensajes de la zona inferior.

```
83      @FXML
84      void borrarDatos(ActionEvent event) {
85          vaciarCampos();
86          reiniciaBotonesBD();
87          etMensajes.setText("");
88      }
89
90      private void vaciarCampos() {
91          textDNI.setText("");
92          textNombre.setText("");
93          textApellidos.setText("");
94          textEmail.setText("");
95          pwPassword.setText("");
96          areaNotas.clear();
97      }
```

Método para reiniciar los botones y habilitar la edición del campo textDNI, después de cada operación sobre la BD:

```
99
100     private void reiniciaBotonesBD() {
101         botonActualizar.setDisable(true);
102         botonInsertar.setDisable(true);
103         botonEliminar.setDisable(true);
104         textDNI.setEditable(true);
105     }
106
```

Método asociado al botón Insertar. Intentamos la inserción en la BD. Indicamos en la zona de mensajes si la operación tuvo éxito o no:

```
107     @FXML
108     void insertar(ActionEvent event) {
109         boolean exito = App.bd.insertar("usuario", new String[][]{
110             {"dni", "'" + textDNI.getText() + "'"},
111             {"nombre", "'" + textNombre.getText() + "'"},
112             {"apellido", "'" + textApellidos.getText() + "'"},
113             {"email", "'" + textEmail.getText() + "'"},
114             {"password", "'" + pwPassword.getText() + "'"},
115             {"notas", "'" + areaNotas.getText() + "'"}
116         });
117         etMensajes.setText(
118                 exito ? "Registro " + textDNI.getText() + " insertado"
119                       : "Error en la inserción del registro " + textDNI.getText());
120         vaciarCampos();
121         reiniciaBotonesBD();
122     }
123
```

Método asociado al botón Actualizar. Intentamos la actualización en la BD. Indicamos en la zona de mensajes si la operación tuvo éxito o no:

```
123
124     @FXML
125     void actualizar(ActionEvent event) {
126         boolean exito = App.bd.actualizar("usuario", new String[][]{
127             {"nombre", "'" + textNombre.getText() + "'"},
128             {"apellido", "'" + textApellidos.getText() + "'"},
129             {"email", "'" + textEmail.getText() + "'"},
130             {"password", "'" + pwPassword.getText() + "'"},
131             {"notas", "'" + areaNotas.getText() + "'"}
132         }, "dni = '" + textDNI.getText() + "'");
133         etMensajes.setText(
134                 exito ? "Registro " + textDNI.getText() + " actualizado"
135                       : "Error en la actualización del registro " + textDNI.getText());
136         vaciarCampos();
137         reiniciaBotonesBD();
138     }
139
```

Método asociado al botón Eliminar. Intentamos la eliminación en la BD. Indicamos en la zona de mensajes si la operación tuvo éxito o no:

```
    @FXML
    void eliminar(ActionEvent event) {
        boolean exito = App.bd.eliminar("usuario", "dni = '" + textDNI.getText() + "'");
        etMensajes.setText(
                exito ? "Registro " + textDNI.getText() + " borrado"
                      : "Error en la eliminación del registro " + textDNI.getText());
        vaciarCampos();
        reiniciaBotonesBD();
    }
```

Asignación de un Listener al evento asociado a la pérdida del foco en el control textDNI. El nuevo Listener será de la case DNIChangeListerner que crearemos a continuación:

```
    @FXML
    private void initialize() {
        textDNI.focusedProperty().addListener(new DNIChangeListener());
    }
```

Clase DNIChangeListener que implementa los métodos a ser llamados cuando se produce una pérdida del foco en el control textDNI. Si el campo queda vacío, se bloquea el foco. Solo podemos salir de este control cuando haya texto introducido. Cuando salimos del control, se comprueba si existe ya ese registro en la BD (método **recupera**()); si existe, se recuperan sus datos, se bloquea el botón de Insertar además de no permitir la edición del propio campo por ser la clave primaria, y se activan los de Actualizar y Eliminar. Si no existe se bloquean los botones de Actualizar y Eliminar y se activa el de Insertar.

```
    private class DNIChangeListener implements ChangeListener<Boolean> {

        @Override
        public void changed(ObservableValue<? extends Boolean> arg0,
                Boolean oldPropertyValue, Boolean newPropertyValue) {
            if (!newPropertyValue) {
                reiniciaBotonesBD();
                if (textDNI.getText().isBlank()) {
                    textDNI.requestFocus();
                } else {
                    if (recupera()) {
                        botonActualizar.setDisable(false);
                        botonEliminar.setDisable(false);
                        textDNI.setEditable(false);
                    } else {
                        botonInsertar.setDisable(false);
                        textDNI.setEditable(true);
                    }
                }
            }
        }
```

Continuamos con la clase DNIChangeListener. Método **recupera**() encargado de comprobar la existencia en la BD del registro indicado y recuperación de sus datos en caso afirmativo.

```
176         private boolean recupera() {
177             try {
178                 ResultSet rs = App.bd.recuperarTodo("usuario",
179                         "dni = '" + textDNI.getText() + "'", "");
180                 if (rs.next()) {
181                     textNombre.setText(rs.getString(2));
182                     textApellidos.setText(rs.getString(3));
183                     textEmail.setText(rs.getString(4));
184                     pwPassword.setText(rs.getString(5));
185                     areaNotas.setText(rs.getString(6));
186                     etMensajes.setText("Registro " + textDNI.getText() + " recuperado");
187                     return true;
188                 } else {
189                     return false;
190                 }
191             } catch (SQLException ex) {
192                 BD.LOG.log(Level.SEVERE, null, ex);
193             }
194             return false;
195         }
196     }
197
```

Pueba de la Aplicación

Antes de nada, al proceder a probar el programa, me encontré con fallos en las operaciones sobre BD, que eran debidos a haber creado la tabla con unos tamaños demasiado pequeños para los campos. En concreto, email estaba con 15 caracteres, con lo cual al introducir: abelleira@edu.xunta.es ya estaba desbordando su capacidad. Así que modificad la tabla poniendo valores adecuados.

Si intentamos pulsar en cualquier campo sin haber introducido un DNI, veremos que no nos deja. El foco estará bloqueado en textDNI hasta que se introduzca algún valor.

Introducimos un DNI, el sistema comprueba si ya existe en la BD. En este caso no existe todavía, con lo cual se libera el botón de Insertar.

Rellenamos el resto de los campos

Pulsamos Insertar

Se muestra el mensaje, de éxito en este caso, y se reinicia la interfaz. Volvemos a introducir el mismo DNI, y pulsamos tabular o Intro o pulsamos sobre cualquier otro campo. Como el registro ya existía, se recuperan sus datos, se bloquea el campo DNI (no permitimos su modificación), se desactiva la opción de Insertar, y se permite Actualizar y Eliminar.

Vamos a modificar alguna información en intentar la actualización del registro

Volvemos a inroducir el mismo DNI para comprobar que la actualización fue correcta. Y a continuación borramos el registro.

Si ahora volvemos de nuevo a introducir el mismo DNI, no se debería recupera nada, si este fue realmente borrado. Con lo cual la única operación permitida será ahora Insertar.

CAMPOS OBLIGATORIOS Y REQUERIMIENTOS ESPECÍFICOS

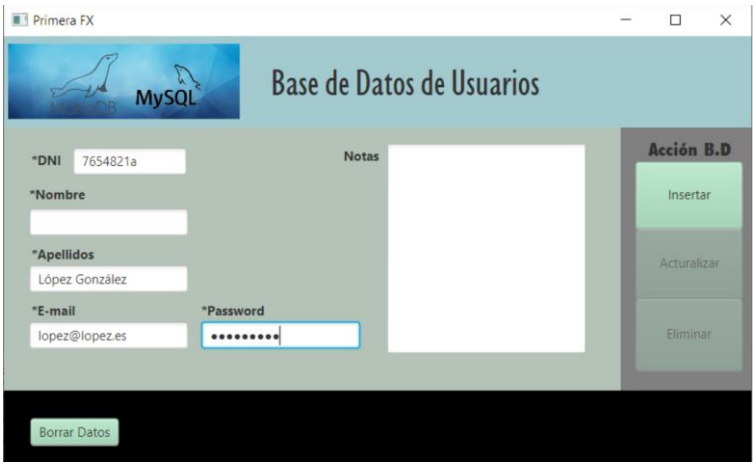
Tenemos una tabla en la que hemos definido una serie de campos como **NOT NULL** (todas menos notas), sin embargo, como podemos comprobar, tanto la inserción como la actualización se realizarán correctamente, aunque no metamos ningún valor en alguno de esos campos.

Vamos a obligar al usuario a rellenarlos si quiere que funcionen la inserción o la actualización.

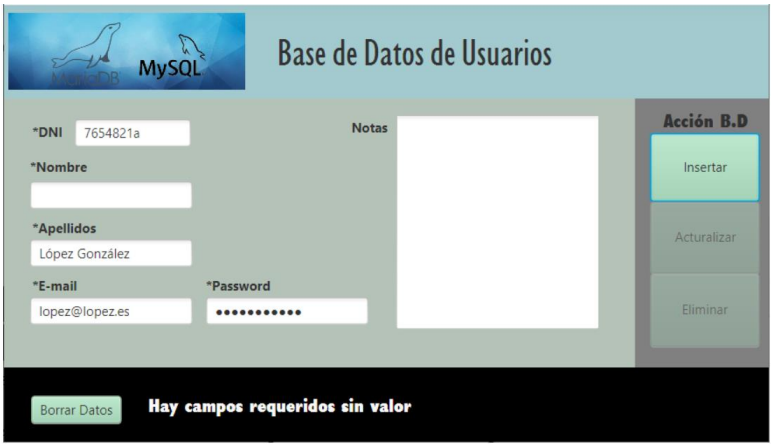
En primer lugar, para los campos NOT NULL de la BD modificamos su etiqueta en la interfaz gráfica, anteponiendo a su nombre un *, indicando con ello que son campos requeridos.

En segundo lugar hacemos que los botones Insertar y Actualizar comprueben que no hay campos obligatorios vacíos, y en caso de haberlos muestren un mensaje de advertencia y aborten la operación.

Ejemplo, rellenamos dejando algún campo en blanco:



Al pulsar Insertar:



También queremos que la contraseña tenga una longitud determinada y que el email tenga un formato válido.

Creamos un método que compruebe si al algún campo obligatorio vacío:

```
    /**
     * ´Comprueba si algún campo de los obligatorios está vacío
     *
     * @return true si hay algún campo vacío. fals en caso contrario
     */
    private boolean hayNulos() {
        return textNombre.getText().isBlank()
                || textApellidos.getText().isBlank()
                || textEmail.getText().isBlank()
                || pwPassword.getText().isBlank();
    }
```

Que será llamado desde otro método dedicado a comprobar la validez de los campos:

```
  /**
   * Método auxiliar para verificar la validez de los datos antes de insertar o
   * actualizar
   *
   * @return true si lso datos son válidos.fasle si no lo son
   */
  private boolean datosValidos() {
    if (hayNulos()) {
      etMensaje.setText("Hay campos requeridos sin valor");
      return false;
    }
    Pattern pattern = Pattern.compile(patronEmail);
    Matcher matcher = pattern.matcher(textEmail.getText());
    if (!matcher.matches()) {
      etMensaje.setText("Email no válido");
      return false;
    }
    if (passsword.getText().length() < 8) {
      etMensaje.setText("Contraseña demasiado corta. Mínimo 8");
      return false;
    }
    return true;
  }
```

Y en los métodos relativos BD de insertar y actualizar, añadiremos al principio de cada método:

```
        if (!datosValidos()) {
            return;
        }
```

Por último, añadiremos listeners a los campos de password y email, para que la comprobación se realice automáticamente sobre cada campo.

CÓDIGO COMPLETO

pantalla.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.Cursor?>
<?import javafx.scene.canvas.Canvas?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.DatePicker?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleButton?>
<?import javafx.scene.control.ToolBar?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>

<BorderPane maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308"
minHeight="686.0" minWidth="1024.0" prefHeight="686.0" prefWidth="1024.0"
xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="es.amador.lineas.Controlador">
  <top>
    <Pane maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308"
BorderPane.alignment="CENTER">
      <children>
        <ToolBar layoutX="790.0" maxWidth="1.7976931348623157E308"
prefHeight="40.0" prefWidth="233.0">
          <items>
            <Button fx:id="botonOpciones" mnemonicParsing="false"
onAction="#opciones" text="Opciones" />
            <Button fx:id="botonManual" mnemonicParsing="false"
onAction="#mostrarManual" text="Manual" />
            <Button mnemonicParsing="false" text="Info." />
            <Button mnemonicParsing="false" onAction="#salir" text="Salir" />
          </items>
        </ToolBar>
        <ToolBar maxWidth="1.7976931348623157E308" prefHeight="40.0"
prefWidth="793.0">
          <items>
            <Button fx:id="botonCargaImagen" onAction="#cargarImagen" text="Cargar
Imagen" />
            <ToggleButton fx:id="botonLinea" disable="true"
mnemonicParsing="false" onAction="#activaLinea" text="Fijar Línea" />
            <ToggleButton fx:id="botonReferencias1" disable="true"
mnemonicParsing="false" onAction="#activaReferencias1" text="Referencia1" />
            <ComboBox fx:id="comboReferencias1" prefHeight="26.0"
prefWidth="115.0" />
```

```
            <ToggleButton fx:id="botonReferencias2" disable="true" layoutX="182.0"
layoutY="12.0" mnemonicParsing="false" onAction="#activaReferencias2"
text="Referencia2" />
            <ComboBox fx:id="comboReferencias2" layoutX="263.0" layoutY="12.0"
prefHeight="26.0" prefWidth="118.0" />
            <ToggleButton fx:id="botonPuntos" disable="true"
mnemonicParsing="false" onAction="#activaPuntos" text="Marcas" />
            <Button fx:id="botonLimpiar" mnemonicParsing="false"
onAction="#limpiar" text="Limpiar" />
            <Button fx:id="botonGrabar" disable="true" mnemonicParsing="false"
onAction="#grabarFichero" text="Grabar" />
          </items>
        </ToolBar>
      </children>
    </Pane>
  </top>
  <right>
    <AnchorPane prefHeight="646.0" prefWidth="245.0"
BorderPane.alignment="CENTER">
      <children>
        <VBox layoutX="1.0" prefHeight="646.0" prefWidth="241.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="1.0"
AnchorPane.rightAnchor="4.0" AnchorPane.topAnchor="0.0">
          <children>
            <Pane prefHeight="211.0" prefWidth="245.0">
              <children>
                <ImageView fitHeight="57.0" fitWidth="237.0" layoutX="11.0"
layoutY="3.0" pickOnBounds="true" preserveRatio="true">
                  <image>
                    <Image url="@lineco.gif" />
                  </image>
                </ImageView>
                <Label layoutX="7.0" layoutY="125.0" prefHeight="18.0"
prefWidth="79.0" text="Descripción" />
                <Label layoutX="8.0" layoutY="97.0" prefHeight="18.0"
prefWidth="79.0" text="Fecha" />
                <Label layoutX="7.0" layoutY="70.0" prefHeight="18.0"
prefWidth="80.0" text="Nombre" />
                <TextField fx:id="textoNombre" layoutX="72.0" layoutY="64.0"
prefHeight="25.0" prefWidth="170.0" />
                <DatePicker fx:id="dateFecha" layoutX="73.0" layoutY="92.0"
prefHeight="26.0" prefWidth="112.0" />
                <TextArea fx:id="textoDescripcion" layoutX="74.0" layoutY="122.0"
prefHeight="80.0" prefWidth="170.0" />
              </children>
            </Pane>
            <AnchorPane prefHeight="358.0" prefWidth="251.0" VBox.vgrow="ALWAYS">
              <children>
                <TableView fx:id="tablaR1" maxHeight="1.7976931348623157E308"
prefHeight="366.0" prefWidth="250.0" AnchorPane.bottomAnchor="0.0"
AnchorPane.topAnchor="0.0">
                  <columns>
                    <TableColumn fx:id="colPuntoR1" editable="false"
prefWidth="60.0" text="Ref-1" />
                    <TableColumn fx:id="colDistanciaR1" editable="false"
prefWidth="60.0" text="Dist." />
                    <TableColumn fx:id="colReferenciaR1" prefWidth="60.0"
text="Size" />
                    <TableColumn fx:id="colBotonR1" editable="false"
prefWidth="55.0" sortable="false" text="Borrar" />
                  </columns>
```

```
                </TableView>
                <TableView fx:id="tabla" maxHeight="1.7976931348623157E308"
prefHeight="366.0" prefWidth="250.0" visible="false" AnchorPane.bottomAnchor="0.0"
AnchorPane.topAnchor="0.0">
                  <columns>
                    <TableColumn fx:id="colPunto" editable="false"
prefWidth="40.0" text="Marca" />
                    <TableColumn fx:id="colDistancia" editable="false"
prefWidth="50.0" text="Dist." />
                    <TableColumn fx:id="colReferencia1" prefWidth="50.0"
text="Ref-1" />
                    <TableColumn fx:id="colReferencia2" prefWidth="50.0"
text="Ref-2" />
                    <TableColumn fx:id="colBoton" editable="false"
prefWidth="55.0" sortable="false" text="Borrar" />
                  </columns>
                </TableView>
                <TableView fx:id="tablaR2" maxHeight="1.7976931348623157E308"
prefHeight="366.0" prefWidth="250.0" visible="false" AnchorPane.bottomAnchor="0.0"
AnchorPane.topAnchor="0.0">
                  <columns>
                    <TableColumn fx:id="colPuntoR2" editable="false"
prefWidth="60.0" text="Ref-2" />
                    <TableColumn fx:id="colDistanciaR2" editable="false"
prefWidth="60" text="Dist." />
                    <TableColumn fx:id="colReferenciaR2" prefWidth="60.0"
text="Size" />
                    <TableColumn fx:id="colBotonR2" editable="false"
prefWidth="55.0" sortable="false" text="Borrar" />
                  </columns>
                </TableView>
              </children>
            </AnchorPane>
            <Pane prefHeight="93.0" prefWidth="238.0">
              <children>
                <ImageView fitHeight="86.0" fitWidth="228.0" layoutX="16.0"
layoutY="1.0" pickOnBounds="true" preserveRatio="true">
                  <image>
                    <Image url="@logoAllerCifp.jpg" />
                  </image>
                </ImageView>
              </children>
            </Pane>
          </children>
        </VBox>
      </children>
    </AnchorPane>
  </right>
  <center>
    <Pane fx:id="pane" maxWidth="1.7976931348623157E308" prefHeight="646.0"
prefWidth="769.0" style="-fx-background-color: #abbad1;"
BorderPane.alignment="CENTER">
      <children>
        <Canvas fx:id="canvas" height="${pane.height}" layoutX="0.0" layoutY="1.0"
onMousePressed="#dibuja" onWidthChange="#redraw" width="${pane.width}">
          <cursor>
            <Cursor fx:constant="CROSSHAIR" />
          </cursor>
        </Canvas>
      </children>
    </Pane>
```

```
  </center>
</BorderPane>
```

App.java

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="686.0" prefWidth="766.0" stylesheets="@miCSS.css"
xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="es.amador.bdfx.Controlador">
  <top>
    <Pane prefHeight="118.0" prefWidth="766.0" style="-fx-background-color:
#66757a;" BorderPane.alignment="CENTER">
      <children>
        <ImageView fitHeight="116.0" fitWidth="210.0" layoutX="39.0" layoutY="1.0"
pickOnBounds="true" preserveRatio="true">
          <image>
            <Image url="@mysql-mariadb.png" />
          </image>
        </ImageView>
        <Label layoutX="269.0" layoutY="35.0" text="Base de Datos de Usuarios">
          <font>
            <Font name="Linux Libertine Display O" size="44.0" />
          </font>
        </Label>
      </children>
    </Pane>
  </top>
  <center>
    <Pane prefHeight="450.0" prefWidth="592.0" style="-fx-background-color:
#e9eef0;" BorderPane.alignment="CENTER">
      <children>
        <Label fx:id="etNombre" layoutX="27.0" layoutY="86.0" text="Nombre">
          <font>
            <Font size="15.0" />
          </font>
        </Label>
        <Label fx:id="etApellido1" layoutX="25.0" layoutY="146.0" text="Apellido
1">
          <font>
            <Font size="15.0" />
          </font>
        </Label>
        <Label fx:id="etApellido2" layoutX="24.0" layoutY="201.0" text="Apellido
2">
          <font>
            <Font size="15.0" />
          </font>
        </Label>
        <Label fx:id="etPassword" layoutX="24.0" layoutY="301.0" text="Password">
```

```
            <font>
              <Font size="15.0" />
            </font>
          </Label>
          <Label fx:id="etEmail" layoutX="24.0" layoutY="249.0" text="E-mail">
            <font>
              <Font size="15.0" />
            </font>
          </Label>
          <Label fx:id="etNotas" layoutX="369.0" layoutY="34.0" text="Notas">
            <font>
              <Font size="15.0" />
            </font>
          </Label>
          <TextField fx:id="textDni" layoutX="106.0" layoutY="35.0" />
          <TextField fx:id="textNombre" layoutX="104.0" layoutY="83.0">
            <font>
              <Font size="15.0" />
            </font>
          </TextField>
          <TextField fx:id="textApellido1" layoutX="106.0" layoutY="141.0">
            <font>
              <Font size="15.0" />
            </font>
          </TextField>
          <TextField fx:id="textApellido2" layoutX="105.0" layoutY="195.0">
            <font>
              <Font size="15.0" />
            </font>
          </TextField>
          <TextField fx:id="textEmail" layoutX="104.0" layoutY="242.0">
            <font>
              <Font size="15.0" />
            </font>
          </TextField>
          <PasswordField fx:id="passsword" layoutX="104.0" layoutY="296.0"
promptText="mínimo 8 caracteres">
            <font>
              <Font size="15.0" />
            </font>
          </PasswordField>
          <TextArea fx:id="textAreaNota" layoutX="365.0" layoutY="79.0"
prefHeight="319.0" prefWidth="224.0" wrapText="true">
            <font>
              <Font size="15.0" />
            </font>
          </TextArea>
          <Label layoutX="31.0" layoutY="44.0" text="DNI" />
              <Label layoutX="19.0" layoutY="46.0" text="*" textFill="#ec0303" />
              <Label layoutX="14.0" layoutY="304.0" text="*" textFill="#ec0303" />
              <Label layoutX="13.0" layoutY="251.0" text="*" textFill="#ec0303" />
              <Label layoutX="13.0" layoutY="88.0" text="*" textFill="#ec0303" />
              <Label layoutX="12.0" layoutY="151.0" text="*" textFill="#ec0303" />
        </children>
      </Pane>
    </center>
    <right>
      <Pane prefHeight="450.0" prefWidth="162.0" style="-fx-background-color:
#b29fe3;" BorderPane.alignment="CENTER">
        <children>
          <VBox layoutY="-2.0" prefHeight="450.0" prefWidth="160.0" spacing="10.0">
```

```
          <children>
            <Label alignment="CENTER" contentDisplay="CENTER" text="Operaciones
B.D">
              <font>
                <Font name="DejaVu Serif Bold" size="15.0" />
              </font>
            </Label>
            <Button fx:id="botonInsertar" disable="true" mnemonicParsing="false"
onAction="#accionInsertar" prefHeight="72.0" prefWidth="166.0" text="Insertar"
textFill="#ee1212">
              <font>
                <Font size="15.0" />
              </font>
            </Button>
            <Button fx:id="botonActualizar" disable="true" mnemonicParsing="false"
onAction="#accionActualizar" prefHeight="80.0" prefWidth="160.0" text="Actualizar"
textFill="#ee1212">
              <font>
                <Font size="15.0" />
              </font>
            </Button>
            <Button fx:id="botonBorrar" disable="true" mnemonicParsing="false"
onAction="#accionBorrar" prefHeight="86.0" prefWidth="164.0" text="Borrar"
textFill="#ee1212">
              <font>
                <Font size="15.0" />
              </font>
            </Button>
          </children>
          <padding>
            <Insets left="5.0" right="5.0" top="10.0" />
          </padding>
        </VBox>
      </children>
    </Pane>
  </right>
  <bottom>
    <Pane prefHeight="118.0" prefWidth="766.0" style="-fx-background-color:
#436b47;" BorderPane.alignment="CENTER">
      <children>
        <Button fx:id="botonResetear" layoutX="23.0" layoutY="34.0"
mnemonicParsing="false" onAction="#accionResetear" prefHeight="53.0"
prefWidth="114.0" text="Borrar Datos" />
        <Label fx:id="etMensaje" layoutX="165.0" layoutY="50.0"
textFill="#faefef">
          <font>
            <Font size="18.0" />
          </font>
        </Label>
      </children>
    </Pane>
  </bottom>
</BorderPane>
```

Controlador.java

```
package es.amador.bdfx;

import es.amador.bdrelacionesm.SGBD;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.paint.Color;

public class Controlador {

  private static final String PATRON_EMAIL = "^(.+)@(.+)$";

  private static final String patronEmail
          = "^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-]+)*@(?:"
          + "[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$";
  private String nombreTabla;

  @FXML
  private Label etNombre;

  @FXML
  private Label etApellido1;

  @FXML
  private Label etApellido2;

  @FXML
  private Label etPassword;

  @FXML
  private Label etEmail;

  @FXML
  private Label etNotas;

  @FXML
  private Label etMensaje;

  @FXML
  private TextField textDni;

  @FXML
  private TextField textNombre;

  @FXML
  private TextField textApellido1;
```

```
@FXML
private TextField textApellido2;

@FXML
private TextField textEmail;

@FXML
private PasswordField passsword;

@FXML
private TextArea textAreaNota;

@FXML
private Button botonInsertar;

@FXML
private Button botonActualizar;

@FXML
private Button botonBorrar;

@FXML
private Button botonResetear;

private void reiniciarBotonesBD() {
  botonActualizar.setDisable(true);
  botonInsertar.setDisable(true);
  botonBorrar.setDisable(true);
  textDni.setEditable(true);
}

private void vaciarCampos() {
  textDni.setText("");
  textNombre.setText("");
  textApellido1.setText("");
  textApellido2.setText("");
  textEmail.setText("");
  passsword.setText("");
  textAreaNota.clear();
}

void mensaje(boolean error, String mensaje) {
  if (error) {
    etMensaje.setTextFill(Color.RED);
  } else {
    etMensaje.setTextFill(Color.WHITE);
  }
  etMensaje.setText(mensaje);
}

@FXML
void accionInsertar(ActionEvent event) {
  if (!datosValidos()) {
    return;
  }
  boolean exito = App.bd.insertarRegistros(nombreTabla, new String[][]{
    {"dni", "'" + textDni.getText() + "'"},
    {"nombre", "'" + textNombre.getText() + "'"},
    {"apellido1", "'" + textApellido1.getText() + "'"},
    {"apellido2", "'" + textApellido2.getText() + "'"},
```

```
      {"email", "'" + textEmail.getText() + "'"},
      {"password", "'" + passsword.getText() + "'"},
      {"notas", "'" + textAreaNota.getText() + "'"},});
    if (exito) {
      mensaje(false, "Registro " + textDni.getText() + " insertado");
    } else {
      mensaje(true, "Error en la inserción del registro " + textDni.getText());
    }
    reseteo();
  }

  @FXML
  void accionActualizar(ActionEvent event) {
    if (!datosValidos()) {
      return;
    }
    boolean exito = App.bd.actualizarRegistros(nombreTabla, new String[][]{
      {"dni", "'" + textDni.getText() + "'"},
      {"nombre", "'" + textNombre.getText() + "'"},
      {"apellido1", "'" + textApellido1.getText() + "'"},
      {"apellido2", "'" + textApellido2.getText() + "'"},
      {"email", "'" + textEmail.getText() + "'"},
      {"password", "'" + passsword.getText() + "'"},
      {"notas", "'" + textAreaNota.getText() + "'"},},
            "dni = '" + textDni.getText() + "'");
    if (exito) {
      mensaje(false, "Registro " + textDni.getText() + " actualizado");
    } else {
      mensaje(true, "Error en la actualización del registro " +
textDni.getText());
    }
    reseteo();
  }

  @FXML
  void accionBorrar(ActionEvent event) {
    boolean exito = App.bd.eliminarRegistros(nombreTabla, "dni = '"
            + textDni.getText() + "'");
    if (exito) {
      mensaje(false, "Registro " + textDni.getText() + " eliminado");
    } else {
      mensaje(true, "Error en la eliminación del registro " + textDni.getText());
    }
    reseteo();
  }

  void reseteo() {
    vaciarCampos();
    reiniciarBotonesBD();
  }

  @FXML
  void accionResetear(ActionEvent event) {
    reseteo();
    etMensaje.setText("");
  }

  @FXML
  private void initialize() {
    nombreTabla = "usuario";
    textDni.focusedProperty().addListener(new DNIChangeListener());
```

```
  }

  /**
   * ´Comprueba si algún campo de los obligatorios está vacío
   *
   * @return true si hay algún campo vacío. fals en caso contrario
   */
  private boolean hayNulos() {
    return textNombre.getText().isBlank()
            || textApellido1.getText().isBlank()
            || textEmail.getText().isBlank()
            || passsword.getText().isBlank();
  }

  /**
   * Método auxiliar para verificar la validez de los datos antes de insertar o
   * actualizar
   *
   * @return true si lso datos son válidos.fasle si no lo son
   */
  private boolean datosValidos() {
    if (hayNulos()) {
      mensaje(true, "Hay campos requeridos sin valor");
      return false;
    }
    Pattern pattern = Pattern.compile(patronEmail);
    Matcher matcher = pattern.matcher(textEmail.getText());
    if (!matcher.matches()) {
      mensaje(true, "Email no válido");
      return false;
    }
    if (passsword.getText().length() < 8) {
      mensaje(true, "Contraseña demasiado corta. Mínimo 8");
      return false;
    }
    return true;
  }

  private class DNIChangeListener implements ChangeListener<Boolean> {

    @Override
    public void changed(ObservableValue<? extends Boolean> ov, Boolean oldPV,
Boolean newPV) {
      if (!newPV) {
        reiniciarBotonesBD();
        if (textDni.getText().isBlank()) {
          textDni.requestFocus();
        } else {
          if (recupera()) {
            botonActualizar.setDisable(false);
            botonBorrar.setDisable(false);
            textDni.setEditable(false);
          } else {
            botonInsertar.setDisable(false);
            textDni.setEditable(true);
          }
        }
      }
    }

    private boolean recupera() {
```

```
      try {
        ResultSet rs = App.bd.recuperarTodo(nombreTabla,
                "dni = '" + textDni.getText() + "'");
        if (rs.next()) {
          textNombre.setText(rs.getString(2));
          textApellido1.setText(rs.getString(3));
          textApellido2.setText(rs.getString(4));
          textEmail.setText(rs.getString(5));
          passsword.setText(rs.getString(6));
          textAreaNota.setText(rs.getString(7));
          mensaje(false, "Registro " + textDni.getText() + " recuperado");
          return true;
        } else {
          return false;
        }
      } catch (SQLException ex) {
        SGBD.LOG.log(Level.SEVERE, null, ex);
      }
      return false;
    }
  }
}
```