

Ing. Julio Carreño

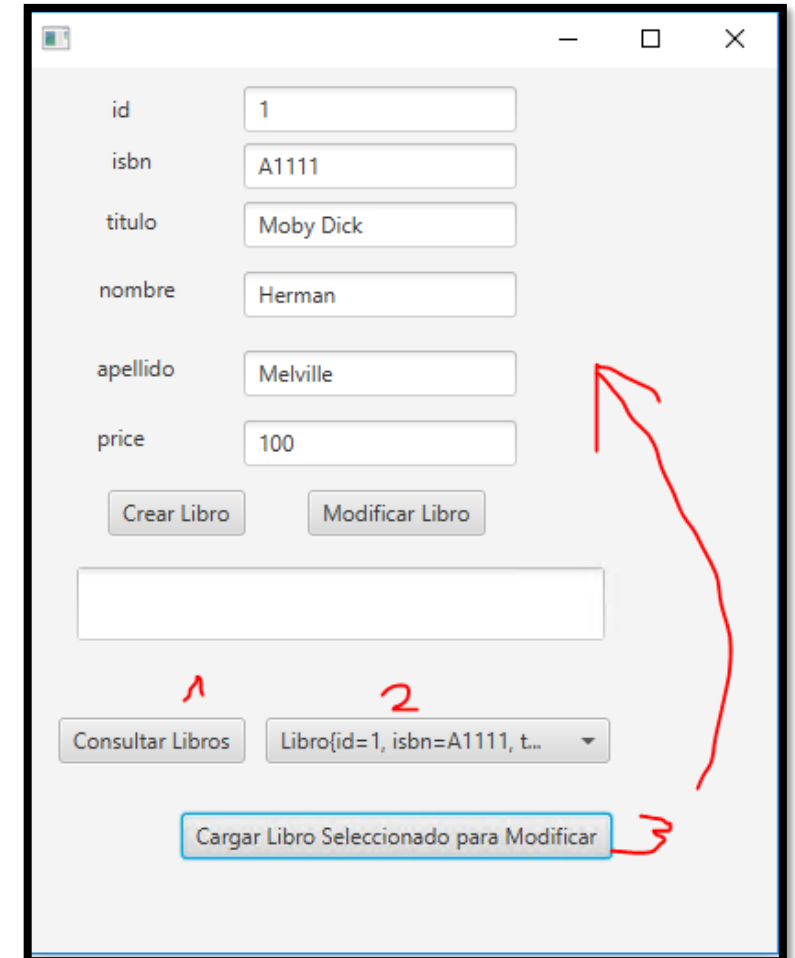
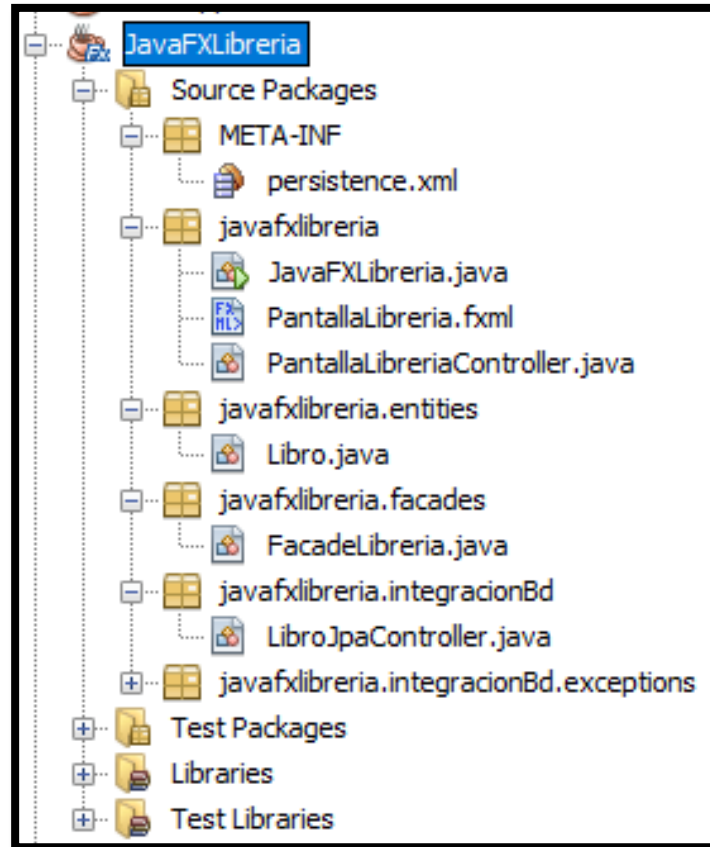
Aplicación Java en Capas usando acceso a BD Derby

Aplicación MVC y en capas usando Java FX

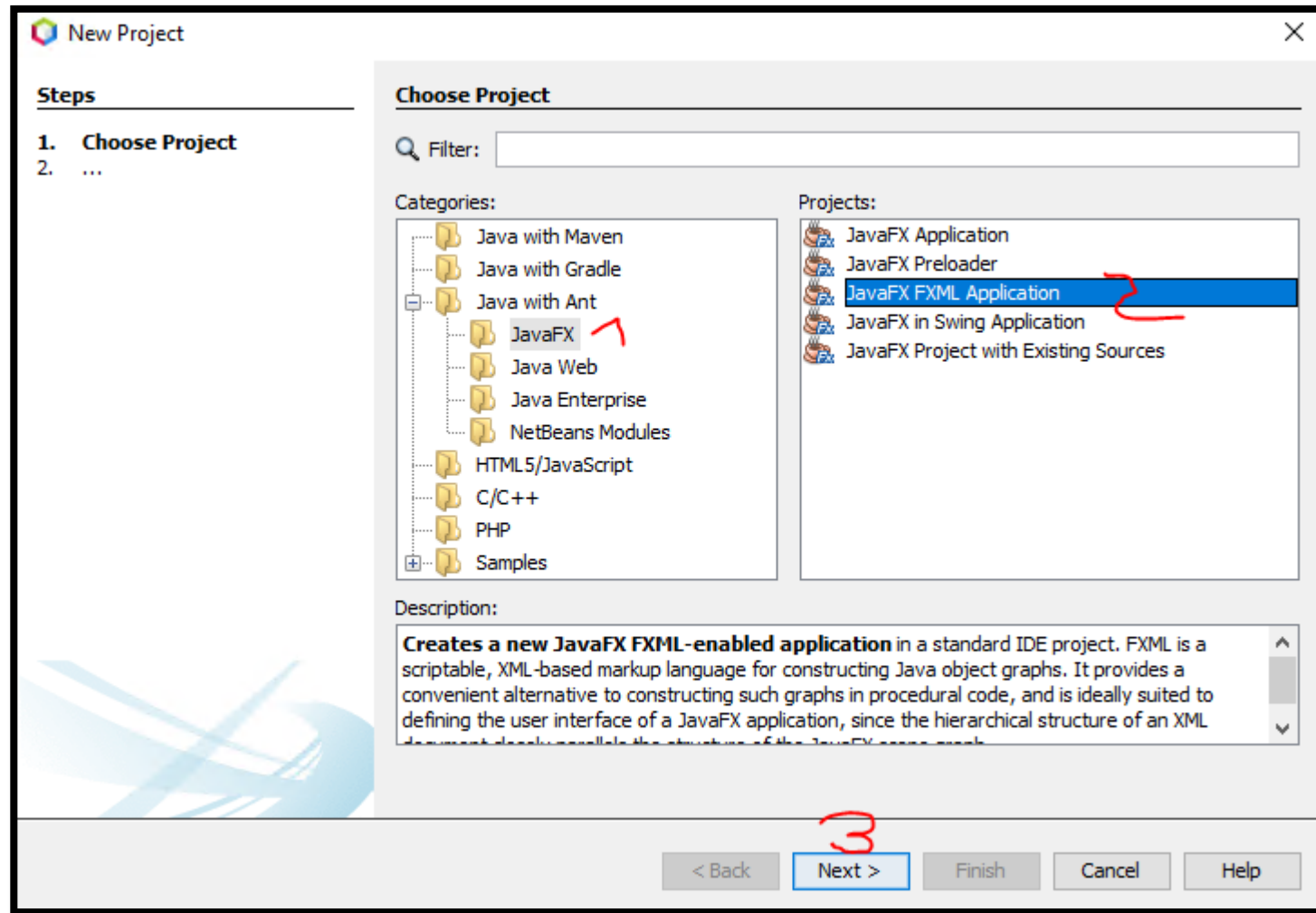
Ing. Julio Carreño

Objetivo

- Crear una aplicación en Java que tenga varias capas
 - Presentacion: Java Fx
 - Negocio: facades
 - Integración: clases controladoras JPA para acceder tablas derby
 - Entidades: clases JPA que representan tablas



Nuevo Proyecto Capa Presentación



Nuevo Proyecto Capa Presentación

New JavaFX Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: JavaFXLibreria 1

Project Location: D:\Tools\nb12proys Browse...

Project Folder: D:\Tools\nb12proys\JavaFXLibreria

JavaFX Platform: JDK 1.8 (Default) Manage Platforms...

☐ Create Custom Preloader

Project Name: JavaFXLibreria-Preloader

FXML name: PantallaLibreria 2

☐ Use Dedicated Folder for Storing Libraries

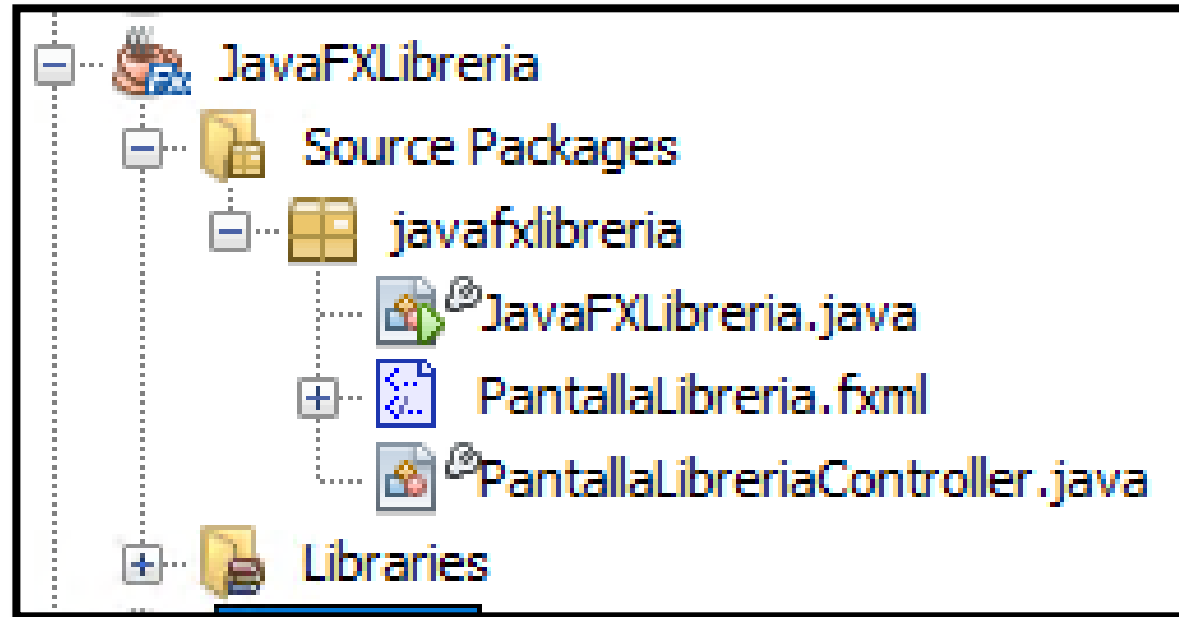
Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Application Class javafxlibreria.JavaFXLibreria

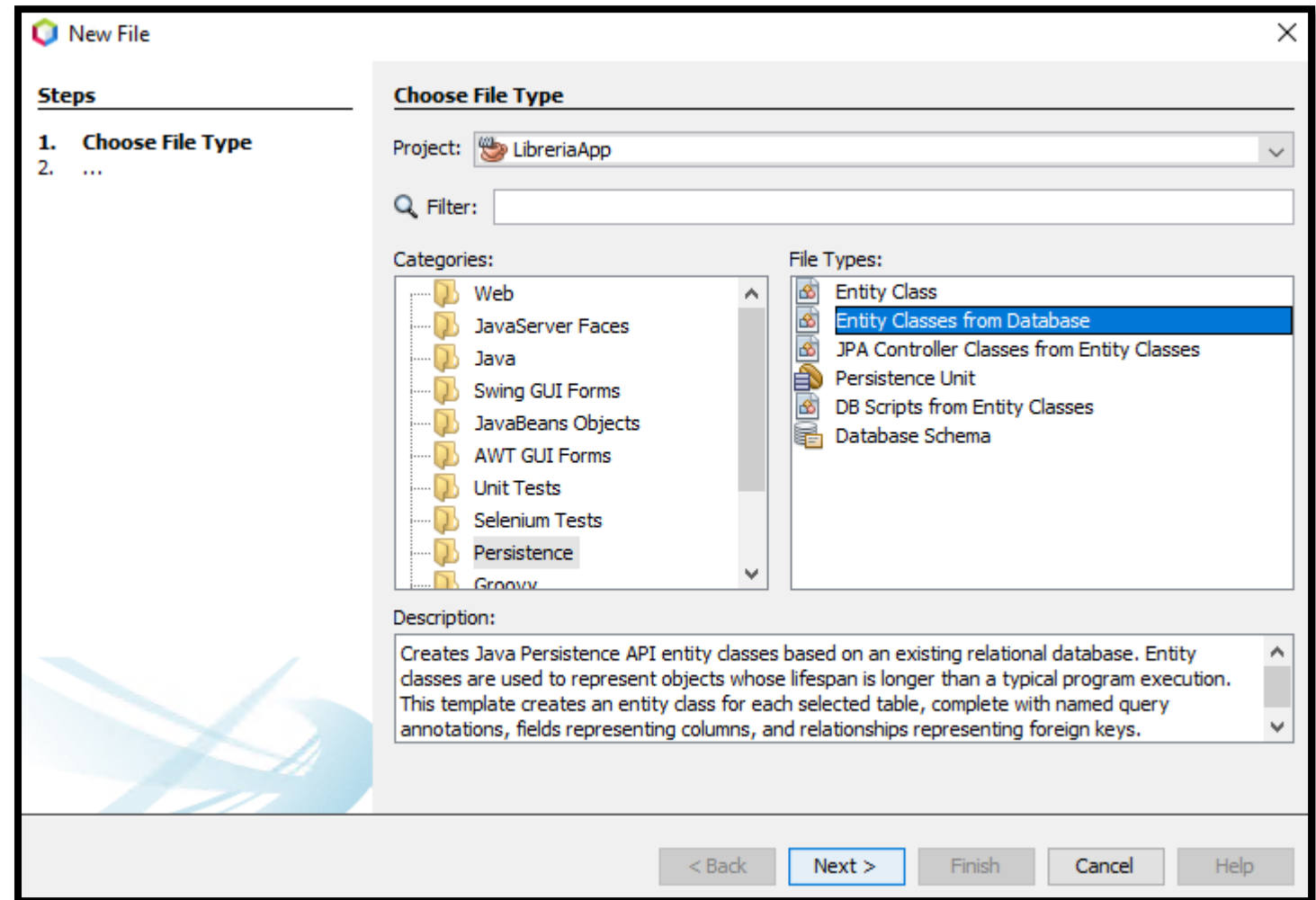
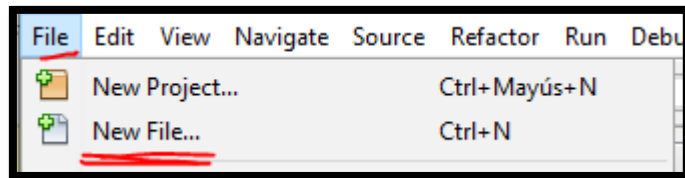
< Back Next > Finish 3 Cancel Help

Estructura del proyecto Creado

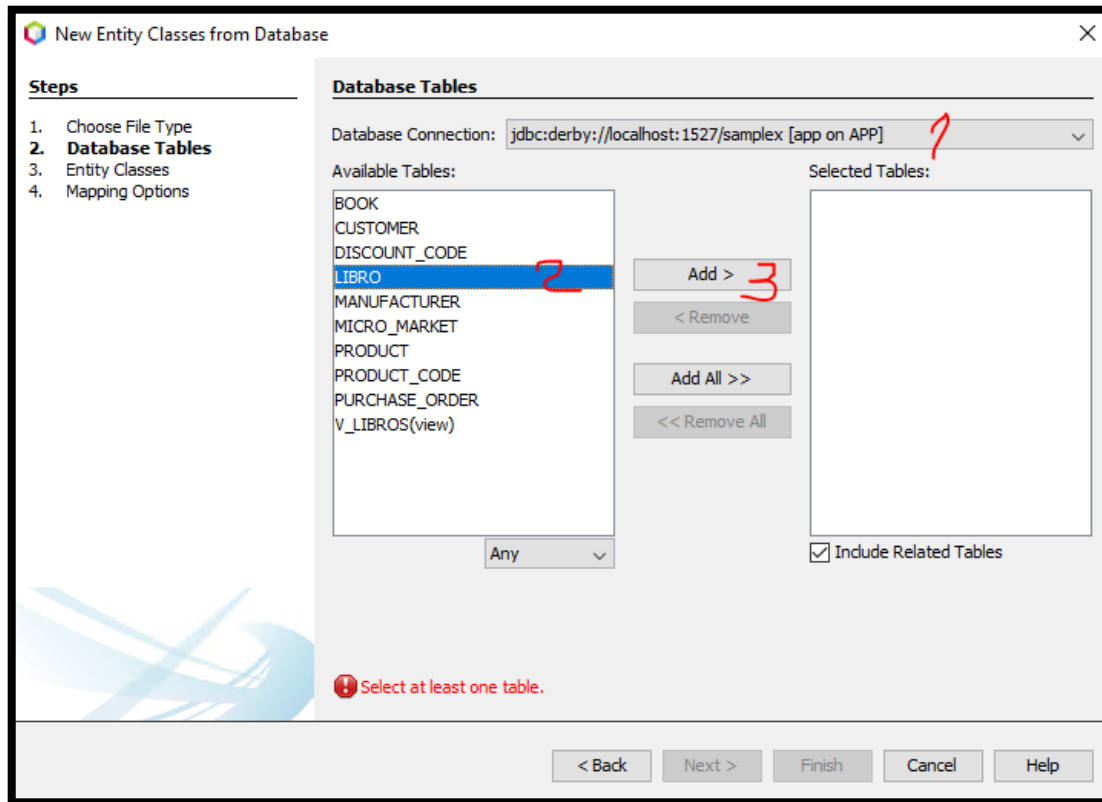


Creando las entidades JPA

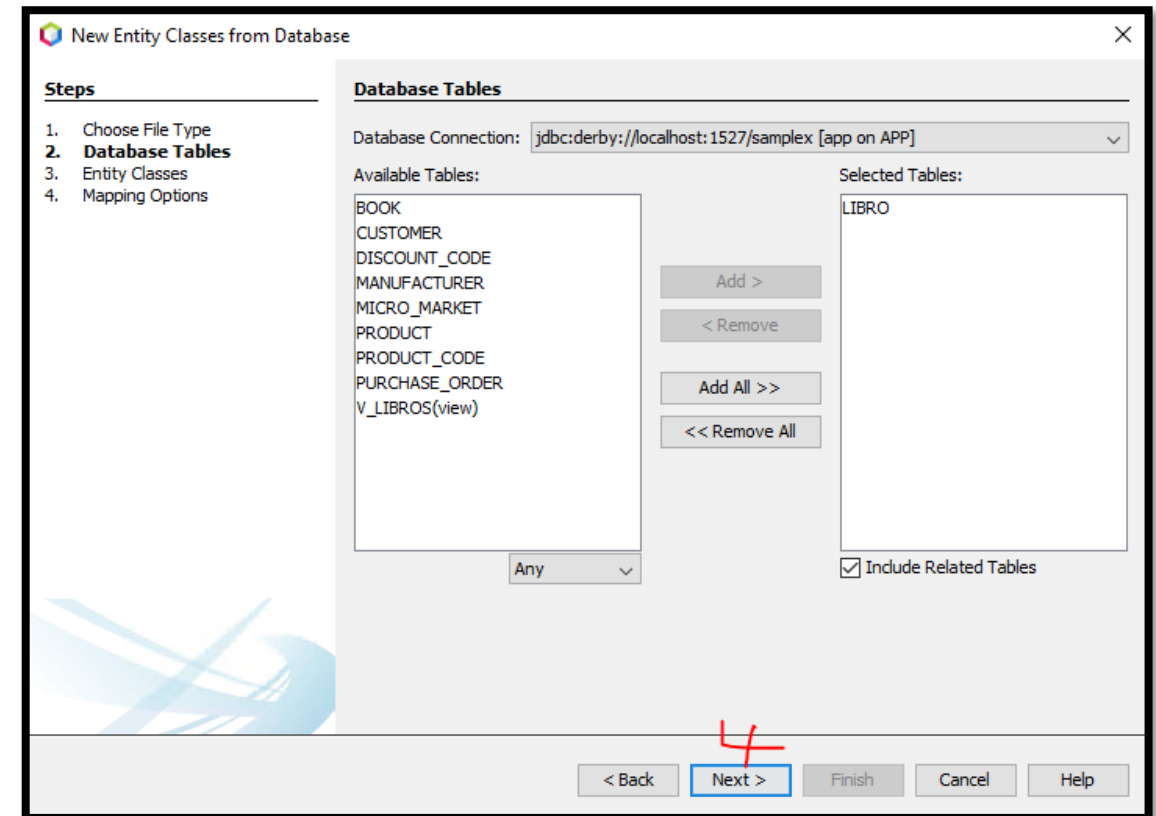
Crear Entidades JPA desde la BD (1)



Crear Entidades JPA desde la BD (2)



La lista que se despliega en (1) aparece porque usted ya creó en el NetBeans una conexión a Derby DB



Crear Entidades JPA desde la BD (3)

New Entity Classes from Database

Steps

1. Choose File Type
2. Database Tables
3. **Entity Classes**
4. Mapping Options

Entity Classes

Specify the names and the location of the entity classes.

Class Names:

| Database Table | Class Name | Generation Type |
|----------------|------------|-----------------|
| LIBRO | Libro | New |
| ... | | |

Project: LibreriaApp

Location: Source Packages

Package: libreriaapp.entities

☒ Generate Named Query Annotations for Persistent Fields

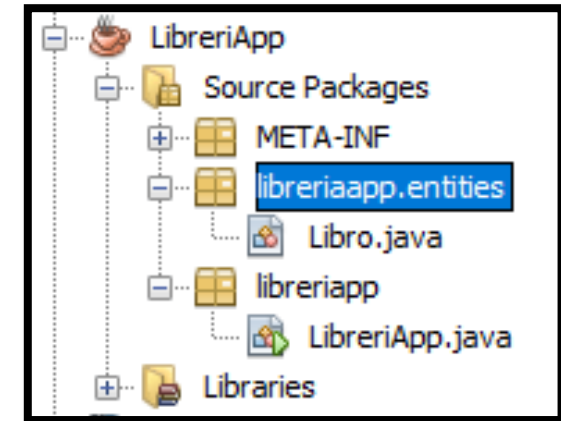
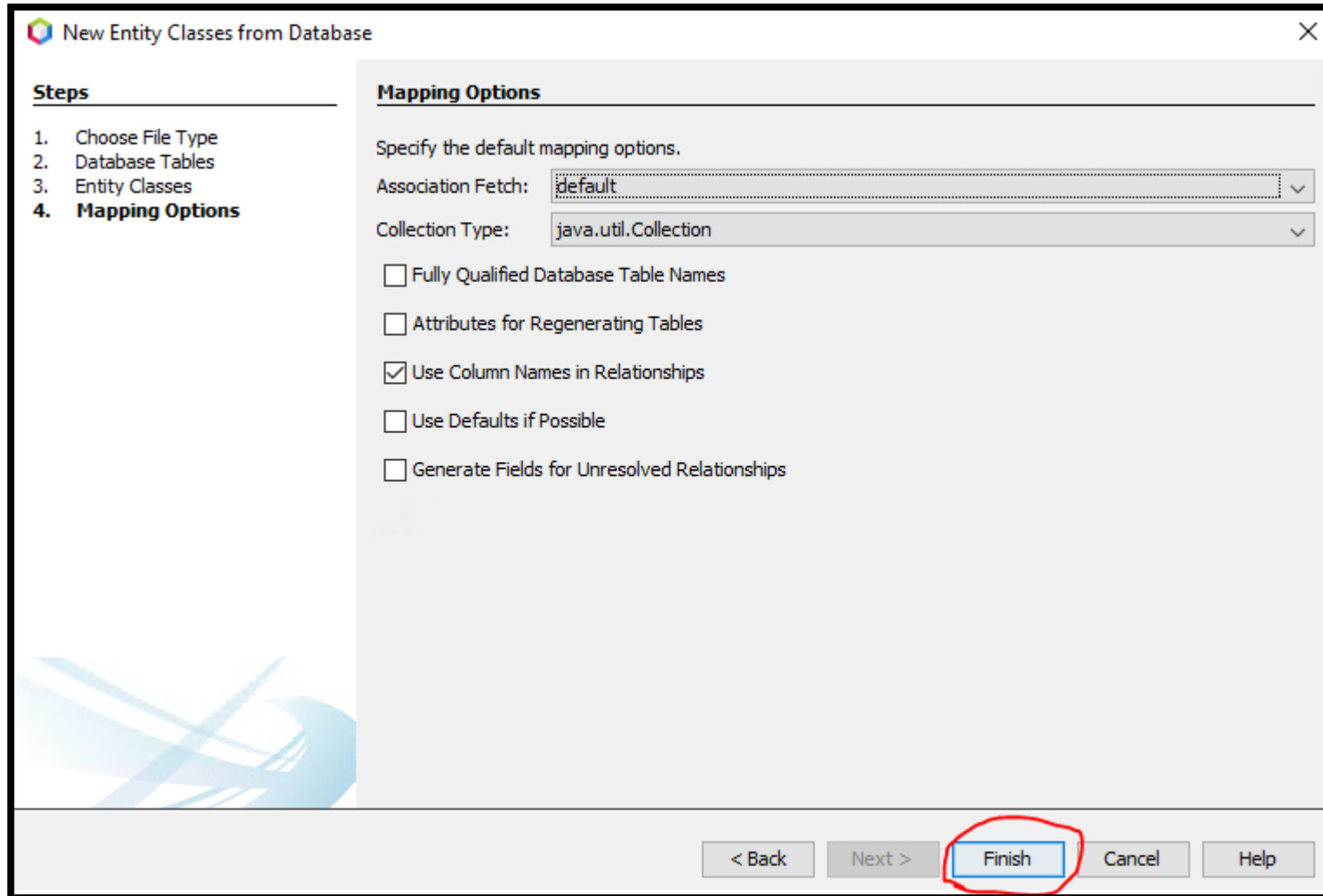
☒ Generate JAXB Annotations

☐ Generate MappedSuperclasses instead of Entities

☒ Create Persistence Unit

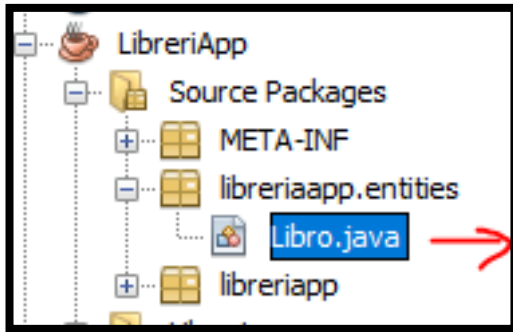
< Back Next > Finish Cancel Help

Crear Entidades JPA desde la BD (4)



Observe que se crea la clase Libro.java

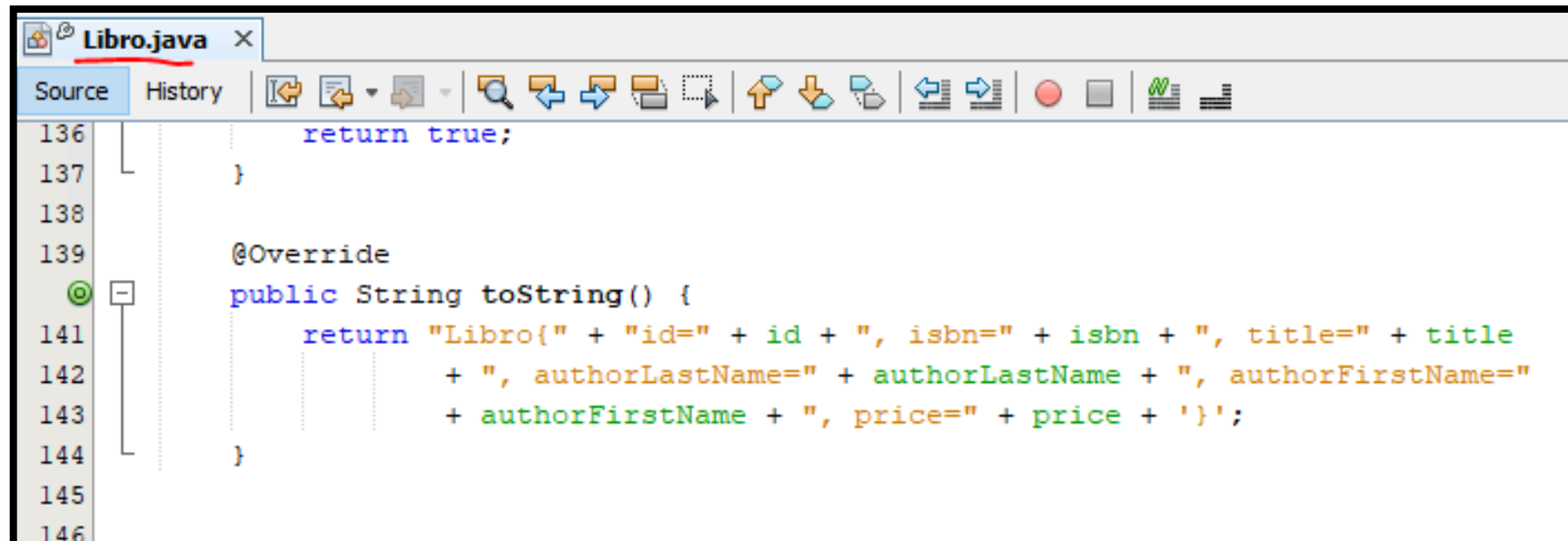
Revise la clase Libro



```
Libro.java
Source History
13 import javax.persistence.GenerationType;
14 import javax.persistence.Id;
15 import javax.persistence.NamedQueries;
16 import javax.persistence.NamedQuery;
17 import javax.persistence.Table;
18 import javax.xml.bind.annotation.XmlRootElement;
19
20 /**...4 lines */
21
22 @Entity
23 @Table(name = "LIBRO")
24 @XmlRootElement
25 @NamedQueries({
26     @NamedQuery(name = "Libro.findAll", query = "SELECT 1 FROM Libro 1"),
27     @NamedQuery(name = "Libro.findById", query = "SELECT 1 FROM Libro 1 WHERE 1.id = :id"),
28     @NamedQuery(name = "Libro.findByIsbn", query = "SELECT 1 FROM Libro 1 WHERE 1.isbn = :isbn"),
29     @NamedQuery(name = "Libro.findByTitle", query = "SELECT 1 FROM Libro 1 WHERE 1.title = :title"),
30     @NamedQuery(name = "Libro.findByAuthorLastName", query = "SELECT 1 FROM Libro 1 WHERE 1.authorLastName = :authorLastName"),
31     @NamedQuery(name = "Libro.findByAuthorFirstName", query = "SELECT 1 FROM Libro 1 WHERE 1.authorFirstName = :authorFirstName"),
32     @NamedQuery(name = "Libro.findByPrice", query = "SELECT 1 FROM Libro 1 WHERE 1.price = :price")})
33
34 public class Libro implements Serializable {
35
36     private static final long serialVersionUID = 1L;
37     @Id
38     @GeneratedValue(strategy = GenerationType.IDENTITY)
39     @Basic(optional = false)
40     @Column(name = "ID")
41     private Integer id;
42     @Column(name = "ISBN")
43     private String isbn;
44     @Basic(optional = false)
45     @Column(name = "TITLE")
46     private String title;
47     @Basic(optional = false)
48     @Column(name = "AUTHOR_LAST_NAME")
49     private String authorLastName;
50     @Column(name = "AUTHOR_FIRST_NAME")
51     private String authorFirstName;
52     @Basic(optional = false)
53     @Column(name = "PRICE")
54     private int price;
```

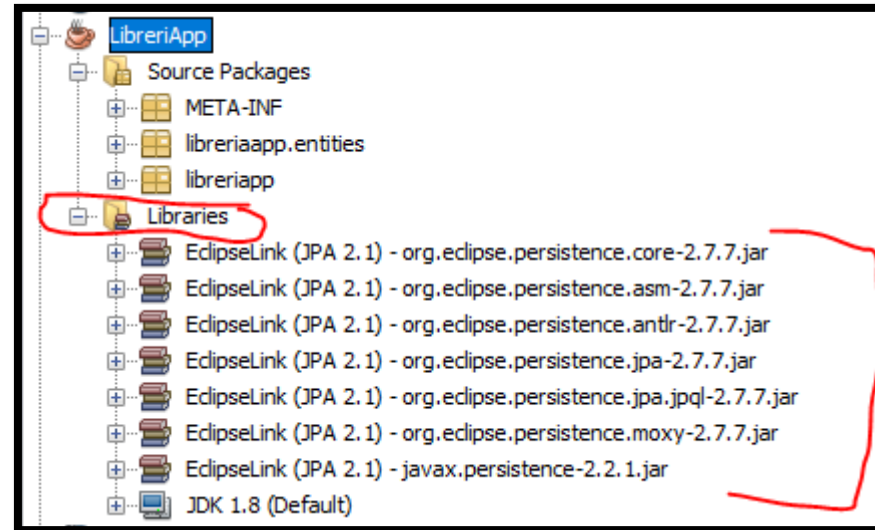
Agregue toString() en la clase Libro con todos los atributos

Use la opción Insert Code → ToString() del NB



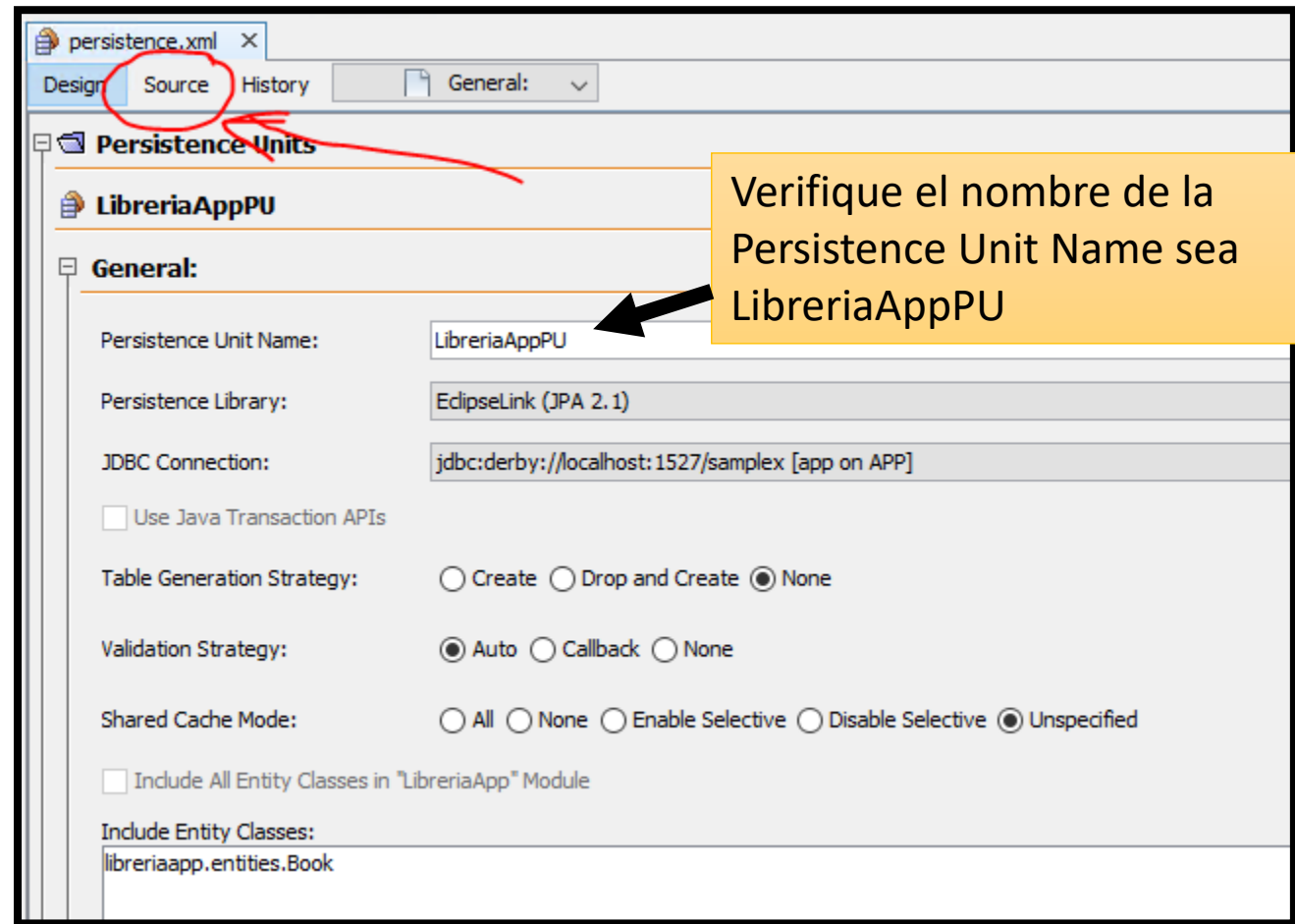
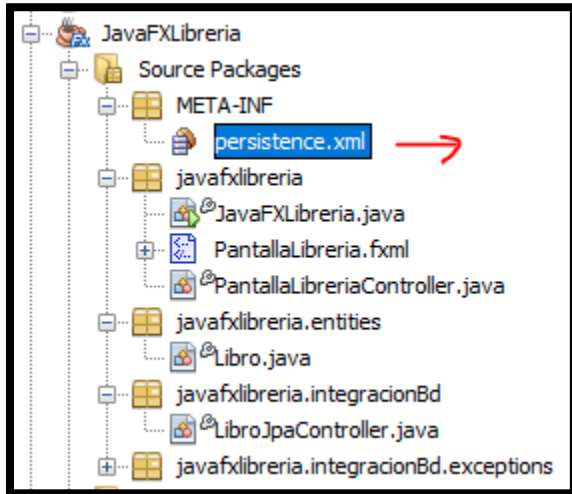
```
Libro.java x
Source History
136         return true;
137     }
138
139     @Override
140     public String toString() {
141         return "Libro{" + "id=" + id + ", isbn=" + isbn + ", title=" + title
142             + ", authorLastName=" + authorLastName + ", authorFirstName="
143             + authorFirstName + ", price=" + price + '}';
144     }
145
146
```

Revise las librerías JPA



Observe que en la carpeta Librerías se agregaron los .jar requeridos para el JPA

Verifique el persistence.xml (1)

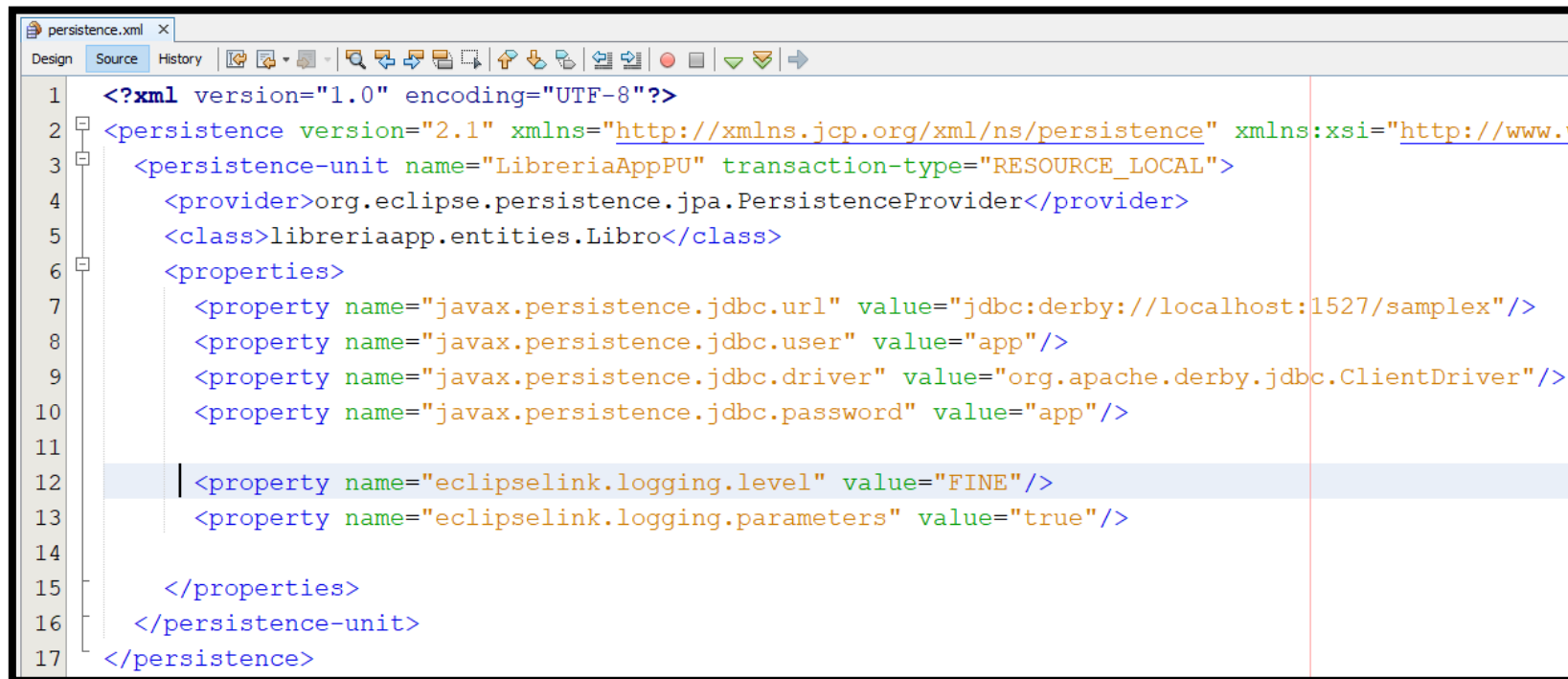


Agregue propiedades al persistence.xml (2)

- Para revisar los SQL generados por el JPA agregue las siguientes propiedades en el persistence.xml (vea líneas 11 y 12)

`<property name="eclipselink.logging.level" value="FINE"/>`

`<property name="eclipselink.logging.parameters" value="true"/>`



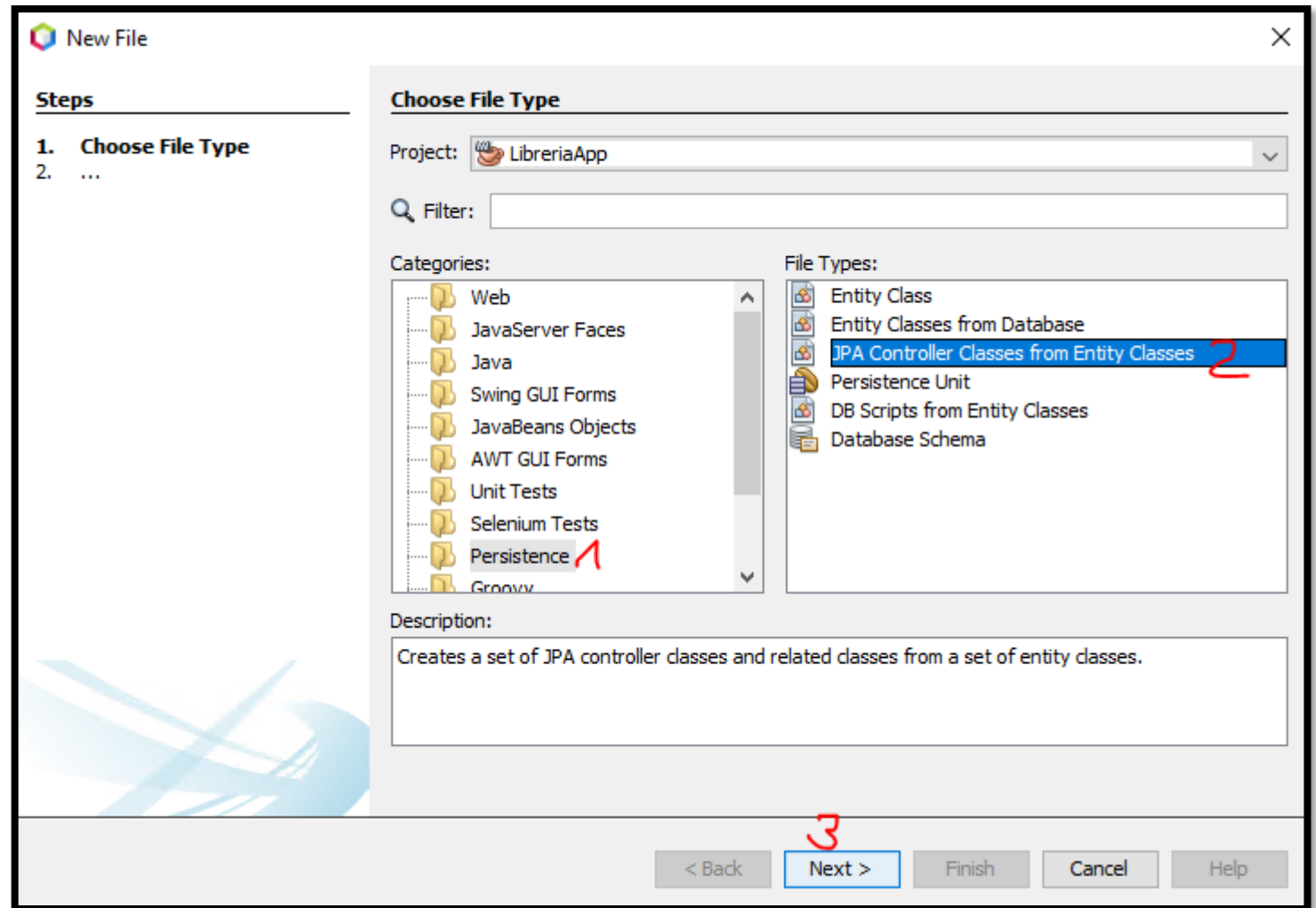
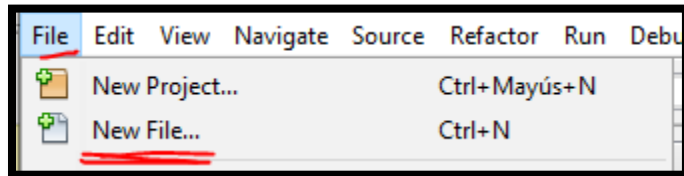
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence.xml">
3    <persistence-unit name="LibreriaAppPU" transaction-type="RESOURCE_LOCAL">
4      <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
5      <class>libreriaapp.entities.Libro</class>
6      <properties>
7        <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/samplex"/>
8        <property name="javax.persistence.jdbc.user" value="app"/>
9        <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
10       <property name="javax.persistence.jdbc.password" value="app"/>
11       <property name="eclipselink.logging.level" value="FINE"/>
12       <property name="eclipselink.logging.parameters" value="true"/>
13     </properties>
14   </persistence-unit>
15 </persistence>
```


Guarde

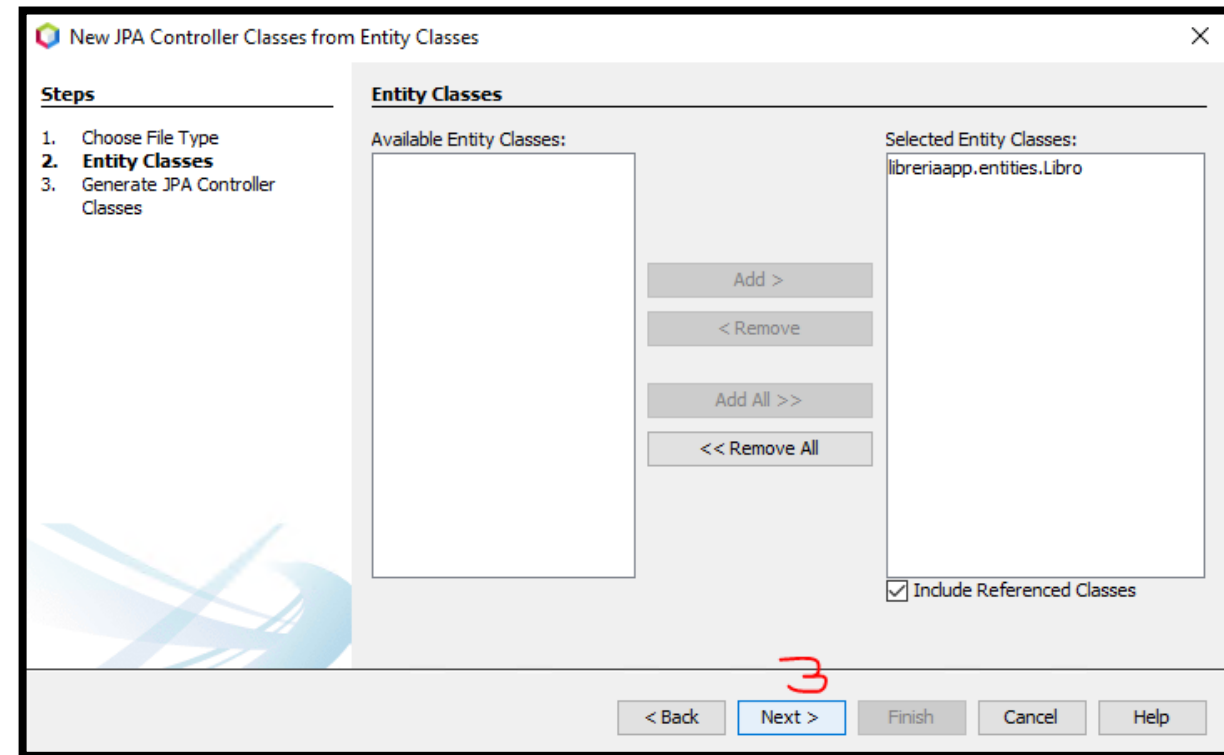
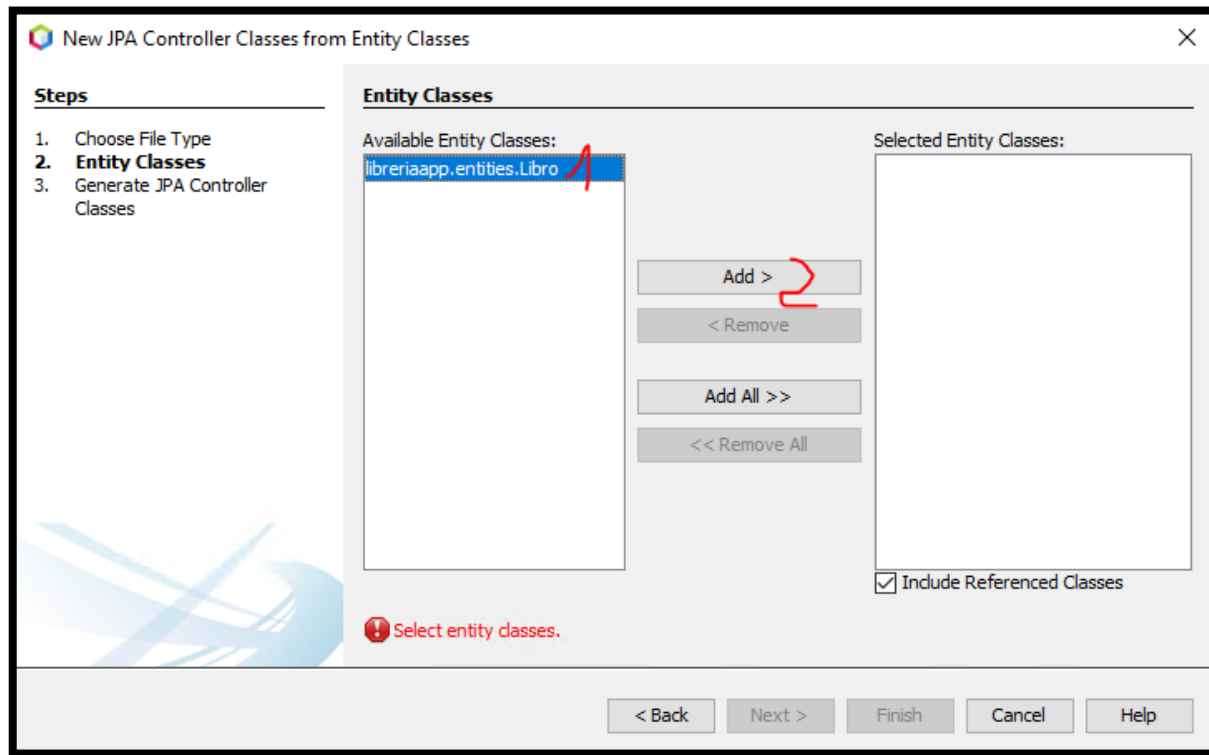
- Guarde su trabajo

Creando los controladores de
Integración a la base de datos

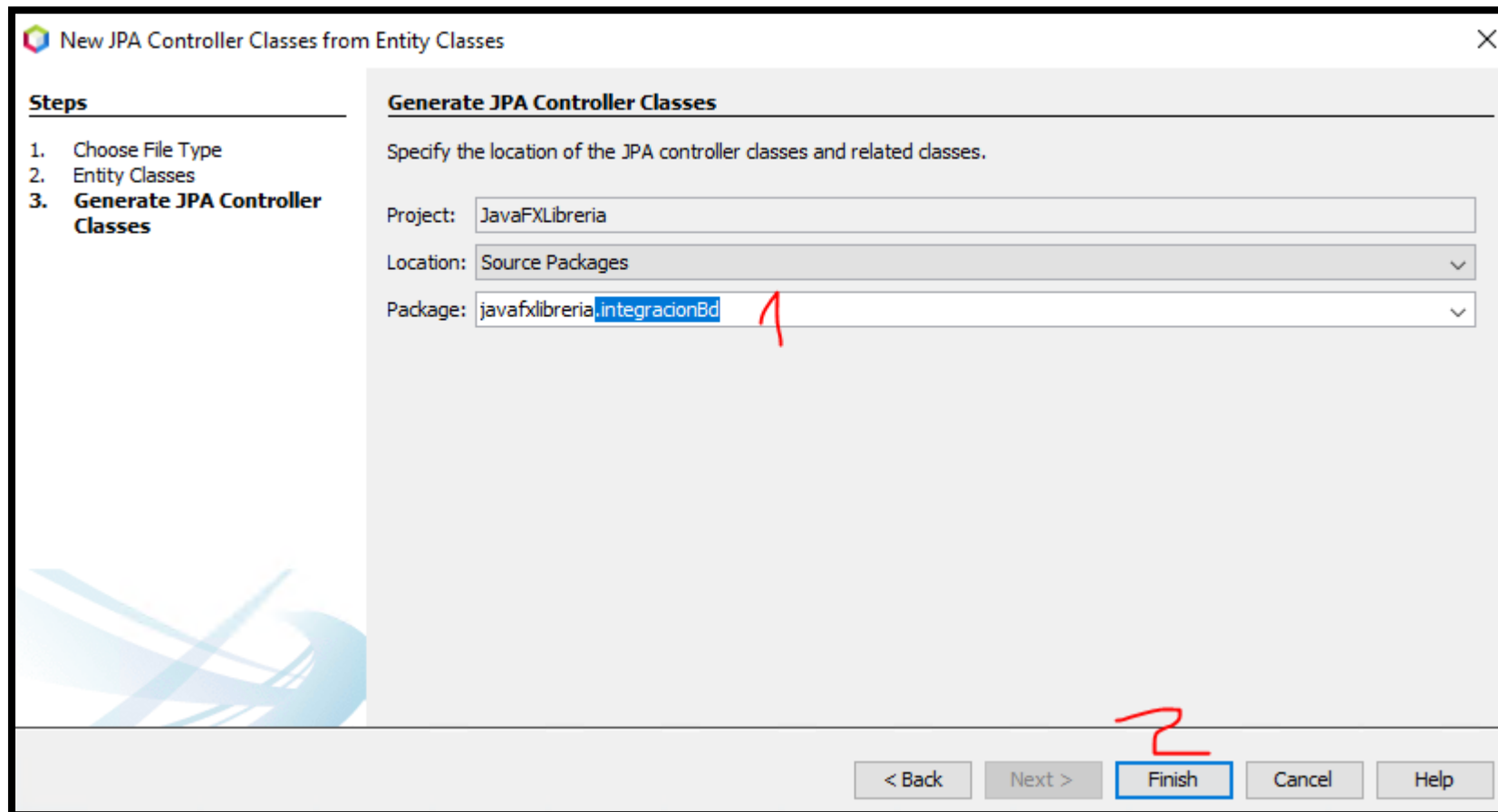
Crear controladores de Integración a BD desde Entidades JPA (1)



Crear controladores de Integración a BD desde Entidades JPA (2)



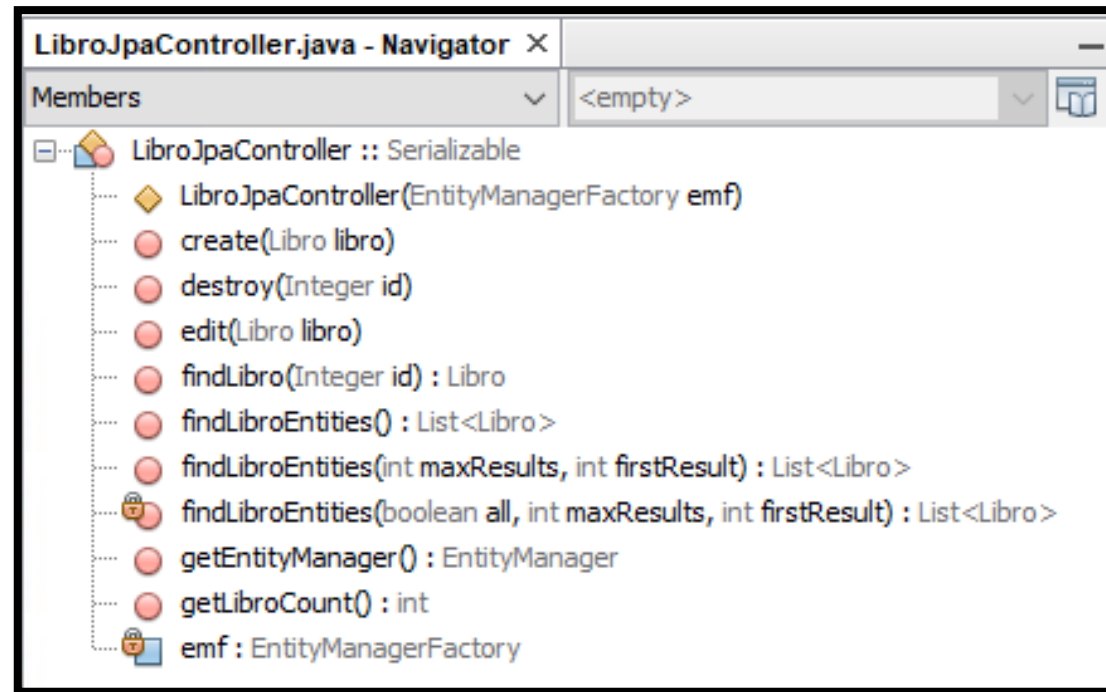
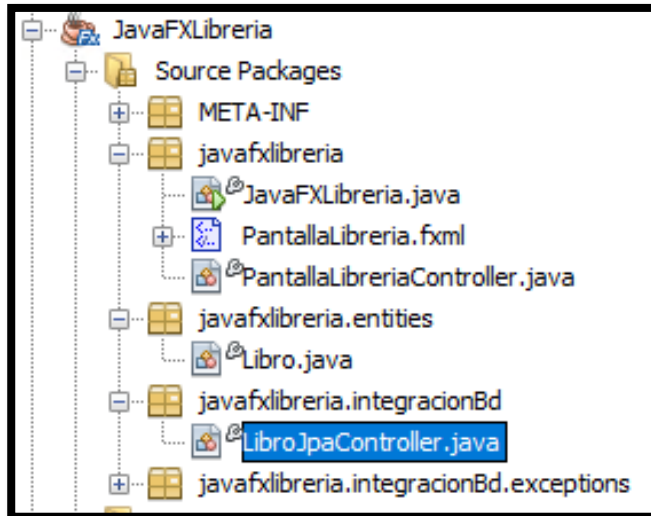
Crear controladores de Integración a BD desde Entidades JPA (3)



Guarde

- Guarde su trabajo

Revisando el controlador de integración a BD



Ajustes a la clase controladora de integración a BD

- Inserte constructor vacío en la clase controladora de integración a BD

Use la opción Insert Code → Constructor del NB

Ajustes a la clase controladora de integración a BD

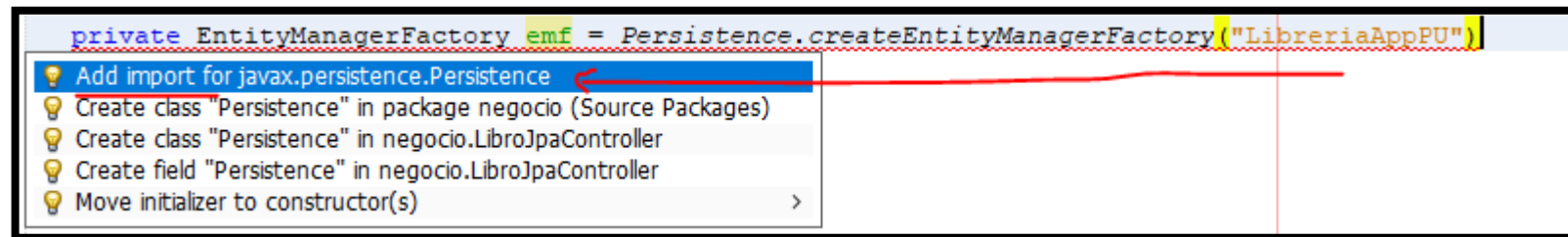
- Busque la línea de código:

```
private EntityManagerFactory emf = null;
```

- Y reemplace por:

```
private EntityManagerFactory emf =  
Persistence.createEntityManagerFactory("LibreriaAppPU");
```

- Agregue el import:

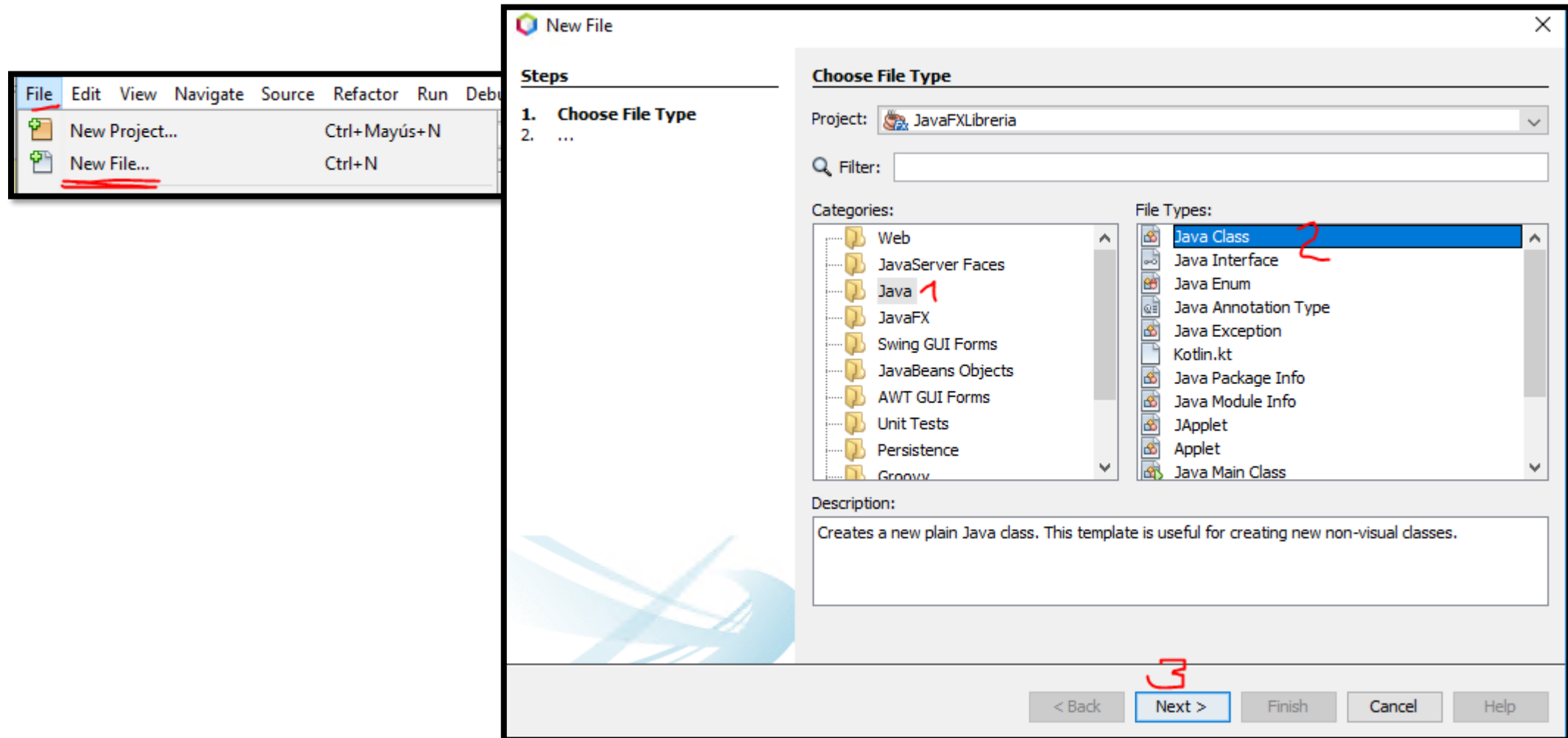


Guarde

- Guarde su trabajo

Capa de Negocio

Agregar la clase Facade (lógica de negocio)



Agregar la clase Facade (lógica de negocio)

- Ajuste el nombre del paquete

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: FacadeLibreria

Project: JavaFXLibreria

Location: Source Packages

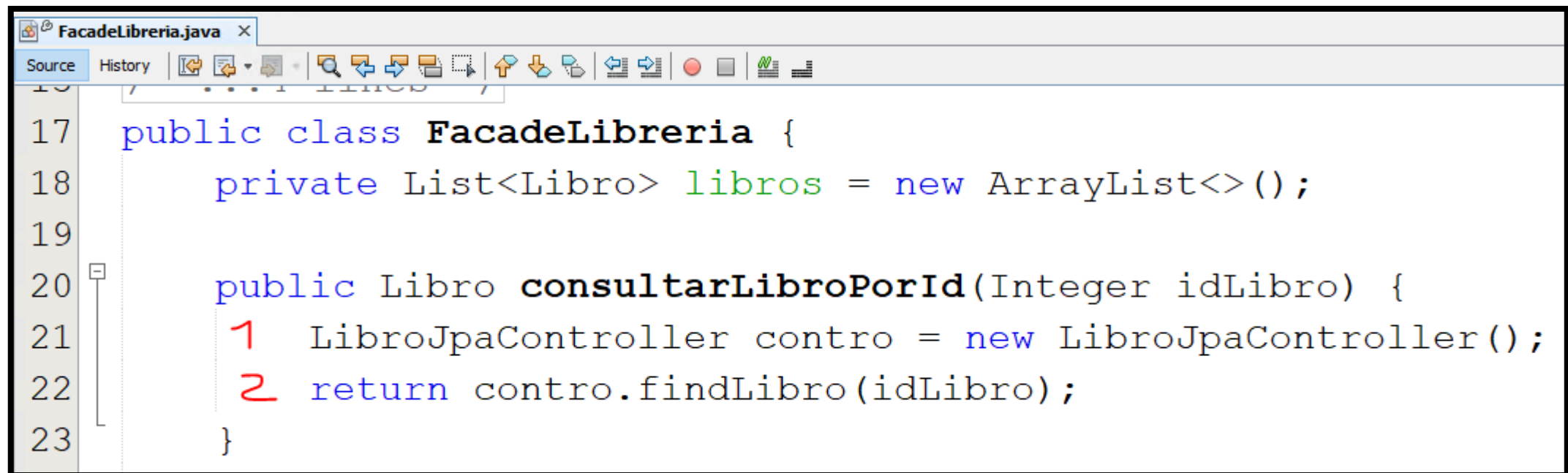
Package: javafxlibreria.facades 1

Created File: D:\Tools\nb12proys\JavaFXLibreria\src\javafxlibreria\facades\FacadeLibreria.java

< Back Next > **Finish** 2 Cancel Help

Agregue Método Negocio

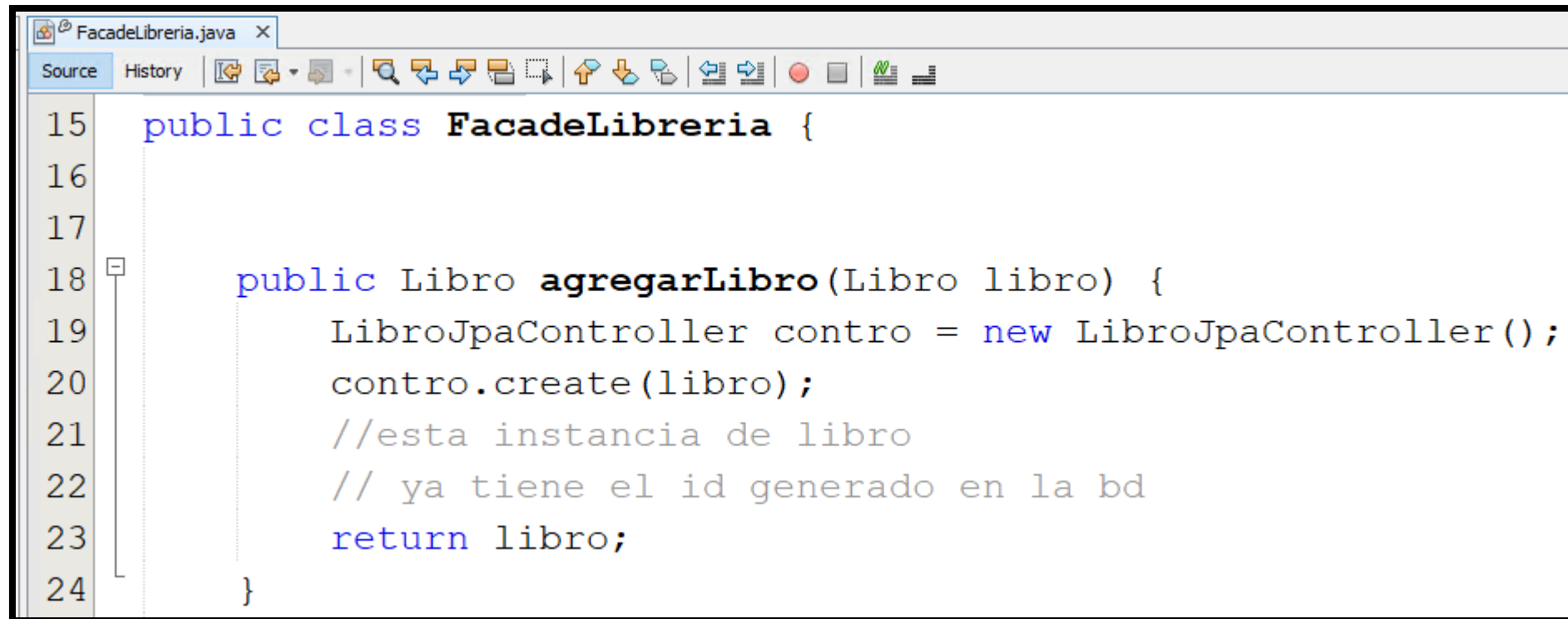
- Agregue y codifique el método consultarLibroPorId(Integer idLibro) de la siguiente forma:
 - (1) cree un controlador de acceso a base de datos
 - (2) retorne el libro usando el controlador



```
FacadeLibreria.java x
Source History
17 public class FacadeLibreria {
18     private List<Libro> libros = new ArrayList<>();
19
20     public Libro consultarLibroPorId(Integer idLibro) {
21         1 LibroJpaController contro = new LibroJpaController();
22         2 return contro.findLibro(idLibro);
23     }
```

Agregue Método Negocio

- Codifique el método agregarLibro(Libro libro) de la siguiente forma:

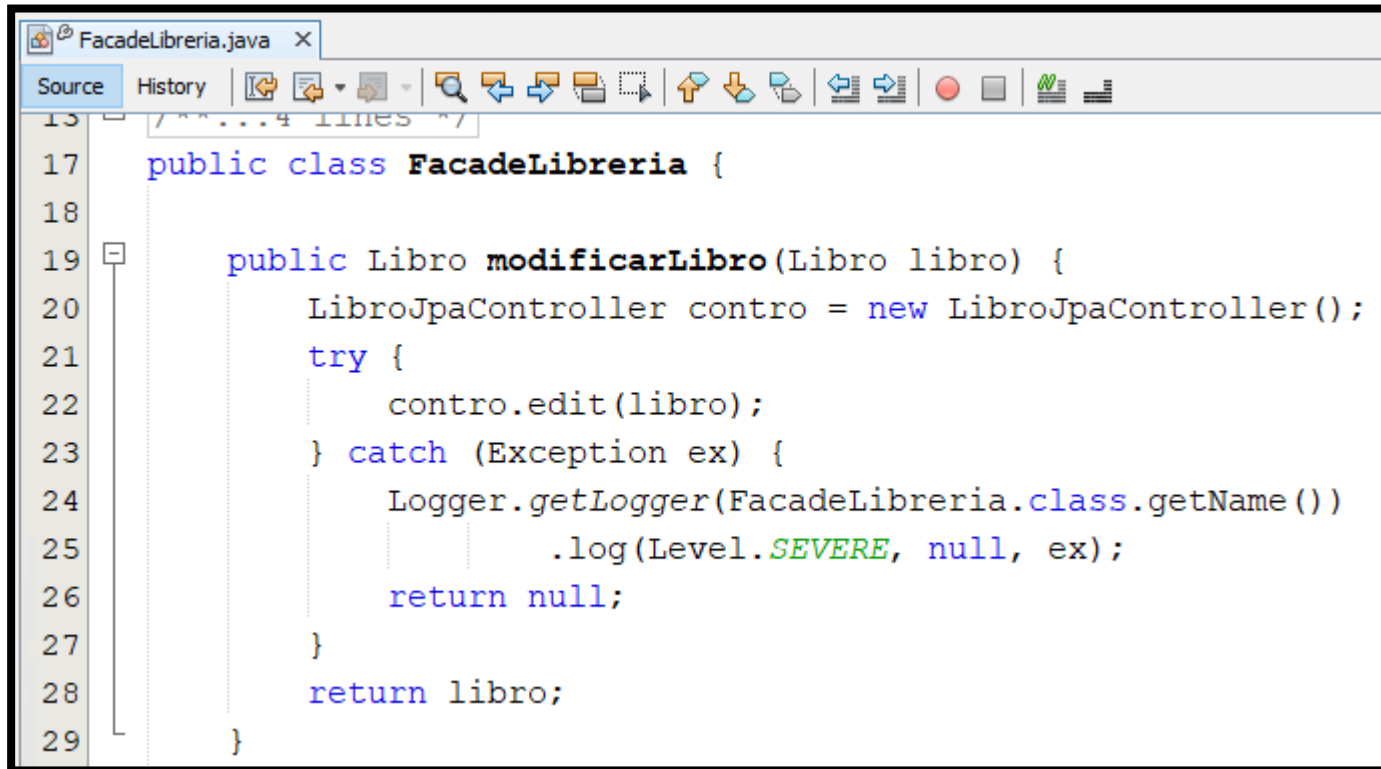


```
15 public class FacadeLibreria {
16
17
18     public Libro agregarLibro(Libro libro) {
19         LibroJpaController contro = new LibroJpaController();
20         contro.create(libro);
21         //esta instancia de libro
22         // ya tiene el id generado en la bd
23         return libro;
24     }
```

La instancia libro llega sin el id del libro; cuando se agrega el libro en la base de datos el atributo id ya trae el valor de la llave surrogate

Agregue Método Negocio

- Codifique el método `modificarLibro(Libro libro)` de la siguiente forma:

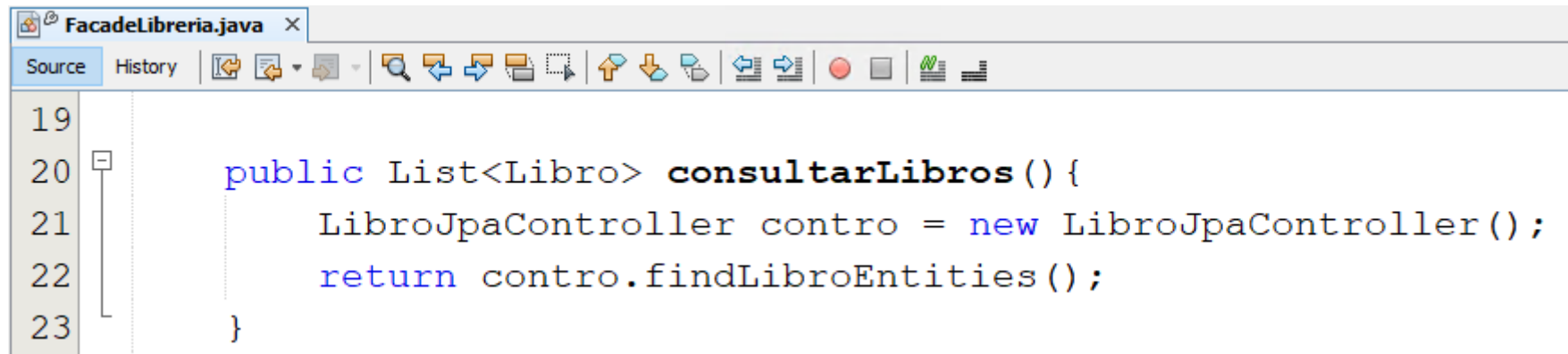


```
13  /** ...4 lines ... */
17  public class FacadeLibreria {
18
19      public Libro modificarLibro(Libro libro) {
20          LibroJpaController contro = new LibroJpaController();
21          try {
22              contro.edit(libro);
23          } catch (Exception ex) {
24              Logger.getLogger(FacadeLibreria.class.getName())
25                  .log(Level.SEVERE, null, ex);
26              return null;
27          }
28          return libro;
29      }
```

La instancia libro debe llegar con el id del libro

Agregue Método Negocio

- Codifique el método consultarLibros() de la siguiente forma:



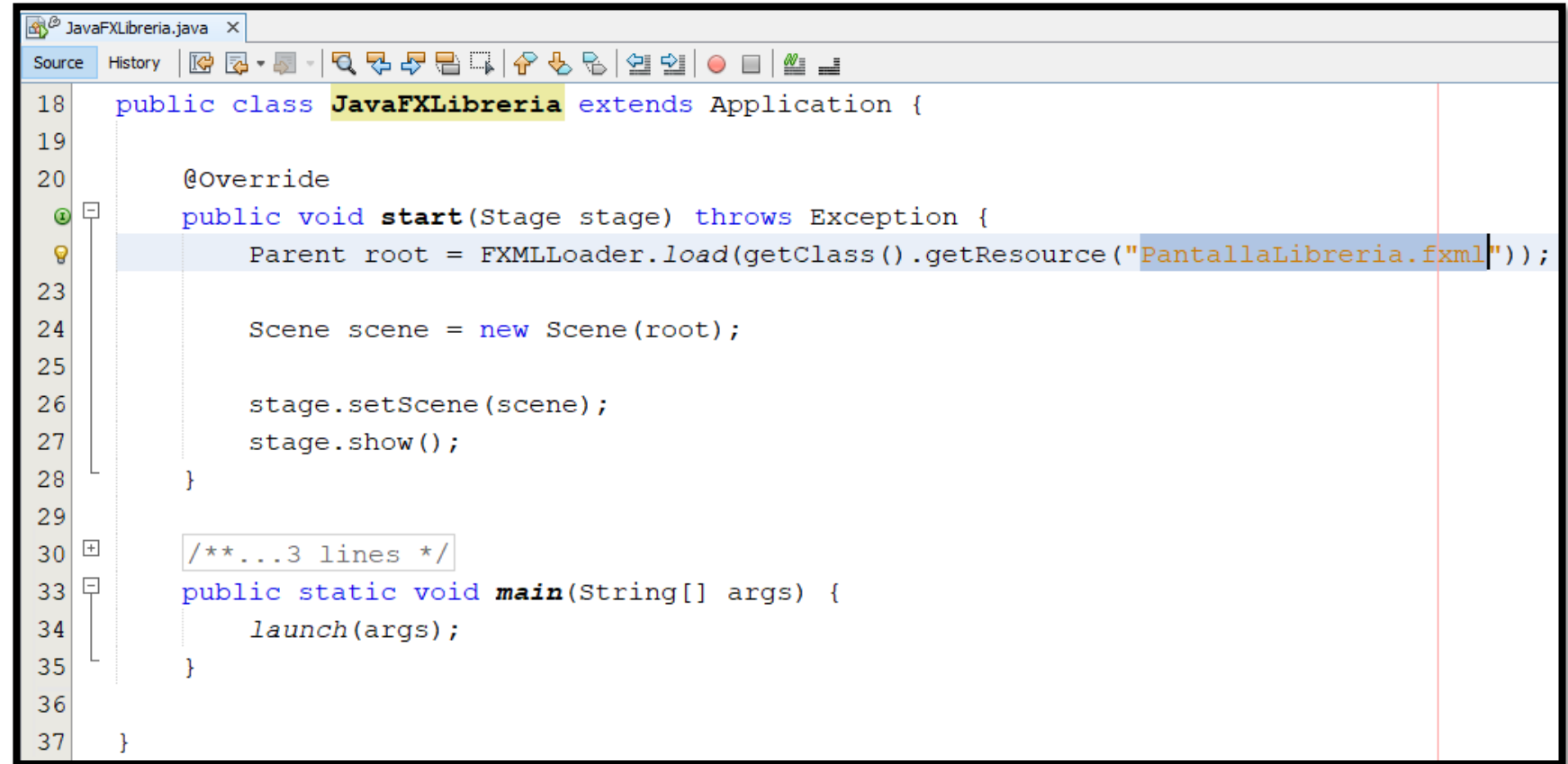
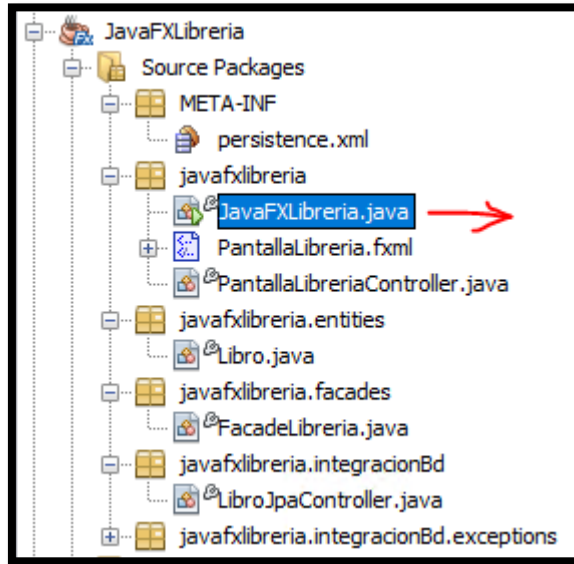
The screenshot shows an IDE window titled "FacadeLibreria.java". The editor displays the following Java code:

```
19  
20 public List<Libro> consultarLibros() {  
21     LibroJpaController contro = new LibroJpaController();  
22     return contro.findLibroEntities();  
23 }
```

Capa de Presentación

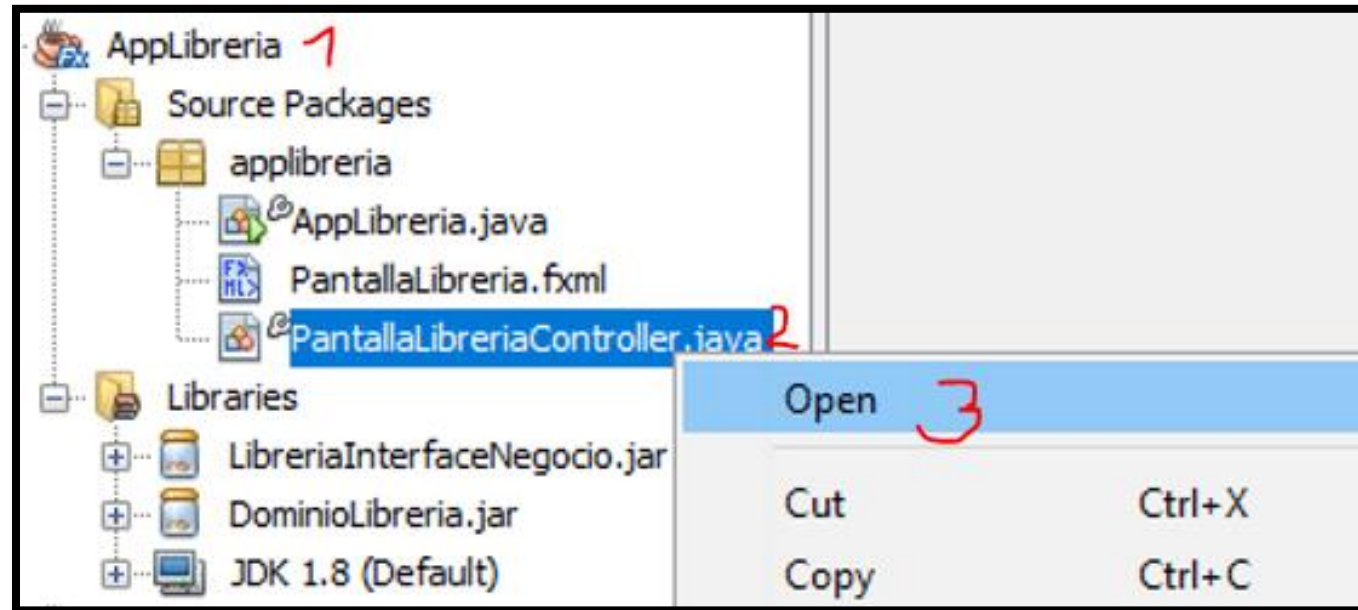
JavaFXLibreria.java

- Es la clase inicial que inicia el programa y carga la PantallaLibreria.fxml



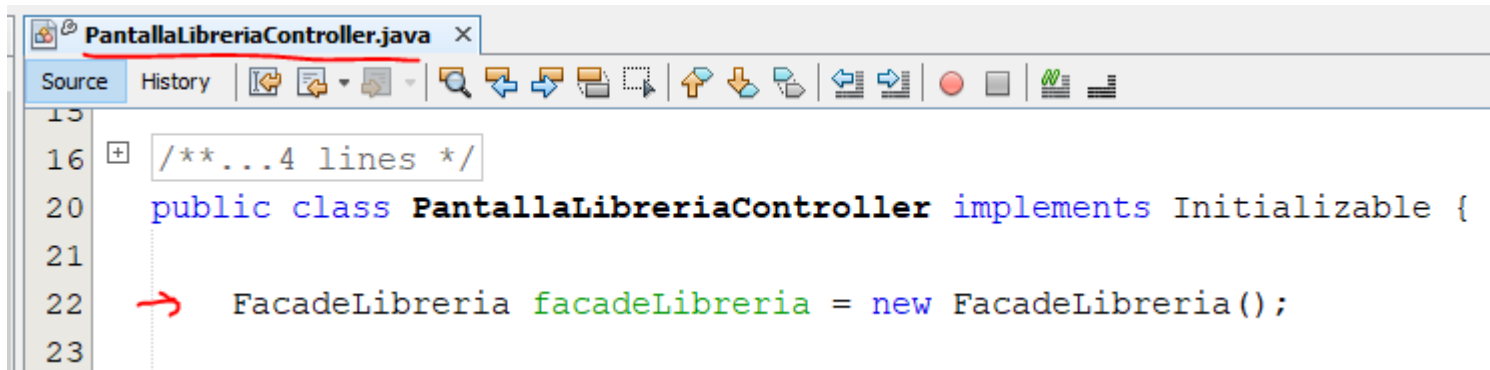
PantallaLibreriaController

- Abra el controlador de eventos



PantallaLibreriaController

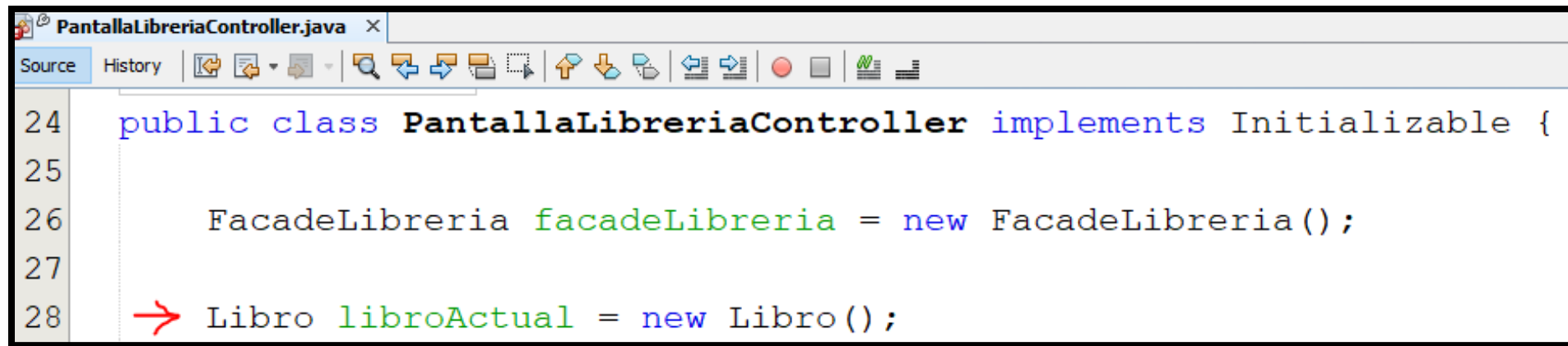
- Declare e instancie un objeto de tipo FacadeLibreria



```
15
16  /**...4 lines */
20  public class PantallaLibreriaController implements Initializable {
21
22  →  FacadeLibreria facadeLibreria = new FacadeLibreria();
23
```

PantallaLibreriaController

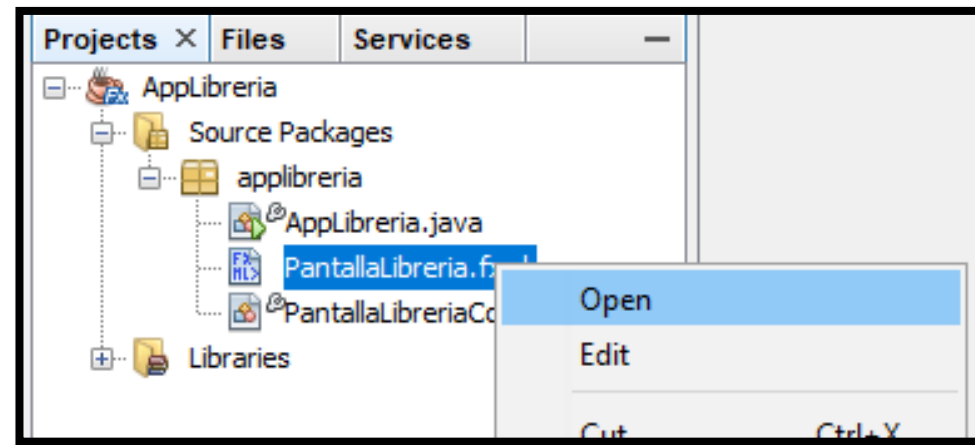
- Declare e instancie un objeto de tipo Libro que se mantendrá en memoria



```
PantallaLibreriaController.java x
Source History
24 public class PantallaLibreriaController implements Initializable {
25
26     FacadeLibreria facadeLibreria = new FacadeLibreria();
27
28     ➔ Libro libroActual = new Libro();
```

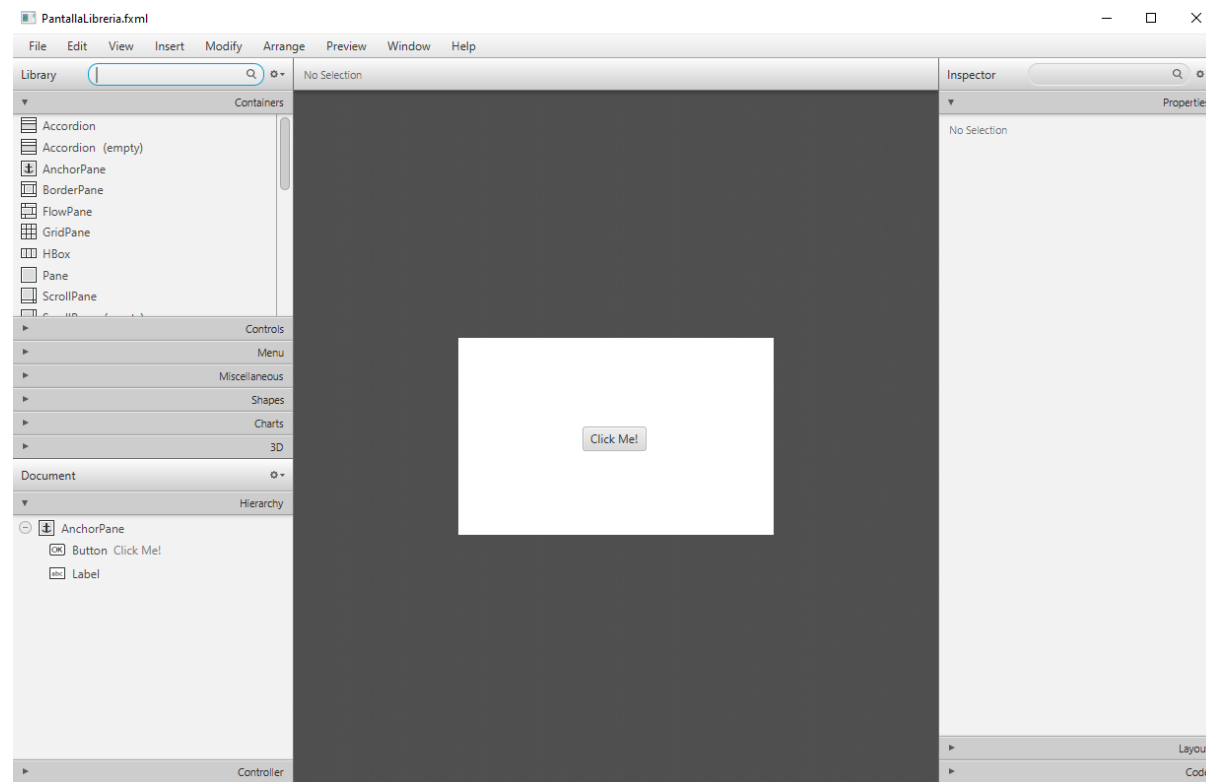
Cambiando la PantallaLibreria

- Abrir el Scene Builder para construir los objetos visuales en la Pantalla para solicitar datos y crear un nuevo Libro



Cambiando la PantallaLibreria

- Se abre el Scene Builder mostrando un botón creado por defecto por el NetBeans



Cambiando la PantallaLibreria

- Agrande el lienzo desde la esquina (vea la figura)



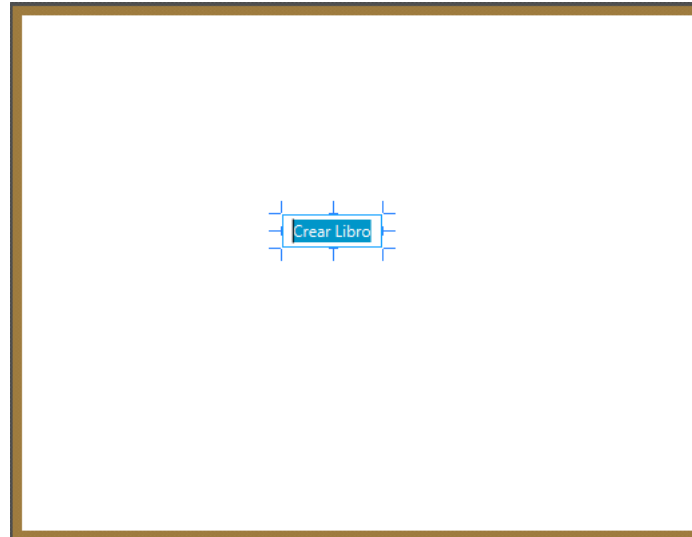
Cambiando la PantallaLibreria

- Arrastre y suelte para mover objetos en la Pantalla



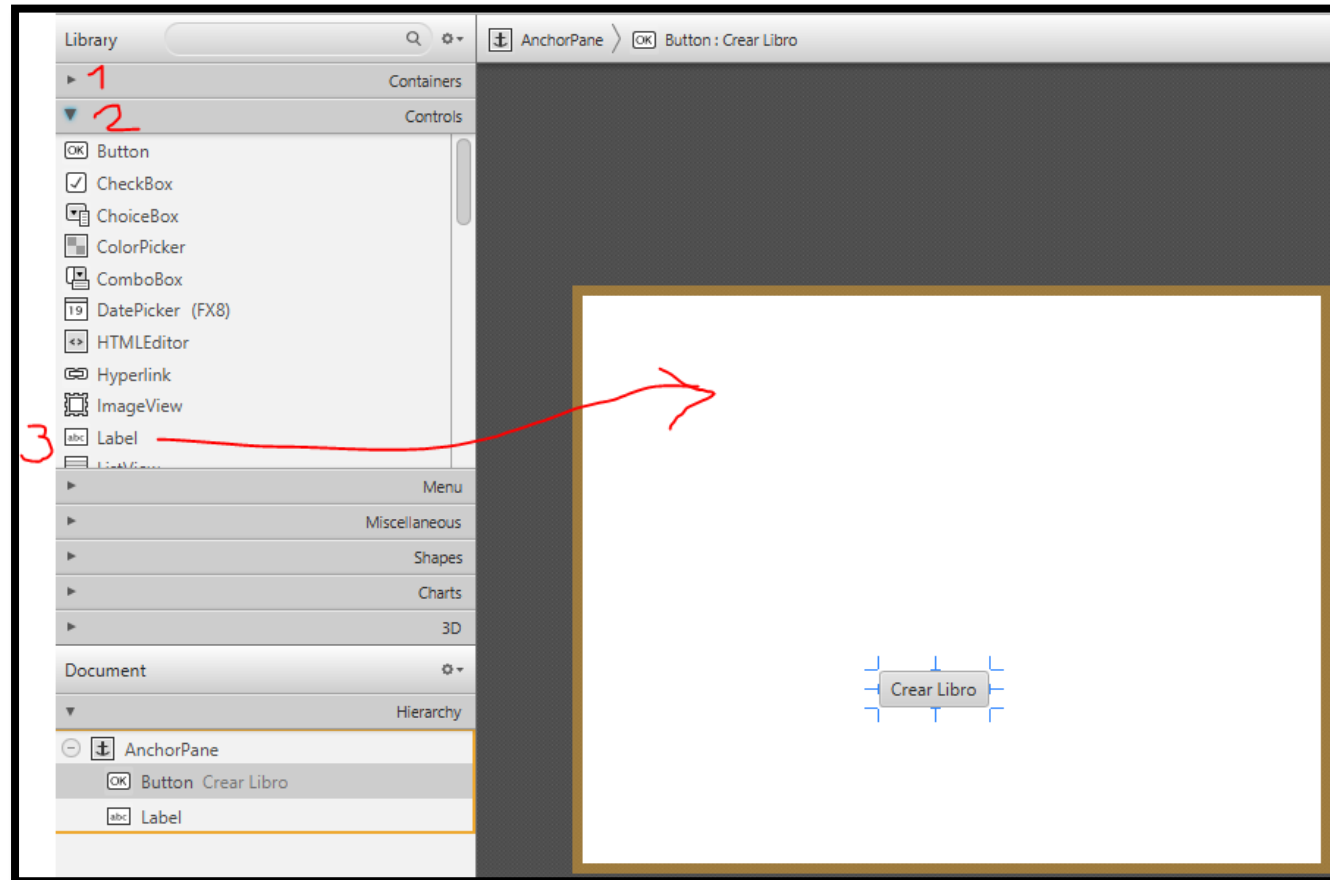
Cambiando la PantallaLibreria

- Doble click en el botón y cambie el texto a 'Crear Libro'



Cambiando la PantallaLibreria

- Expanda los controles ; busque Label y arrastre y suelte al lienzo



Cambiando la PantallaLibreria



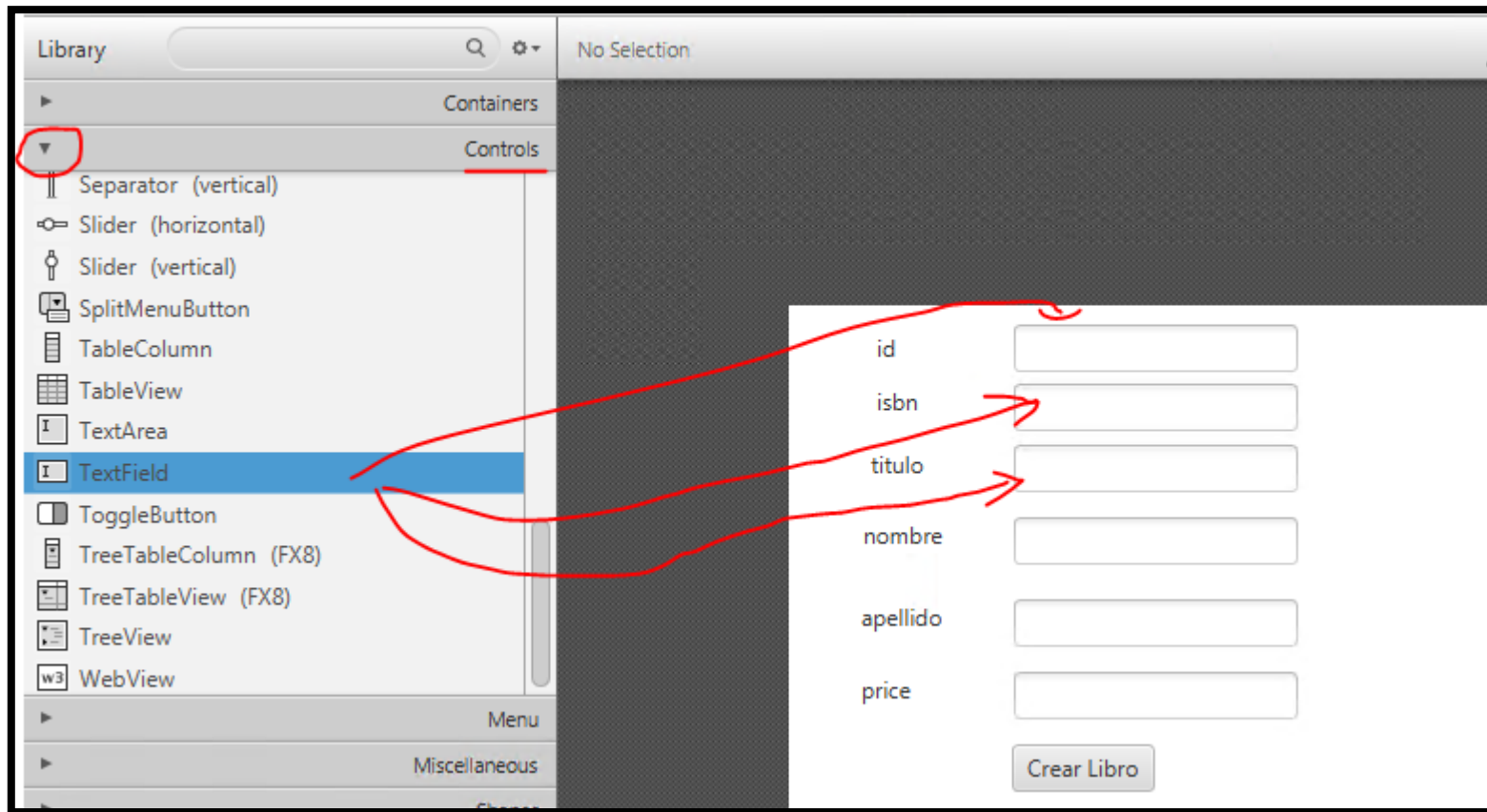
A screenshot of a web form titled "Crear Libro". The form contains six text input fields labeled "id", "isbn", "titulo", "nombre", "apellido", and "price". A "Crear Libro" button is located at the bottom right of the form.



El id no se introduce desde la pantalla; se genera como llave surrogate en la BD

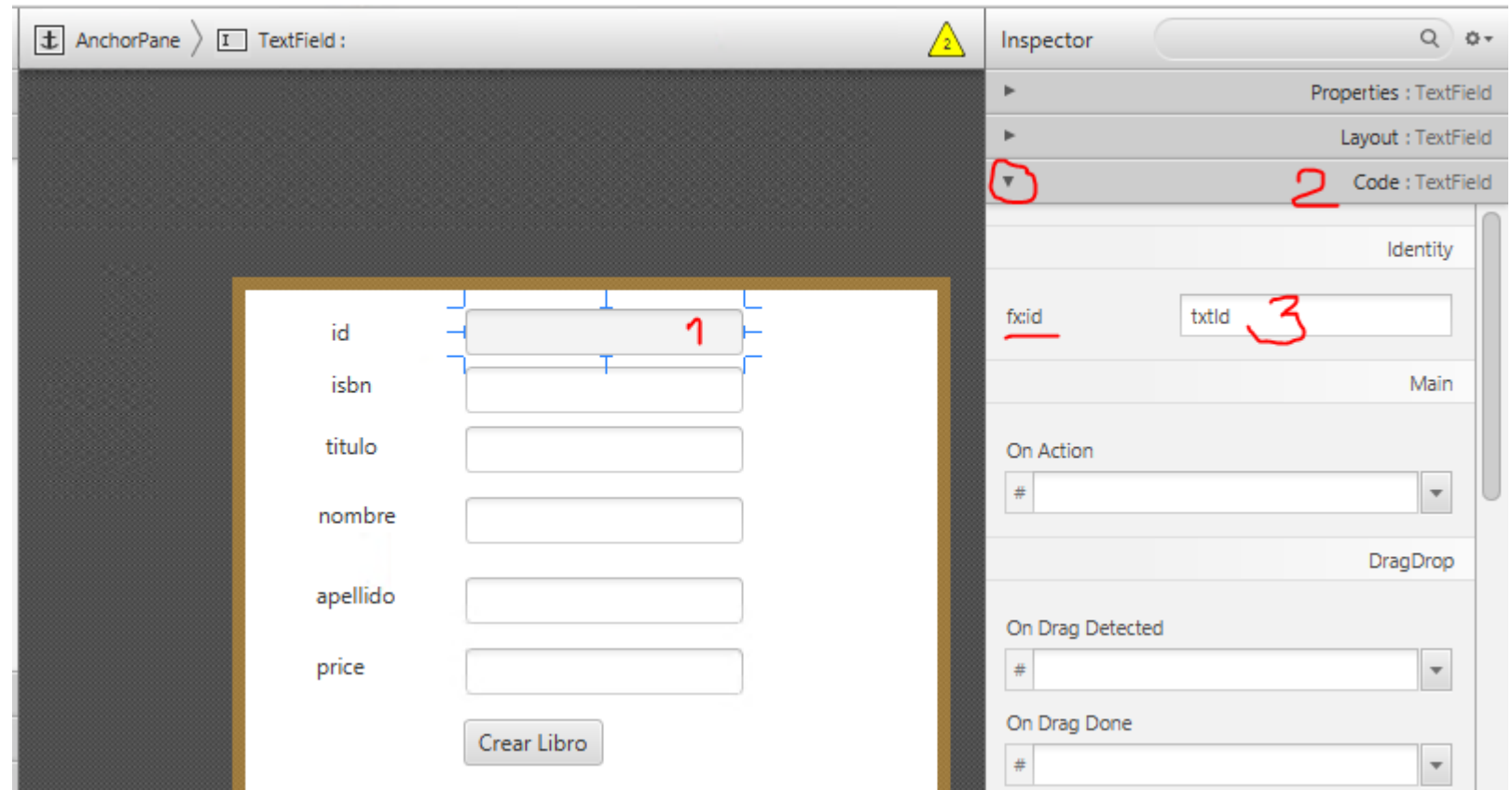
Cambiando la PantallaLibreria

- Expanda los controles ; busque TextField y arrastre y suelte al lienzo para completar la entrada de datos de los atributos del Libro



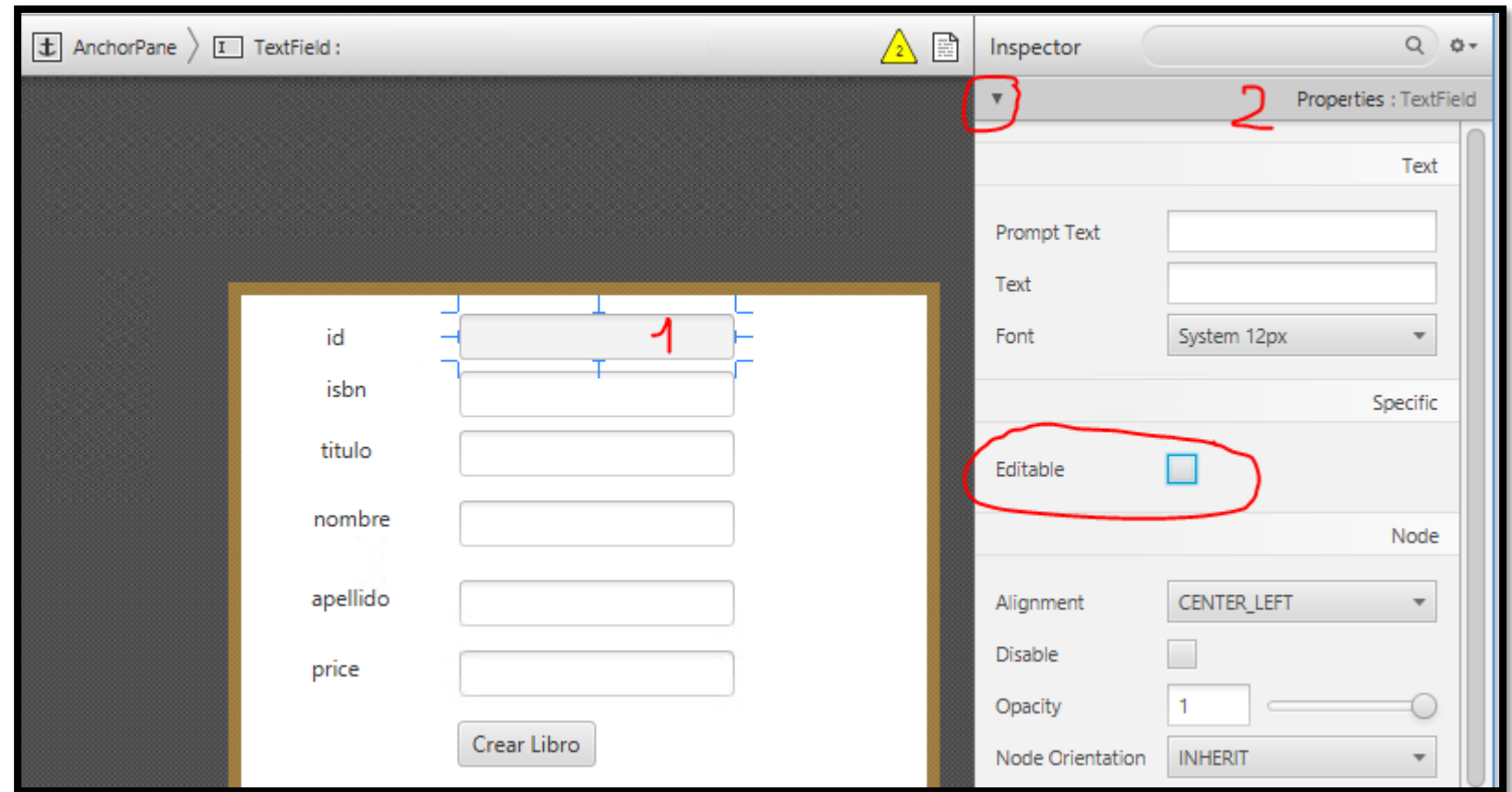
Cambiando la PantallaLibreria

- Asigne un nombre único a cada TextField
 - Use la notación txtXXX; donde XXX es el nombre del elemento visual; para el ejemplo **txtId**



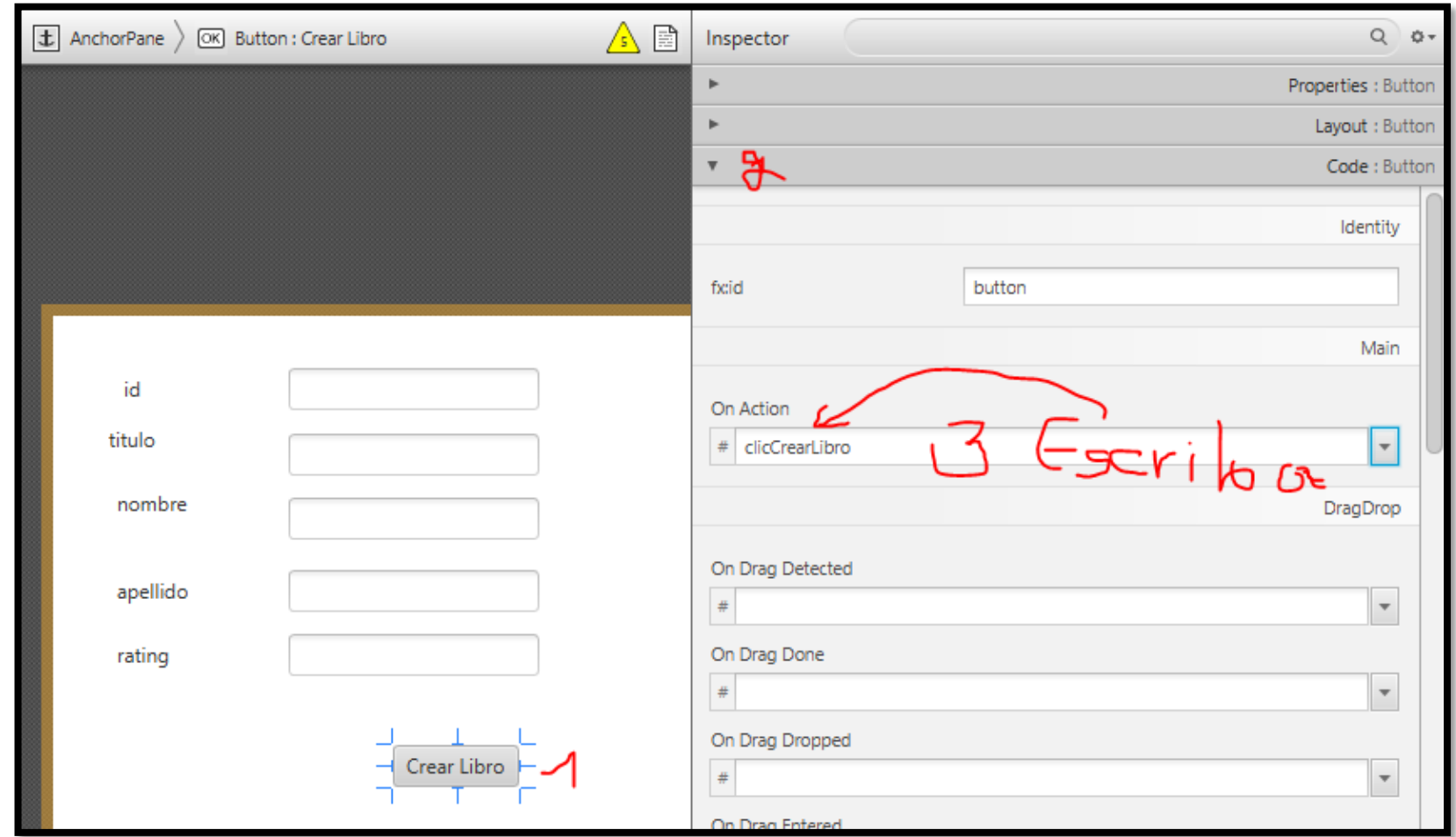
Cambiando la PantallaLibreria

- El txtId es de solo lectura (la llave surrogate no se debería modificar)

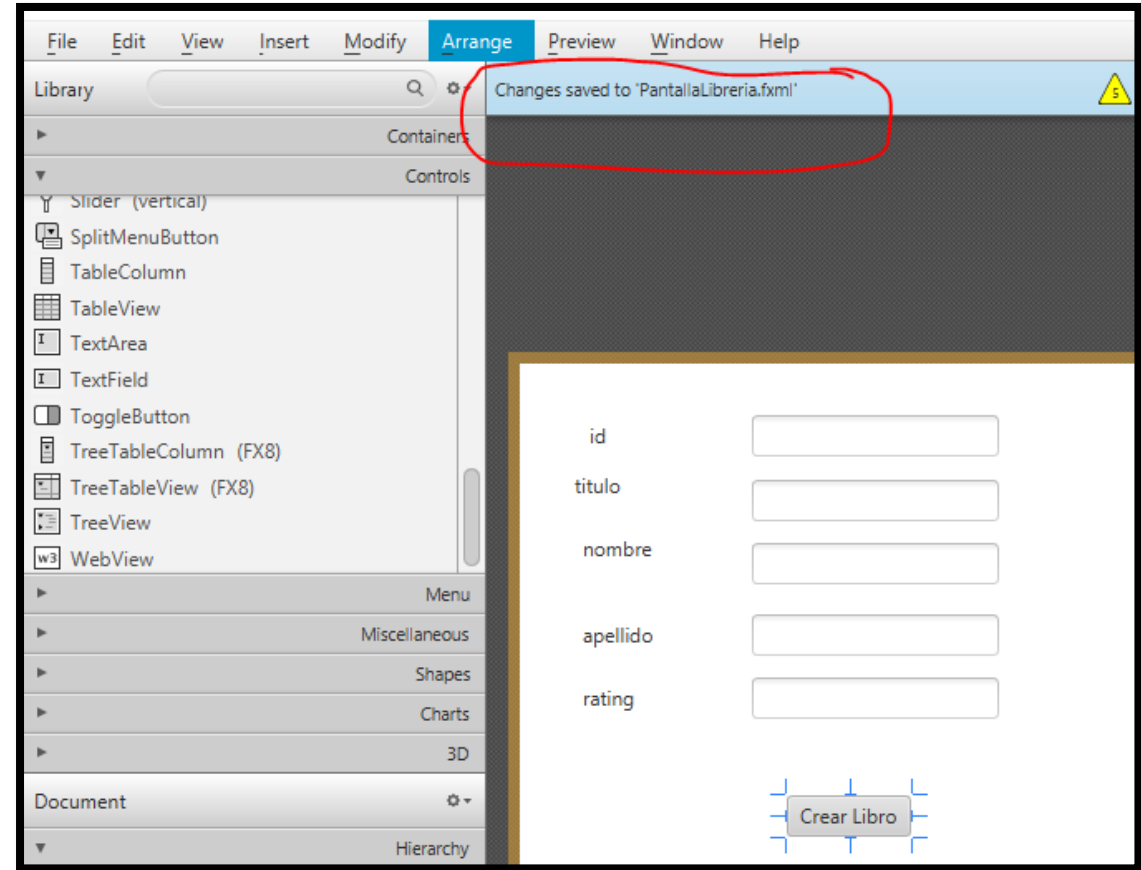
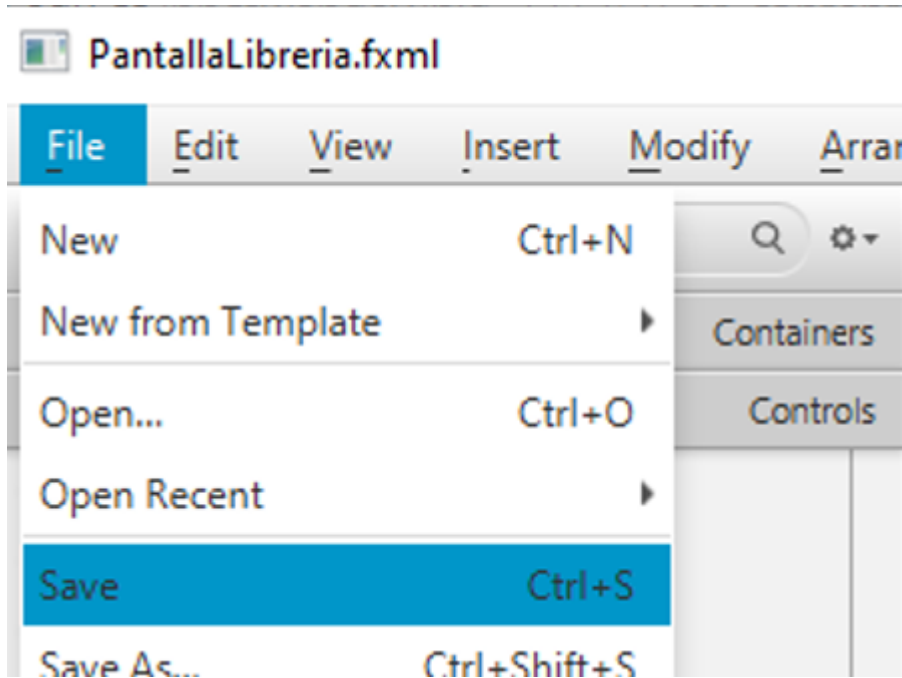


Cambiando la Pantalla Libreria

- Asigne un método para el manejar el evento de hacer click en el botón

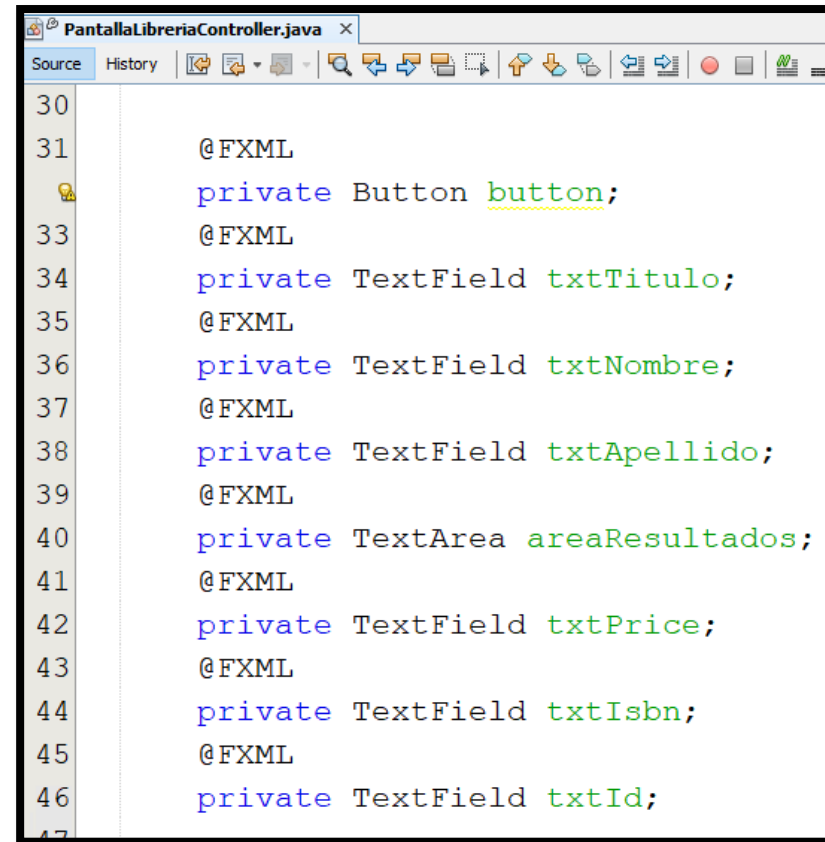
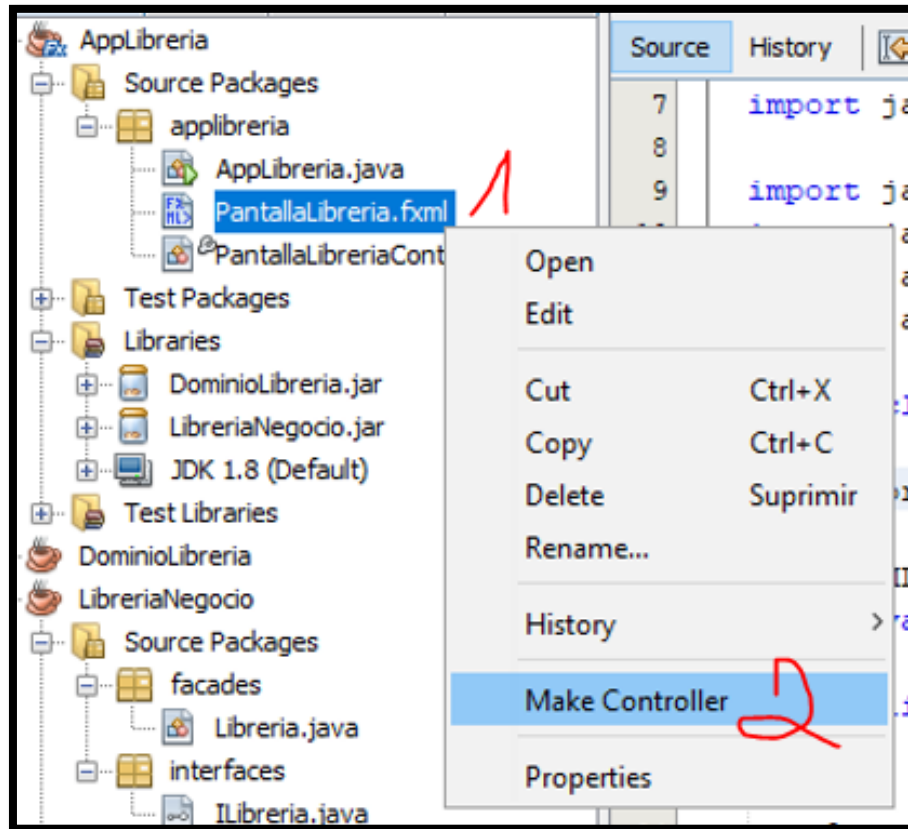


Sincronizar Scene con el código en el controlador



Sincronizar Scene con el código en el controlador

- Ubíquese sobre la Vista, esto es, el archivo PantallaLibreria.fxml; el archivo PantallaLibreriaController.java se actualizará creando atributos para cada elemento visual

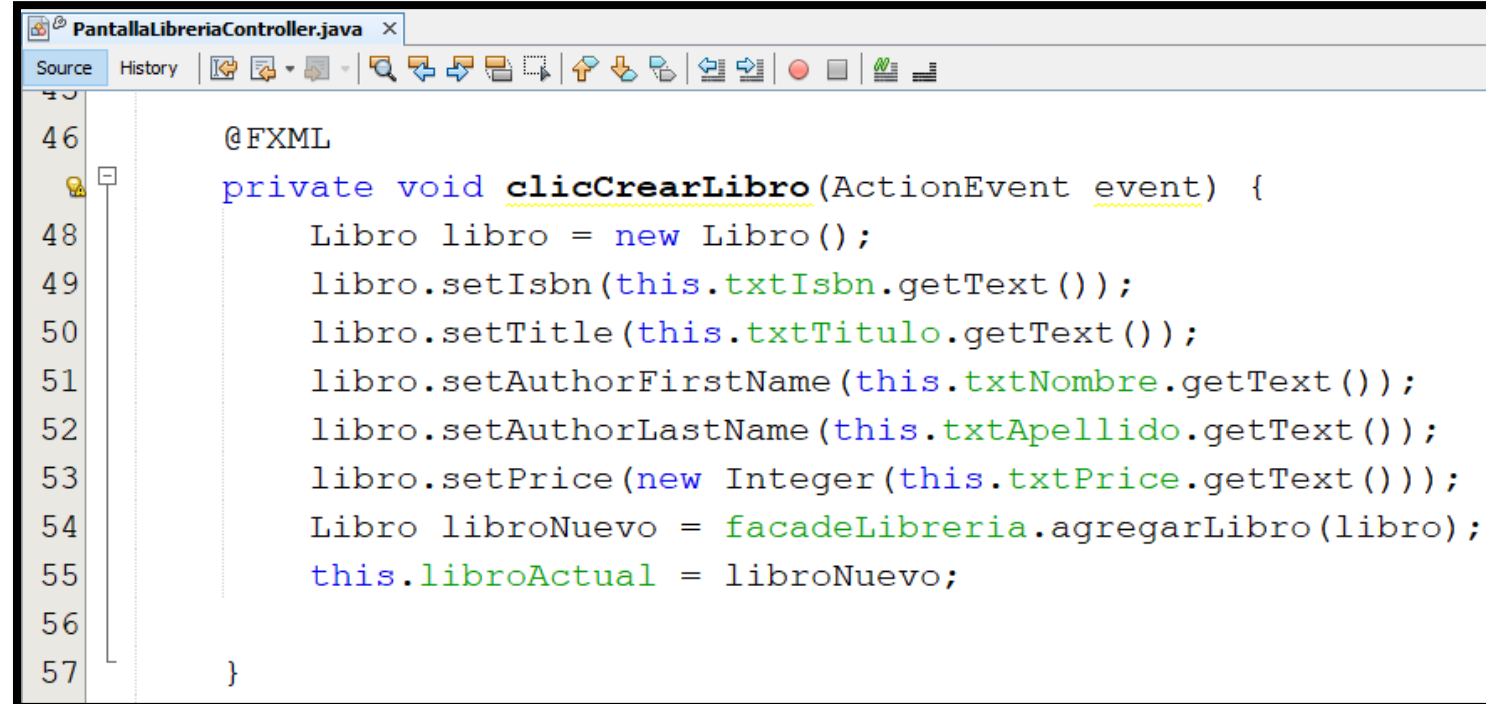


Codificando el método de clicCrearLibro()

- Observe:
 - Se creó un Libro
 - Se asignan valores
 - Para obtener el valor de la caja de texto :

this.txtIsbn.getText()

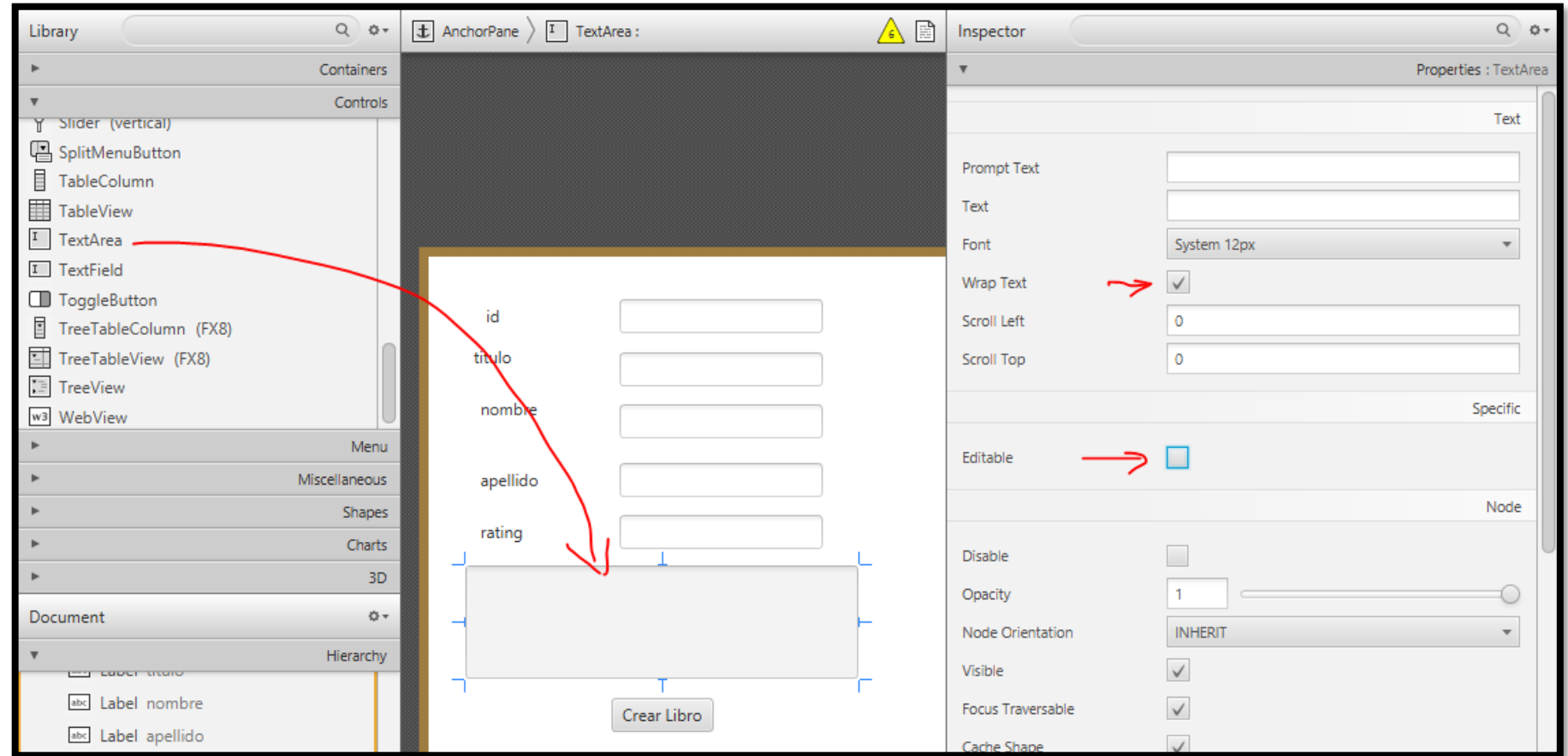
- Para el precio se usa una transformación de cadena a Integer
- Al final del método se invoca la fachada de librería usando el método agregarLibro
- Se asigna el libro creado a la instancia libroActual



```
PantallaLibreriaController.java
Source History
46 @FXML
47 private void clicCrearLibro(ActionEvent event) {
48     Libro libro = new Libro();
49     libro.setIsbn(this.txtIsbn.getText());
50     libro.setTitle(this.txtTitulo.getText());
51     libro.setAuthorFirstName(this.txtNombre.getText());
52     libro.setAuthorLastName(this.txtApellido.getText());
53     libro.setPrice(new Integer(this.txtPrice.getText()));
54     Libro libroNuevo = facadeLibreria.agregarLibro(libro);
55     this.libroActual = libroNuevo;
56
57 }
```

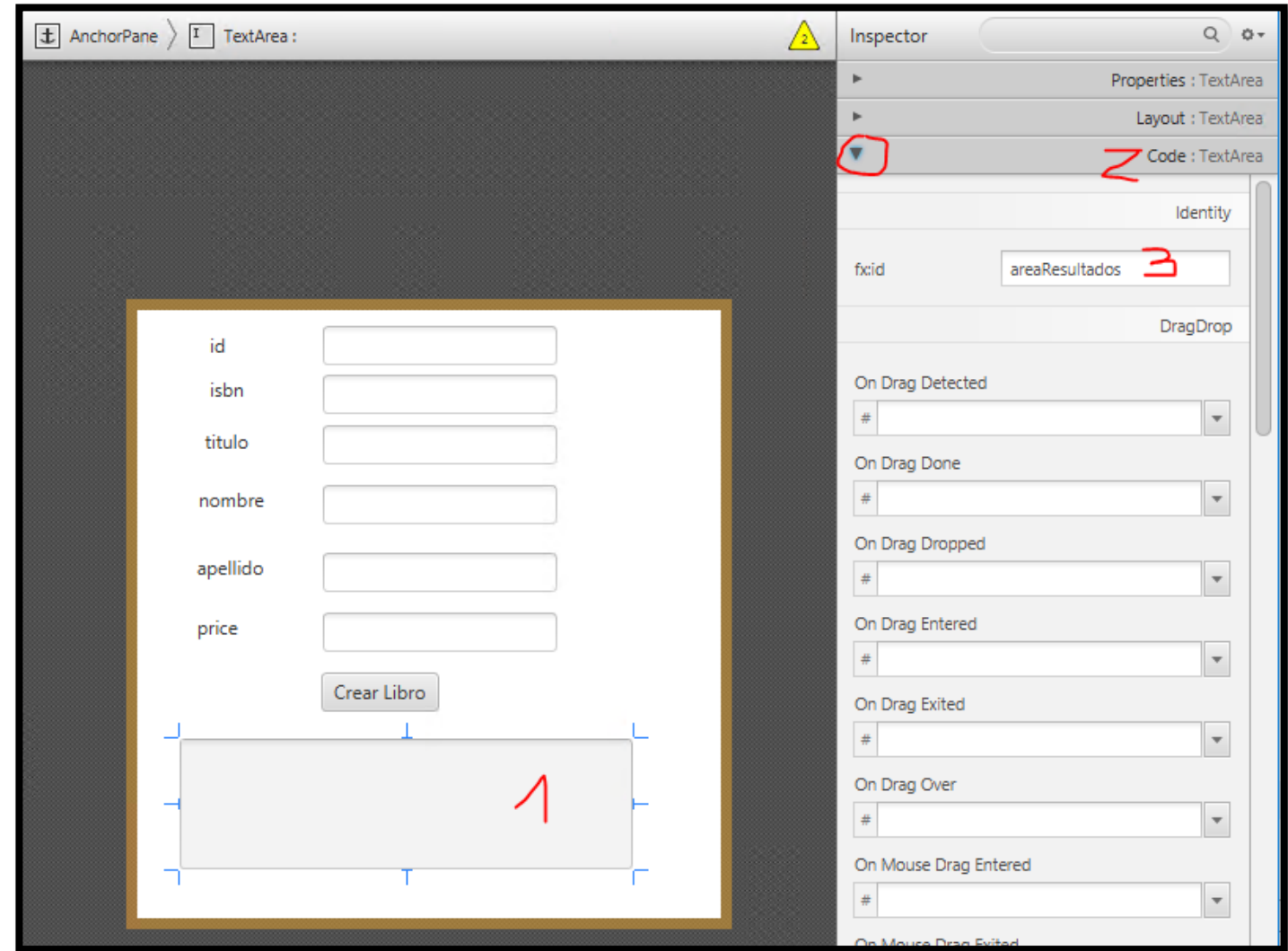
Cambiando la PantallaLibreria

- Agreguemos una TextArea para mostrar el libro agregado
- Actualice el código en el controlador
 - Guardar y Make Controller



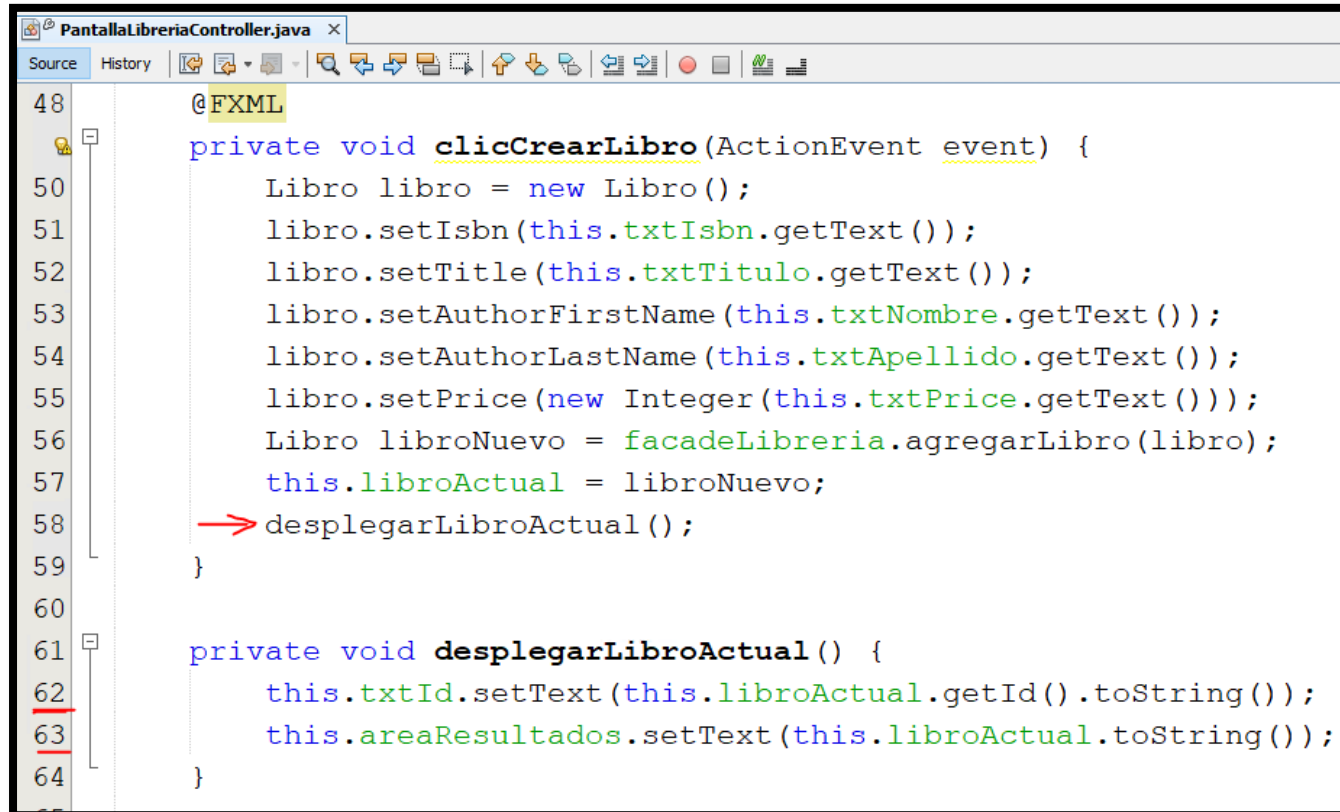
Cambiando la PantallaLibreria

- Cambie la propiedad Wrap Text y Editable de la etiqueta de resultados



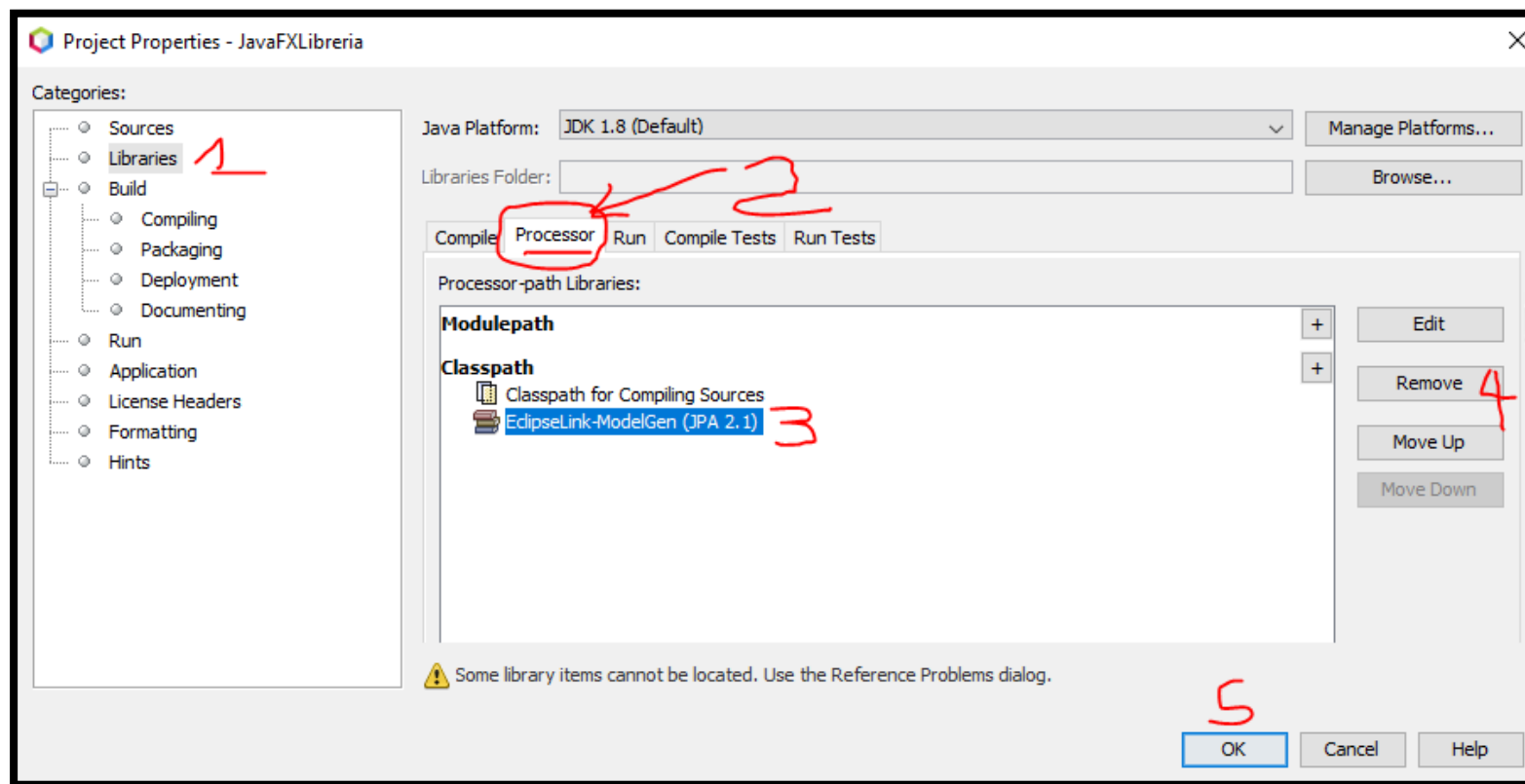
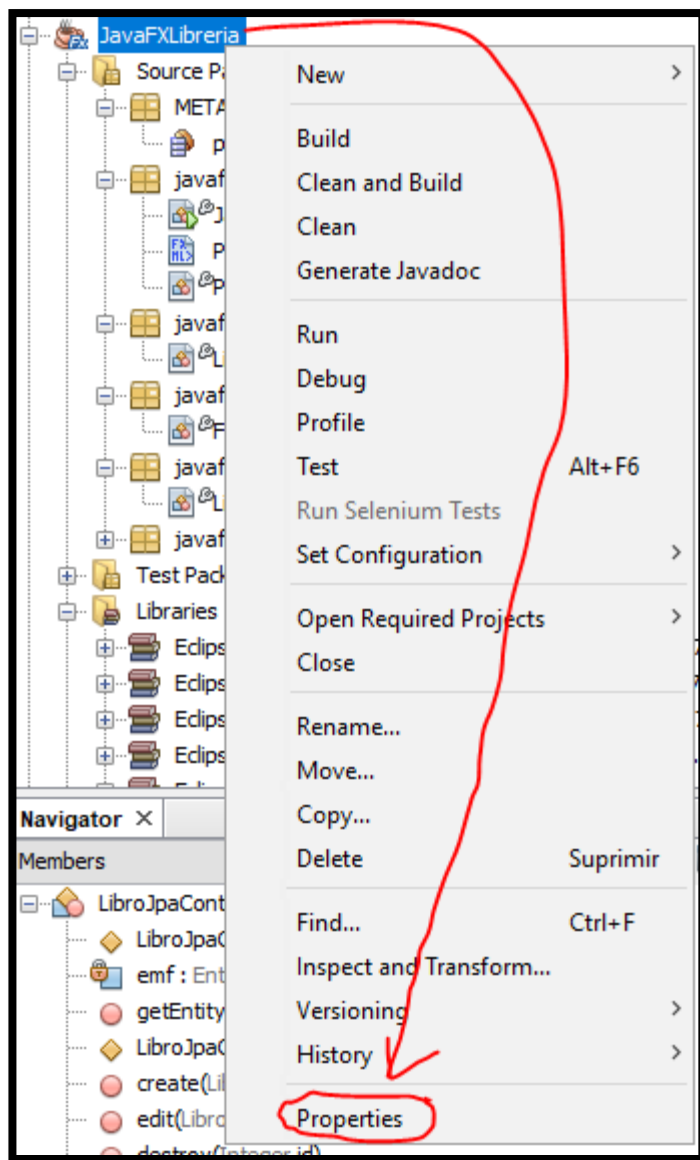
Codificando el método de CrearLibro()

- Agregue la línea marcada en rojo en la figura
- Línea 62: se muestra el valor del id generado en la BD
- Línea 63: Observe que para asignar el valor a la TextArea se usa el método setText

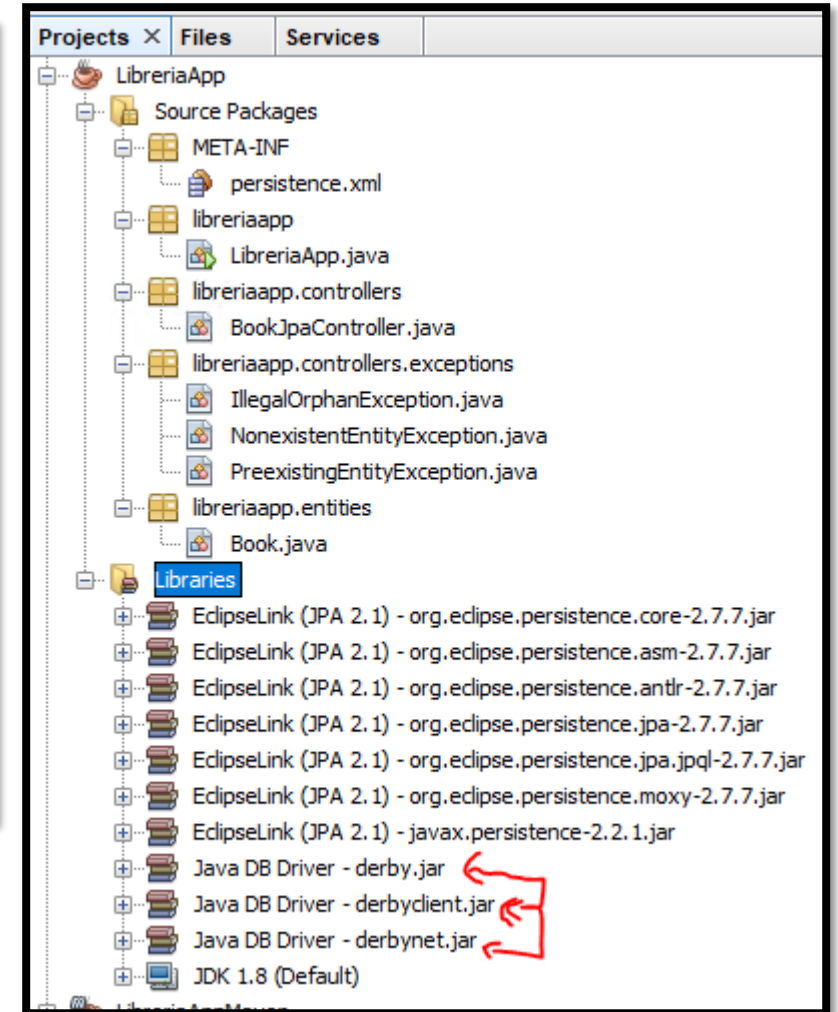
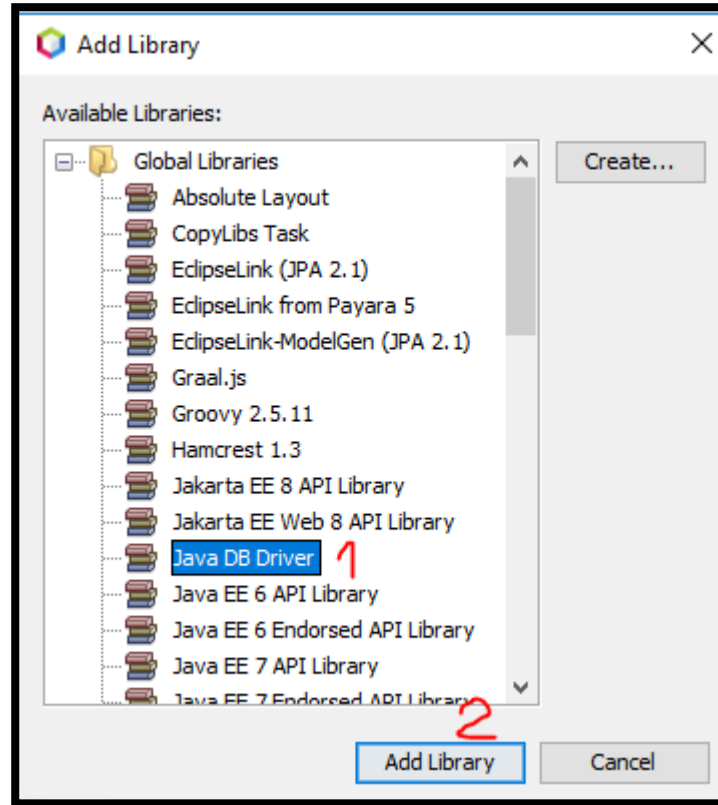
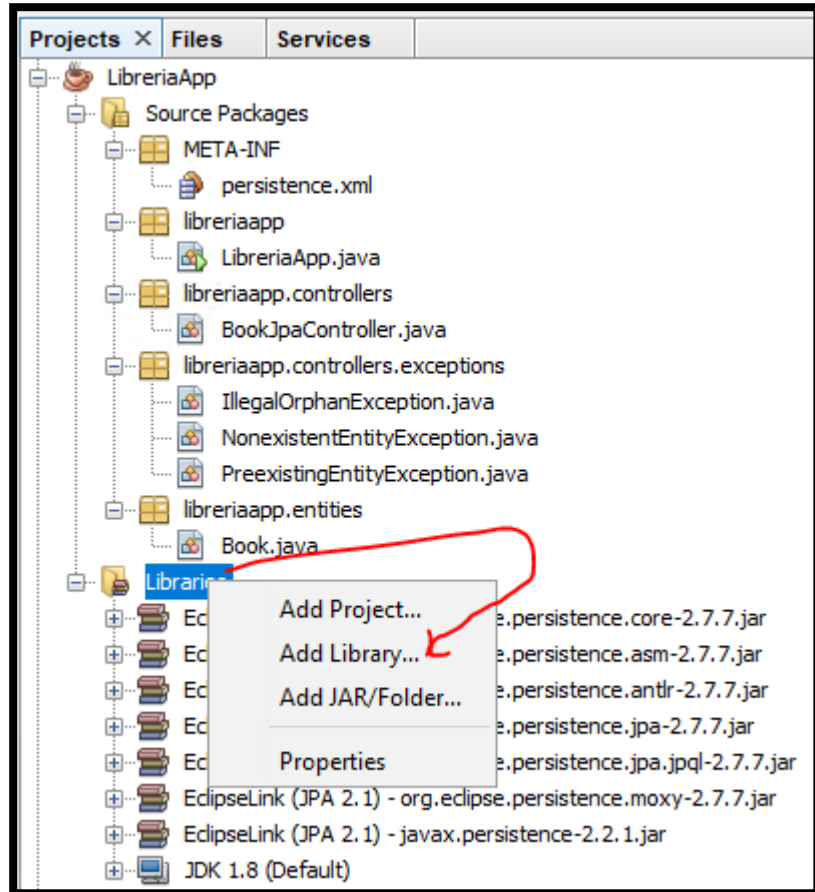


```
PantallaLibreriaController.java
Source History
48 @FXML
49 private void clicCrearLibro(ActionEvent event) {
50     Libro libro = new Libro();
51     libro.setIsbn(this.txtIsbn.getText());
52     libro.setTitle(this.txtTitulo.getText());
53     libro.setAuthorFirstName(this.txtNombre.getText());
54     libro.setAuthorLastName(this.txtApellido.getText());
55     libro.setPrice(new Integer(this.txtPrice.getText()));
56     Libro libroNuevo = facadeLibreria.agregarLibro(libro);
57     this.libroActual = libroNuevo;
58     ➔ desplegarLibroActual();
59 }
60
61 private void desplegarLibroActual() {
62     this.txtId.setText(this.libroActual.getId().toString());
63     this.areaResultados.setText(this.libroActual.toString());
64 }
```

Compilando y construyendo
(Build) el JAR

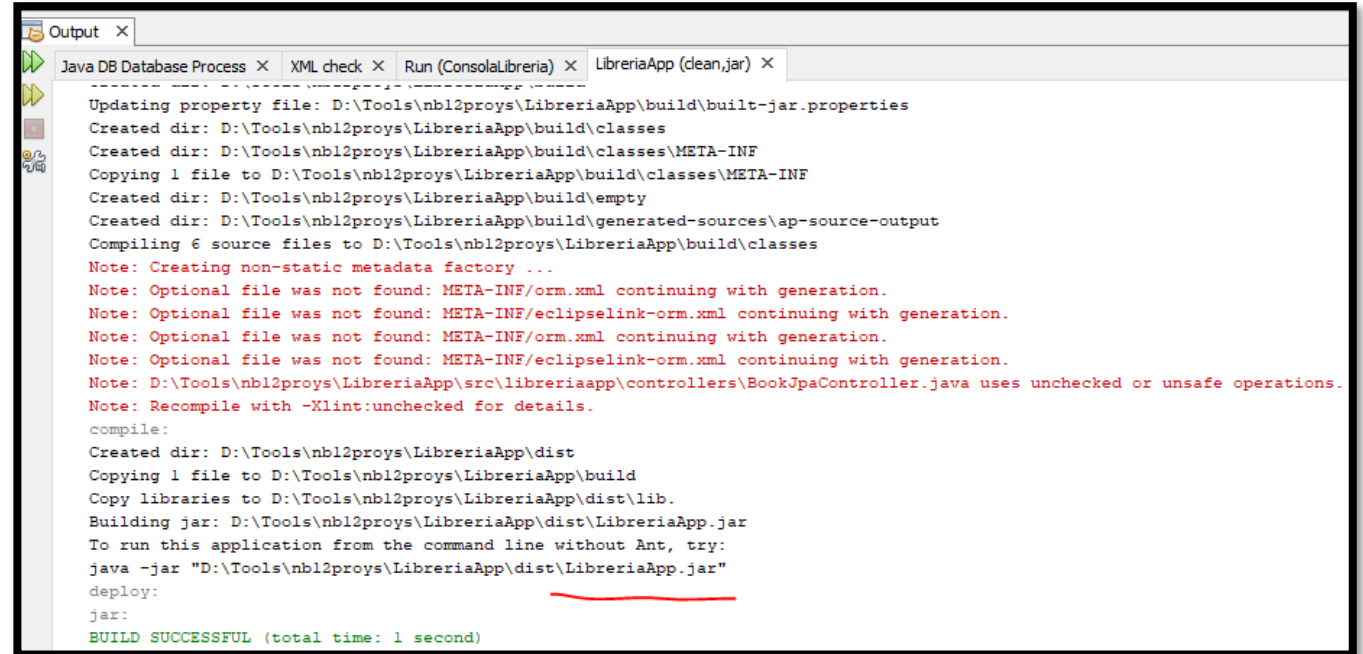
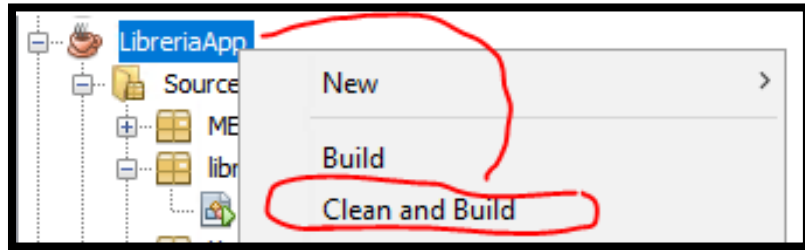


Agregue la librería del Driver JDBC



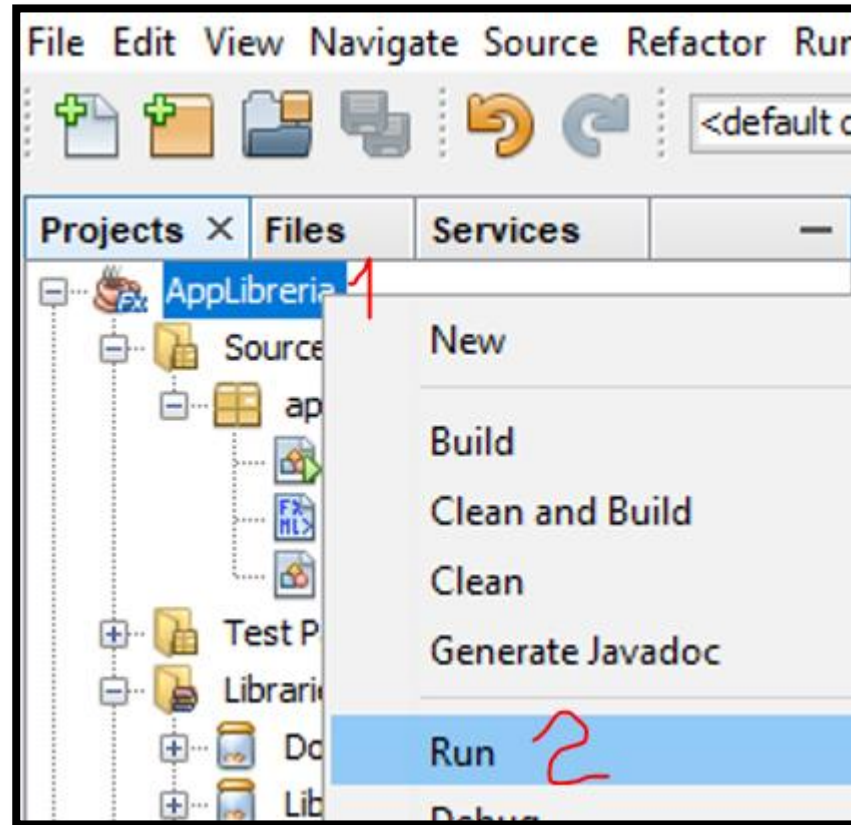
Limpie y Construya

- Clean and Build



Ejecute

- Al ejecutar NB compila, empaqueta (crea el jar) y ejecuta



Ejecute

- Agregue un Libro

A screenshot of a web application window titled "Agregar Libro". The form contains the following fields and values:

| Field | Value | Annotation |
|--|---------|------------|
| id | | |
| isbn | A12345 | 1 |
| titulo | bd | 2 |
| nombre | julio | 3 |
| apellido | ernesto | 4 |
| price | 200 | 5 |
| <input type="button" value="Crear Libro"/> | | 6 |

Below the form is a large empty rectangular box with a blue border.

A screenshot of the same web application window after the "Crear Libro" button has been clicked. The "id" field now contains the value "12", which is underlined in red. A red "8" is written next to the "id" field. The "Crear Libro" button is now disabled (grayed out). A confirmation message is displayed in a box at the bottom:

```
Libro{id=12, isbn=A12345, title=bd, authorLastName=ernesto, authorFirstName=julio, price=200}
```

A red plus sign is written next to the confirmation message box.

Termine el Programa



Agregue botón Modificar; agregue el código mostrado

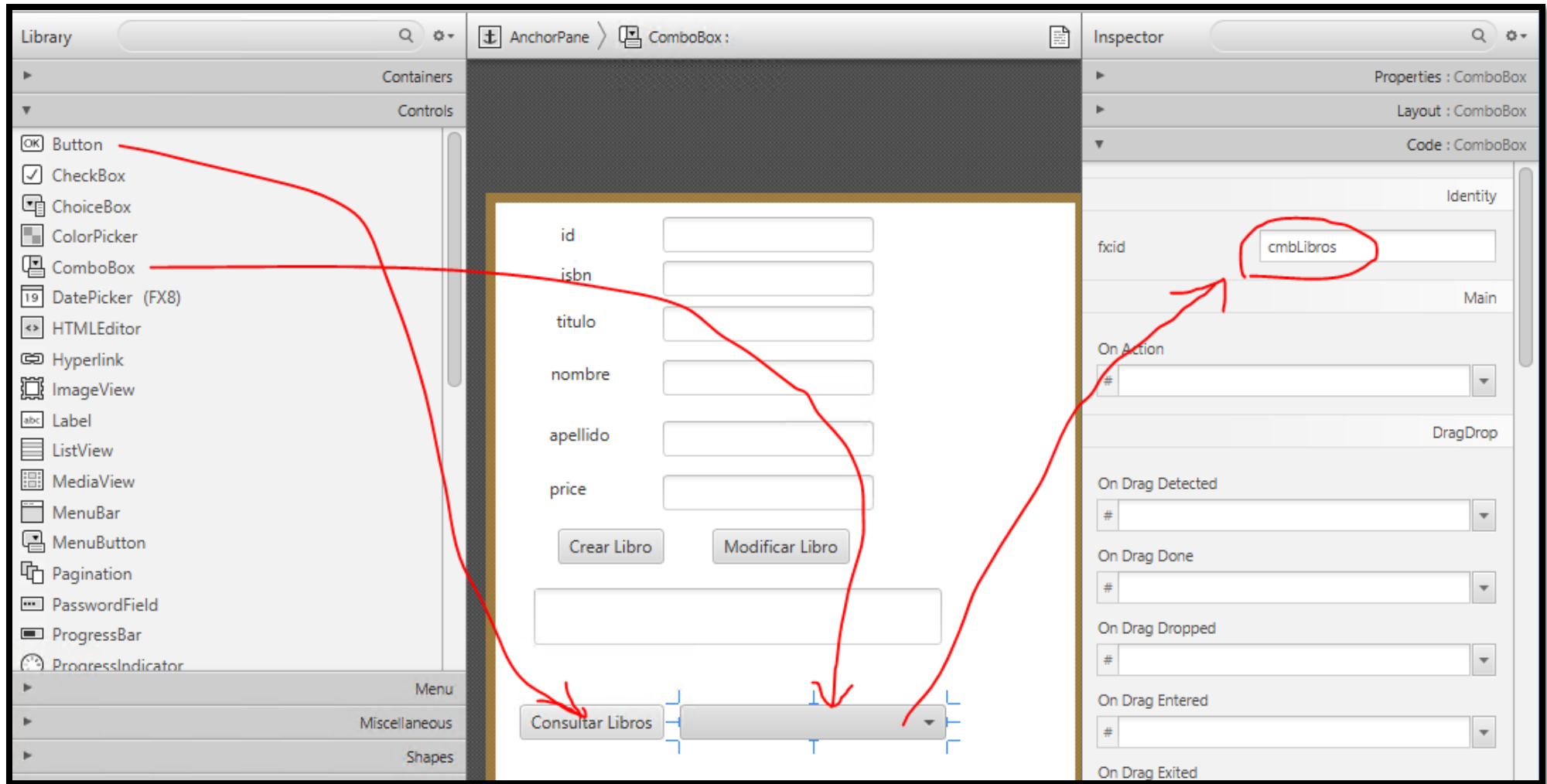
| | |
|---|----------------------|
| id | <input type="text"/> |
| isbn | <input type="text"/> |
| titulo | <input type="text"/> |
| nombre | <input type="text"/> |
| apellido | <input type="text"/> |
| price | <input type="text"/> |
| <input type="button" value="Crear Libro"/> <input type="button" value="Modificar Libro"/> | |

```
PantallaLibreriaController.java
Source History
81 @FXML
82 private void clicModificarLibro(ActionEvent event) {
83     libroActual.setIsbn(this.txtIsbn.getText());
84     libroActual.setTitle(this.txtTitulo.getText());
85     libroActual.setAuthorFirstName(this.txtNombre.getText());
86     libroActual.setAuthorLastName(this.txtApellido.getText());
87     libroActual.setPrice(new Integer(this.txtPrice.getText()));
88     libroActual = facadeLibreria.modificarLibro(libroActual);
89     desplegarLibroActual();
90 }
91
92 }
```

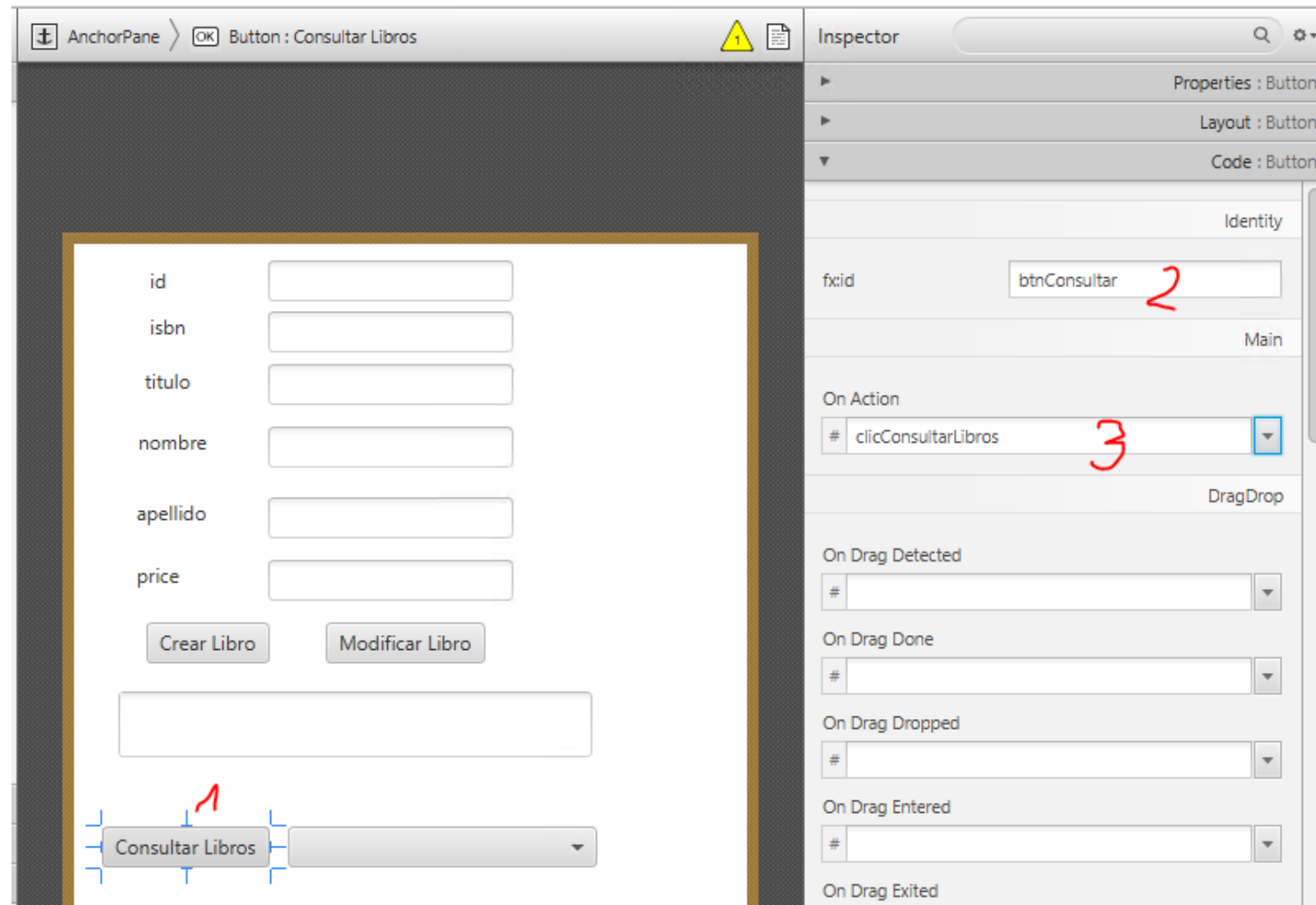
Ejecute

- Guarde
- Clean & Build
- Ejecute

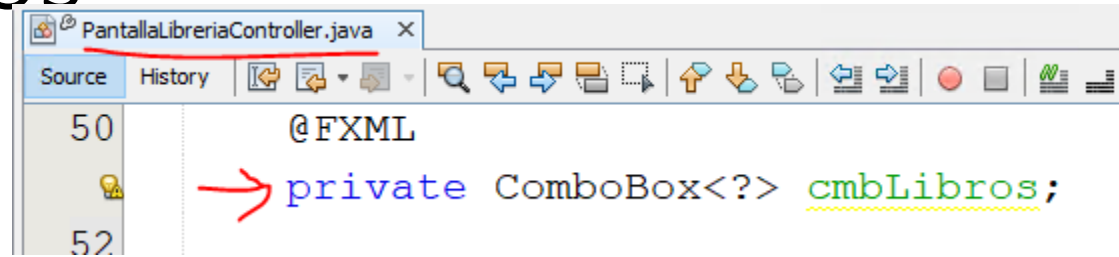
Agregue un Combo de Libros



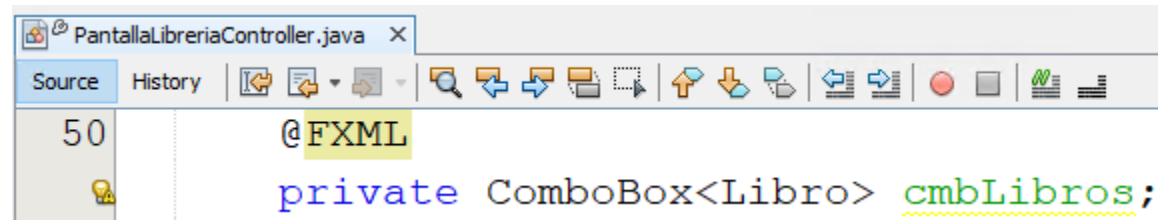
Agregue Botón 'para consultar Libros



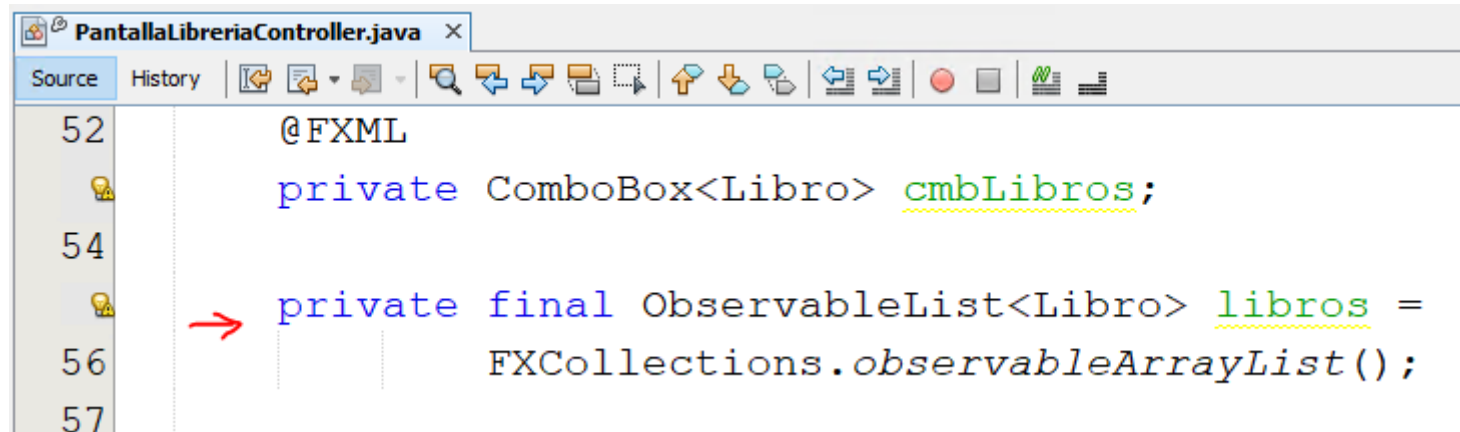
Complete siguiente Código en el controlador de eventos



```
PantallaLibreriaController.java x
Source History
50 @FXML
    private ComboBox<?> cmbLibros;
52
```

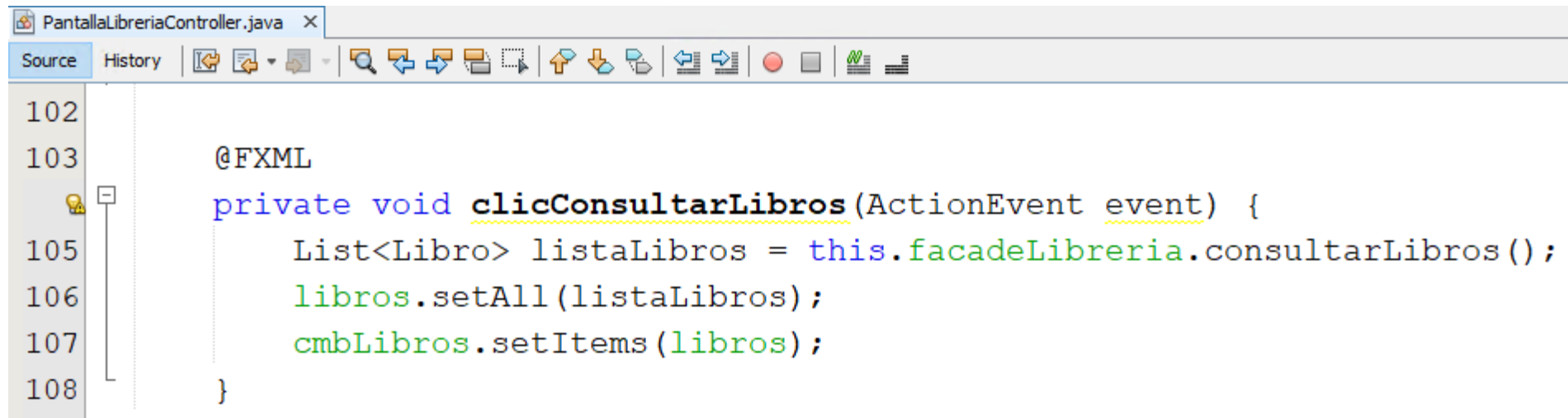


```
PantallaLibreriaController.java x
Source History
50 @FXML
    private ComboBox<Libro> cmbLibros;
```



```
PantallaLibreriaController.java x
Source History
52 @FXML
    private final ObservableList<Libro> libros =
54
56     FXCollections.observableArrayList();
57
```

Complete siguiente Código en el controlador de eventos



```
PantallaLibreriaController.java
Source History
102
103 @FXML
104 private void clicConsultarLibros(ActionEvent event) {
105     List<Libro> listaLibros = this.facadeLibreria.consultarLibros();
106     libros.setAll(listaLibros);
107     cmbLibros.setItems(libros);
108 }
```

Agregue botón

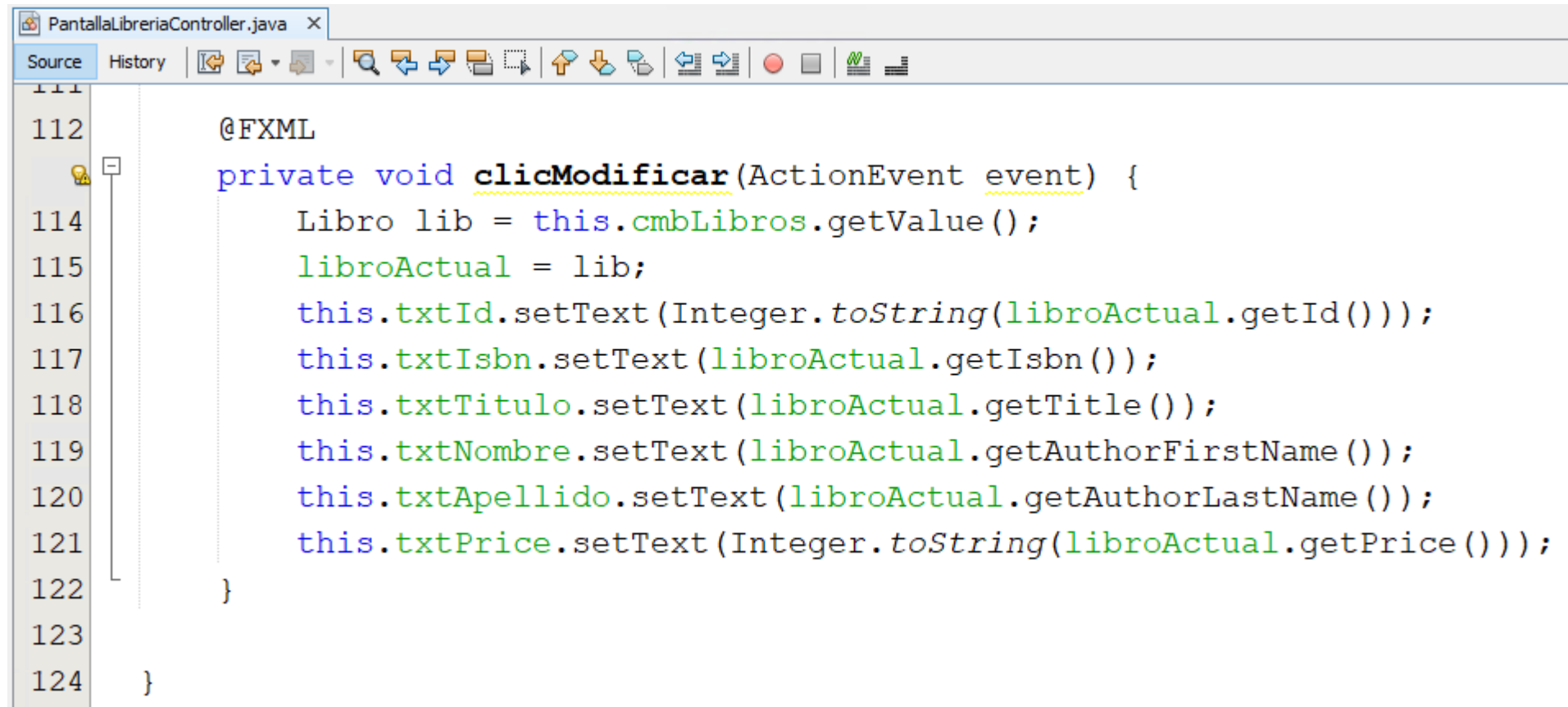
- Botón para tomar un libro seleccionado en el combo y montarlo en la pantalla para modificar datos

The screenshot displays a software development environment with two main components: a form and an Inspector panel.

Form: The form is titled "AnchorPane" and contains several input fields for book data: "id", "isbn", "titulo", "nombre", "apellido", and "price". Below these fields are two buttons: "Crear Libro" and "Modificar Libro". At the bottom of the form, there is a "Consultar Libros" button and a dropdown menu. A new button, "Cargar Libro Seleccionado para Modificar", is being added to the form, indicated by a red "1" and blue selection handles.

Inspector Panel: The Inspector panel is titled "Inspector" and shows the properties of the selected button. The "Identity" section shows the "fxid" as "btnModificar", marked with a red "2". The "Main" section shows the "On Action" event set to "# clicModificar", marked with a red "3". The "DragDrop" section shows various drag-and-drop events, each with a dropdown menu for selecting an action.

Complete siguiente Código en el controlador de eventos



The screenshot shows an IDE window titled "PantallaLibreriaController.java". The code is in the "Source" view. The method `clicModificar` is being completed. The code is as follows:

```
112 @FXML
113 private void clicModificar(ActionEvent event) {
114     Libro lib = this.cmbLibros.getValue();
115     libroActual = lib;
116     this.txtId.setText(Integer.toString(libroActual.getId()));
117     this.txtIsbn.setText(libroActual.getIsbn());
118     this.txtTitulo.setText(libroActual.getTitle());
119     this.txtNombre.setText(libroActual.getAuthorFirstName());
120     this.txtApellido.setText(libroActual.getAuthorLastName());
121     this.txtPrice.setText(Integer.toString(libroActual.getPrice()));
122 }
123
124 }
```

Ejecute

- Guarde
- Clean & Build
- Ejecute

Ejecute

id: 1

isbn: A1111

titulo: Moby Dick

nombre: Herman

apellido: Melville

price: 100

Crear Libro Modificar Libro

1 2

Consultar Libros Libro{id=1, isbn=A1111, t...}

3

Cargar Libro Seleccionado para Modificar