Tutorial JavaFX

Libreta de direcciones con JavaFX

Enlace del proyecto original

Mayo 2020

Índice general

Introducción

Por que el proyecto

El tutorial original fue escrito en inglés y traducido al español por Mario Gómez Martínez, en este documento la intención es realizar el mismo proyecto con las actulizaciones a la última versión del JDK, la codificación se actualizará al español con los respectivos comentarios.

Introducción

JavaFX proporciona a los desarrolladores de Java una nueva plataforma gráfica. JavaFX 2.0 se publicó en octubre del 2011 con la intención de reemplazar a Swing en la creación de nuevos interfaces gráficos de usuario (IGU). Cuando empecé a enseñar JavaFX en 2011 era una tecnología muy incipiente todavía. No había libros sobre JavaFX que fueran adecuados para estudiantes de programación noveles, así es que empecé a escribir una serie de tutoriales muy detallados sobre JavaFX.
El tutorial te guía a lo largo del diseño, programación y publicación de una aplicación de contactos (libreta de direcciones) mediante JavaFX. Este es el aspecto que tendrá la aplicación final:
Colocar la imagen después

Lo que aprenderás

- Creación de un nuevo projecto JavaFX
- Uso de Scene Builder para diseñar la interfaz de usuario
- Estructuración de una aplicación según el patrón MVC (Modelo, Vista, Controlador)
- Uso de `ObservableList` para la actualización automática de la interfaz de usuario
- Uso de `TableView` y respuesta a cambios de selección en la tabla
- Creación de un diálogo personalizado para editar personas
- Validación de la entrada del usuario

- Aplicación de estilos usando CSS
- Persistencia de datos mediante XML
- Guardado del último archivo abierto en las preferencias de usuario
- Creación de un gráfico JavaFX para mostrar estadísticas
- Despliegue de una aplicación JavaFX nativa

Después de completar esta serie de tutoriales deberías estar preparado para desarrollar aplicaciones sofisticadas con JavaFX.

Cómo usar este tutorial

Hay dos formas de utilizarlo

- **Máximo-aprendizaje:** Crea tu propio proyecto JavaFX desde cero.
- **Máxima-rápidez:** Importa el código fuente de una parte del tutorial en tu entorno de desarrollo favorito (es un proyecto Eclipse, pero puedes usar otros entornos, como Netbeans, con ligeras modificaciones). Después revisa el tutorial para entender el código. Este enfoque también resulta útil si te quedas atascado en la creación de tu propio código.

Parte 1

Eclipse y Scene Builder

1.1. Contenidos en Parte 1

- Familiarizándose con JavaFX.
- Crear y empezar un proyecto JavaFX.
- Uso de Scene builder para diseñar la interfaz de usuario
- Estructura básica de una aplicación mediante el patrón Modelo Vista Controlador (MVC).

1.2. Prerrequisitos

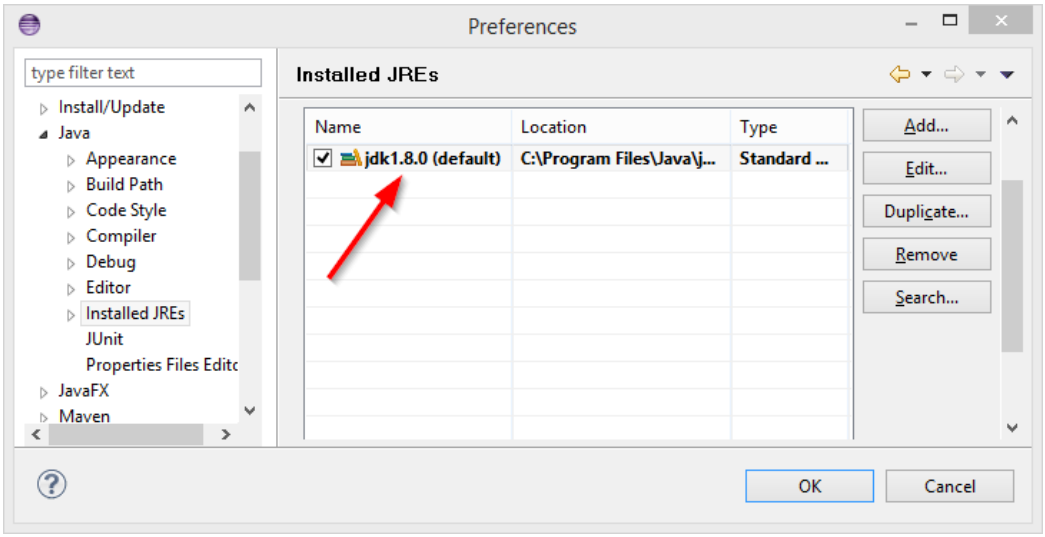
- Última versión de Java JDK 8 o superior.
- Eclipse 4.3 o superior con el plugin e(fx)clipse. La forma más sencilla de obtenerlo es descargarse la distribución preconfigurada desde e(fx)clipse website. Como alternativa puedes usar un sitio de actualización para tu instalación de Eclipse.
- Scene Builder 2.0 o superior

1.3. Configuración de Eclipse

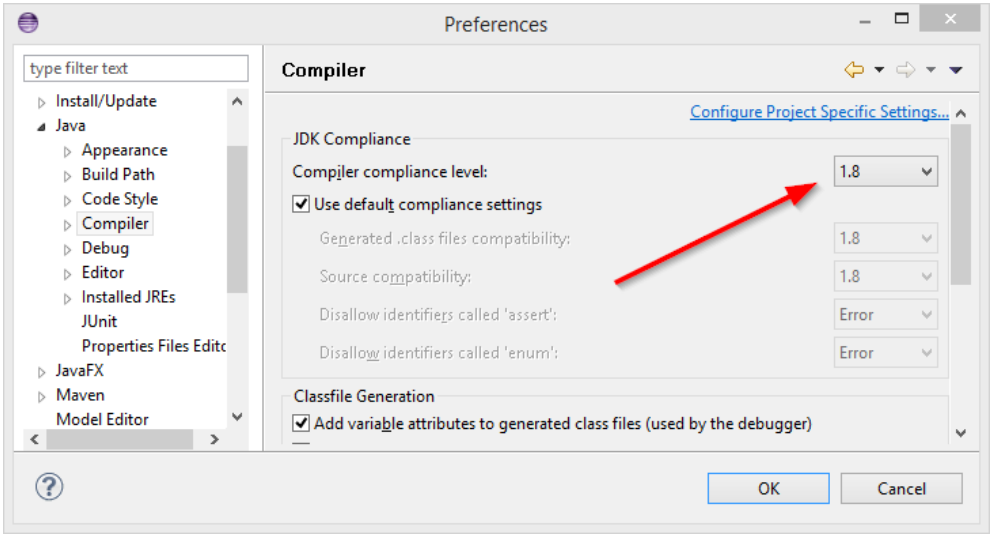
Hay que indicarle a Eclipse que use JDK 8 y también dónde se encuentra el ejecutable del Scene Builder:

1. Abre las Preferencias de Eclipse (menú *Window — Preferences* y navega hasta *Java — Installed JREs*.
2. Si no lo tienes el jre1.8 en la lista de JREs, entonces pulsa *Add...*, selecciona *Standard VM* y elige el Directorio de instalación (JRE Home directory) de tu JDK 8.

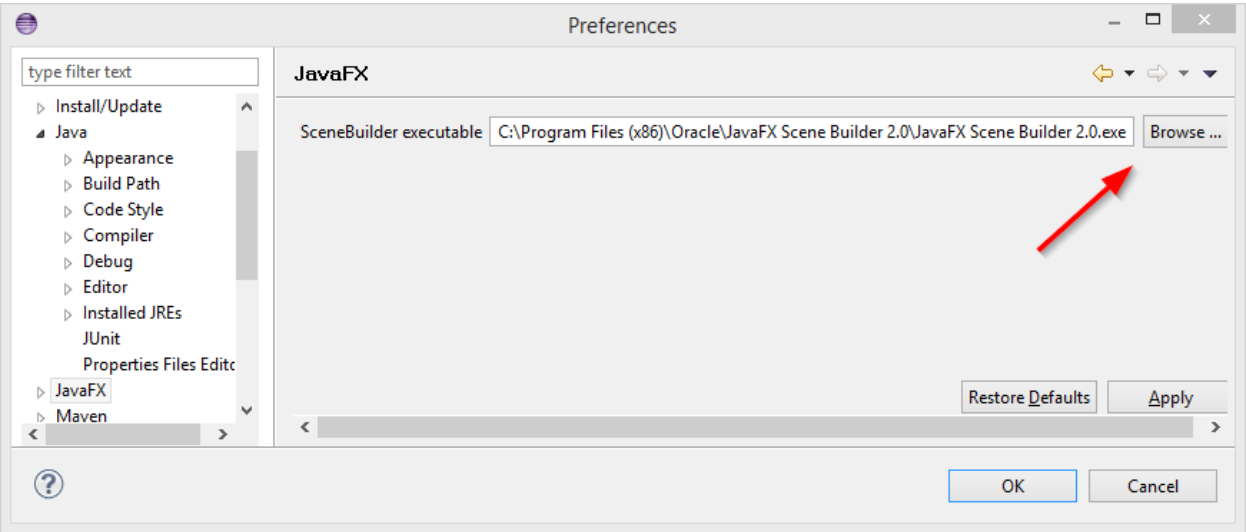
3. Elimina otros JREs o JDKs de tal manera que **JDK 8 se convierta en la opción por defecto**.



4. Navega a *Java* — *Compiler*. Establece el **nivel de cumplimiento del compilador en 1.8** (Compiler compliance level).



5. Navega hasta *Java* — *JavaFX*. Especifica la ruta al ejecutable del Scene Builder.



1.4. Enlaces útiles

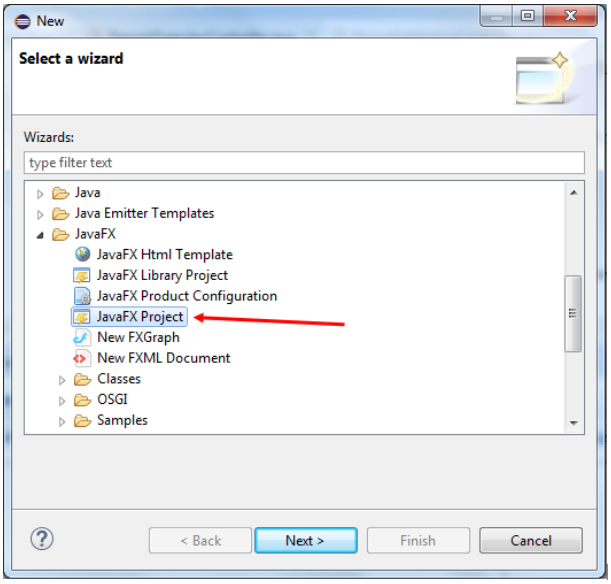
Te podría interesar los siguientes enlaces:

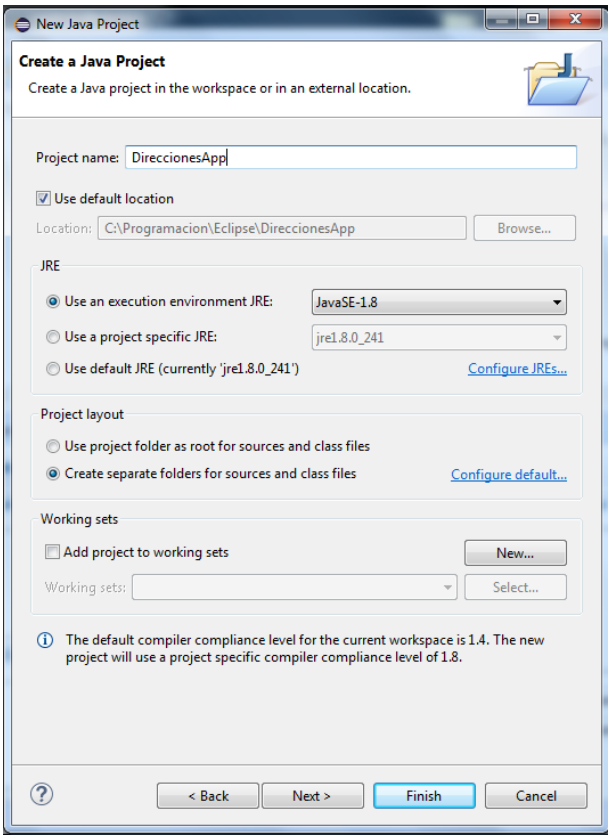
1. Java 8 API - Documentación (JavaDoc) de las clases estándar de Java
2. JavaFX 8 API - Documentación de las clases JavaFX
3. ControlsFX API - Documentación (JavaDoc) - Documentación para el proyecto ControlsFX, el cual ofrece controles JavaFX adicionales
4. Oracle's JavaFX Tutorials - Tutoriales oficiales de Oracle sobre JavaFX

`¡Y ahora, manos a la obra!`

1.5. Crea un nuevo proyecto JavaFX

En Eclipse (con e(fx)clipse instalado) ve a *File — New — Other...* y elige *JavaFX Project*. Especifica el nombre del proyecto (ej. DireccionesApp) y aprieta *Finish*.





Borra el paquete *application* y su contenido que ha sido creado automáticamente.

1.5.1. Crea los paquetes

Desde el principio vamos a seguir buenos principios de diseño de software. Algunos de estos principios se traducen en el uso de la arquitectura denominada Modelo-Vista-Controlador

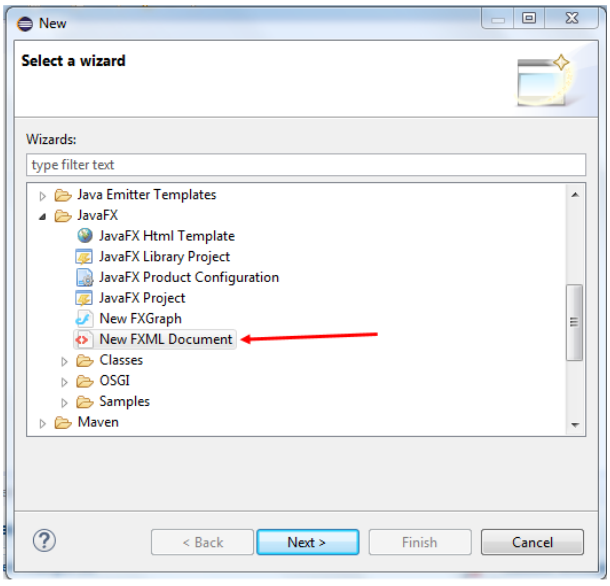
(MVC). Esta arquitectura promueve la división de nuestro código en tres apartados claramente definidos, uno por cada elemento de la arquitectura. En Java esta separación se logra mediante la creación de tres paquetes separados.
En el ratón hacemos clic derecho en la carpeta *src, New — Package:*

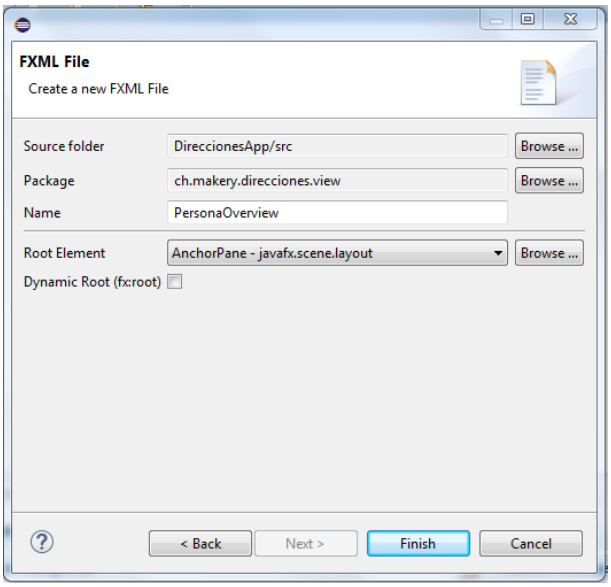
- `ch.makery.direcciones` - contendrá la mayoría de clases de control (C).
- `ch.makery.direcciones.model` - contendrá las clases del modelo (M).
- `ch.makery.direcciones.view` - contendrá las vistas (V)

Nota: Nuestro paquete dedicado a las vistas contendrá también algunos controladores dedicados exclusivamente a una vista. Les llamaremos **controladores-vista**.

1.6. Crea el archivo FXML de diseño

Hay dos formas de crear la interfaz de usuario. Programándolo en Java o mediante un archivo XML. Si buscas en Internet encontrarás información relativa a ambos métodos. Aquí usaremos XML (archivo con la extensión .fxml) para casi todo. Encuentro más claro mantener el controlador y la vista separados entre sí. Además, podemos usar la herramienta de edición visual Scene Builder, la cual nos evita tener que trabajar directamente con el XML.
Haz clic derecho el paquete *view* y crea un nuevo archivo FXML (*New — Other — FXML — New FXML Document*) llamado `PersonaOverview`.

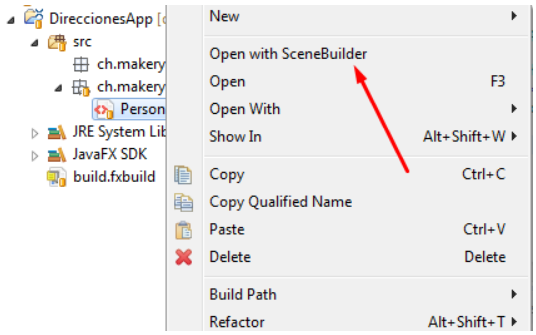




1.7. Diseño mediante Scene Builder

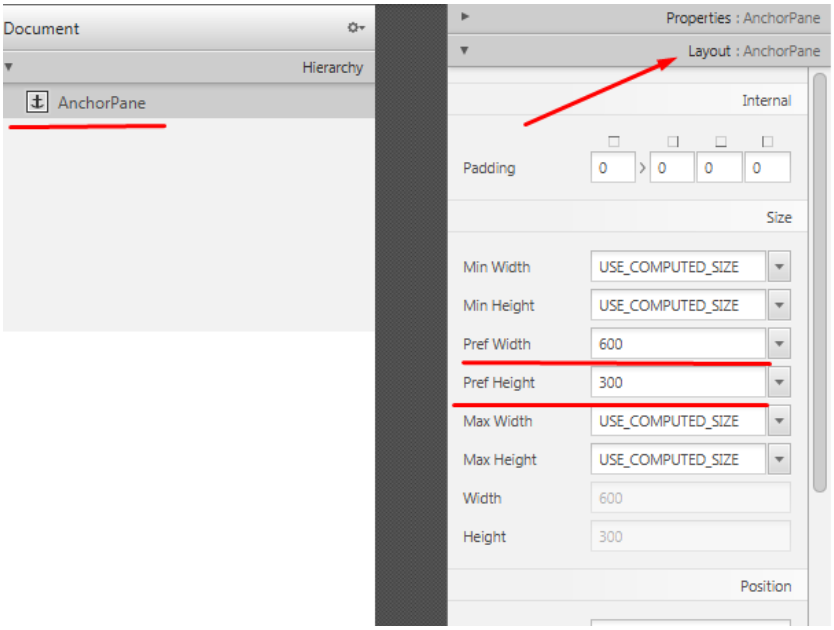
> **Nota:** Si no puedes hacerlo funcionar, descarga las fuentes para esta parte del tutorial e inténtalo con el archivo fxml incluido.

Haz clic derecho sobre PersonaOverview.fxml y elige *Open with Scene Builder*. Ahora deberías ver el Scene Builder con un AnchorPane (visible en la jerarquía de componentes (herramienta Hierarchy) situada a la izquierda).

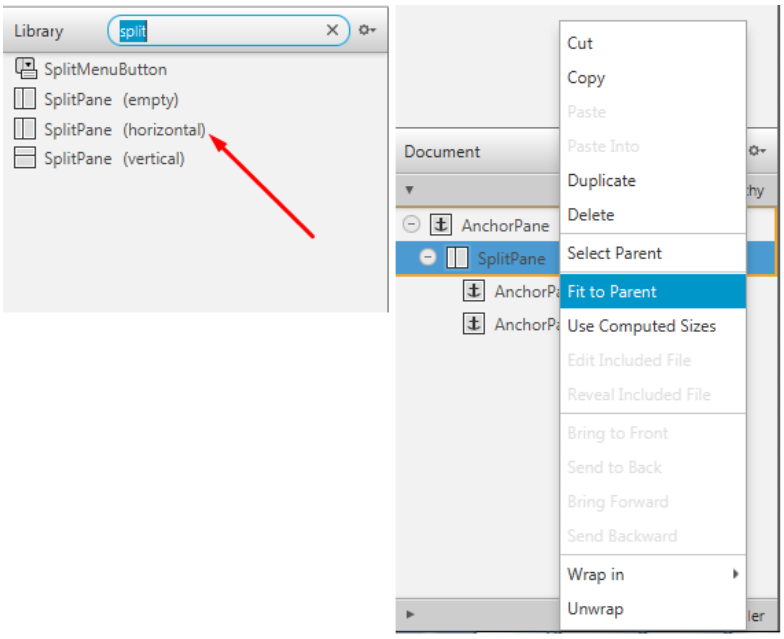


Ya abierto Scene Builder

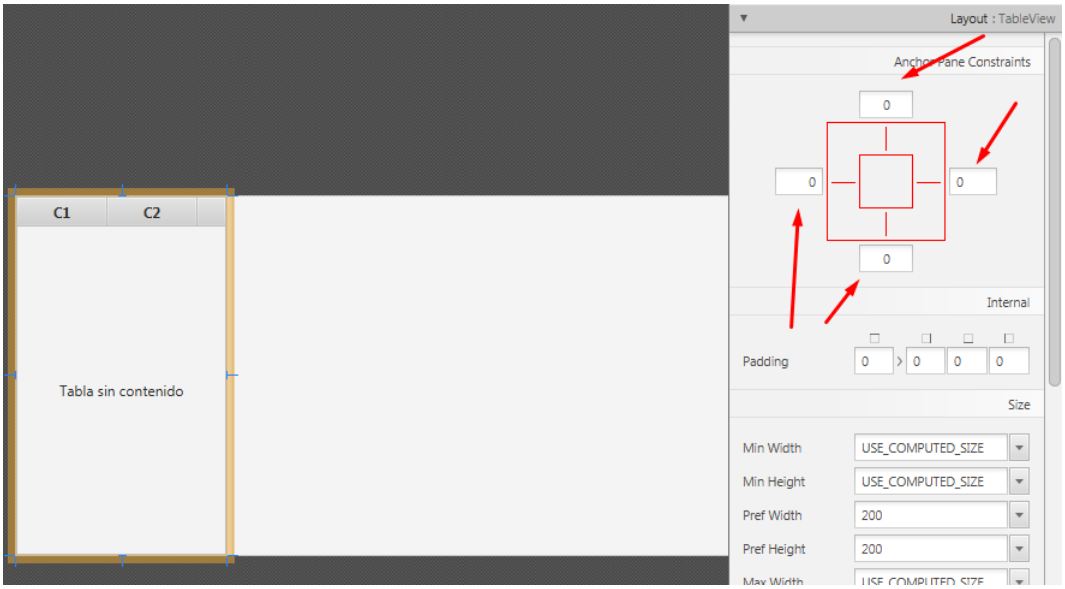
1. Selecciona el *AnchorPane* en tu jerarquía y ajusta el tamaño en el apartado Layout (a la derecha):



2. Añade un *SplitPane (Horizontal Flow)* arrastrándolo desde la librería (Library) al área principal de edición. Haz clic derecho sobre el SplitPane en la jerarquía y elige Fit to Parent.

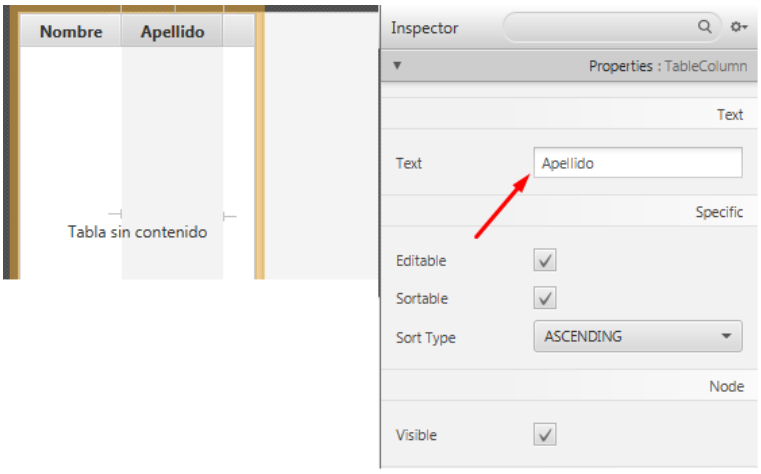


3. Arrastra un TableView (bajo Controls) al lado izquierdo del SplitPane. Selecciona la TableView (no una sola columna) y establece las siguientes restricciones de apariencia (Layout) para la TableView. Dentro de un AnchorPane siempre se pueden establecer anclajes (anchors) para los cuatro bordes (más información sobre Layouts).

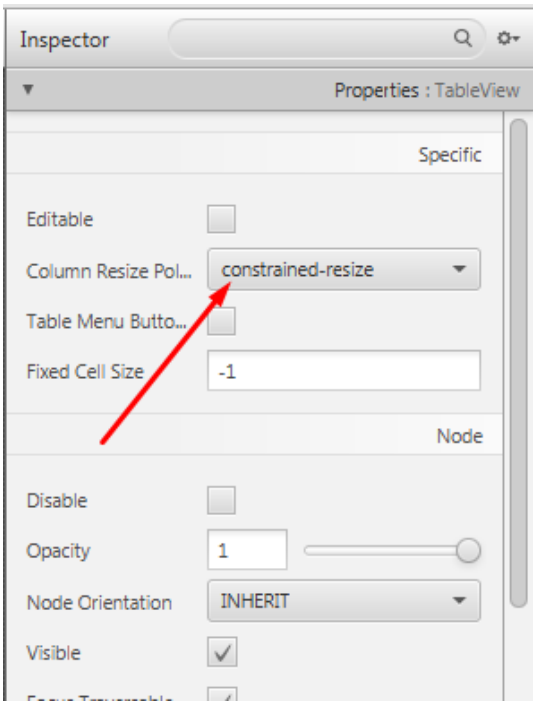


4. Ve al menú *Preview — Show Preview in Window* para comprobar si se visualiza correctamente. Intenta cambiar el tamaño de la ventana. La TableView debería ajustar su tamaño al tamaño de la ventana, pues está "anclada" a sus bordes.

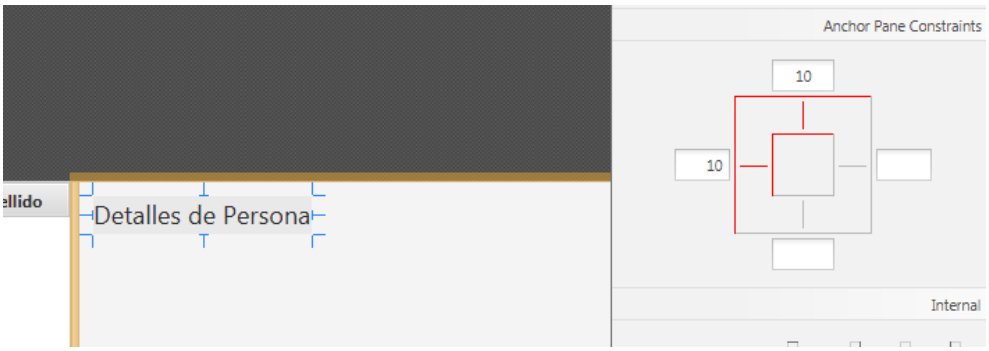
5. Cambia el texto de las columnas (en *Properties*) a "Nombre" y "Apellido".



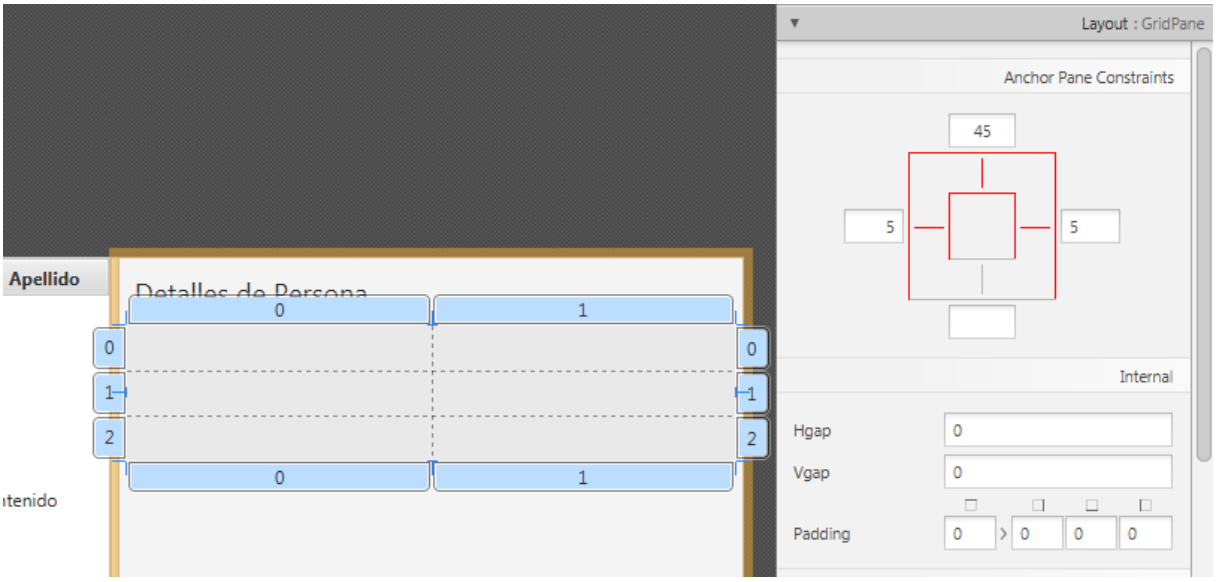
6. Selecciona la *TableView* y elige *constrained-resize* para la *Column Resize Policy* (en *Properties*). Esto asegura que las columnas usarán siempre todo el espacio que tengan disponible.



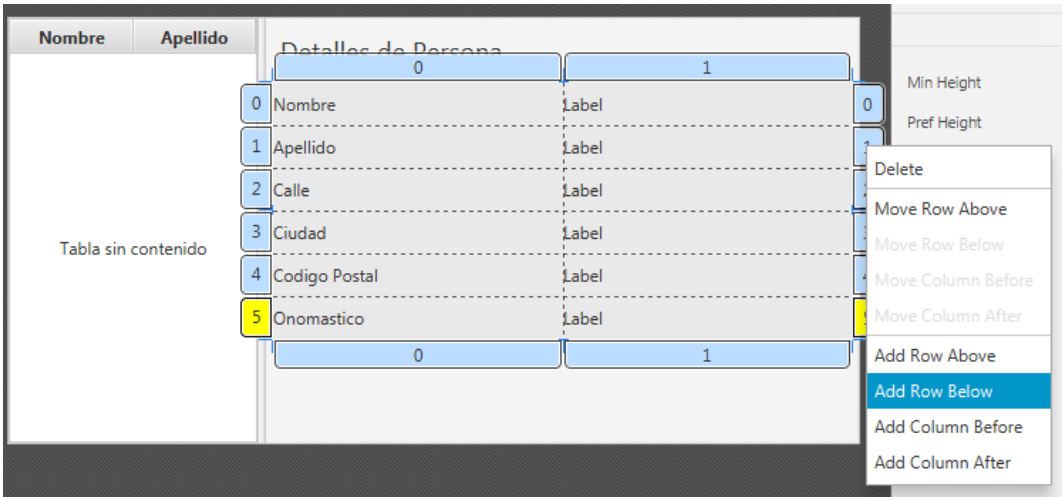
7. Añade una *Label* al lado derecho del *SplitPane* con el texto "Detalles de Persona" (truco: usa la búsqueda en la librería para encontrar el componente *Label*). Ajusta su apariencia usando anclajes.



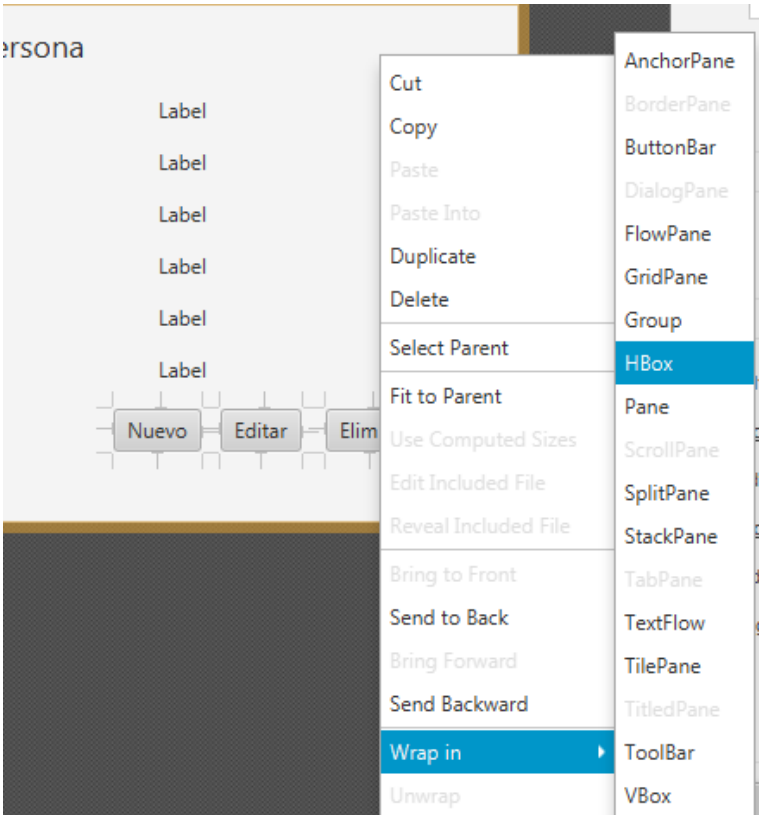
8. Añade un *GridPane* al lado derecho, selecciónalo y ajusta su apariencia usando anclajes (superior, derecho e izquierdo).

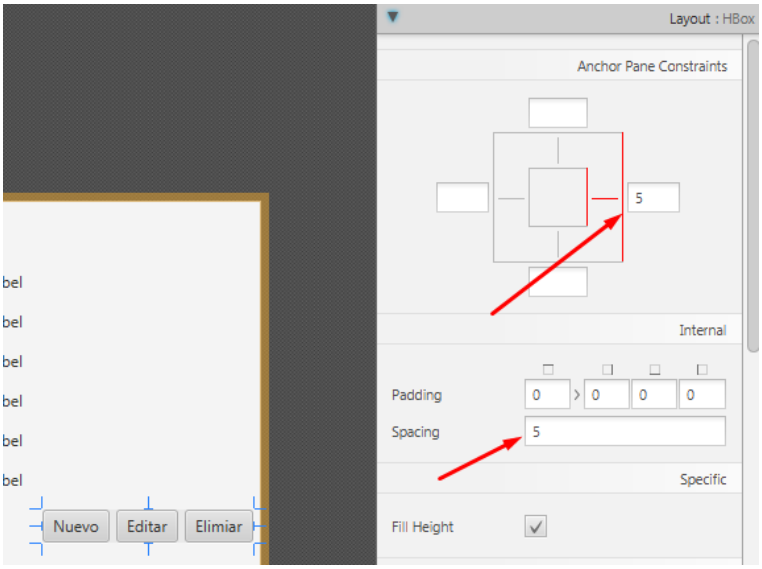


9. Añade las siguientes etiquetas (*Label*) a las celdas del GridPane.
   Nota: Para añadir una fila al *GridPane* selecciona un número de fila existente (se volverá de color amarillo), haz clic derecho sobre el número de fila y elige "Add Row".

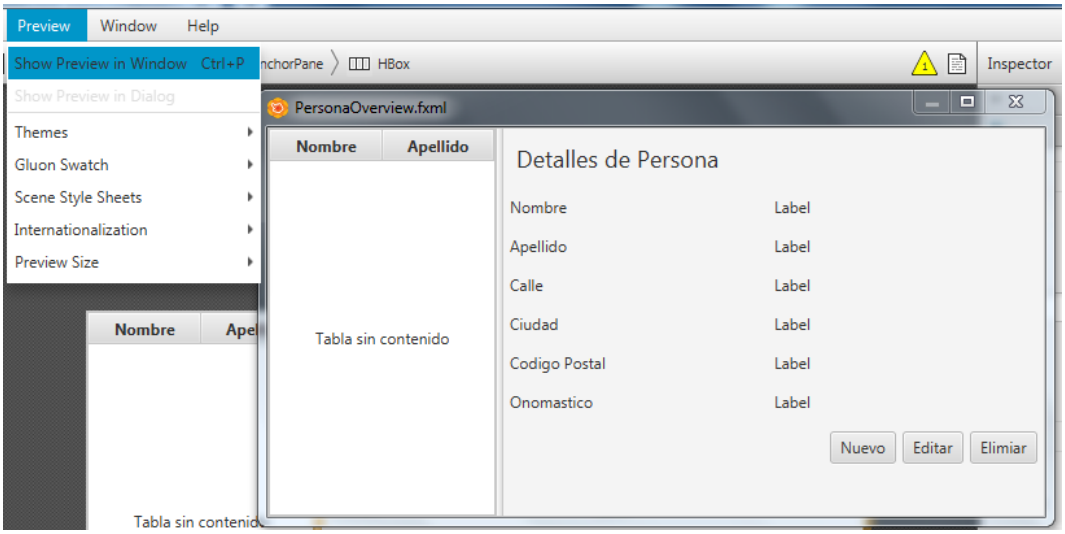


10. Añade tres botones a la parte inferior. Truco: Selecciónalos todos, haz click derecho e invoca *Wrap In — HBox*. Esto los pondrá a los 3 juntos en un *HBox*. Podrías necesitar establecer un espaciado spacing dentro del HBox. Después, establece también anclajes (derecho e inferior) para que se mantengan en el lugar correcto.





11. Ahora deberías ver algo parecido a lo siguiente. Usa el menú *Preview* para comprobar su comportamiento al cambiar el tamaño de la ventana.

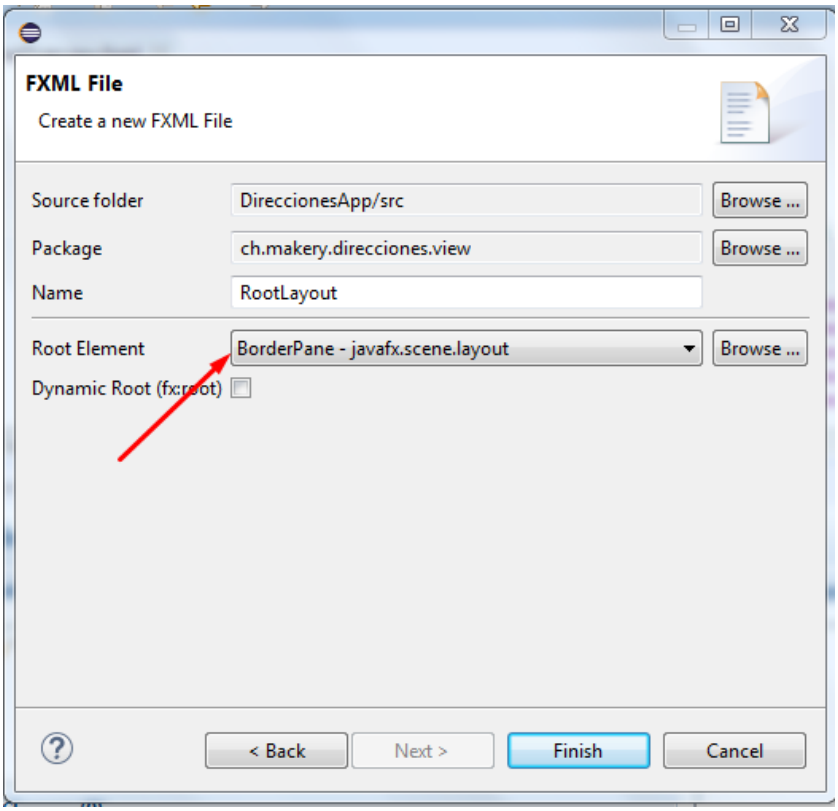


> **Nota:** Guarda los cambios en SceneBuilder, el archivo se actualizará automáticamente en Eclipse, si esto no sucede, puedes presionar F5 para actualizar el archivo.

1.8. Crea la aplicación principal

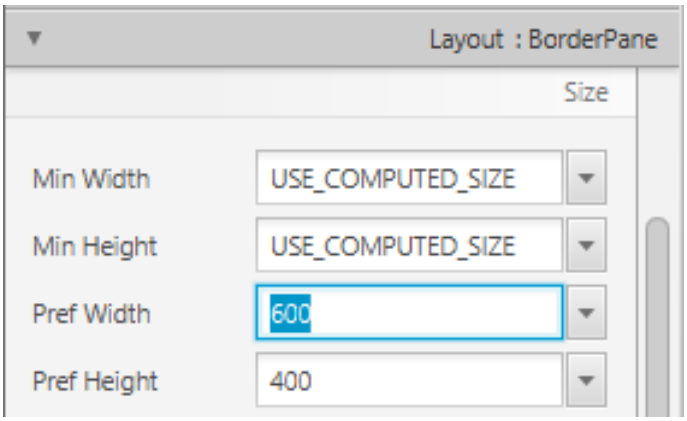
Necesitamos otro archivo FXML para nuestra vista raíz, la cual contendrá una barra de menús y encapsulará la vista recién creada `PersonaOverview.fxml`.

1. Crea otro archivo FXML dentro del paquete view llamado `RootLayout.fxml`. Esta vez, elige BorderPane como elemento raíz.



2. Abre `RootLayout.fxml` en el Scene Builder.

3. Cambia el tamaño del *BorderPane* con la propiedad *Pref Width* establecida en 600 y *Pref Height* en 400.

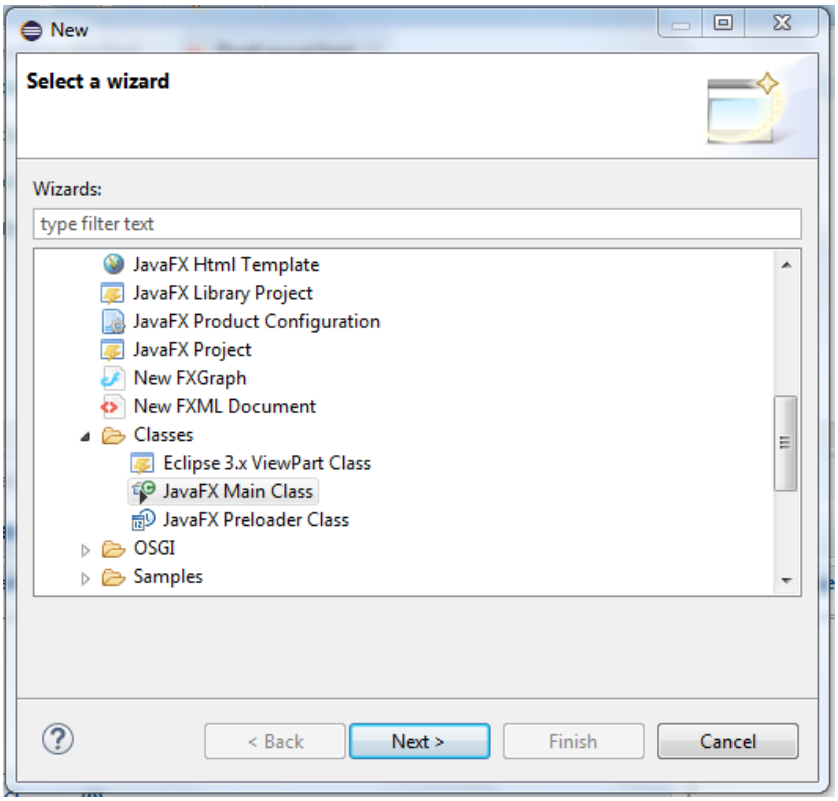


4. Añade una *MenuBar* en la ranura superior del *BorderPane*. De momento no vamos a implementar la funcionalidad del menú.

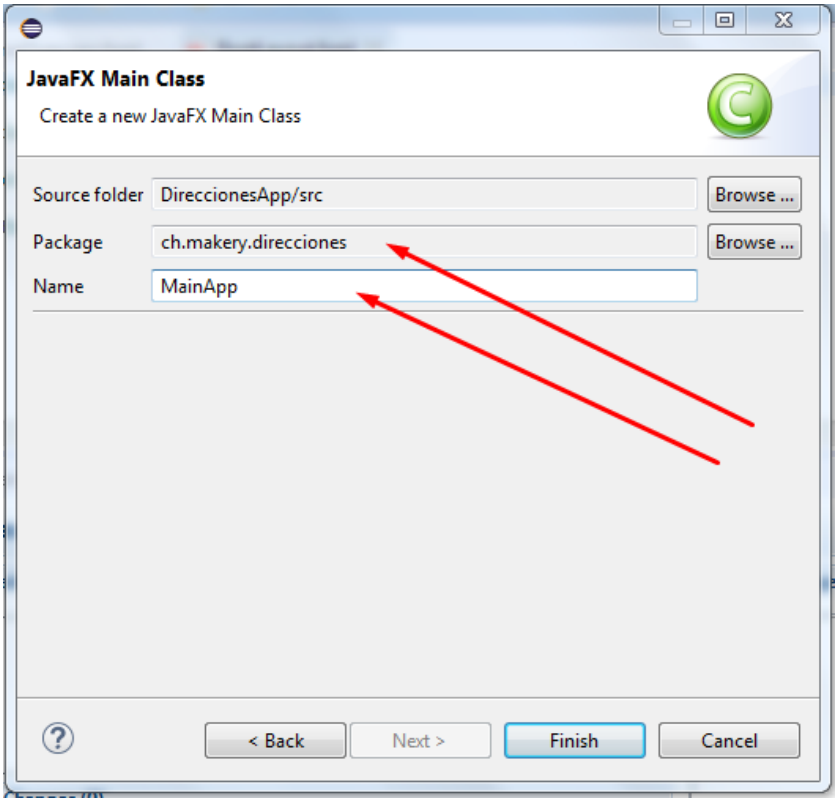
1.9. La clase principal en JavaFX

Ahora necesitamos crear la clase java principal, la cual iniciará nuestra aplicación mediante `RootLayout.fxml` y añadirá la vista `PersonOverview.fxml` en el centro.

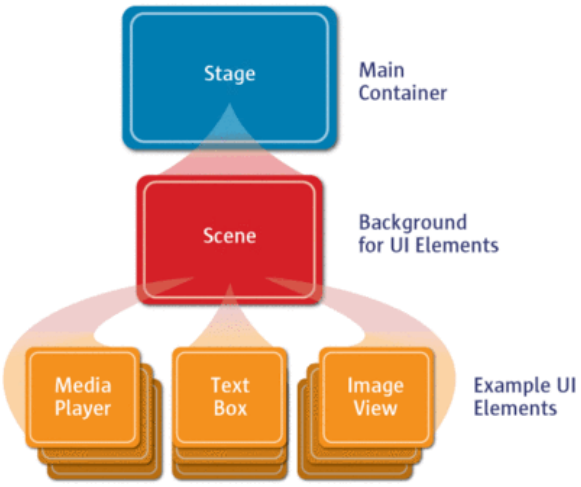
1. Haz clic derecho en el proyecto y elige *New* — *Other* — *JavaFX* — *classes* — *JavaFX Main Class*.



2. Llama a esta clase `MainApp` y ponla en el paquete de controladores `ch.makery.address` (nota: este es el paquete padre de los paquetes `view` y `model`).



La clase generada (`MainApp.java`) extiende a la clase `Application` y contiene dos métodos. Esta es la estructura básica que necesitamos para ejecutar una Aplicación JavaFX. La parte más importante para nosotros es el método `start(Stage primaryStage)`. Este método es invocado automáticamente cuando la aplicación es lanzada desde el método `main`. Como puedes ver, el método `start(...)` tomo un `Stage` como parámetro. El gráfico siguiente muestra la estructura de cualquier aplicación JavaFX:



Fuente de la imagen: http://www.oracle.com

Es como una obra de teatro: El `Stage` (escenario) es el contenedor principal, normal-

mente una ventana con borde y los típicos botones para maximizar, minimizar o cerrar la ventana. Dentro del `Stage` se puede añadir una `Scene` (escena), la cual puede cambiarse dinámicamente por otra `Scene`. Dentro de un `Scene` se añaden los nodos JavaFX, tales como `AnchorPane`, `TextBox`, etc.
Para tener más información puedes consultar Working with the JavaFX Scene Graph.
Abre el archivo `MainApp.java` y reemplaza todo su código con el código siguiente:

```
1  package ch.makery.direcciones;
2
3  import java.io.IOException;
4
5  import javafx.application.Application;
6  import javafx.fxml.FXMLLoader;
7  import javafx.scene.Scene;
8  import javafx.scene.layout.AnchorPane;
9  import javafx.scene.layout.BorderPane;
10 import javafx.stage.Stage;
11
12 public class MainApp extends Application {
13         private Stage primaryStage;
14         private BorderPane rootLayout;
15
16         @Override
17         public void start(Stage primaryStage) {
18                 this.primaryStage = primaryStage;
19                 this.primaryStage.setTitle("Directorios App");
20
21                 initRootLayout();
22                 showPersonaOverview();
23         }
24         /**
25         * Inicializa el diseño raíz.
26         */
27         public void initRootLayout(){
28                 try{
29                         //Carga el diseño raíz del archivo fxml.
30                         FXMLLoader loader = new FXMLLoader();
31                         loader.setLocation(MainApp.class.getResource("view/RootLayout.fx
32                         rootLayout = (BorderPane) loader.load();
33                         //Muestra la escena que contiene el diseño raíz.
34                         Scene scene = new Scene(rootLayout);
35                         primaryStage.setScene(scene);
36                         primaryStage.show();
37                 }catch (IOException e) {
```

```
38                          e.printStackTrace();
39                  }
40          }
41          /**
42          * Muestra la descripción general de la persona dentro del diseño
    ↪  raíz.
43          */
44          public void showPersonaOverview(){
45                  try{
46                          //Carga datos de persona
47                          FXMLLoader loader = new FXMLLoader();
48                          loader.setLocation(MainApp.class.getResource("view/PersonaOvervi
49                          AnchorPane personaOverview = (AnchorPane)
                             ↪  loader.load();
50                          //Carga los datos de la persona en el centro del
                             ↪  diseño raíz.
51                          rootLayout.setCenter(personaOverview);
52                  }catch (IOException e) {
53                          e.printStackTrace();
54                  }
55          }
56          /**
57          * Retorna el escenario principal
58          * @return
59          */
60          public Stage getPriStage(){
61                  return primaryStage;
62          }
63          public static void main(String[] args) {
64                  launch(args);
65          }
66  }
```

Los diferentes comentarios deben darte pistas sobre lo que hace cada parte del código. Si ejecutas la aplicación ahora, verás algo parecido a la captura de pantalla incluida al principio de este artículo.

1.10. Problemas frecuentes

Si JavaFX no encuentra un archivo `fxml` puedes obtener el siguiente mensaje de error: `java.lang.IllegalStateException: Location is not set.`
Para resolverlo comprueba otra vez que no hayas escrito mal el nombre de tus archivos `fxml`.

Si todavía no te funciona, descárgate el código de esta parte del tutorial y pruébalo con el fxml proporcionado.