DOI: 10.36910/6775-2524-0560-2020-39-30
УДК: 004.415.2
Муляр Вадим Петрович, к. пед. н., доцент
https://orcid.org/0000-0003-4774-3947
Східноєвропейський національний університет імені Лесі Українки

РОЗРОБКА JAVAFX-ДОДАТКІВ ІЗ ВИКОРИСТАННЯМ SCENE BUILDER

Муляр В. П. Розробка JavaFX-додатків із використанням Scene Builder. У статті розкрито особливості створення додатків на основі технології JavaFX. Доведено високу ефективність середовища Scene Builder у процесі розробки графічного інтерфейсу користувача. На прикладі створення діалогового середовища розглянуто основні етапи проектування JavaFX-додатків в інтегрованому середовищі розробки NetBeans із використанням Scene Builder.
Ключові слова: платформа JavaFX, інтегроване середовище розробки NetBeans, конструктор макетів Scene Builder, графічний інтерфейс користувача, компонування.

Муляр В. П. Разработка JavaFX-приложений с использованием Scene Builder. В статье раскрыты особенности создания приложений на основе технологии JavaFX. Доказана высокая эффективность среды Scene Builder в процессе разработки графического интерфейса пользователя. На примере создания диалоговой среды рассмотрены основные этапы проектирования JavaFX-приложений в интегрированной среде разработки NetBeans с использованием Scene Builder.
Ключевые слова: платформа JavaFX, интегрированная среда разработки NetBeans, конструктор макетов Scene Builder, графический интерфейс пользователя, компоновка.

Muliar V. P. Developing JavaFX applications using Scene Builder. The features of JavaFX based application development are discussed in the article. High efficiency of Scene Builder environment in the process of developing a graphical user interface has been proven. The example of creating a dialog discusses the basic stages of designing JavaFX applications in an integrated NetBeans development environment using Scene Builder.
Keywords: JavaFX platform, NetBeans integrated development environment, Scene Builder layout designer, graphical user interface, layout.

Постановка наукової проблеми. Створення сучасного, ефективного та повнофункціонального інструментарію для розробки клієнтських додатків на основі Java неможливо уявити без технології JavaFX. За її допомогою можна створювати програми як для різних операційних систем (Windows, MacOS, Linux), так і для різноманітних пристроїв (десктопів, смартфонів, планшетів, вбудованих пристроїв, ТБ). Однак тепер JavaFX розширює свою підтримку на Android і iOS за допомогою таких технологій, як JavaFX Ports і Gluon Mobile. Остання є платформою для написання, компіляції та підготовки додатків JavaFX для розгортання на iOS і Android. Для кінцевого користувача додаток виглядає і поводиться точно так само, як і нативний (native) додаток [12].

JavaFX дозволяє створювати програми з багатою насиченою графікою завдяки використанню апаратного прискорення графіки і можливостей графічного процесора. JavaFX має великий набір елементів управління та широкі можливості для роботи з мультимедіа, двомірною і тримірною графікою. Характерним для JavaFX є декларативний спосіб опису інтерфейсу за допомогою мови розмітки FXML, можливість стилізації інтерфейсу за допомогою CSS і багато іншого.

На даний час для розробки програмного забезпечення мовою Java потрібно встановити на комп'ютері декілька основних програмних засобів. Першим є JDK (Java Development Kit – комплект для Java-розробки), який, як правило, містить у собі ще й віртуальну машину Java (пакет JRE – Java Runtime Environment). Його можна встановити з офіційного сайту Oracle: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Другим засобом є JavaFX SDK, який завантажують за адресою https://gluonhq.com/products/javafx/. Третім є інтегроване середовище розробника Java-програм (NetBeans), яке завантажують безпосередньо з сайту NetBeans (https://netbeans.org). Останнім засобом є візуальний конструктор графічного інтерфейсу користувача Scene Builder, який завантажують за адресою https://gluonhq.com/products/scene-builder/.

Аналіз досліджень. Огляд сучасних технологій створення RIA-додатків (Rich Internet Applications) здійснено у статті К. Афанасьєва та К. Лебедєва [1]. Розгляду архітектури платформи JavaFX 2.0, її основним компонентам графічного інтерфейсу користувача, використанню CSS-стилів, створенню візуальних ефектів, трансформації й анімації зображень, використанню компонентів JavaFX NetBeanes, мові FXML присвячена робота Т. Машніна [4]. Створенню насичених інтернет-додатків за допомогою JavaFX присвячено дослідження В. Герасимова та В. Левицької [2], Ю. Парфенова та В. Федорченко [6]. Особливості побудови графічного контенту додатків із використанням JavaFX і даних, взятих із баз даних, розкрито в дослідженні В. Карашецького [3]. На думку дослідників, JavaFX є інструментарієм наступного покоління для створення графічного

інтерфейсу користувача. Технологія забезпечує кросплатформні додатки з графічним інтерфейсом такими складними функціями як плавна анімація, веб-представлення, відтворення аудіо та відео, стилі на основі CSS [7; 8; 11].

Формулювання цілей статті. Мета статті – розкрити особливості створення JavaFX-додатків із використанням Scene Builder.

Виклад основного матеріалу й обґрунтування отриманих результатів. JavaFX-технології представлені екземпляром класу GUI-компонента. Компоненти графічного інтерфейсу користувача (графічний інтерфейс користувача, GUI) JavaFX-додатка створюють сцену, логічна структура якої описується графом сцени. Відображення GUI-інтерфейсу JavaFX-додатка включає в себе графічне представлення графа зі сценами. Графом сцени є структура даних, колекція вузлів (вузол) дерева, яка використовує логічну структуру сцен. Сцена – це скомпонований у робочих областях набір моделей та об'єктів, які викликають різні ефекти, наприклад, джерело світла та камера, які створюють ефекти освітленості та перспективи. Під моделлю розуміють опис або набір даних, що представляє форму об'єкта. Моделі всередині сцени характеризуються розміром і взаємним розташуванням. Сцена має знаходитися на підмостках, які є вікном верхнього рівня, якщо програма виконується на робочому столі операційної системи, або прямокутною областю, якщо програма виконується в вигляді аплета [4, с. 16].

Для забезпечення гнучкого і динамічного розміщення елементів управління в графі сцени JavaFX-додатку використовуються контейнери схем компонування або панелі. JavaFX Layout API включає в себе наступні контейнерні класи, які автоматизують загальні моделі схем компонування.

Клас BorderPane розміщує вузли його контенту у верхній, нижній, правій, лівій, або центральній області.

Клас HBox розміщує вузли його контенту горизонтально в один рядок.

Клас VBox розміщує вузли його контенту вертикально в один стовпець.

Клас StackPane розміщує вузли його контенту в стек.

Клас GridPane дозволяє розробнику створити гнучку сітку з рядків і стовпців, в яких розміщуються вузли контенту.

Клас FlowPane розміщує вузли його контенту в горизонтальний або вертикальний «потік», огинаючи зазначені межі по ширині або висоті.

Клас TilePane розміщує вузли його контенту в комірках однакового розміру.

Клас AnchorPane дозволяє розробникам створювати вузли-якорі для прив'язки до верхньої, нижньої або лівої сторони, або в центрі макета.

Для досягнення бажаної структури розташування, різні контейнери можуть бути вкладені.

У JavaFX підтримуються різні способи компонування елементів управління графічного інтерфейсу. Проте, основним засобом розробки візуального інтерфейсу користувача є JavaFX Scene Builder, оскільки:

– інтерфейс перетягування дозволяє швидко створити макет інтерфейсу користувача без необхідності запису вихідного коду;

– можна додавати, комбінувати та редагувати елементи керування інтерфейсом JavaFX у свій макет за допомогою бібліотеки елементів управління інтерфейсом та панелі вмісту;

– інтеграція з будь-яким Java IDE є простою, оскільки це окремий інструмент розробки;

– автоматичне генерування коду FXML відбувається під час створення та зміни макета інтерфейсу користувача: створений код FXML зберігається в окремому файлі;

– функції редагування в реальному часі та попереднього перегляду дозволяють швидко візуалізувати зміни макета інтерфейсу без необхідності компіляції;

– отримуємо доступ до повної бібліотеки управління інтерфейсом JavaFX: підтримка CSS дозволяє гнучко керувати зовнішнім виглядом інтерфейсу програми.

За умовчанням головне вікно JavaFX Scene Builder містить наступні розділи (рис. 1).

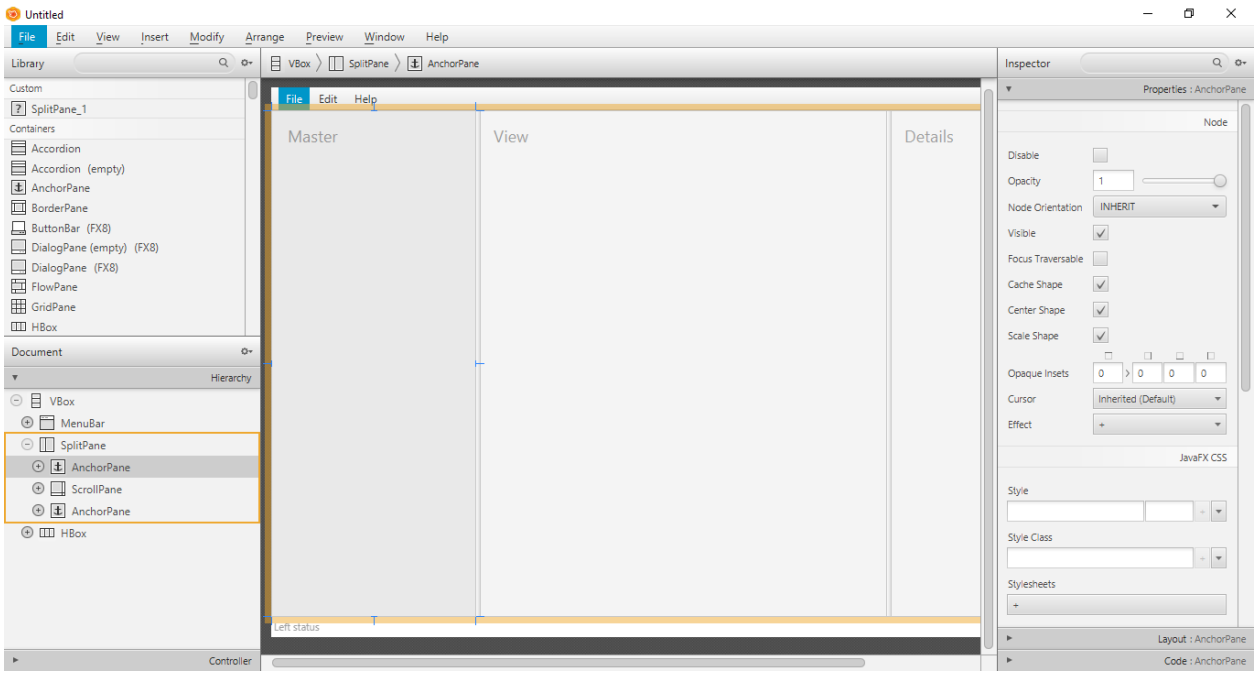

Рис. 1. Головне вікно Scene Builder

Рядок меню: надає доступ до меню команд, доступних у JavaFX Scene Builder.

Вибір та панель повідомлень: відображає шлях до обраного елемента. Він також відображає повідомлення про помилки або статус.

Панель вмісту: контейнер сцени для елементів інтерфейсу, що складають макет FXML. За умовчанням новий файл FXML, відкритий у JavaFX Scene Builder, містить кореневий (верхній) контейнер AnchorPane.

Панель бібліотеки: містить доступні елементи інтерфейсу або елементи управління JavaFX, які можна використовувати для створення макета FXML. Елементи інтерфейсу вибирають на цій панелі та додають їх на панель «Зміст» або «Ієрархія».

Панель ієрархії: відображає подання з дерева на макет FXML, який будують на панелі «Зміст». Елементи, яких не видно на панелі «Зміст», можна поставити у фокус, вибравши його на панелі «Ієрархія».

Панель інспектора: містить розділи Властивості, Макет та Код. Розділи «Властивості та макет» допомагають керувати властивостями вибраного елемента інтерфейсу на панелі «Зміст» або на панелі «Ієрархія». Розділ «Код» дозволяє визначити ім'я об'єкта (fx: id), а також для кожного компонента задати реакцію на події.

Панель «Інспектор» також містить текстове поле пошуку, яке дозволяє виділити конкретні властивості, які можна змінити.

Наступна панель відображається в головному вікні, коли в головному меню вибрати пункт Перегляд, а потім Показати CSS Analyzer.

Панель аналізатора CSS: дозволяє вивчити усі властивості CSS, доступні для компонента JavaFX на макеті FXML, та допомагає створити правила CSS [10].

Файл FXML можна об'єднати з проєктом Java наступним чином:

```
public void start(Stage stage) {
try {
Parent root = FXMLLoader.load(getClass().getResource("dialog.fxml"));
stage.setScene(new Scene(root));
stage.show();
} catch (IOException ex) {
ex.printStackTrace();
System.exit(0);
}
}
```

Розглянемо основні можливості JavaFX як засобу розробки графічного інтерфейсу користувача на прикладі лабораторного практикуму з комп'ютерного моделювання фізичних процесів і явищ [5].

Лабораторна робота
Створення діалогового середовища

Мета: ознайомитись із основними етапами розробки *JavaFX*-додатків з використанням конструктора макетів Scene Builder під час створення проекту й головної форми графічних побудов.

Створення головної форми

1. Відрийте середовище *NetBeans* із підтримкою платформи *JavaFX 2.0*. У меню *Файл* виберіть *Створити проект* | *JavaFX* | *JavaFX FXML Application*, натисніть кнопку *Далі*, введіть ім'я проекту *model*, дайте назву FXML-файлу *FXMLModel*, установіть прапорець *Create Application Class* і натисніть кнопку *Готово*. В результаті середовищем *NetBeans* буде згенеровано проект, що містить у каталозі src папку пакета головного класу додатка з трьома файлами – Java-файлом головного класу додатка *Model.java*, *FXMLModel.fxml* і *FXMLModelController.java*.

2. Відкрийте файл *FXMLModel.fxml* у середовищі JavaFX Scene Builder, клацнувши двічі по *FXMLModel.fxml*. Внесіть зміни в макет FXML-файлу (рис. 2).

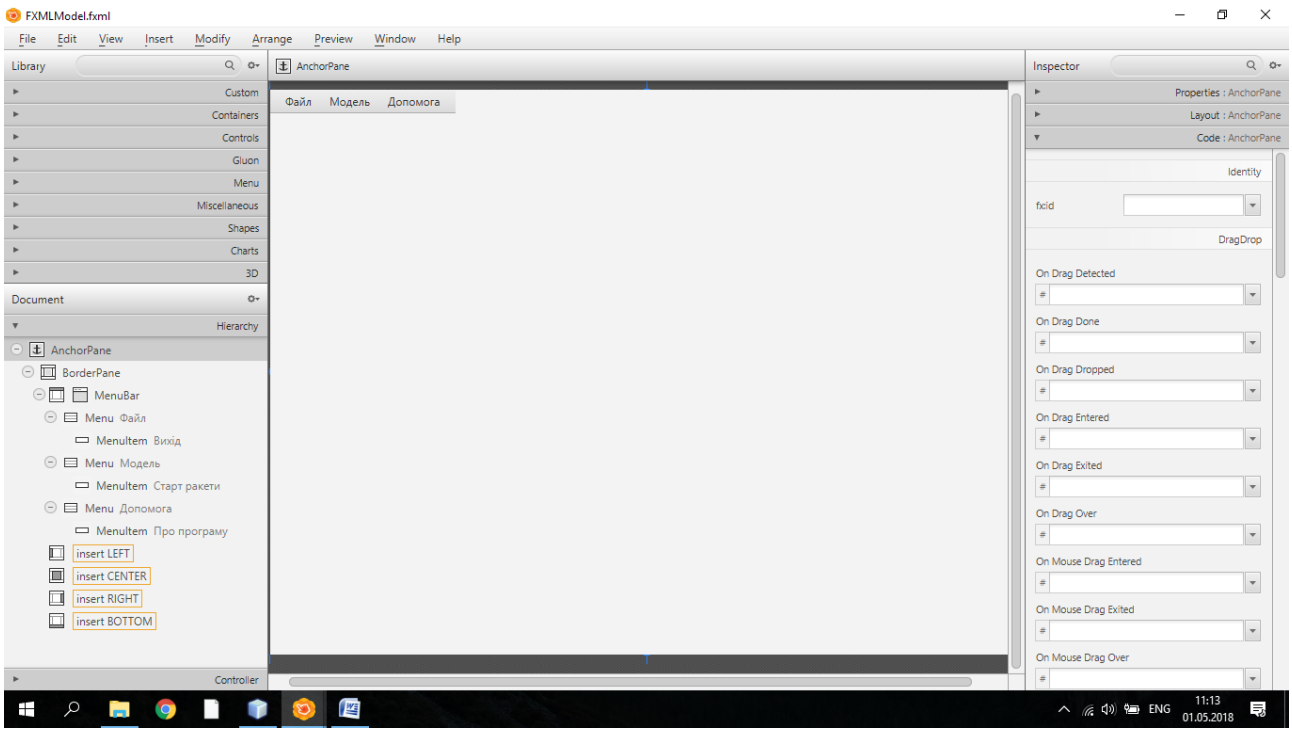

Рис. 2. Макет файлу *FXMLModel.fxml* у середовищі *JavaFX Scene Builder*

3. Виберіть компонент *AnchorPane* на вкладці *Hierarchy*, перейдіть на вкладку *Layout* і в полі *Pref Width* установіть значення 800, а в полі *Pref Height* – значення 600.

Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *root*.

Для компонента *MenuItem Вихід* у полі *fx:id* установіть значення *menuItemClose*, а в полі *On Action* задайте значення *menuItemCloseOnAction*. Аналогічно для компонента *MenuItem Старт ракети* в полі *fx:id* установіть значення *menuItemRocket*, а в полі *On Action* задайте значення *menuItemRocketOnAction*.

4. У меню *View* виберіть *Show Sample Controller Skeleton*. Далі виберіть у вікні діалогу наступний текст.

```
@FXML
private AnchorPane root;
@FXML
private MenuItem menuItemClose;
@FXML
private MenuItem menuItemRocket;
@FXML
void menuItemCloseOnAction(ActionEvent event) {
}
```

```
@FXML
void menuItemRocketOnAction(ActionEvent event) {
}
```

Після цього замініть код у *public class FXMLModelController implements Initializable { ...}* виділеним фрагментом.

У результаті *FXMLModelController.java* матиме такий вигляд.

```
package model;
import static com.sun.corba.se.impl.util.Utility.printStackTrace;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.AnchorPane;
public class FXMLModelController implements Initializable {
@FXML
private AnchorPane root;
@FXML
private MenuItem menuItemClose;
@FXML
private MenuItem menuItemRocket;
@FXML
void menuItemCloseOnAction(ActionEvent event) {
}
@FXML
void menuItemRocketOnAction(ActionEvent event) throws IOException {
}
@Override
public void initialize(URL url, ResourceBundle rb) {
// TODO
}
}
```

5. Вставте в обробник події *menuItemCloseOnAction* наступний код:

```
printStackTrace();
System.exit(0);
```

6. Внесіть зміни у файл *Model.java*, вставивши в метод *start* наступний код:

```
stage.setResizable(false);
stage.setTitle("Комп'ютерне моделювання фізичних процесів і явищ");
```

7. Запустіть додаток на виконання. Загальний вигляд головної форми подано на рис. 3.

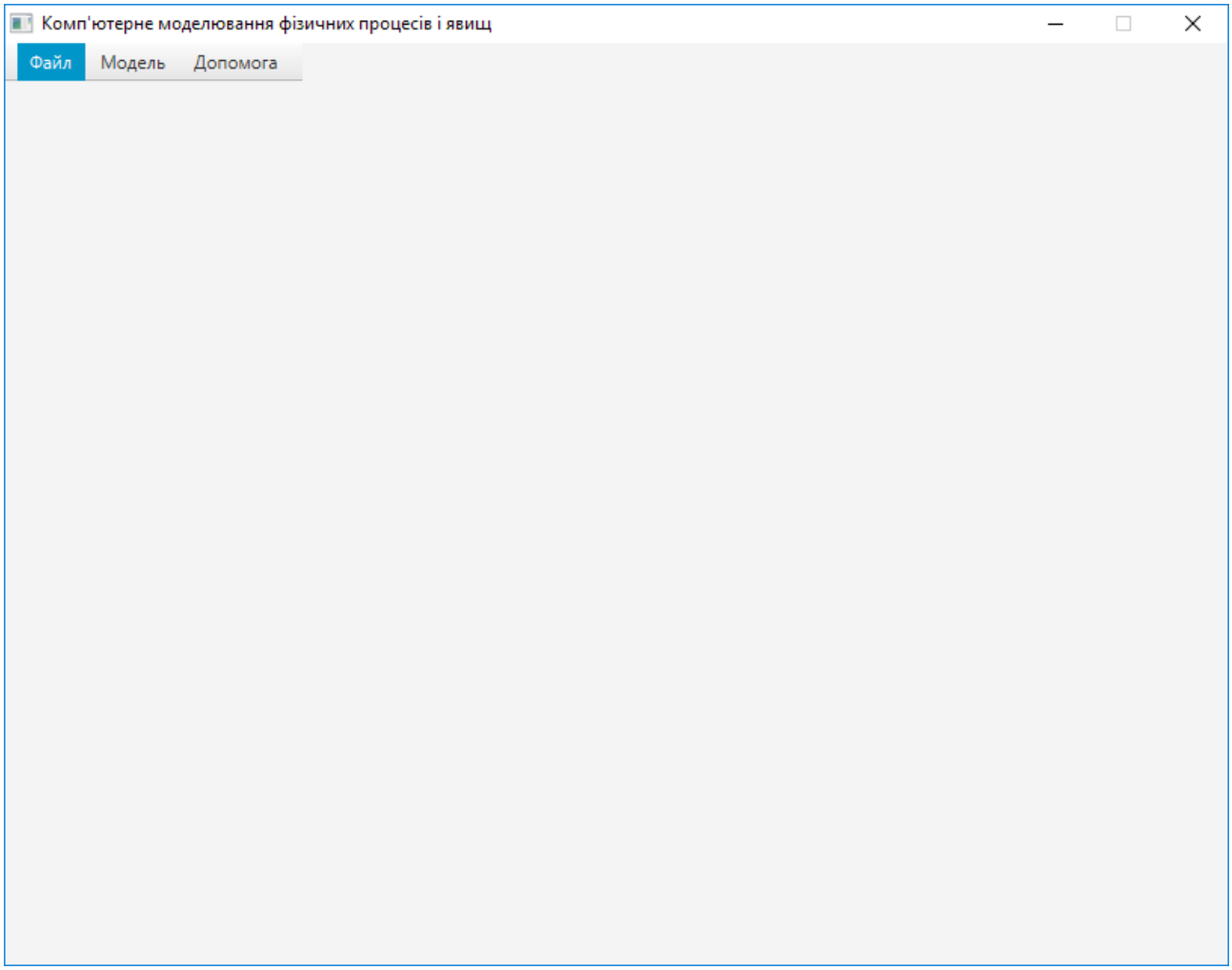

Рис. 3. Загальний вигляд головної форми

Створення макету форми графічних побудов

1. У меню *Файл* виберіть *Створити файл | JavaFX | Empty FXML*, натисніть кнопку *Далі.* Дайте назву FXML-файлу *FXMLAll*, натисніть кнопку *Далі.* Встановіть прапорець *Use Java Controller* і натисніть кнопку *Далі*. Натисніть кнопку *Готово*. В результаті середовищем *NetBeans* буде згенеровано два файли – *FXMLAll.fxml* і *FXMLAllController.java*.

2. Відкрийте файл *FXMLAll.fxml* у середовищі *JavaFX Scene Builder*, клацнувши двічі по *FXMLAll.fxml*. Створіть макет FXML-файлу як показано на рис. 4.

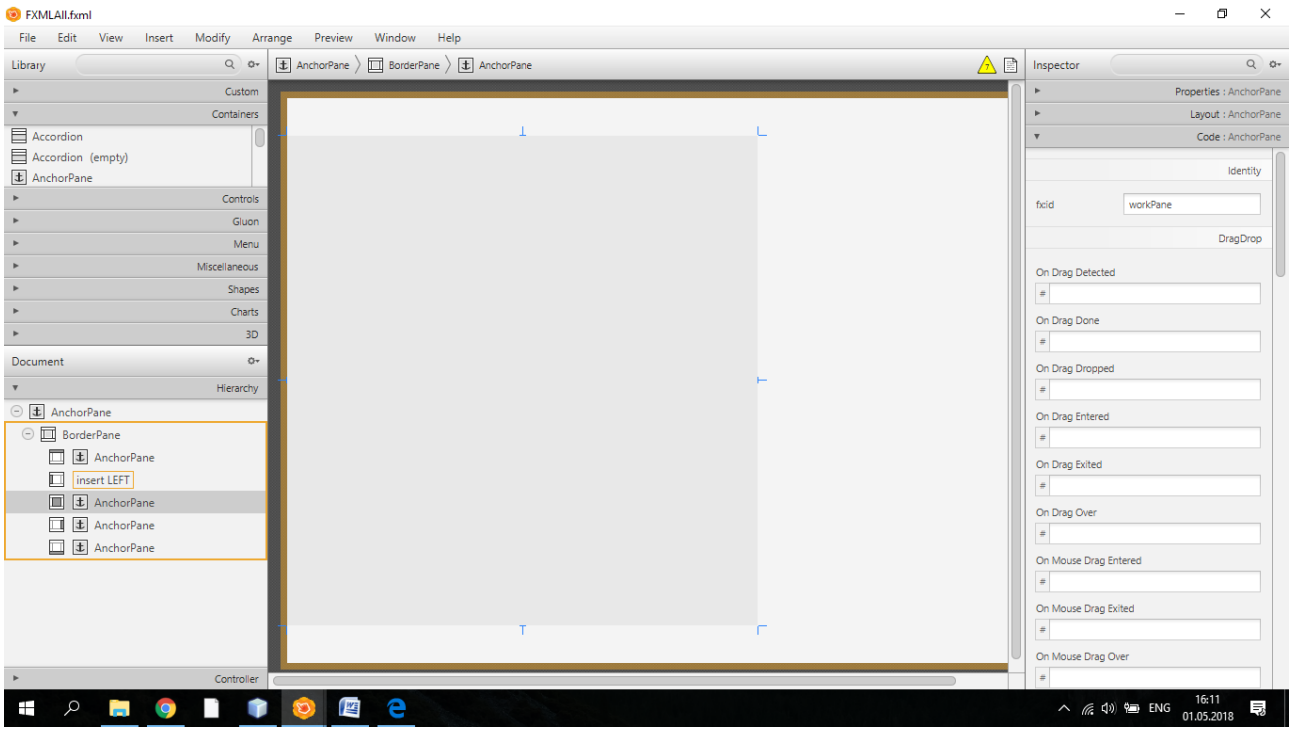

Рис. 4. Макет файлу *FXMLAll.fxml* у середовищі *JavaFX Scene Builder*

3. Виберіть компонент *AnchorPane* на вкладці *Hierarchy*, перейдіть на вкладку *Layout* і в полі *Pref Width* установіть значення 800, а в полі *Pref Height* – значення 600. Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *root.*

Виберіть компонент *AnchorPane*, який знаходиться в *TOP BorderPane*, перейдіть на вкладку *Layout* і в полі *Pref Width* установіть значення 800, а в полі *Pref Height* – значення 40. Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *topPane.*

Аналогічно виберіть компонент *AnchorPane*, який знаходиться в *BOTTOM BorderPane*, перейдіть на вкладку *Layout* і в полі *Pref Width* встановіть значення 800, а в полі *Pref Height* – значення 40. Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *bottomPane.*

Виберіть компонент *AnchorPane*, який знаходиться в *CENTER BorderPane*, перейдіть на вкладку *Layout* і в полі *Pref Width* установіть значення 500, а в полі *Pref Height* – значення 520. Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *workPane.*

Виберіть компонент *AnchorPane*, який знаходиться в *RIGHT BorderPane*, перейдіть на вкладку *Layout* і в полі *Pref Width* установіть значення 300, а в полі *Pref Height* – значення 520. Далі перейдіть на вкладку *Code* і для компонента *AnchorPane* в полі *fx:id* установіть значення *rightPane.*

4. У меню *View* виберіть *Show Sample Controller Skeleton*. Виберіть у вікні діалогу наступний текст.

```
@FXML
private AnchorPane root;
@FXML
private AnchorPane topPane;
@FXML
private AnchorPane rightPane;
@FXML
private AnchorPane bottomPane;
@FXML
private AnchorPane workPane;
```

Вставте виділений код у *public class FXMLAllController implements Initializable { ...}*. Замініть *private* на *public*.

У результаті *FXMLAllController.java* матиме такий вигляд.

```
package model;
import java.net.URL;
```

```
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.layout.AnchorPane;
public class FXMLAllController implements Initializable {
@FXML
public AnchorPane root;
@FXML
public AnchorPane topPane;
@FXML
public AnchorPane rightPane;
@FXML
public AnchorPane bottomPane;
@FXML
public AnchorPane workPane;
@Override
public void initialize(URL url, ResourceBundle rb) {
// TODO
}
}
```

5. Запустіть додаток на виконання *(F6)* та збережіть проект *(File\Save All)*.

Висновки та перспективи подальшого дослідження. У роботі розкрито особливості розробки додатків на основі технології JavaFX з використанням конструктора макетів Scene Builder. Показано, що програмний інтерфейс JavaFX API дає можливість створювати RIA-додатки, код яких поєднує широкі можливості платформи Java з багатою графікою та медіафункціональністю платформи JavaFX. Основним засобом розробки візуального інтерфейсу користувача є JavaFX Scene Builder. Його можуть використовувати як Java-розробники, так і дизайнери. Перші можуть швидко створювати прототипи інтерфейсу користувача і окремо розробляти логіку додатку. Другі можуть не тільки швидко створювати візуальний інтерфейс без написання будь-якого коду, а й проектувати і переглядати макет візуального інтерфейсу, змінювати зовнішній вигляд інтерфейсу користувача за допомогою таблиць стилів CSS.

На прикладі створення діалогового середовища розглянуто основні етапи проектування JavaFX-додатків в інтегрованому середовищі розробки NetBeans засобами Scene Builder.

Список бібліографічного опису

1. Афанасьев К.С., Лебедев К.С. Обзор современных технологий создания RIA-приложений. *Кибернетика. Управление в сложных системах. Вестник ИрГТУ.* № 4 (44). 2010. С. 6–12.

2. Герасимов В. В., Левицька В. Я. Аналіз технологій розробки насичених інтернет-додатків на платформі Java. *ИТ проектирования, моделирования, дизайна, WEB,* 2017. С. 355–363.

3. Карашецький В. П. Побудова графічного контенту додатків з використанням JavaFX і Swing компонентів і даних, взятих із баз даних. *Науковий вісник НЛТУ.* 2015. Вип. 25.1. С. 386–392.

4. Машнин Т. С. *JavaFX 2.0: разработка RIA-приложений.* СПб.: БХВ-Петербург, 2012. 320 с.

5. Муляр В. П., Федонюк А. А. *Комп'ютерне моделювання фізичних процесів і явищ: навч. посіб.* Луцьк: ПП Іванюк В. П., 2018. 212 с.

6. Парфенов Ю. Э., Федорченко В. Н. Разработка «насыщенных» интернет-приложений с помощью JavaFX. *Системи обробки інформації.* 2012. Вип. 8 (106). С. 40–46.

7. Хорстманн К. *Java SE 8. Вводный курс.* М.: Вильямс, 2014. 208 с.

8. Carl Dea. *JavaFX 2.0: Introduction by Example.* Apress, 2011. 181 p.

9. JavaFX – Application. URL: https://www.tutorialspoint.com/javafx/javafx_application.htm

10. JavaFX Scene Builder. URL: http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

11. Kishori Sharan. *Learn JavaFX 8 (The Expert's Voice in Java): Building User Experience and Interfaces with Java 8*. New York, 2015. P. 1–1173.

12. Stephen Chin, Johan Vos, James Weaver. *The Definitive Guide to Modern Java Clients with JavaFX: Cross-Platform Mobile and Cloud Development.* Apress, 2019. 621 p.

References

1. Afanas'ev K.S., Lebedev K.S. Obzor sovremennyh tehnologij sozdanija RIA-prilozhenij. Kibernetika. Upravlenie v slozhnyh sistemah. Vestnik IrGTU. № 4 (44). 2010. S. 6–12.

2. Herasymov V. V., Levytska V. Ya. Analiz tekhnolohii rozrobky nasychenykh internet-dodatkiv na platformi Java. IT proektirovanija, modelirovanija, dizajna, WEB, 2017. S. 355–363.

3. Karashetskyi V. P. Pobudova hrafichnoho kontentu dodatkiv z vykorystanniam JavaFX i Swing komponentiv i danykh, vziatykh iz baz danykh. Naukovyi visnyk NLTU. 2015. Vyp. 25.1. S. 386–392.

4. Mashnin T. S. JavaFX 2.0: razrabotka RIA-prilozhenij. SPb.: BHV-Peterburg, 2012. 320 s.

5. Muliar V. P., Fedoniuk A. A. Kompiuterne modeliuvannia fizychnykh protsesiv i yavyshch: navch. posib. Lutsk: PP Ivaniuk V. P., 2018. 212 s.

6. Parfenov Ju. Je., Fedorchenko V. N. Razrabotka «nasyshhennyh» internet-prilozhenij s pomoshh'ju JavaFX. Systemy obrobky informatsii. 2012. Vyp. 8 (106). S. 40–46.

7. Horstmann K. Java SE 8. Vvodnyj kurs. M.: Vil'jams, 2014. 208 s.

8. Carl Dea. *JavaFX 2.0: Introduction by Example.* Apress, 2011. 181 p.

9. JavaFX – Application. URL: https://www.tutorialspoint.com/javafx/javafx_application.htm

10. JavaFX Scene Builder. URL: http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

11. Kishori Sharan. *Learn JavaFX 8 (The Expert's Voice in Java): Building User Experience and Interfaces with Java 8.* New York, 2015. P. 1–1173.

12. Stephen Chin, Johan Vos, James Weaver. *The Definitive Guide to Modern Java Clients with JavaFX: Cross-Platform Mobile and Cloud Development.* Apress, 2019. 621 p.

Рецензенти:

Пех П. А., завідувач кафедри комп'ютерної інженерії та кібербезпеки Луцького національного технічного університету, кандидат технічних наук, доцент.

Яцюк С. М., в.о. завідувача кафедри вищої математики та інформатики Східноєвропейського національного університету імені Лесі Українки, кандидат педагогічних наук, доцент.