Тема: розробка простого desktop-застосунку для обліку товарів для крамниці телефонів

Додатки:

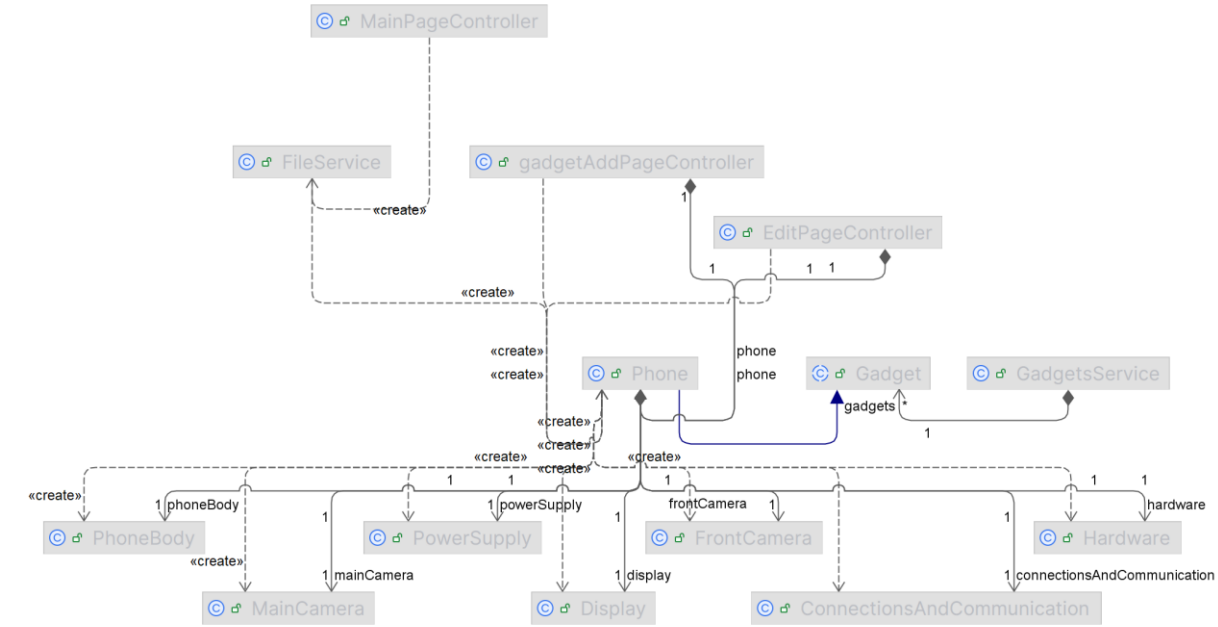
ВСТУП

У сучасних умовах цифрової трансформації бізнесу ефективне управління товарним обліком є важливою складовою успішної діяльності будь-якого торговельного підприємства. Особливо це стосується невеликих крамниць, які потребують простих, доступних та надійних засобів для організації внутрішніх процесів.

Ручне ведення обліку товарів призводить до зростання кількості помилок, втрати інформації та ускладнення контролю за наявністю продукції. Впровадження програмного забезпечення дозволяє значно підвищити точність обліку, прискорити процес обробки даних і знизити навантаження на персонал.

Метою даної роботи є розробка простого настільного застосунку для обліку товарів у крамниці мобільних телефонів. Застосунок має забезпечувати базові функції управління товарним асортиментом: додавання, редагування, видалення, перегляд та пошук товарів. Програма повинна бути зручною у користуванні, автономною, тобто не вимагати підключення до інтернету чи зовнішньої бази даних, і зберігати дані у форматі CSV.

У процесі реалізації було використано мову програмування **Java** та бібліотеку **JavaFX** для побудови графічного інтерфейсу. Такий вибір технологій дозволив створити легке та надійне рішення, придатне до подальшого розширення функціоналу відповідно до потреб бізнесу.


MainPageController
FileService
gadgetAddPageController
EditPageController
«create»
1
1 1
1 1
«create»
«create»
«create»
phone
phone
Phone
Gadget
GadgetsService
gadgets
*
1
«create»
«create»
«create»
«create»
«create»
1 1
1 1
1 1
«create»
1 phoneBody
1 powerSupply
frontCamera 1
hardware
PhoneBody
PowerSupply
FrontCamera
Hardware
«create»
1 mainCamera
1 display
1 connectionsAndCommunication
MainCamera
Display
ConnectionsAndCommunication

Розділ 1. Опис та Аналіз предметної області

1.1. Актуальність теми

У сучасних умовах динамічного розвитку роздрібної торгівлі, особливо в галузі електроніки, зокрема мобільних телефонів, зростає потреба в автоматизації обліку товарів. Більшість невеликих крамниць телефонів стикаються з проблемами ведення товарного обліку вручну: це призводить до втрати часу, помилок у документації, труднощів у контролі залишків та веденні звітності.

Використання спеціалізованого програмного забезпечення дозволяє суттєво оптимізувати бізнес-процеси, підвищити точність обліку та полегшити управління асортиментом. Desktop-застосунок є особливо зручним для малих підприємств через простоту використання, відсутність потреби в інтернет-з’єднанні та невисокі технічні вимоги.

Отже, розробка простого у використанні desktop-застосунку для обліку товарів є актуальним і практично значущим завданням.

1.2. Аналіз існуючих рішень

На ринку існує низка рішень для обліку товарів: 1С, Poster POS, Торгсофт, та інші. Однак більшість із них мають або надлишкову функціональність, або складні в освоєнні для малого бізнесу. Такі системи вимагають додаткового навчання персоналу, налаштувань або підписки.

Безкоштовні або умовно-безкоштовні програми часто мають обмежений функціонал, рекламу або низьку стабільність. Крім того, багато з них є веб-застосунками, що вимагають постійного доступу до інтернету, що не завжди зручно.

Таким чином, є потреба у простому, автономному desktop-застосунку, який забезпечує базові функції обліку — додавання, редагування, пошук та

видалення товарів, зберігання інформації про моделі, виробників, ціни та кількість на складі.

1.3. Висновки до розділу 1

У ході аналізу встановлено, що проблема обліку товарів залишається актуальною для малих крамниць мобільних телефонів. Існуючі рішення або занадто складні, або не повністю відповідають потребам малого бізнесу. Розробка власного desktop-застосунку дозволить створити зручний інструмент, адаптований під конкретні задачі користувача, з інтуїтивно зрозумілим інтерфейсом та базовим функціоналом для ефективного управління товарними залишками.

Розділ 2. Постановка задачі на розробку програмного забезпечення

2.1. Мета розробки

Метою розробки є створення простого у використанні desktop-застосунку для обліку товарів у крамниці, що спеціалізується на продажі мобільних телефонів. Застосунок має забезпечити зручний інтерфейс для щоденного використання продавцем або адміністратором магазину, автоматизувати основні операції з базою товарів, зменшити кількість помилок під час обліку, спростити пошук інформації та вести звітність.

2.2. Завдання, які вирішуються

У межах розробки даного застосунку вирішуються наступні завдання:

- створення та підтримка локальної бази даних товарів;
- додавання, редагування та видалення записів про товари;
- пошук товару за назвою, ціною або іншими параметрами;
- створення простого інтерфейсу користувача з формами та таблицями;
- забезпечення надійного зберігання даних без втрати інформації при перезапуску програми.

2.3. Основні вимоги до системи

Основні вимоги до програмного забезпечення поділяються на функціональні та нефункціональні:

Функціональні вимоги:

- ведення обліку товарів з полями: назва, ціна, опис;
- можливість редагування записів;
- швидкий пошук по товарах;

Нефункціональні вимоги:

- простий і зрозумілий інтерфейс;
- стабільна робота на ОС Windows;
- мінімальні апаратні вимоги.

2.4. Вхідні та вихідні дані

Вхідні дані:

Користувач (адміністратор магазину) вводить дані про товари: назву, ціну , тощо.

Вихідні дані:

Програма виводить:

- список наявних товарів;
- результати пошуку;
- повідомлення про успішне додавання, редагування чи видалення;

2.5. Середовище виконання

Розробка застосунку буде здійснюватися для середовища Windows 10 або вище.

Програмне забезпечення буде реалізоване мовою програмування Java з використанням бібліотеки **Javafx** для побудови графічного інтерфейсу. Для зберігання даних використовується текстовий файл у форматі csv.

Розділ 3. Проєктування програмного забезпечення

Проєктування є ключовим етапом розробки, оскільки дозволяє визначити структуру системи, її компоненти та взаємозв’язки між ними. У цьому розділі буде представлено основні UML-діаграми, які ілюструють логіку та архітектуру застосунку: діаграму класів та діаграму станів.

Діаграма класів:

Діаграма класів відображає основні сутності системи, їх атрибути та методи.

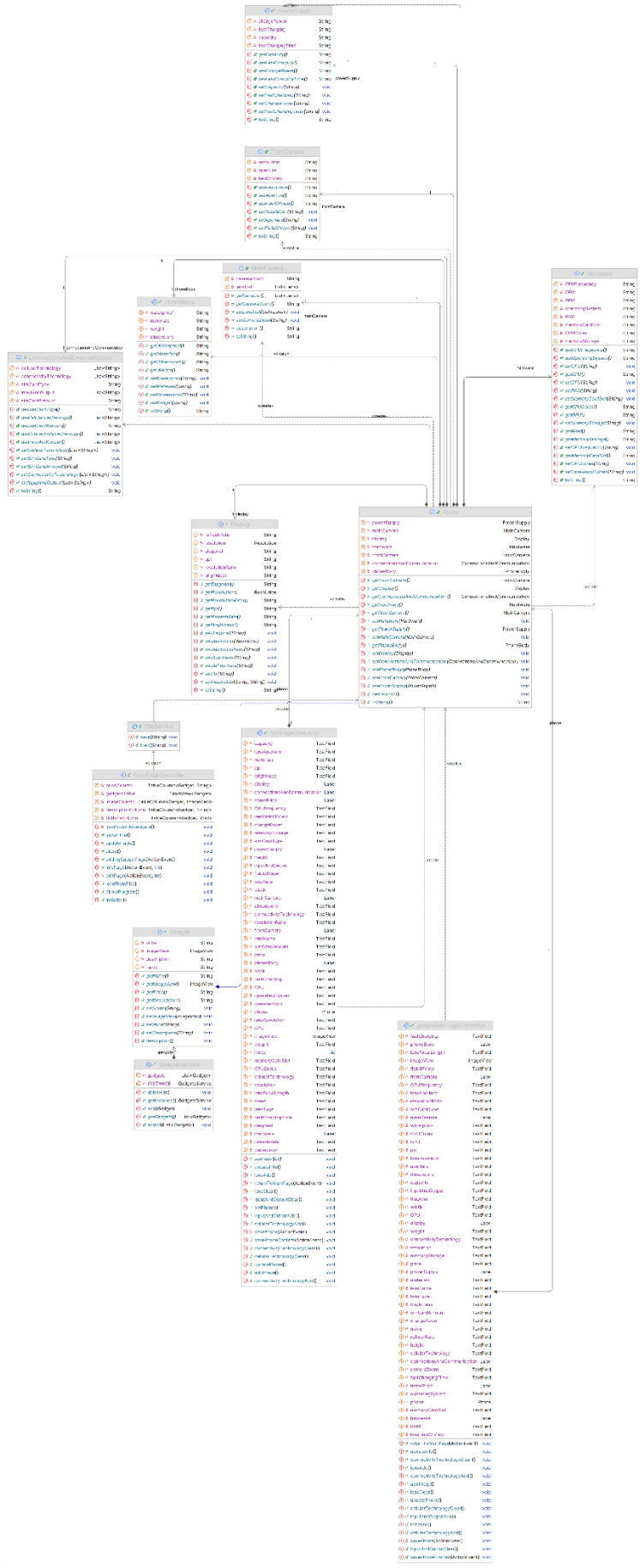


Рисунок 3. 1 – діаграма класів

Діаграма станів:

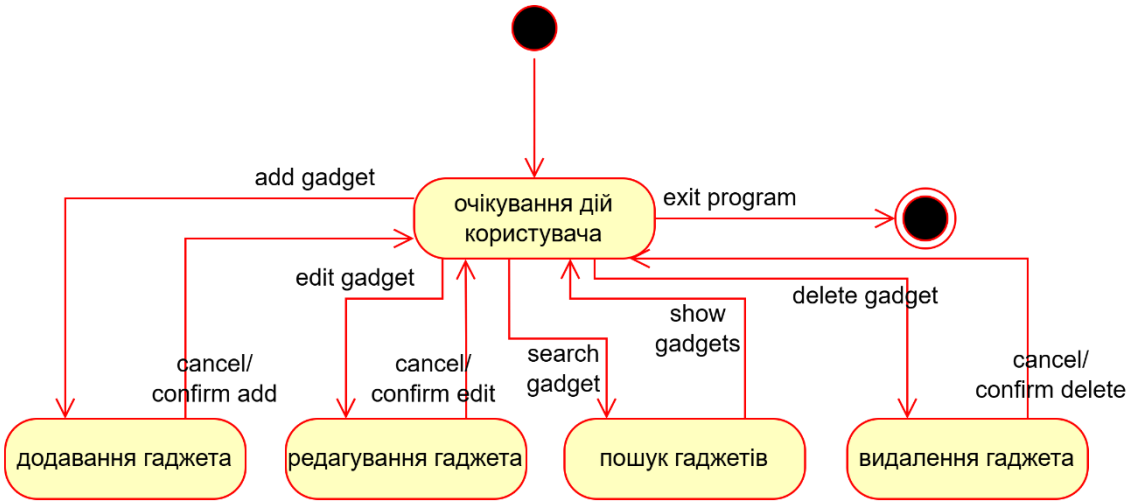


Рисунок 3. 2 – діаграма станів

Ця діаграма демонструє зміни станів застосунку в процесі його використання.

Розділ 4. Розробка програмного забезпечення

4.1. Обґрунтування вибору технологій

Для реалізації застосунку було обрано мову програмування **Java**, яка є стабільною, об'єктно-орієнтованою та кросплатформеною мовою, що забезпечує надійність та гнучкість у розробці настільних додатків.
Для створення графічного інтерфейсу використано **JavaFX** — сучасну бібліотеку для побудови GUI, яка дозволяє створювати зручні, інтерактивні та естетично привабливі вікна.
Як сховище даних обрано **CSV-файл**, що забезпечує простий та легкий у реалізації формат зберігання структурованої інформації без потреби в зовнішніх базах даних.

4.2. Архітектура застосунку

Архітектура застосунку поділена на три основні компоненти:

- **Модель (Model)** —включає класи, які описують структуру об’єктів.
- **Обробка данних (Service)** — зберігання даних про гаджети, робота с файлами.
- **Логіка управління (Controller)** — забезпечує взаємодію між моделлю та інтерфейсом: реагує на події користувача (кнопки «Додати», «Видалити», «Зберегти» тощо) та виконує відповідні дії з файлами й відображенням.

Такий поділ сприяє підтримуваності та масштабованості застосунку.

4.3. Реалізація основного функціоналу

У програмі реалізовано наступні функції:

- **Додавання товару:**

  Користувач заповнює форму (назва, ціна, дисплей, камера і т. д.), після чого інформація додається у внутрішній список і записується у CSV-файл.
- **Редагування товару:**

  Після вибору товару з таблиці дані завантажуються у форму. Зміни зберігаються як у пам’яті програми, так і в CSV-файлі (перезапис усього файлу).
- **Видалення товару:**

  Товар можна видалити зі списку та CSV-файлу. Перед видаленням програма запитує підтвердження.
- **Пошук товарів:**

  Реалізовано сортування за різними параметрами, пошук за ключовим словом
- **Табличне відображення:**

  Усі товари відображаються в JavaFX-таблиці (TableView) з можливістю сортування та прокручування. Після кожної зміни (додавання/редагування/видалення) таблиця оновлюється автоматично.

Розділ 5. Аналіз отриманих результатів

У результаті виконаної роботи було створено настільний застосунок для обліку товарів крамниці мобільних телефонів. Програма повністю відповідає поставленим у розділі 2 завданням та реалізована згідно з проєктними рішеннями, описаними у розділі 3.

5.1. Досягнуті результати

Основні досягнення:

- Реалізовано зручний графічний інтерфейс користувача за допомогою JavaFX;
- Забезпечено основний функціонал: додавання, редагування, видалення та пошук товарів;
- Інформація про товари надійно зберігається у CSV-файлі, що завантажується при запуску програми;
- Застосунок є автономним і не потребує зовнішньої бази даних або підключення до інтернету;

Завдяки використанню JavaFX застосунок має сучасний інтерфейс, а збереження даних у CSV-файлі забезпечує простоту обслуговування та зрозумілу структуру для резервного копіювання або переносу.

5.2. Висновки

Розроблений застосунок показав себе як ефективний інструмент для обліку товарів у невеликій крамниці. Простота використання, автономність та зрозуміла структура збереження даних роблять його придатним для практичного застосування в умовах малого бізнесу.

Також завдяки відкритій архітектурі, програму легко масштабувати — за потреби її можна розширити підтримкою експорту у PDF/Excel, статистикою

продажів або переходом до повноцінної бази даних (наприклад, SQLite чи MySQL).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Icon packs for Java applications URL:https://kordamp.org/ikonli/

2. draw.io URL: https://app.diagrams.net/

3. Introduction to JSON-Java URL: https://www.baeldung.com/java-org-json

4. JavaFX Dialogs (official) URL: https://code.makery.ch/blog/javafx-dialogs-official/?authuser=1

5. Joshua Bloch Effective java Third Edition URL: https://kea.nu/files/textbooks/new/Effective%20Java%20%282017%2C%20Addison-Wesley%29.pdf

Додаток А. Приклади екранних форм

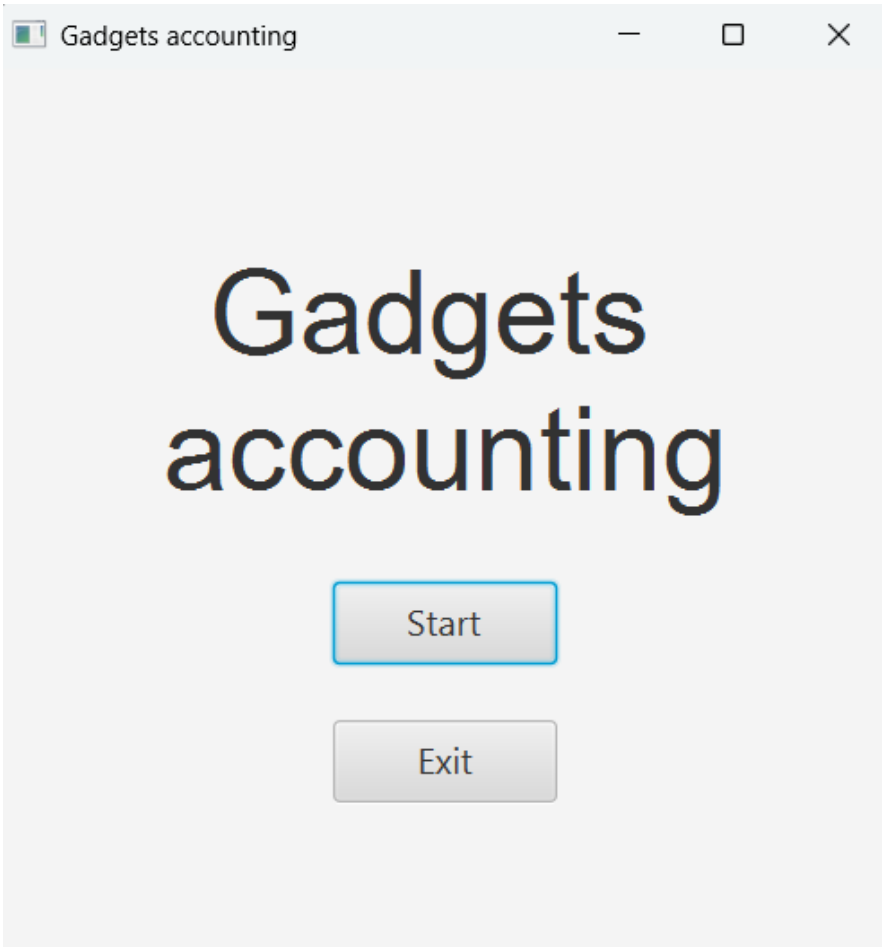


Рисунок 1 – Стартовий екран

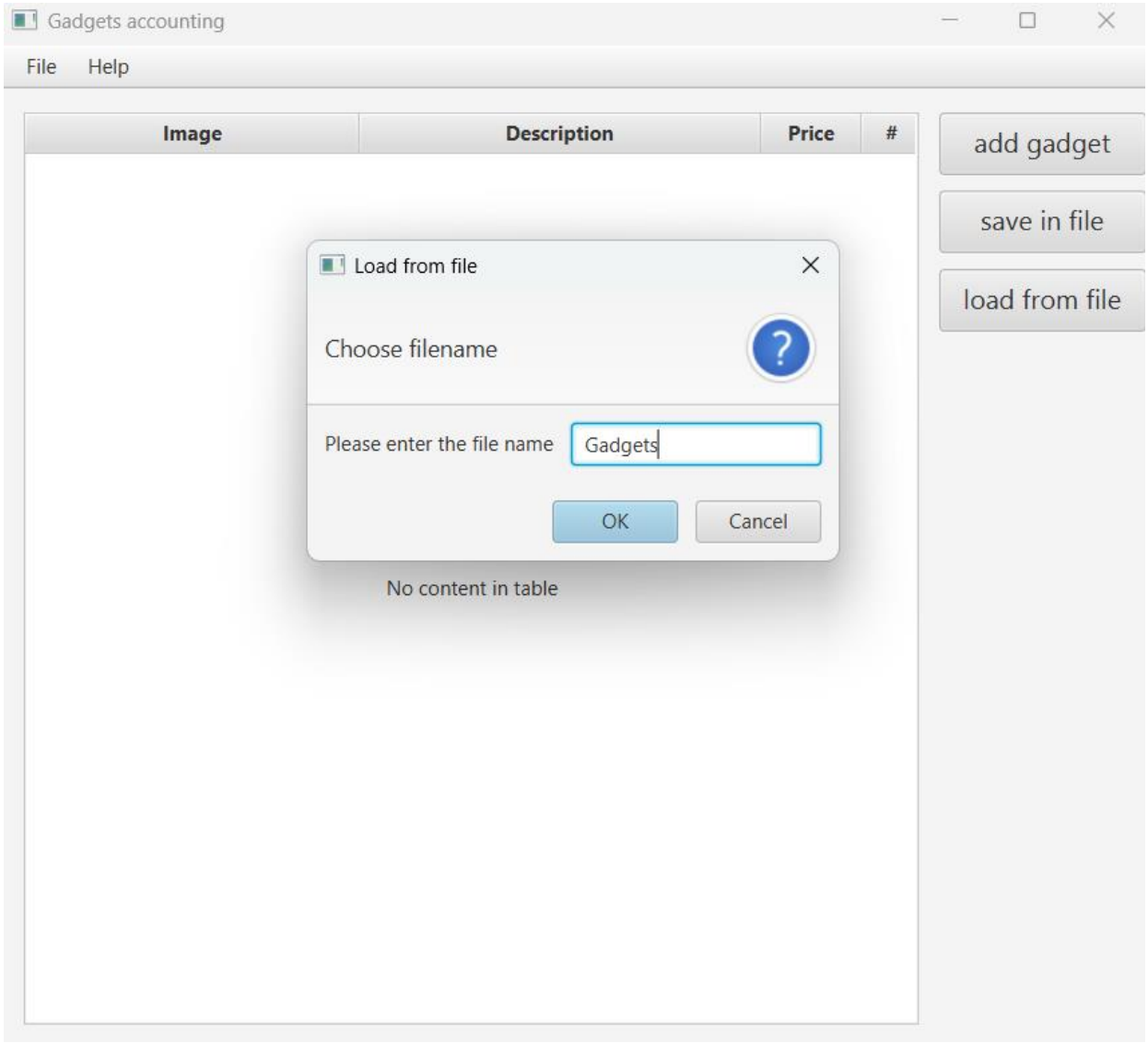


Рисунок 2 – вибір файлу для завантаження

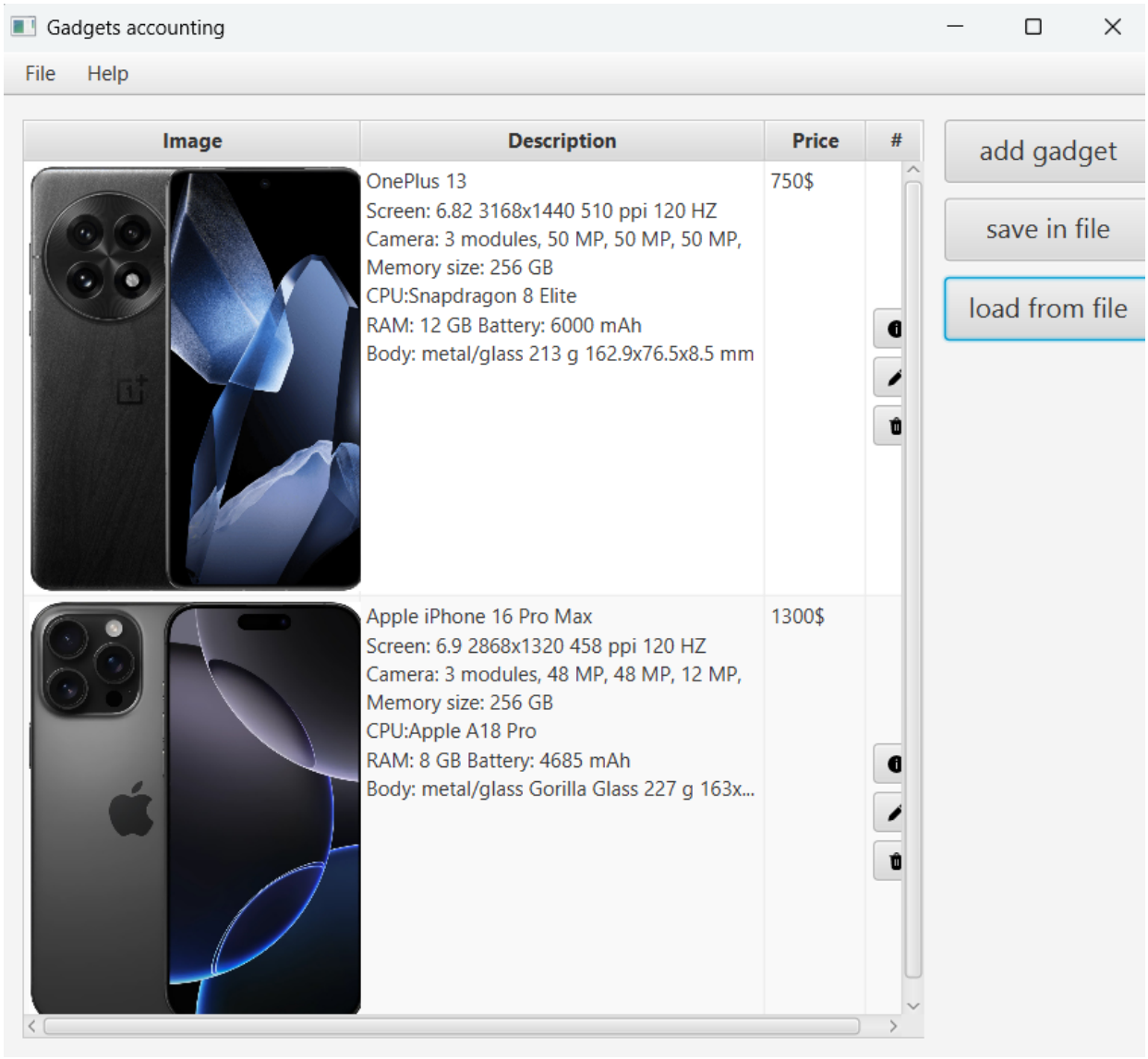


Рисунок 3 – показ гаджетів

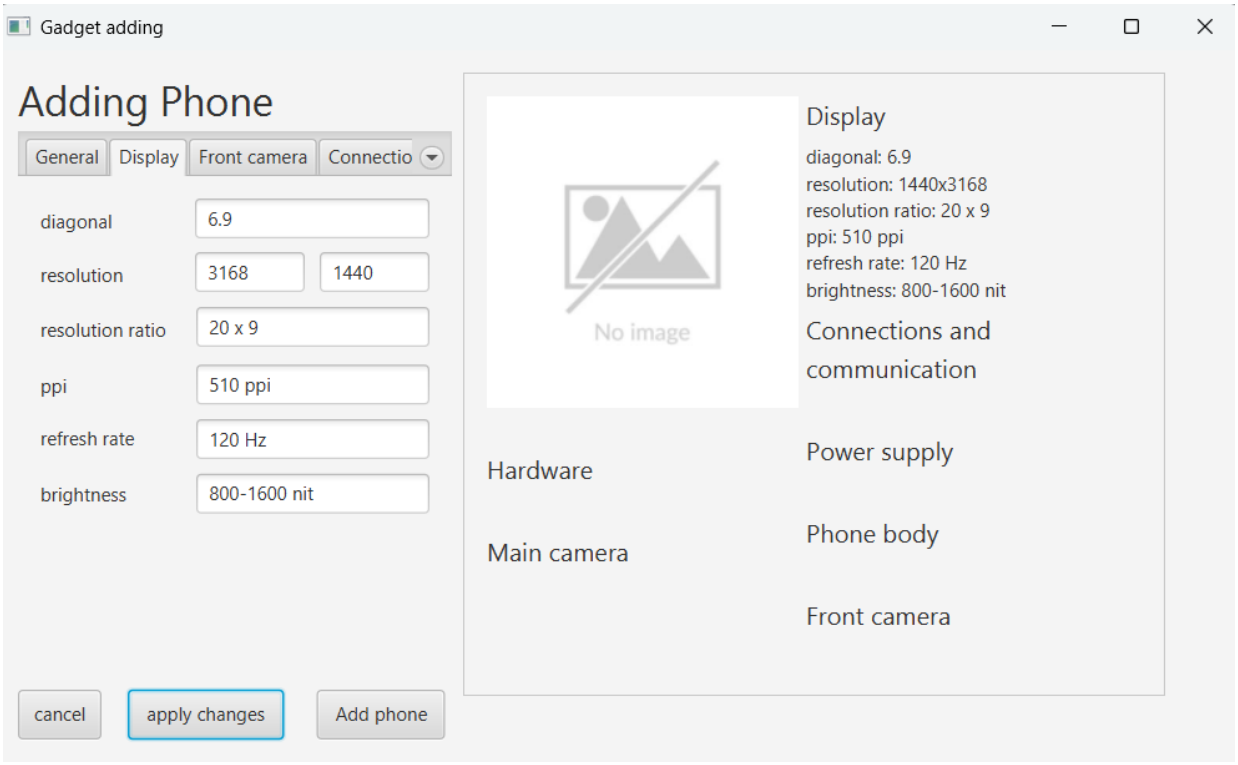


Рисунок 4 – додавання гаджету

Додаток Б. Коди програмних модулів

Код класу MainPageController:

```
package main.controllers;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import main.MainApp;
import main.model.Gadget;
import main.service.FileService;
import main.service.GadgetsService;
import org.kordamp.ikonli.javafx.FontIcon;
```

```
import java.io.IOException;
import java.util.Comparator;
import java.util.Optional;

public class MainPageController {
   @FXML
   private TableView<Gadget> gadgetsTable;
   @FXML
   private TableColumn<Gadget, Pane> imageColumn;
   @FXML
   private TableColumn<Gadget, String> descriptionColumn;
   @FXML
   private TableColumn<Gadget, String> priceColumn;
   @FXML
   private TableColumn<Gadget, Void> buttonsColumn;

   public  void initialize() {
      imageColumn.setCellValueFactory(new PropertyValueFactory<>("imageViewTable"));
      imageColumn.setSortable(false);
      descriptionColumn.setCellValueFactory(new PropertyValueFactory<>("description"));
      descriptionColumn.setSortable(false);
      priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));
      Comparator<String> priceColumnComparator = (o1, o2) -> {
         String q1="",q2="";
         for(Character ch : o1.toCharArray()){
            if((int)ch>47&(int)ch<58) q1 += ch.toString();
         }
         for(Character ch : o2.toCharArray()){
            if((int)ch>47&(int)ch<58) q2 += ch.toString();
         }
         if(Integer.parseInt(q1)>Integer.parseInt(q2)) return 1;
         else if(Integer.parseInt(q1)<Integer.parseInt(q2)) return -1;
         return 0;
      };
      priceColumn.setComparator(priceColumnComparator);
      buttonsColumn.setSortable(false);
      buttonsColumn.setCellFactory(col ->new TableCell<>(){
         private final Button infoButton=new Button("",new FontIcon("fltfal-info-28"));
         private final Button editButton=new Button("",new FontIcon("fltfal-edit-24"));
         private final Button deleteButton=new Button("",new FontIcon("fltfal-delete-28"));
         private final VBox vbox = new VBox(5,infoButton,editButton,deleteButton);
         {
            vbox.setAlignment(Pos.CENTER);
            infoButton.setOnAction(event -> infoPage(event,getIndex()));
            editButton.setOnAction(event -> editPage((event),getIndex()));
            deleteButton.setOnAction(event -> {
               Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
               alert.setTitle("Confirm phone deleting");
               alert.setHeaderText(null);
               alert.setContentText("Are you sure?");
```

```
            Optional<ButtonType> result = alert.showAndWait();
            result.ifPresent(buttonType -> {
              if (buttonType == ButtonType.OK) {
                GadgetsService.getInstance().delete(getIndex());
                updateTable();
              }
            });
          });
        }
        @Override
        protected void updateItem(Void item, boolean empty) {
          super.updateItem(item, empty);
          if (empty) {
            setGraphic(null);
          } else {
            setGraphic(vbox);
          }
        }
      });
      updateTable();
    }

    private void updateTable() {
      gadgetsTable.setItems(FXCollections.observableList(GadgetsService.getInstance().getGadgets()));
    }

    public void infoPage(ActionEvent actionEvent, int index) {
      FXMLLoader loader = new FXMLLoader(MainApp.class.getResource("infoPage.fxml"));
      Parent root;
      try {
        root = loader.load();
      } catch (IOException e) {
        throw new RuntimeException(e);
      }
      InfoPageController controller = loader.getController();
      controller.setIndex(index);
      controller.gadgetView();
      Stage newStage=new Stage();
      newStage.setScene(new Scene(root));
      newStage.setTitle("Info");
      newStage.setMinWidth(400);
      newStage.setMinHeight(500);
      newStage.show();
      Stage currentStage = (Stage)((Node)actionEvent.getSource()).getScene().getWindow();
      currentStage.close();
    }

    private void editPage(ActionEvent actionEvent, int index) {
      FXMLLoader loader = new FXMLLoader(MainApp.class.getResource("editPage.fxml"));
      Parent root;
```

```
        try {
            root = loader.load();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        EditPageController controller = loader.getController();
        controller.setIndex(index);
        controller.setPhone();
        controller.updateInfo();
        Stage newStage=new Stage();
        newStage.setScene(new Scene(root));
        newStage.setTitle("Editing gadget");
        newStage.setMinWidth(400);
        newStage.setMinHeight(500);
        newStage.show();
        Stage currentStage = (Stage)((Node)actionEvent.getSource()).getScene().getWindow();
        currentStage.close();
    }

    public void addingGadgetPage(ActionEvent actionEvent) throws IOException {
        Stage newStage=new Stage();
        FXMLLoader loader = new FXMLLoader(MainApp.class.getResource("gadgetAddPage.fxml"));
        Scene scene = new Scene(loader.load(), 800, 450);
        newStage.setScene(scene);
        newStage.setTitle("Gadget adding");
        newStage.setMinWidth(770);
        newStage.setMinHeight(500);
        newStage.show();
        Stage currentStage = (Stage)((Node)actionEvent.getSource()).getScene().getWindow();
        currentStage.close();
    }

    public void aboutProgram() {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("About Program");
        alert.setHeaderText("OOP_Kursova");
        alert.setContentText("This program was developed for a coursework assignment");
        alert.showAndWait();
    }

    public void close() {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Confirm Exit");
        alert.setHeaderText(null);
        alert.setContentText("Are you sure?");
        alert.showAndWait();
        if (alert.getResult() == ButtonType.OK) Platform.exit();
    }

    public void saveInFile() {
```

```
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Save in file");
        dialog.setHeaderText("Choose filename");
        dialog.setContentText("Please enter the file name");
        Optional<String> result = dialog.showAndWait();
        if (result.isPresent()) {
            String fileName = result.get();
            FileService fileService = new FileService();
            fileService.save(fileName);
        }
    }

    public void loadFromFile() {
        TextInputDialog dialog = new TextInputDialog("Gadgets");
        dialog.setTitle("Load from file");
        dialog.setHeaderText("Choose filename");
        dialog.setContentText("Please enter the file name");
        Optional<String> result = dialog.showAndWait();
        if (result.isPresent()) {
            String fileName = result.get();
            FileService fileService = new FileService();
            fileService.load(fileName);
            updateTable();
        }
    }
}
```