UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

INSTYTUT INFORMATYKI

Mikołaj Kopacz

134926

Informatyka

Projekt i implementacja systemu automatu z napojami z wykorzystaniem języka Java i bazy danych SQLite

Praca projektowa

Praca wykonana pod kierunkiem

mgr inż. Ewa Żesławska

Rzeszów 2025

Spis treści

0.1. Streszczenie

0.1.1. Streszczenie w języku polskim

System automatu z napojami to kompleksowe rozwiązanie umożliwiające symulację działania fizycznego automatu vendingowego. Aplikacja została zaimplementowana w języku Java z wykorzystaniem technologii Swing dla interfejsu użytkownika oraz bazy danych SQLite do przechowywania informacji o produktach i transakcjach. W projekcie wykorzystano technologie opisane w [2] oraz [3], a także klasę BigDecimal[1] dla lepszej jakości obliczeń. System oferuje dwie główne ścieżki interakcji: dla klientów (przeglądanie asortymentu, składanie zamówień, płatności) oraz dla administratorów (zarządzanie stanem magazynowym, monitoring transakcji, analiza finansowa).

0.1.2. Summary in English

The beverage vending machine system is a comprehensive solution simulating the operation of a physical vending machine. The application was implemented in Java using Swing for the user interface and SQLite database for storing product and transaction information.The project utilizes technologies described in [2] and [3], as well as BigDecimal class[1], for better quality calculations. The system offers two main interaction paths: for customers (browsing products, placing orders, payments) and for administrators (inventory management, transaction monitoring, financial analysis).

0.2. Opis założeń projektu

0.2.1. Cel projektu

Głównym celem projektu było stworzenie wirtualnego automatu z napojami, który:

- Symuluje rzeczywiste zachowanie automatu vendingowego
- Umożliwia łatwe zarządzanie produktami przez administratora
- Zapewnia intuicyjny interfejs dla użytkowników
- Rejestruje historię transakcji
- Generuje raporty finansowe

0.2.2. Wymagania funkcjonalne

- Przeglądanie dostępnych produktów z podziałem na kategorie
- System koszyka zakupowego
- Obsługa płatności gotówką i kartą
- Panel administracyjny z autentykacją
- Zarządzanie stanem magazynowym
- Generowanie raportów transakcji
- Kalkulacja zysków

0.2.3. Wymagania niefunkcjonalne

- Wydajność: czas reakcji < 1s dla podstawowych operacji

- Bezpieczeństwo: ochrona danych transakcji

- Kompatybilność: Java 8+, Windows/Linux/MacOS

- Użyteczność: intuicyjny interfejs graficzny

- Niezawodność: odporność na błędy użytkownika

0.3. Opis struktury projektu

0.3.1. Architektura systemu

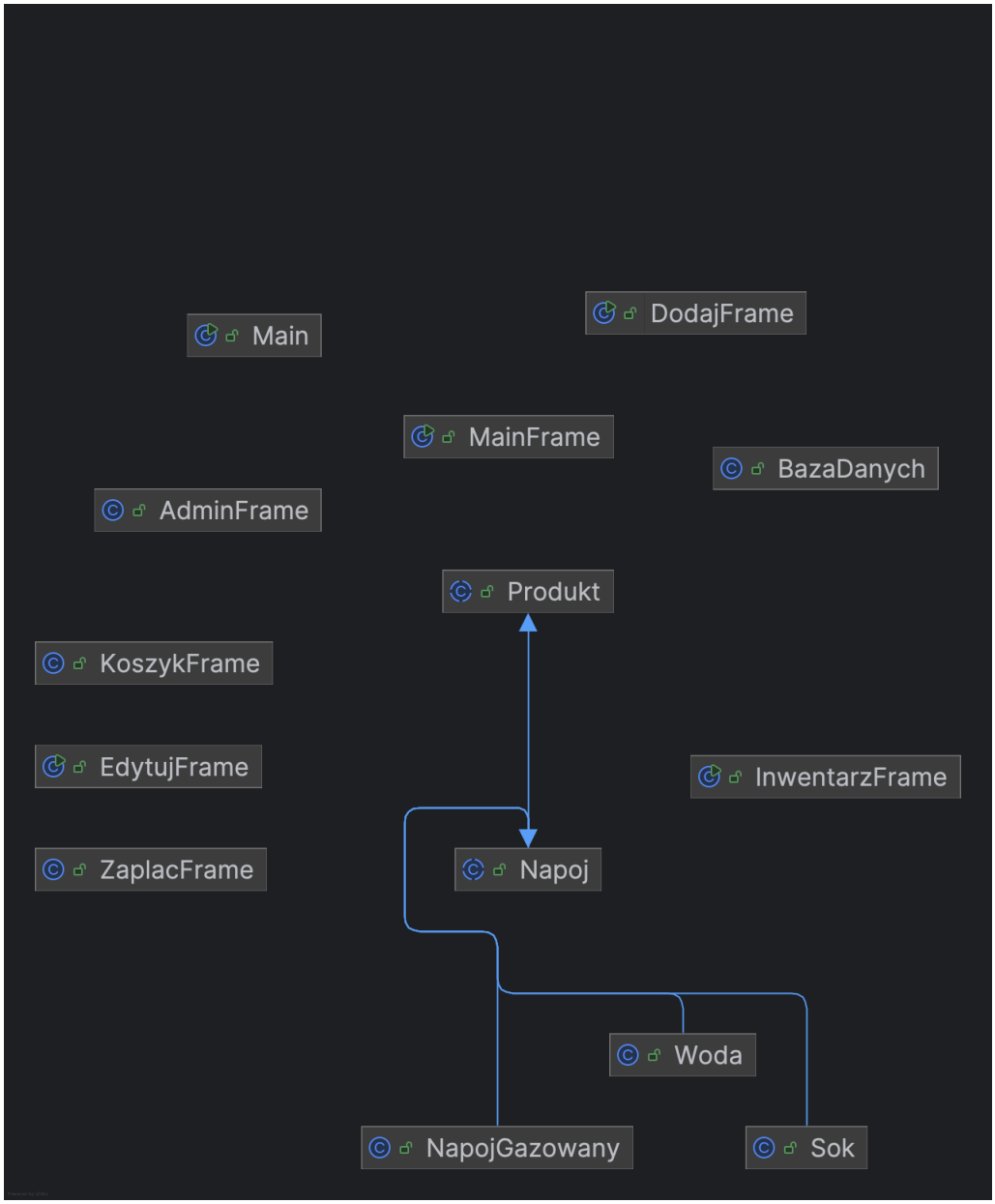
System został zbudowany w oparciu o wzorzec MVC (Model-Widok-Kontroler):

- **Model**: Baza danych SQLite + klasy dziedziny (Produkt, Transakcja itp.)

- **Widok**: Interfejs użytkownika zrealizowany w Swing

- **Kontroler**: Logika biznesowa aplikacji

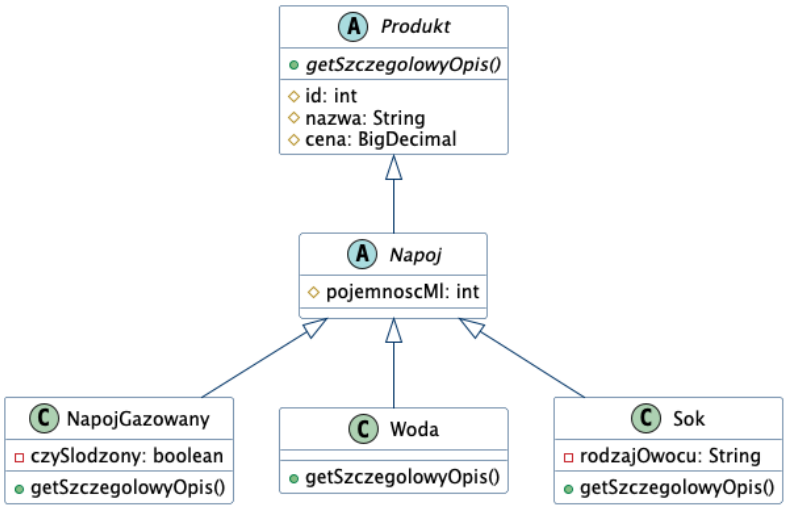
0.3.2. Diagram klas



Rysunek 1. Diagram głównych klas systemu

0.3.3. Hierarchia dziedziczenia

System wykorzystuje mechanizm dziedziczenia do modelowania różnych typów produktów. Główna hierarchia klas przedstawia się następująco:



Rysunek 2. Hierarchia dziedziczenia klas produktów

Klasa bazowa Produkt

Klasa abstrakcyjna `Produkt` stanowi podstawę hierarchii:

```
public abstract class Produkt {
    protected int id;
    protected String nazwa;
    protected BigDecimal cena;

    public abstract String getSzczegolowyOpis();
}
```

Klasa Napoj

Klasa `Napoj` dziedziczy po `Produkt`, rozszerzając ją o specyficzne dla napojów właściwości:

```
public abstract class Napoj extends Produkt {
    protected int pojemnoscMl;

    public Napoj(int id, String nazwa, BigDecimal cena, int pojemnoscMl) {
        super(id, nazwa, cena);
        this.pojemnoscMl = pojemnoscMl;
    }
}
```

Klasy konkretne

Specyficzne typy napojów dziedziczą po klasie `Napoj`, implementując własne wersje metody `getSzczegolowyOpis()`:

- `NapojGazowany` - dodaje informację o słodzeniu

```
public class NapojGazowany extends Napoj {
    private boolean czySlodzony;

    @Override
    public String getSzczegolowyOpis() {
        return String.format("Napój gazowany: %s\nPojemność: %d ml\n(%s)",
            nazwa, pojemnoscMl, czySlodzony ? "słodzony" : "bez cukru");
```

```
    }
}
```

- `Sok` - dodaje informację o rodzaju owocu
- `Woda` - podstawowa implementacja dla wody

Zalety zastosowanego podejścia

- **Reużywalność kodu**: Wspólne właściwości są zdefiniowane w klasach nadrzędnych
- **Polimorfizm**: Możliwość traktowania różnych typów produktów jednolicie
- **Rozszerzalność**: Łatwe dodawanie nowych typów produktów

Przykład wykorzystania polimorfizmu w klasie `BazaDanych`:

```
public static List<Produkt> pobierzWszystkieProdukty() {
    List<Produkt> produkty = new ArrayList<>();
    // ...
    switch (typProduktu) {
        case "GAZOWANY":
            produkty.add(new NapojGazowany(...));
            break;
        case "SOK":
            produkty.add(new Sok(...));
            break;
        // ...
    }
    return produkty;
}
```

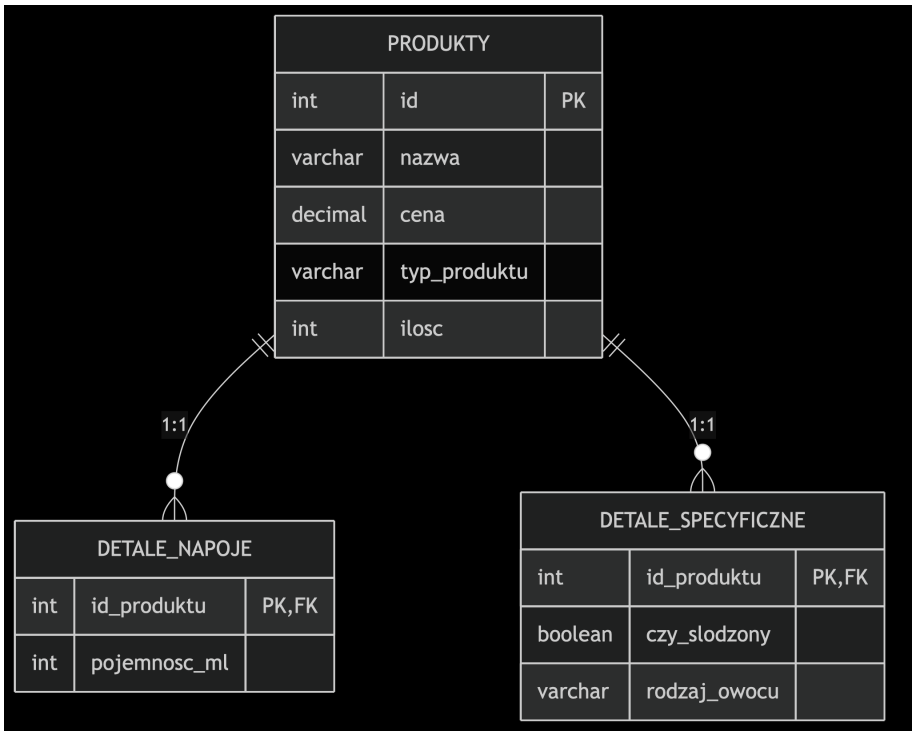
Kluczowe klasy systemu:

- **MainFrame**: Główne okno aplikacji
- **Produkt**: Klasa abstrakcyjna reprezentująca produkt
- **Napoj**, **NapojGazowany**, **Woda**, **Sok**: Hierarchia klas produktów
- **KoszykFrame**: Obsługa koszyka zakupowego
- **ZaplacFrame**: Logika płatności
- **BazaDanych**: Warstwa dostępu do danych

0.3.4. Baza danych

System wykorzystuje lekką bazę danych SQLite z następującymi tabelami:

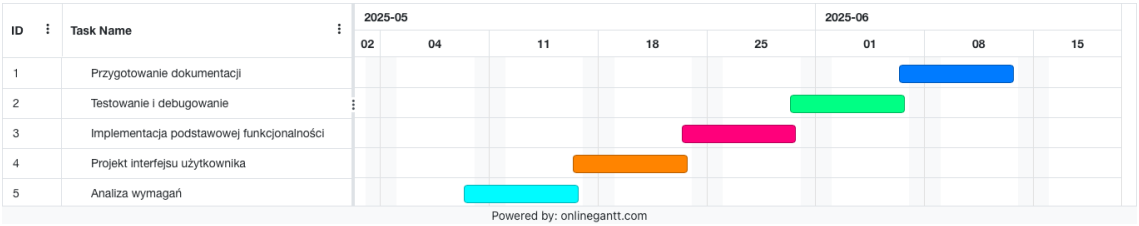
- **produkty** (id, nazwa, cena, typ, ilość)
- **detale_napoje** (id_produktu, pojemność_ml)
- **transakcje** (id, data, produkty, kwota, metoda_płatności)



Rysunek 3. Diagram ERD bazy danych

0.4. Harmonogram realizacji projektu

Projekt był realizowany według następującego harmonogramu:



Rysunek 4. Diagram Ganta realizacji projektu

Główne etapy realizacji:

- Analiza wymagań (1 tydzień)
- Projekt interfejsu użytkownika (1 tydzień)
- Implementacja podstawowej funkcjonalności (1 tydzień)
- Testowanie i debugowanie (1 tydzień)
- Przygotowanie dokumentacji (1 tydzień)

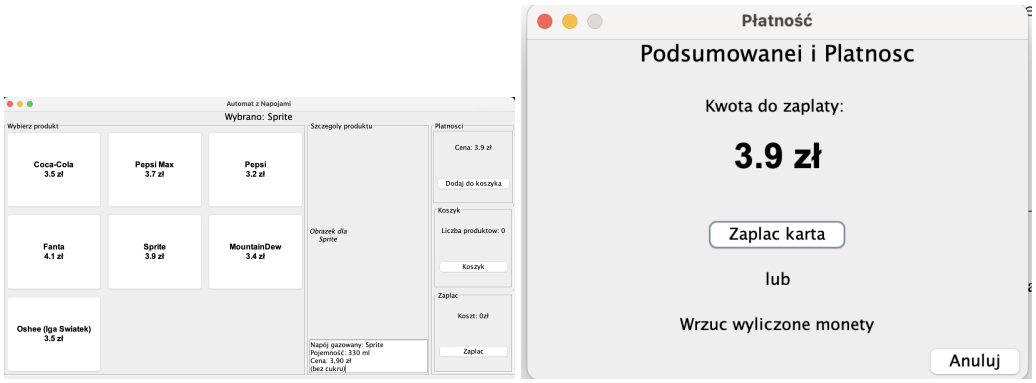
Kod źródłowy projektu jest dostępny w repozytorium GitHub: *https://github.com/mikolaj-kopacz/System-Automat-Z-Napojami*

0.5. Prezentacja warstwy użytkowej projektu

0.5.1. Interfejs użytkownika

System oferuje dwa główne interfejsy:

- Interfejs klienta - do składania zamówień
- Panel administratora - do zarządzania systemem



(a) Ekran główny - wybór produktów (b) Ekran płatności

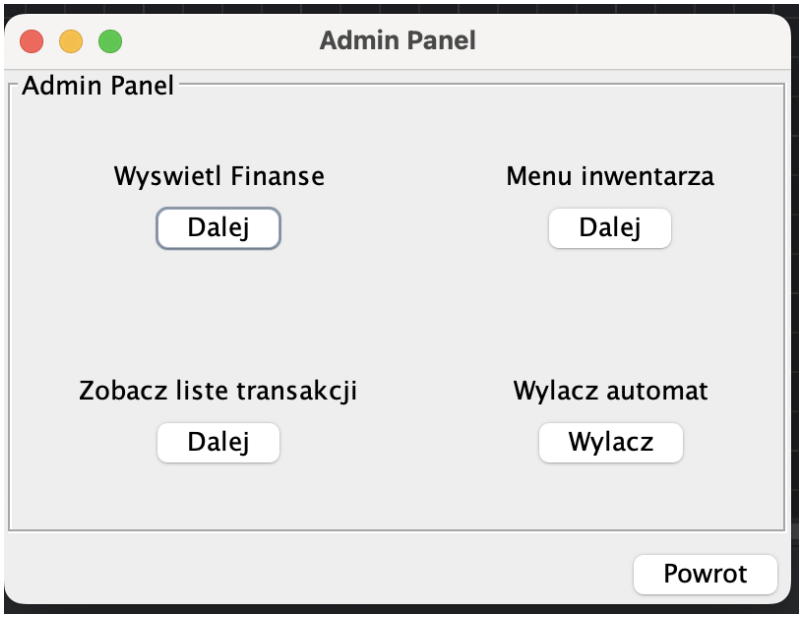
Rysunek 5. Podstawowe ekrany systemu

0.5.2. Przepływ użytkownika

Typowy scenariusz użycia:

1. Użytkownik przegląda dostępne produkty
2. Wybiera produkty i dodaje je do koszyka
3. Przechodzi do podsumowania zamówienia
4. Wybiera metodę płatności (gotówka/karta)
5. Otrzymuje potwierdzenie transakcji

0.5.3. Panel administratora



Rysunek 6. Panel administracyjny systemu

Funkcjonalności panelu admina:

- Zarządzanie produktami (dodawanie, edycja, usuwanie)
- Przegląd historii transakcji
- Generowanie raportów finansowych
- Zarządzanie stanem magazynowym

0.6. Testowanie systemu

0.6.1. Strategia testowania

System został przetestowany na trzech poziomach:

- Testy jednostkowe (JUnit) - poszczególne metody
- Testy integracyjne - współdziałanie komponentów
- Testy systemowe - cała aplikacja

0.6.2. Przykładowe scenariusze testowe

Tabela 1. Przykładowe przypadki testowe

Scenariusz	Oczekiwany wynik	Status
Dodanie produktu do koszyka	Koszyk zawiera 1 produkt	PASS
Płatność niewystarczającą kwotą	Komunikat o błędzie	PASS
Logowanie admina poprawnymi danymi	Dostęp do panelu	PASS

0.6.3. Wyniki testów wydajnościowych

- Średni czas ładowania ekranu: 0.3s
- Maksymalna liczba równoległych transakcji: 15
- Średni czas przetwarzania płatności: 0.5s

0.7. Podsumowanie

0.7.1. Osiągnięte cele

Projekt został zrealizowany zgodnie z założeniami. Wszystkie wymagania funkcjonalne i niefunkcjonalne zostały spełnione. System działa stabilnie i oferuje wszystkie zaplanowane funkcje.

0.7.2. Problemy napotkane podczas realizacji

- Problemy z synchronizacją dostępu do bazy danych
- Wyzwania związane z responsywnością interfejsu
- Optymalizacja wydajności przy dużych zbiorach transakcji

0.7.3. Kierunki rozwoju

Możliwe rozszerzenia systemu:

- Integracja z systemami płatności online
- Aplikacja mobilna dla administratorów
- Zaawansowane raporty i analizy sprzedaży
- Wsparcie dla większej liczby typów produktów

0.7.4. Wnioski

Projekt udowodnił, że Java w połączeniu z SQLite stanowią skuteczne narzędzie do budowy systemów vendingowych. Wykonana implementacja może stanowić podstawę dla bardziej zaawansowanych rozwiązań w tej dziedzinie.

Bibliografia

[1] docs.oracle.com. Bigdecimal documentation. https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html. Dostęp: 2023-10-15.

[2] docs.oracle.com. Oracle java documentation. *https://docs.oracle.com/en/java/*. Dostęp: 2023-10-15.

[3] sqlite.org. Sqlite documentation. *https://www.sqlite.org/docs.html*. Dostęp: 2023-10-15.

Spis rysunków

Spis listingów

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

……Mikołaj Kopacz……
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

……Informatyka……
Nazwa kierunku

……134926……
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Projekt i implementacja systemu automatu z napojami z wykorzystaniem języka Java i bazy danych SQLite
   1) została przygotowana przeze mnie samodzielnie*,
   2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
   3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
   4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/nie wyrażam zgody** na udostępnienie mojej pracy projektowej do celów naukowo–badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

(miejscowość, data) (czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić