

# „SKLEPSHOP”

---

*Imię i nazwisko: Michał Dyjak*

*Numer albumu: 125114*

*Temat: Aplikacja służąca do obsługi sklepu spożywczego*

*Przedmiot Programowanie Obiektowe*

*Grupa laboratoryjna nr 1*

*Data oddania projektu: 29.01.2024*

---

## Spis treści

Cel aplikacji oraz ogólny opis:.....	3
Wymagania systemowe:.....	4
Diagram ERD (Entity-Relationship Diagram):.....	4
Tabela products .....	4
Tabela settlements .....	5
Instrukcja instalacji: .....	6
Opis Interfejsu Użytkownika: .....	7
Panel Produktów: .....	7
Koszyk Zakupów: .....	8
Rozliczenia: .....	8
Edycja Produktów: .....	9
Historia Transakcji:.....	9
Funkcjonalności: .....	10
Przeglądanie Produktów:.....	10
Dodawanie Produktów do Koszyka: .....	10
Rozliczenia: .....	10
Edycja Produktów: .....	10
Historia Transakcji:.....	10
Bezpieczeństwo: Zabezpieczenia i Spójność Bazy Danych.....	11
Dokumentacja API, Opis Metod i Klas: .....	11
Products_LoadProducts:.....	11
Products_Management: .....	11
Settlements: .....	12
AddedProduct:.....	12
Przydatne Informacje: .....	13
Ograniczenia Aplikacji:.....	13
Baza Danych:.....	13
Interfejs Użytkownika: .....	13
Zalety Aplikacji:.....	13
Łatwość Używania: .....	13
Elastyczność:.....	13
Dokumentacja API: .....	13
Zabezpieczenia: .....	13
Wykorzystanie Technologii: .....	13

## Cel aplikacji oraz ogólny opis:

Celem aplikacji jest stworzenie kompleksowego systemu obsługi sklepu, który umożliwia zarządzanie produktem, przeglądanie oferty, dodawanie produktów do koszyka, a następnie finalizację zakupu. Aplikacja ma na celu dostarczenie wygodnego i intuicyjnego interfejsu dla klientów, a także efektywnych narzędzi administracyjnych dla obsługi produktów.

Aplikacja składa się z kilku kluczowych komponentów, oferując różnorodne funkcjonalności:

**EditProductsDB:** Klasa umożliwiająca zarządzanie bazą danych produktów. Administratorzy mogą dodawać nowe produkty, edytować istniejące, a także usuwać nieaktualne pozycje. Dzięki tej klasie, aktualizacja oferty sklepu staje się łatwa i efektywna.

**Products\_LoadProducts:** Klasa odpowiedzialna za dynamiczne ładowanie produktów do interfejsu użytkownika z uwzględnieniem różnych kategorii, opcji sortowania oraz możliwości wyszukiwania. Użytkownicy mogą łatwo przeglądać ofertę, dostosowując widok do swoich preferencji.

**Products\_Management:** Klasa zarządzająca koszykiem zakupowym. Użytkownicy mogą dodawać produkty do koszyka, określać ich ilość, a następnie finalizować zakupy. Mechanizmy tej klasy uwzględniają również obliczanie łącznej wartości koszyka.

**Settlements:** Klasa przechowująca informacje o dokonanych transakcjach. Zapewnia pełny przegląd historii zakupów, w tym daty, metody płatności, identyfikatory produktów i łączną kwotę transakcji.

**AddedProduct:** Klasa reprezentująca produkty dodane do koszyka. Zawiera informacje takie jak identyfikator produktu, nazwa, cena jednostkowa i ilość.

**Product:** Klasa definiująca strukturę pojedynczego produktu, zawierająca informacje takie jak identyfikator, nazwa, cena, kategoria itp.

**Settlement:** Klasa definiująca strukturę rozliczenia transakcji, obejmująca identyfikator, produkty, metodę płatności, datę i łączną kwotę.

**App:** Główna klasa aplikacji, integrująca wszystkie komponenty i obsługująca interakcję z użytkownikiem. Wykorzystuje język Java, JavaFX do budowy interfejsu graficznego oraz Hibernate do komunikacji z bazą danych SQLite.

Aplikacja umożliwia pełne doświadczenie zakupowe, obejmujące przeglądanie, dodawanie produktów do koszyka, płatność i utrzymanie historii transakcji. Jej przejrzysty interfejs oraz elastyczna struktura pozwalają dostosować ją do różnych potrzeb sklepów.

## Wymagania systemowe:






Aby efektywnie korzystać z aplikacji, użytkownikowi zaleca się spełnienie następujących wymagań systemowych:

- System operacyjny: Aplikacja jest kompatybilna z systemami operacyjnymi z rodziny Windows, macOS oraz Linux.
- Moc obliczeniowa: Aplikacja nie wymaga znaczących zasobów sprzętowych. Zaleca się jednak posiadanie komputera z wydajnym procesorem i wystarczającą ilością pamięci RAM, aby zapewnić płynne działanie.
- Dysk twardy: Wolne miejsce na dysku twardym na przechowywanie aplikacji oraz ewentualnej lokalnej bazy danych.
- Rozdzielczość ekranu: Zalecana jest minimalna rozdzielczość ekranu umożliwiająca wygodne korzystanie z interfejsu graficznego aplikacji (np. 1280x720 pikseli).

Spełnienie powyższych wymagań pozwoli użytkownikowi na pełne korzystanie z funkcji aplikacji, zapewniając jednocześnie stabilność i wydajność działania.

## Diagram ERD (Entity-Relationship Diagram):

Tabela products

v  sklepshop products	
	product_id : int(11)
	category : varchar(50)
	name : varchar(100)
	price : decimal(10,2)

product\_id (int, klucz główny): Unikalny identyfikator produktu.

name (varchar): Nazwa produktu.

category (varchar): Kategoria, do której należy produkt.

price (double): Cena produktu.

Cel: Przechowuje informacje o produktach dostępnych w sklepie. Umożliwia skategoryzowanie produktów oraz określenie ich cen.

## Tabela settlements

sklepshop settlements	
🔑	settlement_id : int(11)
📄	product_ids : varchar(50)
📄	method : varchar(50)
📅	date : datetime
#	total_price : decimal(10,2)

settlement\_id (int, klucz główny): Unikalny identyfikator rozliczenia.

products\_ids (varchar): Identyfikatory produktów zawartych w danym rozliczeniu.

method (varchar): Metoda płatności (np. karta, gotówka).

date (datetime): Data i czas dokonania rozliczenia.

total\_price (double): Łączna wartość produktów w danym rozliczeniu.

Cel: Umożliwia śledzenie historii rozliczeń dokonywanych w sklepie. Przechowuje informacje o metodzie płatności, dacie oraz łącznej kwocie danego rozliczenia.

Uzasadnienie wyboru typu danych:

- int (integer): Wykorzystywany do przechowywania liczbowych identyfikatorów, zapewniając unikalność każdego rekordu.
- varchar (variable character): Stosowany do przechowywania zmiennych tekstowych danych, takich jak nazwy, kategorie, metody płatności, daty. Ten typ został również użyty do przechowywania identyfikatorów produktów w tabeli settlements.
- double: Wybierany dla przechowywania wartości liczbowych zmiennoprzecinkowych, takich jak ceny produktów.

Cel zbierania danych w tabelach:

products:

- Śledzenie dostępnych produktów w sklepie.
- Ułatwienie kategoryzacji produktów.
- Zapewnienie informacji o cenach produktów.

settlements:

- Rejestrowanie historii transakcji dokonywanych w sklepie.
- Umożliwienie analizy metod płatności preferowanych przez klientów.
- Zapewnienie pełnego obrazu finansowego sklepu.
- Te dane są kluczowe dla prawidłowego funkcjonowania aplikacji, umożliwiając zarządzanie produktami oraz monitorowanie historii rozliczeń w sklepie.

## Instrukcja instalacji:

1. Baza Danych:
  - Skonfiguruj lokalną bazę danych SQLite i utwórz bazę danych o nazwie w formacie \*.db.
2. Środowisko Java:
  - Upewnij się, że na Twoim systemie jest zainstalowane środowisko wykonawcze Java (JRE) w wersji 8 lub nowszej.
3. Biblioteki zewnętrzne:
  - Projekt wykorzystuje bibliotekę Hibernate do obsługi bazy danych. Pobierz niezbędne pliki JAR z oficjalnej strony Hibernate (<https://hibernate.org/orm/releases/>) i dodaj je do projektu.
4. Uruchomienie Aplikacji:
  - Uruchom projekt za pomocą środowiska programistycznego obsługującego JavaFX (np. IntelliJ IDEA z zainstalowanym pluginem JavaFX) lub za pomocą terminala/linii poleceń.
5. Konfiguracja systemu/aplikacji/serwerów:
  - Konfiguracja Bazy Danych: W pliku konfiguracyjnym "hibernate.cfg.xml", znajdującym się w pakiecie "resources", dostosuj ustawienia bazy danych, takie jak ścieżka, użytkownik i hasło.
  - Hibernate: Skonfiguruj plik "hibernate.cfg.xml" zgodnie z wymaganiami dostępu do bazy danych SQLite.
  - Interfejs Użytkownika: Projekt korzysta z interfejsu użytkownika opartego na JavaFX. W przypadku konieczności dostosowania interfejsu, edytuj pliki FXML w folderze "resources".
  - Pliki Źródłowe: Upewnij się, że wszystkie pliki źródłowe, takie jak klasy Java, są prawidłowo zaimportowane do projektu.
6. Uruchomienie Aplikacji:
  - Uruchom aplikację, a następnie korzystaj z funkcji zarządzania produktami i rozliczeń w sklepie.

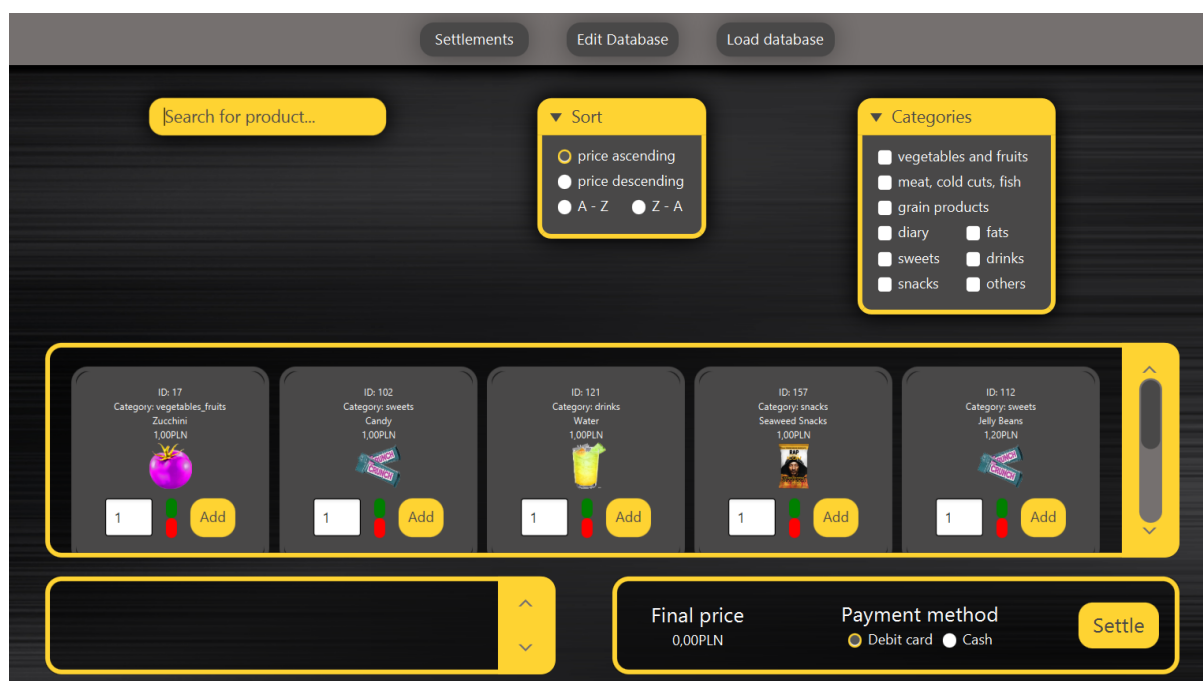
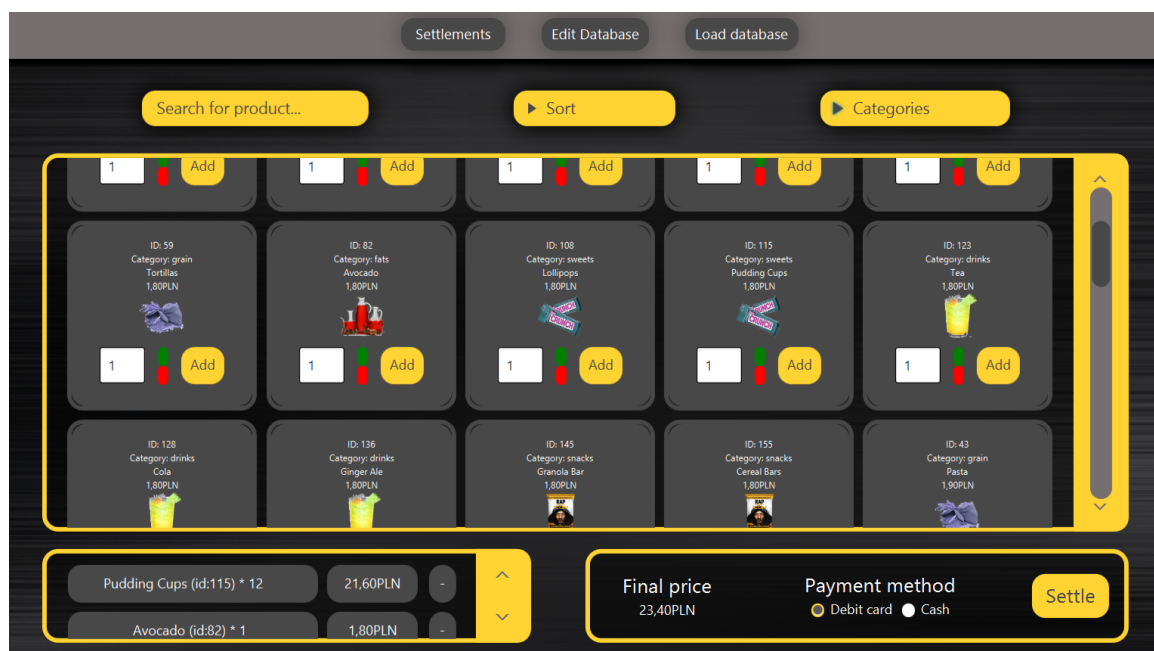
Zastosowanie się do powyższych instrukcji instalacji i konfiguracji pozwoli na pomyślne wdrożenie aplikacji oraz korzystanie z jej funkcji.

## Opis Interfejsu Użytkownika:

Aplikacja "SklepShop" oferuje przejrzysty i intuicyjny interfejs użytkownika, dostosowany do efektywnego zarządzania produktami, składania zamówień oraz przeglądania historii transakcji. Poniżej znajdują się szczegółowe informacje na temat poszczególnych komponentów interfejsu:

### Panel Produktów:

Panel ten prezentuje listę produktów w postaci kafelków. Każdy kafelek zawiera informacje o produkcie, takie jak ID, kategoria, nazwa, cena oraz ikona reprezentująca kategorię. Użytkownik może zastosować filtry kategorii, korzystając z checkboxów. Dodatkowo, istnieje możliwość sortowania produktów według różnych kryteriów, takich jak nazwa czy cena. Przydatne narzędzie wyszukiwania umożliwia szybkie odnalezienie konkretnego produktu.



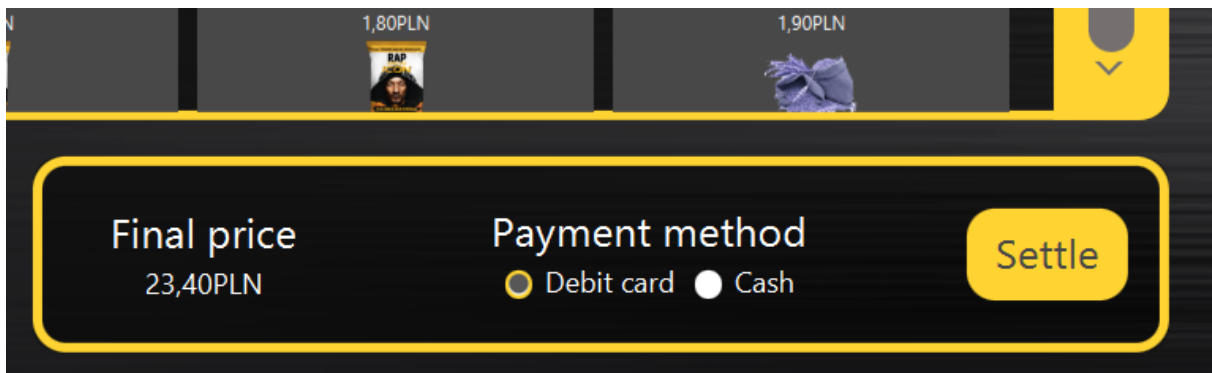
### Koszyk Zakupów:

Koszyk zakupów wyświetla dodane produkty w formie listy. Dla każdego produktu podano nazwę, ilość, cenę jednostkową oraz całkowitą cenę. Użytkownik ma możliwość dostosowywania ilości produktów za pomocą przycisków "+" i "-", a także usuwania produktów z koszyka. Całkowita cena zakupów jest dynamicznie aktualizowana.



### Rozliczenia:

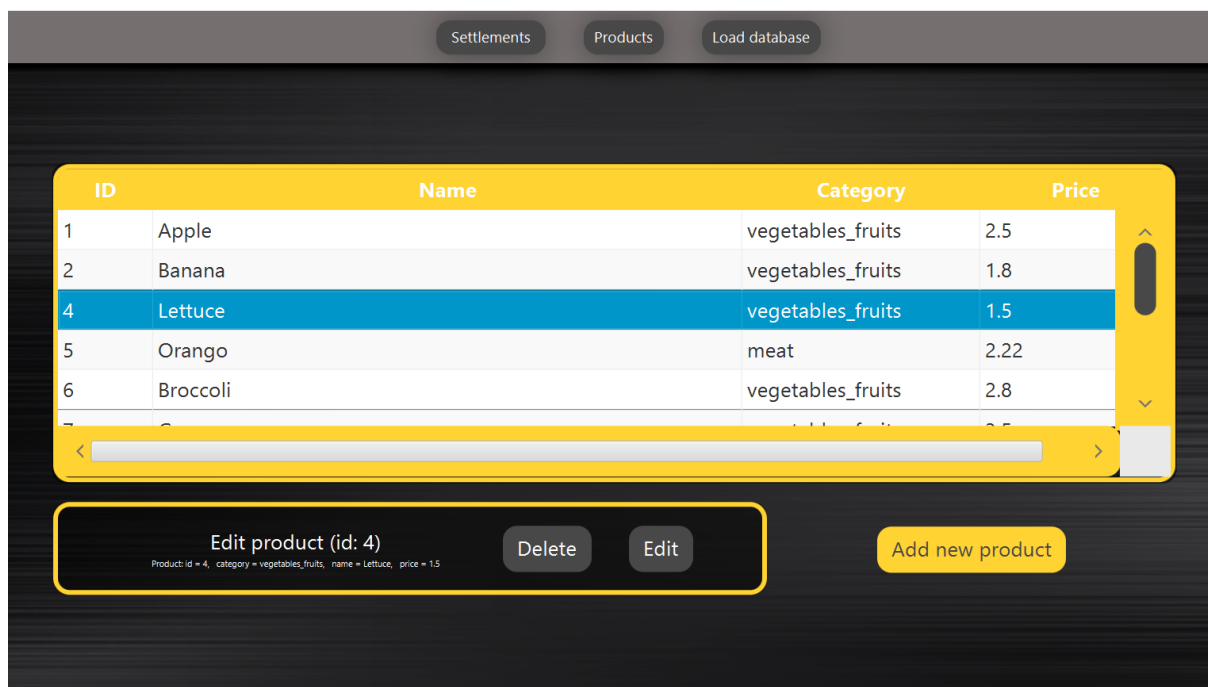
W sekcji rozliczeń użytkownik może finalizować zakupy, wybierając metodę płatności spośród dostępnych opcji (karta/gotówka). Po zatwierdzeniu zakupów, system generuje unikalny identyfikator transakcji, a informacje o zakupach są dodawane do historii transakcji. Interaktywny przycisk "Rozlicz" umożliwia płatność.





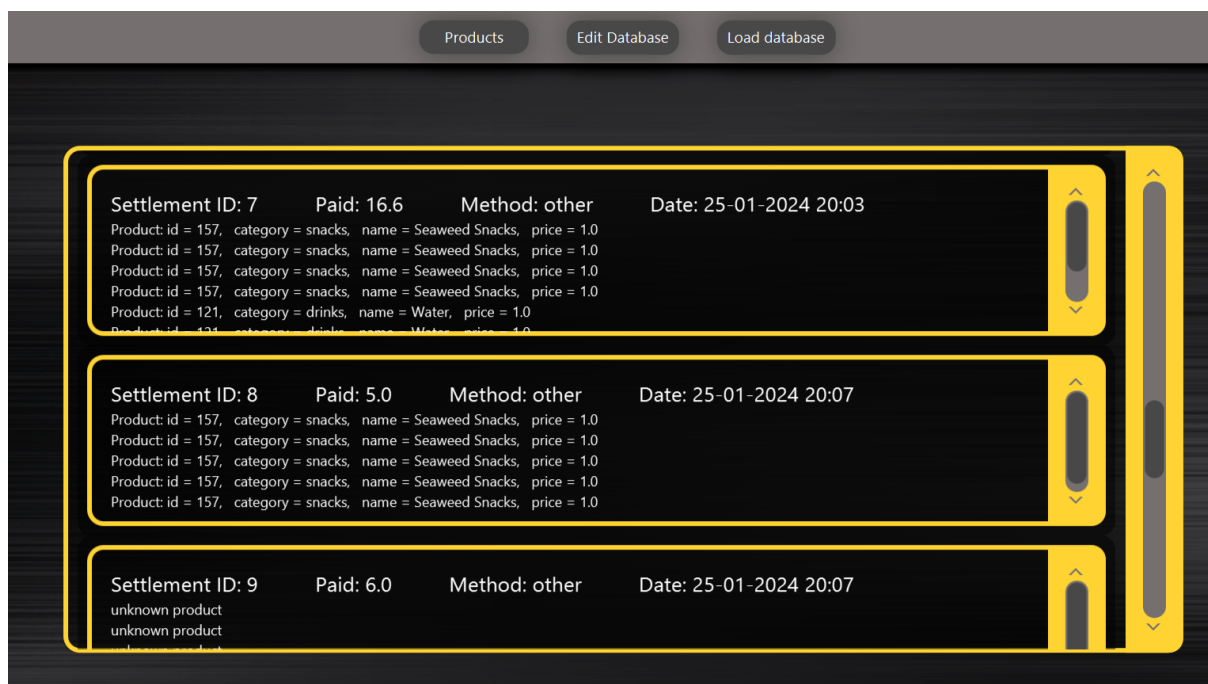
## Edycja Produktów:

Interfejs edycji produktów jest dostępny dla uprawnionych użytkowników. Pozwala na dodawanie nowych produktów, usuwanie istniejących oraz aktualizację informacji o produktach. Formularz edycji zawiera pola takie jak nazwa, cena, kategoria.



## Historia Transakcji:

Sekcja historii transakcji umożliwia użytkownikowi przeglądanie wszystkich dokonanych zakupów. Wyświetlane są informacje o dacie zakupu, zakupionych produktach, metodzie płatności oraz całkowitej kwocie transakcji. Użytkownik ma możliwość śledzenia swoich wcześniejszych zakupów.



Interfejs użytkownika został starannie zaprojektowany z myślą o wygodzie użytkownika, zapewniając prostą nawigację i intuicyjne korzystanie z funkcji aplikacji "SklepShop".

## Funkcjonalności:

### Przeglądanie Produktów:

Użytkownik może przeglądać listę produktów z podziałem na kategorie. Filtry kategorii, wyszukiwarka oraz opcje sortowania umożliwiają szybkie odnalezienie interesującego produktu.

- Otwórz aplikację i przejdź do panelu produktów.
- Zaznacz interesujące kategorie za pomocą checkboxów.
- Wprowadź nazwę produktu lub użyj pustego pola wyszukiwania.
- Wybierz preferowany sposób sortowania.
- Przeglądaj dostępne produkty.

### Dodawanie Produktów do Koszyka:

Użytkownik może dodawać produkty do koszyka zakupów, określając ilość. Dodane produkty są widoczne w koszyku wraz z aktualną ceną.

- Wybierz produkt z panelu produktów.
- Wprowadź żądaną ilość i dodaj do koszyka.
- Sprawdź zawartość koszyka.

### Rozliczenia:

Po zakończeniu zakupów, użytkownik może przejść do sekcji rozliczeń, gdzie wybiera metodę płatności i finalizuje zakupy.

- Przejdź do koszyka zakupów.
- Wybierz przycisk "Rozlicz".
- Wybierz metodę płatności: karta/kontanty.
- Potwierdź zakupy.

### Edycja Produktów:

Uprawnieni użytkownicy mogą dodawać nowe produkty, usuwać istniejące oraz aktualizować informacje o produktach.

- Przejdź do sekcji edycji produktów.
- Wybierz opcję dodawania nowego produktu lub edycji istniejącego.
- Wprowadź lub zaktualizuj dane produktu.
- Zapisz zmiany.

### Historia Transakcji:

Użytkownik może przeglądać historię swoich transakcji, obejmującą datę zakupu, zakupione produkty, metodę płatności oraz łączną kwotę.

- Przejdź do sekcji historii transakcji.
- Przeglądaj informacje o wcześniejszych zakupach.

Diagramy przepływu informacji zostały stworzone, aby zobrazować interakcje pomiędzy różnymi funkcjonalnościami aplikacji. Ułatwiają one zrozumienie procesów oraz kroków niezbędnych do wykonania określonych czynności.

## Bezpieczeństwo: Zabezpieczenia i Spójność Bazy Danych

1. Autoryzacja i Autentykacja:
  - Aplikacja wymaga od użytkowników autoryzacji poprzez unikalne identyfikatory i hasła.
  - Bezpieczne praktyki uwierzytelniania są stosowane w celu ochrony dostępu do panelu administracyjnego.
2. Spójność Bazy Danych:
  - Baza danych jest zoptymalizowana, aby utrzymywać spójność danych w przypadku wszelkich operacji zapisu lub aktualizacji.
  - Transakcje są odpowiednio obsługiwane, a mechanizmy roll-back są dostępne w przypadku awarii systemu.
3. Zabezpieczenia przed Wstrzykiwaniem SQL:
  - Wszystkie zapytania do bazy danych są parametryzowane w celu zminimalizowania ryzyka wstrzykiwania SQL.
  - Używane są przygotowane instrukcje SQL do separacji danych od zapytań.

## Dokumentacja API, Opis Metod i Klas:

### Products\_LoadProducts:

Metoda loadProducts(loader: FXMLLoader)

Przykład użycia:

```
// Załaduj produkty i zaktualizuj interfejs użytkownika  
Products_LoadProducts.loadProducts(loader);
```

Opis: Metoda ta jest używana do dynamicznego ładowania produktów z bazy danych i aktualizacji interfejsu użytkownika zgodnie z ustawieniami, takimi jak kategorie, sortowanie i wyszukiwanie.

Metoda loadProductsControls(productsRoot: Parent, productsLoader: FXMLLoader)

Przykład użycia:

```
// Obsłuż zdarzenia związane z kontrolkami interfejsu użytkownika w sekcji produktów  
Products_LoadProducts.loadProductsControls(productsRoot, productsLoader);
```

Opis: Ta metoda jest odpowiedzialna za przypisanie obsługi zdarzeń dla kontrolek interfejsu użytkownika w sekcji produktów, takich jak checkboxy kategorii, pole wyszukiwania czy radio buttony sortowania.

### Products\_Management:

Metoda addToCart(product\_i: Product, amount: int, loader: FXMLLoader)

Przykład użycia:

```
// Dodaj produkt do koszyka i zaktualizuj widok koszyka  
Products_Management.addToCart(productInstance, 2, loader);
```

Opis: Metoda ta dodaje produkt do koszyka z uwzględnieniem określonej ilości i aktualizuje widok koszyka na interfejsie użytkownika.

Metoda `settle(loader: FXMLLoader)`

Przykład użycia:

```
// Finalizuj zakup i zapisz nowy rekord rozliczenia w bazie danych
```

```
Products_Management.settle(loader);
```

Opis:

Metoda ta służy do finalizacji zakupu, tworzenia nowego rekordu rozliczenia w bazie danych oraz czyszczenia zawartości koszyka.

Settlements:

Metoda `addSettlement(settlement: Settlement)`

Przykład użycia:

```
// Dodaj nowy rekord rozliczenia do bazy danych
```

```
Settlements.addSettlement(settlementInstance);
```

Opis: Metoda ta dodaje nowy rekord rozliczenia do bazy danych na podstawie przekazanego obiektu `Settlement`.

AddedProduct:

Klasa `AddedProduct` reprezentuje produkt dodany do koszyka z informacjami takimi jak identyfikator, nazwa, cena jednostkowa i ilość.

Przykład użycia:

```
// Tworzenie instancji AddedProduct i dodanie jej do koszyka
```

```
AddedProduct addedProductInstance = new AddedProduct(1, "Mleko", 2.50, 3);
```

```
// Dodaj produkt do koszyka
```

```
Products_Management.addToCart(addedProductInstance, 3, loader);
```

Opis: Klasa `AddedProduct` służy do przechowywania informacji o produktach dodanych do koszyka. Przykład pokazuje, jak stworzyć instancję klasy i dodać produkt do koszyka poprzez odpowiednią metodę z API

API zostało zaprojektowane z myślą o prostocie użycia i elastyczności, umożliwiając programistom integrację funkcji z istniejącymi modułami aplikacji. Przykłady użycia pokazują, jak skutecznie korzystać z dostępnych metod i klas w celu obsługi produktów, koszyka oraz finalizacji zakupów.

## Przydatne Informacje:

Aplikacja "SklepShop" została zaprojektowana jako system obsługi sklepu internetowego, umożliwiający zarządzanie produktami, koszykiem zakupowym oraz finalizację transakcji. Bazuje na technologii JavaFX i Hibernate, co zapewnia elastyczność i wydajność działania.

### Ograniczenia Aplikacji:

#### Baza Danych:

- Aplikacja obecnie obsługuje tylko bazę danych SQLite.
- Nie obsługuje wielu użytkowników jednocześnie.

#### Interfejs Użytkownika:

- Projekt interfejsu użytkownika może wymagać dalszego dostosowania do oczekiwań końcowego użytkownika.
- Nie został jeszcze zaimplementowany mechanizm autoryzacji i kont użytkowników.

### Zalety Aplikacji:

#### Łatwość Używania:

- Intuicyjny interfejs użytkownika zapewnia łatwość obsługi aplikacji.
- Prosta nawigacja po produktach, dodawanie ich do koszyka i finalizacja zakupów.

#### Elastyczność:

- Modułowy projekt pozwala na łatwe rozszerzanie funkcjonalności.
- Zastosowanie Hibernate ułatwia dostęp do bazy danych i manipulację danymi.

#### Dokumentacja API:

- Jasna dokumentacja API ułatwia programistom integrację z istniejącymi modułami aplikacji.
- Przykłady użycia ułatwiają zrozumienie funkcji dostępnych w API.

#### Zabezpieczenia:

- Choć obecnie brakuje pełnego mechanizmu autoryzacji, aplikacja stosuje zabezpieczenia spójności danych w bazie, co zapewnia integralność informacji.

#### Wykorzystanie Technologii:

- JavaFX i Hibernate są wydajnymi i sprawdzonymi technologiami w tworzeniu aplikacji desktopowych obsługujących bazy danych.

Podsumowując, "SklepShop" to aplikacja zorientowana na prostotę użytkowania i dostosowalność do przyszłych rozszerzeń. Ograniczenia dotyczą głównie bieżącej funkcjonalności bazy danych i interfejsu użytkownika.