Inżynieria oprogramowania
Laboratorium

Prowadzący: prof. dr hab. inż. Jan Magott

Sprawozdanie z zajęć laboratoryjnych

Skład grupy:
Kamil Dywan
Jakub Płuciennik

Termin zajęć:
czwartek 11:15-13:00

Drugie zajęcia laboratoryjne

1. Zadanie laboratoryjne

Celem laboratorium było wykonanie opisu biznesowego „świata rzeczywistego" projektowanego oprogramowania, oraz specyfikacja wymagań funkcjonalnych i niefunkcjonalnych aplikacji na podstawie stworzonego wcześniej opisu. Dodatkowo wykonany tekst powinno umieścić się w projekcie zawierającym model ULM, oraz powinno stworzyć się diagramy wymagań odpowiadające wymaganiom funkcjonalnym i niefunkcjonalnym.

2. Opis biznesowy „świata rzeczywistego". Wypożyczalnia sprzętu turystycznego

2.1 Opis zasobów ludzkich

Pracownik wypożyczalni może dodawać do, oraz usuwać z katalogu przedmiotów nowe przedmioty lub ich egzemplarze. Przedmioty reprezentowane są przez następujące dane: kategoria, nazwa, marka, model, dane techniczne (np. waga lub litraż) oraz cena za dzień wypożyczenia. Każdy przedmiot opisywany jest również unikatowym kodem pozwalającym go identyfikować. Pracownik może modyfikować, dodawać lub usuwać wymienione wyżej dane. Dodatkowo może on dodawać i usuwać promocje oraz dowolnie przeglądać katalogi przedmiotów. Klient może zakładać konto pod unikalnym numerem identyfikacyjnym oraz zakładać rachunki wypożyczeń. Jeden klient powinien mieć możliwość założenia maksymalnie dwóch osobnych rachunków. Klient może przeglądać katalogi przedmiotów, wyszukując je po dowolnej danej oprócz unikatowego numeru identyfikującego (ten przeznaczony jest tylko dla pracowników).

2.2 Przepisy i strategia firmy

Pracownik ponosi odpowiedzialność za poprawność danych - odpowiada materialnie za niezgodność danych ze stanem wypożyczalni oraz odpowiada za stan wypożyczanego przedmiotu. System przeglądania przedmiotów powinien być sprawny i przyjazny zarówno dla klienta jak i pracownika.

2.3 Dane techniczne

Klient może przeglądać dane wypożyczalni oraz zakładać rachunki za pomocą dedykowanego programu. Zakłada się, że klientów przeglądających dane wypożyczalni może być ponad 500 oraz wypożyczalnia zawierać może dziesiątki kategorii i tysiące przedmiotów. Mogą oni przeglądać dane poprzez stronę **int**ernetową. Do zaimplementowania systemu przeglądania danych wypożyczalni zostanie zastosowana technologia Java. Dodatkowo wypożyczalnia posiadać będzie dedykowaną bazę danych. Wypożyczalnia składa się z kilku ośrodków we Wrocławiu i Jeleniej Górze.

3. Wymagania stawiane tworzonej aplikacji

Na podstawie opisu świata rzeczywistego sformułowano wymagania funkcjonalne i niefunkcjonalne aplikacji.

3.1 Wymagania funkcjonalne

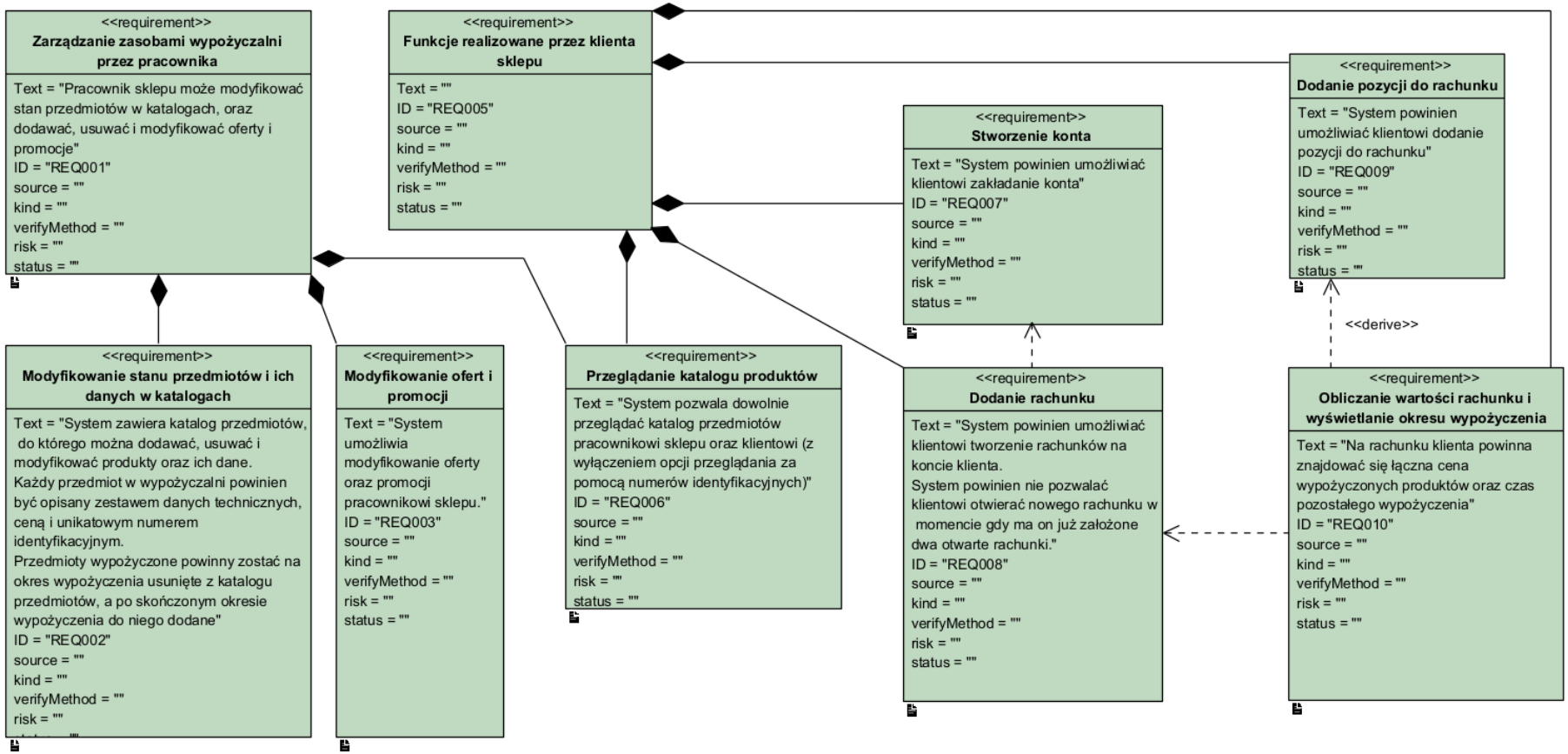
- System zawiera katalog przedmiotów, do którego można dodawać, usuwać i modyfikować produkty oraz ich dane
- Wypożycza się przedmioty tylko użytkownikom posiadającym założone konto i posiadającym nie więcej niż 2 otwarte rachunki
- Każdy przedmiot w wypożyczalni powinien być opisany zestawem danych technicznych, ceną i unikatowym numerem identyfikacyjnym
- System powinien umożliwiać klientowi zakładanie konta oraz zakładanie rachunków na koncie klienta
- Na rachunku klienta powinna znajdować się łączna cena wypożyczonych produktów oraz czas pozostałego wypożyczenia
- Przedmioty wypożyczone powinny zostać na okres wypożyczenia usunięte z katalogu przedmiotów, a po skończonym okresie wypożyczenia do niego dodane
- System umożliwia modyfikowanie oferty oraz promocji pracownikowi sklepu
- System pozwala dowolnie przeglądać katalog przedmiotów pracownikowi sklepu oraz klientowi (z wyłączeniem opcji przeglądania za pomocą numerów identyfikacyjnych)

3.2 Wymaganie niefunkcjonalne

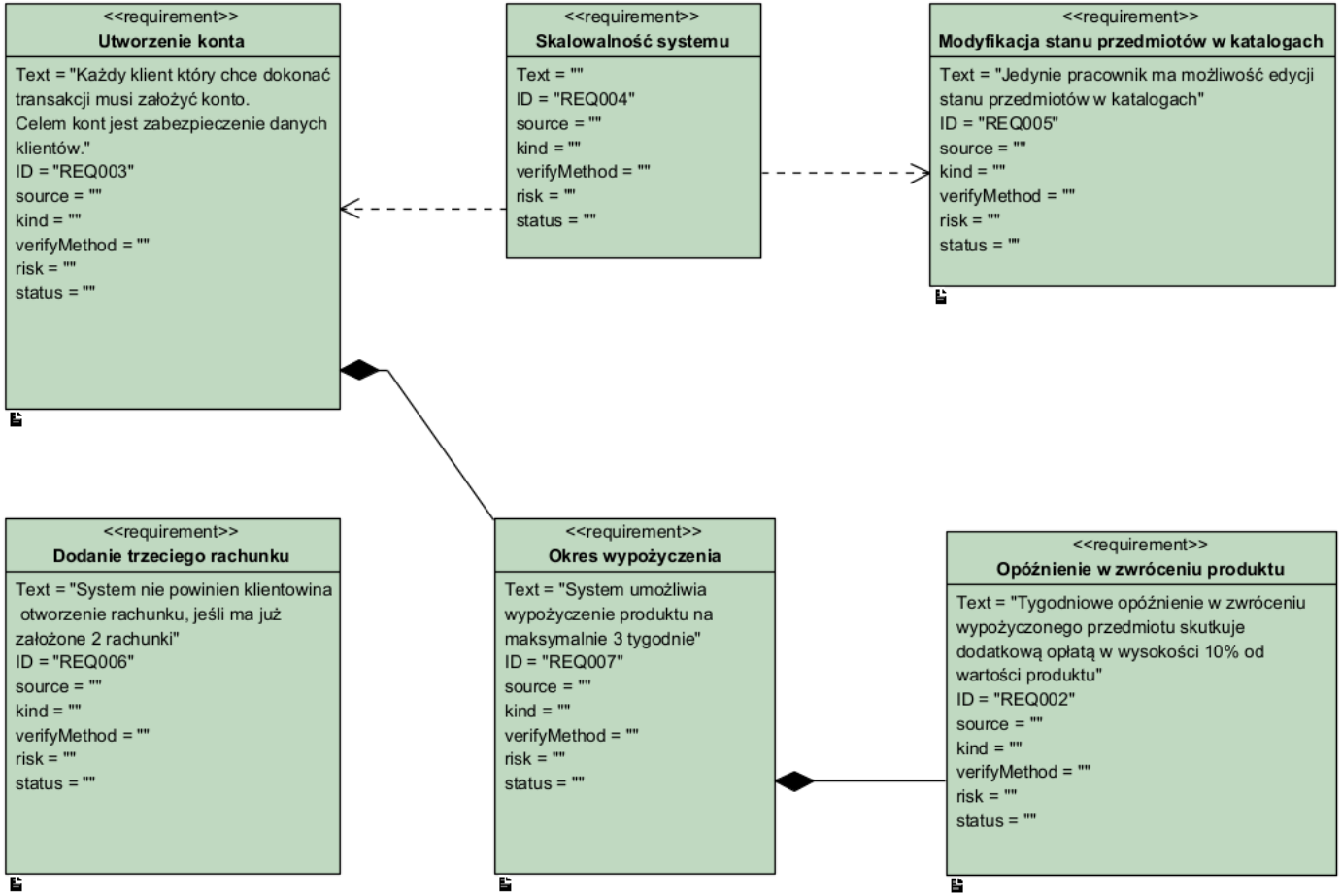
- Modyfikowanie stanu przedmiotów w katalogach powinno być możliwe tylko przez pracowników
- System powinien zabezpieczać dane klientów, a ich podgląd powinni mieć jedynie uprawnieni do tego pracownicy
- System powinien nie pozwalać klientowi otwierać nowego rachunku w momencie gdy ma on już założone dwa otwarte rachunki
- Przedmioty wypożycza się na określone okresy nie dłuższe niż 3 tygodnie.
- Tydzień opóźnienia w zwróceniu wypożyczonego przedmiotu nalicza dodatkową opłatę w wysokości 10% kwoty wypożyczenia tego przedmiotu.
- System powinien być projektowany tak by w każdej chwili mógł zostać rozwinięty

4. Diagramy wymagań

Na podstawie opisu wylistowanych wymagań funkcjonalnych i niefunkcjonalnych stworzono diagramy wymagań.



Rysunek 1: diagram wymagań funkcjonalnych



Rysunek 2: diagram wymagań niefunkcjonalncyh

4. Wnioski

Podczas zajęć udało nam się zrealizować wszystkie wymagane zadania. Na etapie tworzenia diagramów napotkaliśmy na kilka problemów związanych z relacjami pomiędzy diagramami, ale z udało nam się je zrealizować. Laboratorium pomogło nam w opanowaniu wstępnego etapu projektowania aplikacji jakim jest definiowanie wymagań.

Trzecie oraz czwarte zajęcia laboratoryjne

1. Zadanie laboratoryjne

Celem laboratorium była specyfikacja wymagań funkcjonalnych zdefiniowanych na poprzednim laboratorium za pomocą diagramów przypadków użycia oraz sformułowanie ich opisów w języku naturalnym zawierających cele, warunki i przebiegi przypadków.

2. Opisy przypadków użycia

Na podstawie wymagań stworzonych na poprzednim laboratorium sformułowaliśmy opisy przypadków użycia w języku naturalnym.

PU – przypadek użycia

- **PU Zakładanie konta**

  Opis:
  CEL: Zapewnienie bezpieczeństwa
  WS (warunki wstępne): inicjalizacja przez kliknięcie przycisku „Rejestracja”
  WK (warunki końcowe): założenie konta jeśli podane dane są prawidłowe
  Przebieg:
  1. Podanie imienia, nazwiska, wieku, loginu, hasła, adresu e-mail
  2. Kliknięcie Rejestruj aby potwierdzić dane
  3. Jeśli wprowadzone login i e-mail nie znajdują się już bazie danych, zakładane jest nowe konto, a w przeciwnym razie zwracany jest komunikat o błędzie

- **PU Logowanie**

  Opis
  CEL: Umożliwienie dokonania zakupu produktów (zapewnienie bezpieczeństwa)
  WS (warunki wstępne): inicjalizacja przez kliknięcie przycisku „Logowanie”
  WK (warunki końcowe): zalogowanie klienta do systemu jeśli wprowadzone dane są prawidłowe
  Przebieg:
  1. Podanie loginu, hasła
  2. Jeśli podane dane są prawidłowe klient zostanie zalogowany do systemu, a w przeciwnym razie zwracany jest komunikat o błędzie

- **PU Zarządzanie koszykiem**

  Opis
  CEL: Zarządzanie koszykiem
  WS (warunki wstępne): inicjalizacja przez kliknięcie przycisku modyfikującego koszyk
  WK (warunki końcowe): odpowiednia modyfikacja koszyka
  Przebieg:
  1. Wywołanie PU Przeglądanie oferty
  2. Wywołanie jednego z PU Dodanie nowego produktu do koszyka,
  PU Usunięcie produktu z koszyka,
  PU Modyfikacja liczby danego produktu w koszyku
  3. Zapisanie zmian w bazie danych
- **PU Przeglądanie oferty**

  Opis
  CEL: Przeglądanie dostępnych zasobów
  WS (warunki wstępne): inicjalizacja przez kliknięcie przycisku „Szukaj", przez PU Zarządzanie koszykiem lub PU Zarządzanie katalogiem
  WK (warunki końcowe): wyświetlenie dostępnych produktów z uwzględnieniem wprowadzonych filtrów
  Przebieg:
  1. Szukanie odbywa się przez podanie nazwy lub ceny
  2. Wprowadzenie odpowiednich filtrów (opcjonalna opcja dla inicjalizacji przez kliknięcie przycisku „Szukaj")
  3. Zwrócenie dostępnych produktów z uwzględnieniem wprowadzonych filtrów

- **PU Dodanie produktu z koszyka**

  Opis
  CEL: Dodanie produktu do koszyka
  WS (warunki wstępne): inicjalizacja przez PU Zarządzanie koszykiem
  WK (warunki końcowe): dodanie produktu do koszyka (pamięć podręczna) (jeśli produkt ten znajduje się już w koszyku, to zostanie zwiększona liczba tego produktu w koszyku.
  Przebieg:
  1. Kliknięcie przycisku „Dodaj" przy danym produkcie
  2. Dodanie produktu do koszyka
- **PU Usunięcie produktu z koszyka**

  Opis
  CEL: Usunięcie produktu z koszyka
  WS (warunki wstępne): inicjalizacja przez PU Zarządzanie koszykiem
  WK (warunki końcowe): usunięcie wszystkich danych produktów z koszyka (pamięć podręczna)
  Przebieg:
  1. Kliknięcie przycisku „Usuń" przy danym produkcie
  2. Usunięcie wszystkich produktów o danym identyfikatorze z koszyka

- **PU Modyfikacja liczby danego produktu w koszyku**

  Opis
  WS (warunki wstępne): inicjalizacja przez PU Zarządzanie koszykiem
  WK (warunki końcowe): modyfikacja liczby produktów w koszyku (pamięć podręczna), jeśli podana liczba jest nie większa, niż liczba dostępnych produktów w magazynie wypożyczalni
  Przebieg:
  1. Wprowadzenie innej wartości niż obecna w polu „Liczba produktu”
  2. Następuje modyfikacja liczby danego produktu w koszyku, a jeśli podana liczba jest większa, niż liczba dostępnych produktów w magazynie, wtedy zwracana jest informacja o błędzie

- **PU Zakładanie rachunku**
  Opis
  Cel: Założenie nowego rachunku
  WS (warunki wstępne): inicjalizacja poprzez kliknięcie przynisku „załóż nowy rachunek”. Posiadanie maksymalnie jednego otwartego rachunku.
  WK (warunki końcowe): Utworzenie nowego rachunku na koncie klienta, jeżeli spełnione są warunki.
  Przebieg:
  1. Sprawdzenie czy klient nie posiada już dwóch otwartych rachunków. Jeżeli taka sytuacja ma miejsce następuje wywołanie błędu utworzenia nowego rachunku z odpowiednią informacją.
  2. Jeżeli system zezwolił na utworzenie rachunku, rachunek zostaje wstawiony pod unikalnym numerem.

- **PU Obliczanie wartości rachunku**
  Opis
  Cel: Zwrócenie łącznej wartości rachunku klienta
  WS (warunki wstępne): istnienie otwartego rachunku pod danym numerem. Może być wywołany z PU Zakładanie rachunku.
  WK (warunki końcowe): zwrócenie wartości całego rachunku
  Przebieg:
  1. Zsumowanie wartości każdego przedmiotu dodanego do rachunku.
  2. Zwrócenie i wyświetlenie łącznej wartości rachunku

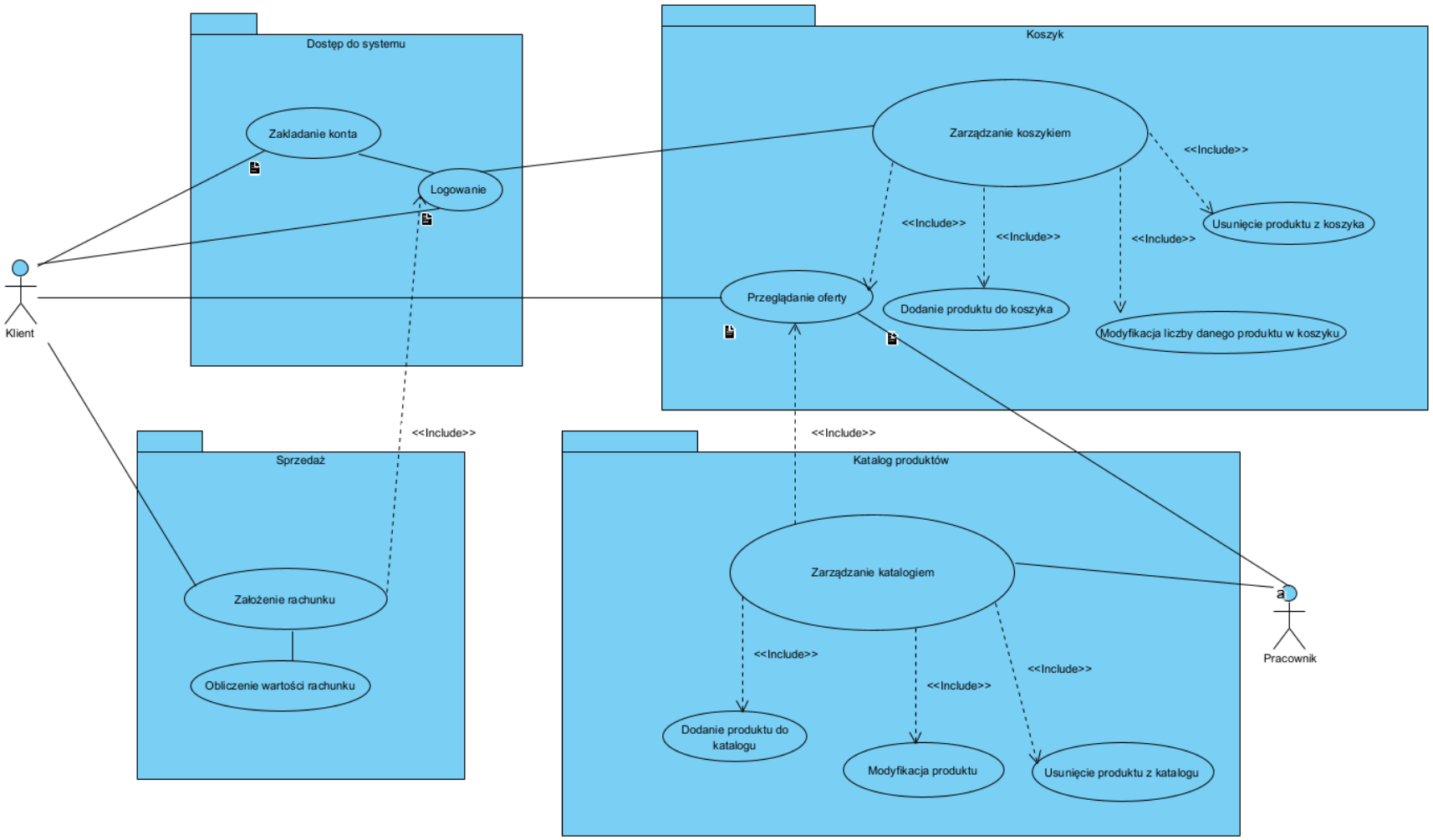
- **PU Zarządzanie katalogiem**
  Opis
  Cel: Modyfikacja stanu katalogu
  WS (warunki wstępne): inicjalizacja przez kliknięcie przycisku modyfikującego katalog
  WK (warunki końcowe): zmodyfikowanie katalogu przedmiotów zgodnie z założeniem
  Przebieg:
  1. Wywołanie PU Przeglądanie oferty
  2. Wywołanie jednego z PU Dodanie produktu do katalogu, PU Usunięcie produktu z katalogu, PU Modyfikacja produktu
  3. Zapisanie zmian w bazie danych

- **PU Dodanie produktu do katalogu**
  Opis
  Cel: Dodanie nowego przedmiotu do katalogu przedmiotów
  WS (warunki wstępne): inicjalizacja przyciskiem „dodaj nowy przedmiot”.
  WK (warunki końcowe): dodanie nowego przedmiotu o unikalnym numerze identyfikacyjnym do katalogu przedmiotów (pamięć podręczna).
  Przebieg:
    1. Podanie parametrów przedmiotu, wraz z ceną i kategorią.
    2. W przypadku gdy dana kategoria dodawanego przedmiotu nie istnieje należy automatycznie utworzyć w bazie nową kategorię.
    3. Podanie liczebności dodawanego przedmiotu.
    4. Wygenerowanie unikatowego numeru identyfikacyjnego dla każdej instancji nowego przedmiotu.
    5. Wstawienie przedmiotu do katalogu.

- **PU Usunięcie produktu z katalogu**
  Opis
  Cel: Usunięcie przedmiotu z katalogu przedmiotów.
  WS (warunki wstępne): inicjalizacja przyciskiem „usuń przedmiot”. Istnienie przedmiotu w katalogu.
  WK (warunki końcowe): usunięcie przedmiotu z katalogu (pamięć podręczna).
  Przebieg:
    1. Usunięcie przedmiotu jeśli przedmiot ten znajduje się w bazie danych
    2. Zapisanie zmian

- **PO Modyfikacja produktu**
  Opis
  Cel: Modyfikacja danych przedmiotu
  WS (warunki wstępne): inicjalizacja przyciskiem „edytuj”
  WK (warunki końcowe): odpowiednia modyfikacja przedmiotu (pamięć podręczna).
  Przebieg:
    1. Modyfikacja danych produktu danych
    2. Zapisanie zmian

3. Diagram przypadków użycia



Rysunek 3: Diagram przypadków użycia

4. Wnioski

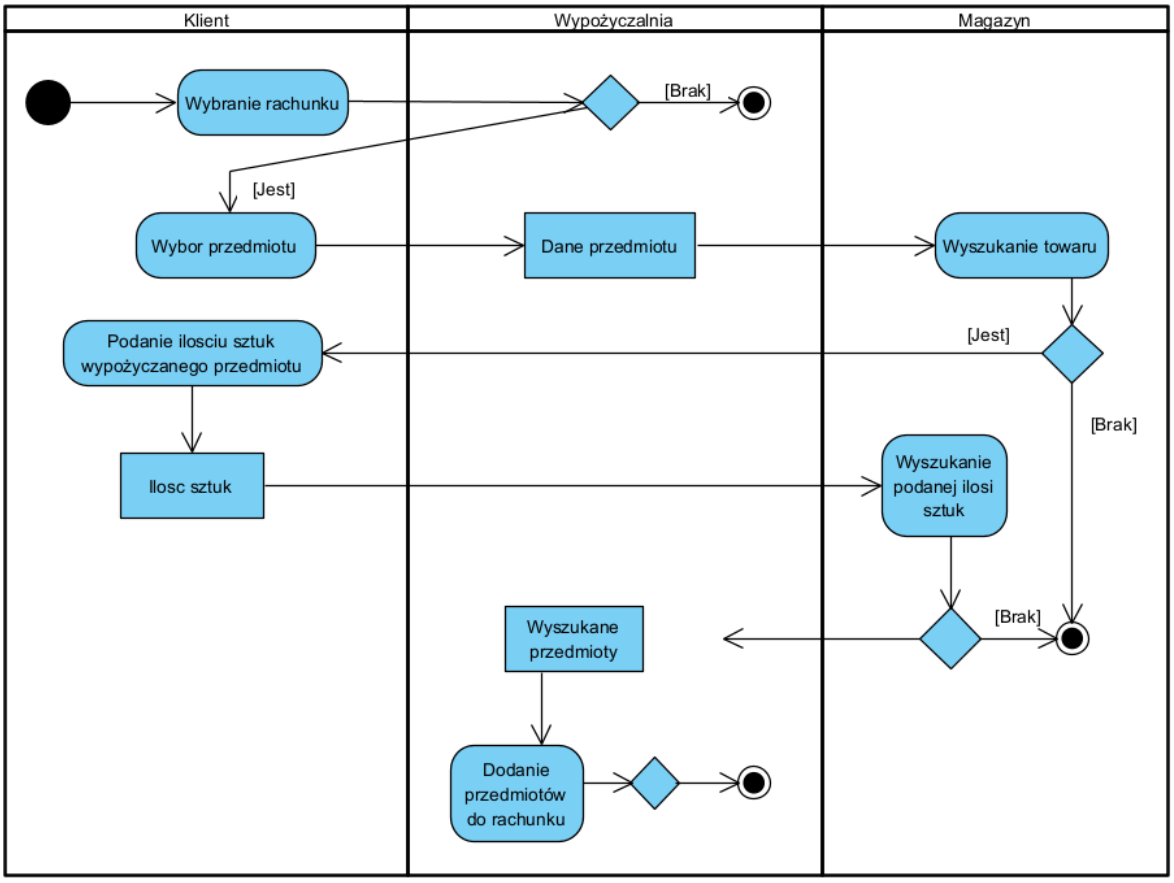
Podczas zajęć udało nam się zrealizować wszystkie wymagane zadania. Na etapie tworzenia diagramu napotkaliśmy problem polegającym na poprawnym użyciu relacji pomiędzy przypadkami. Problem po konsultacji udało się rozwiązać. Laboratorium pomogło nam rozwinąć swoją wiedze na temat modelowania UML oraz rozwinąć projekt rozpoczęty na wcześniejszych laboratoriach.

Piąte oraz szóste zajęcia laboratoryjne

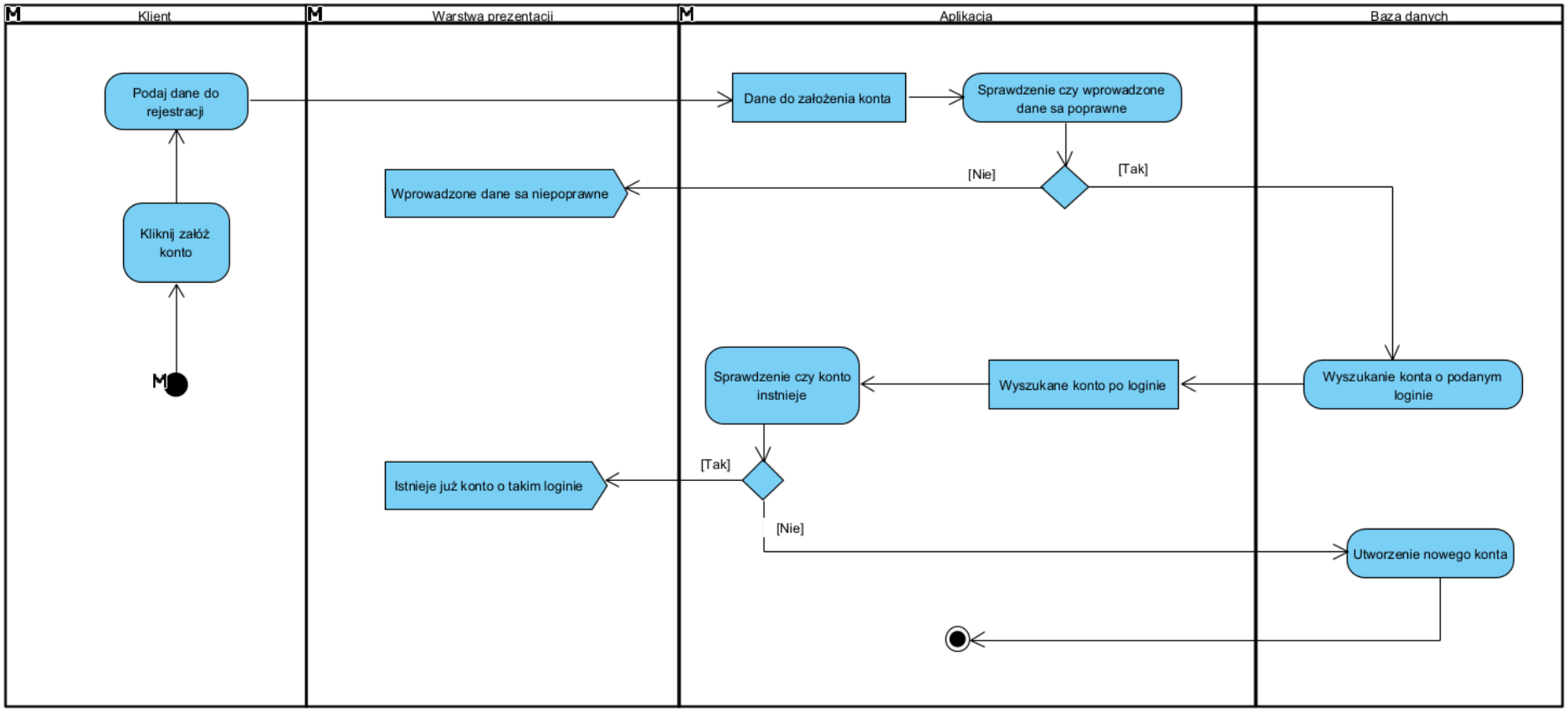
1. Zadanie laboratoryjne:

Celem laboratorium była budowa diagramu czynności reprezentującego model biznesowy „świata rzeczywistego”, oraz budowa diagramów czynności reprezentujących scenariusze wybranych przypadków użycia.

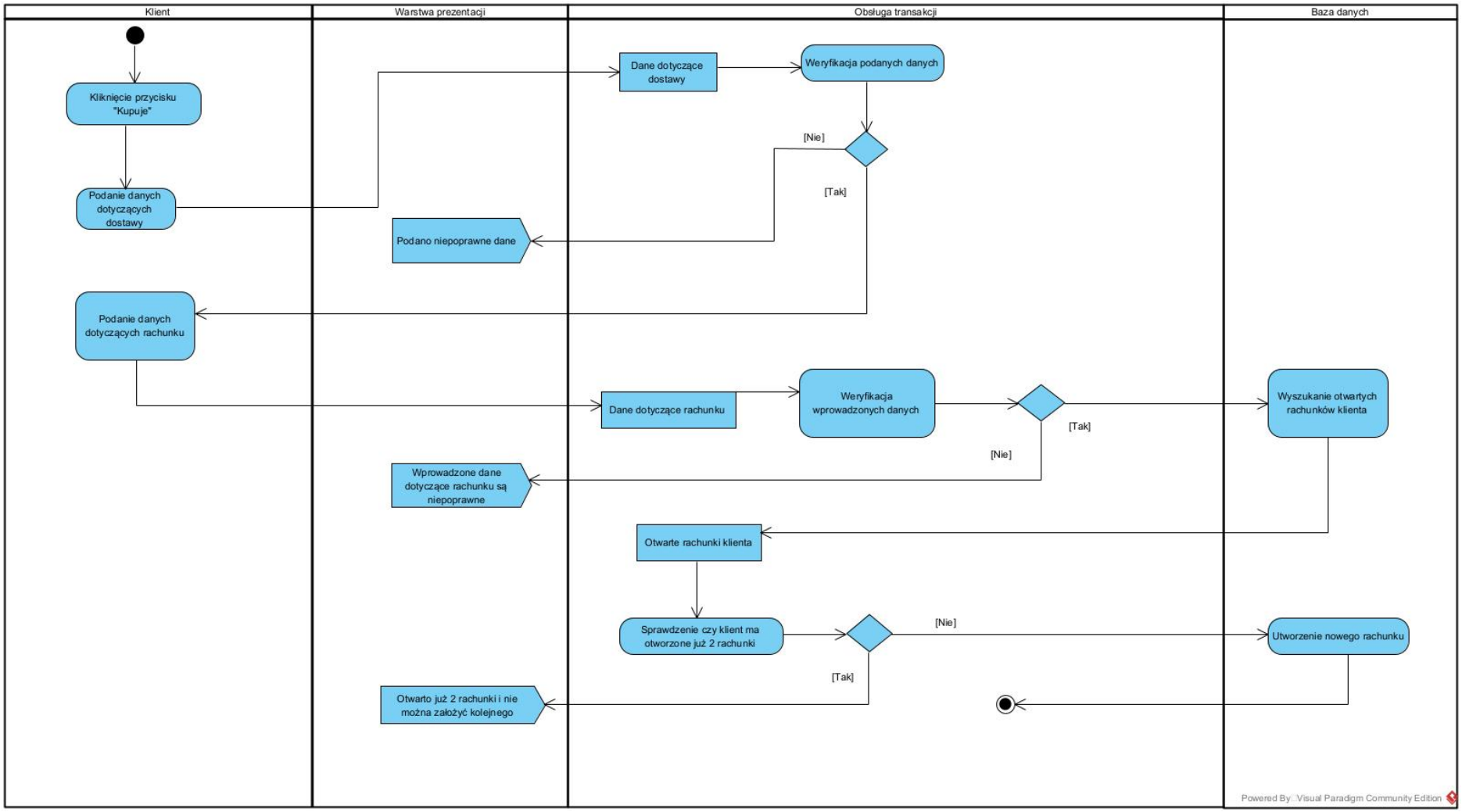
2. Diagramy czynności



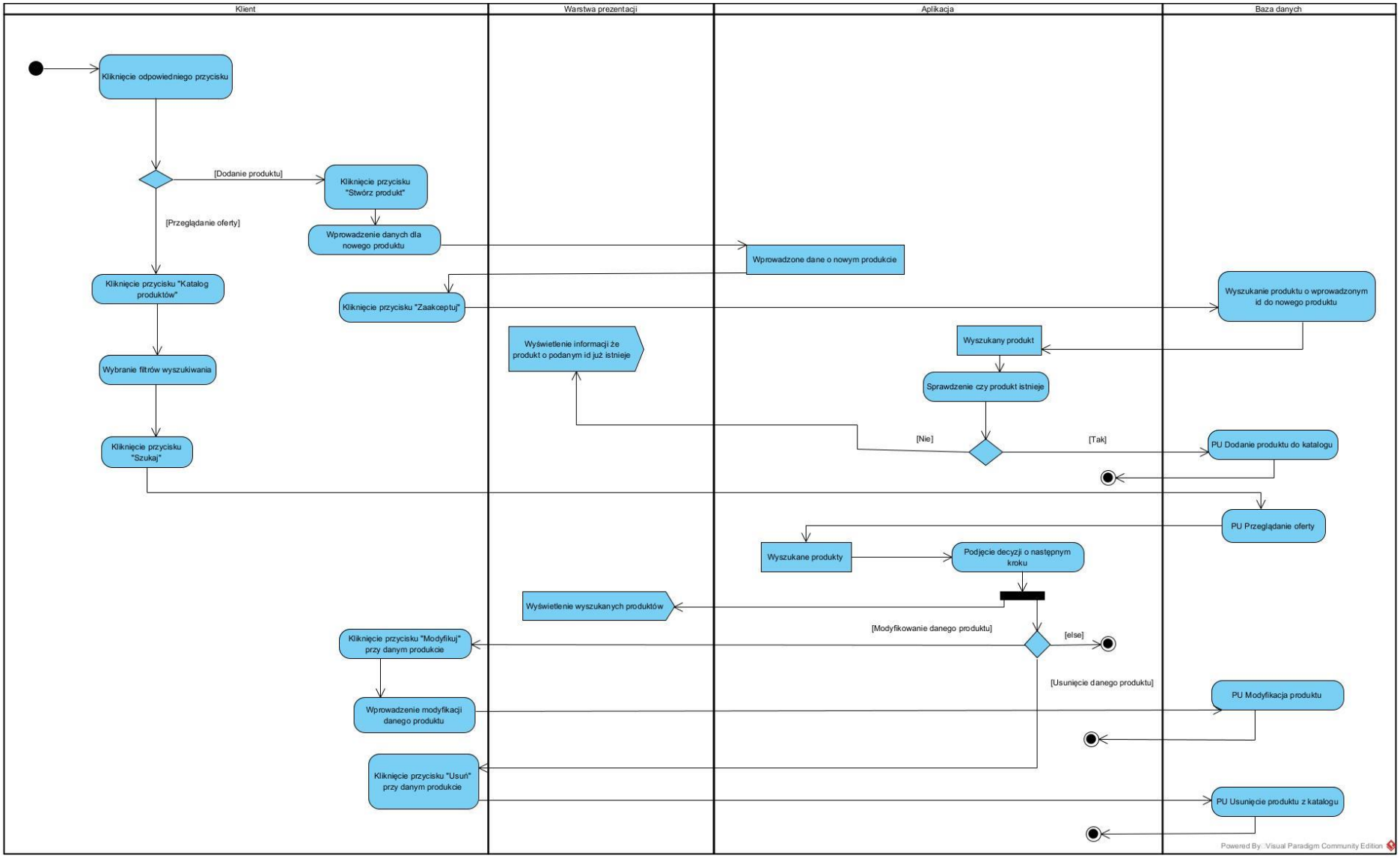
Rysunek 4: Diagram czynności procesu biznesowego ze "świata rzeczywistego" dotyczącego dodawania nowych przedmiotów do rachunku wypożyczeń



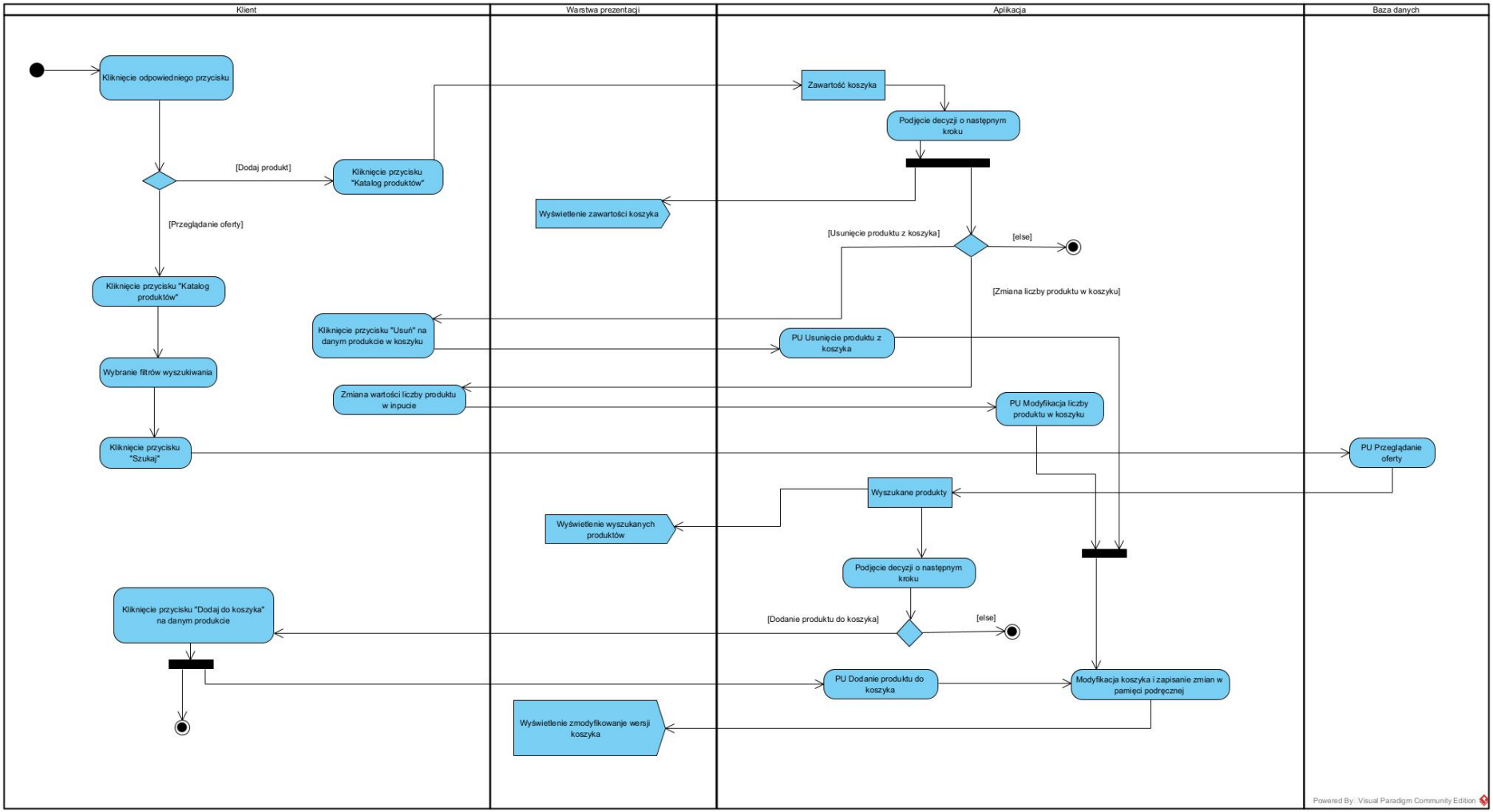
Rysunek 5: Diagram czynności reprezentujący scenariusz PU Zakładanie Konta



Rysunek 6: Diagram czynności reprezentujący scenariusz PU Zakładanie Rachunku



Rysunek 7: Diagram czynności reprezentujący scenariusz PU Zarządzanie Katalogiem



Rysunek 8: Diagram czynności reprezentujący scenariusz PU Zarządzanie Koszykiem

4. Wnioski

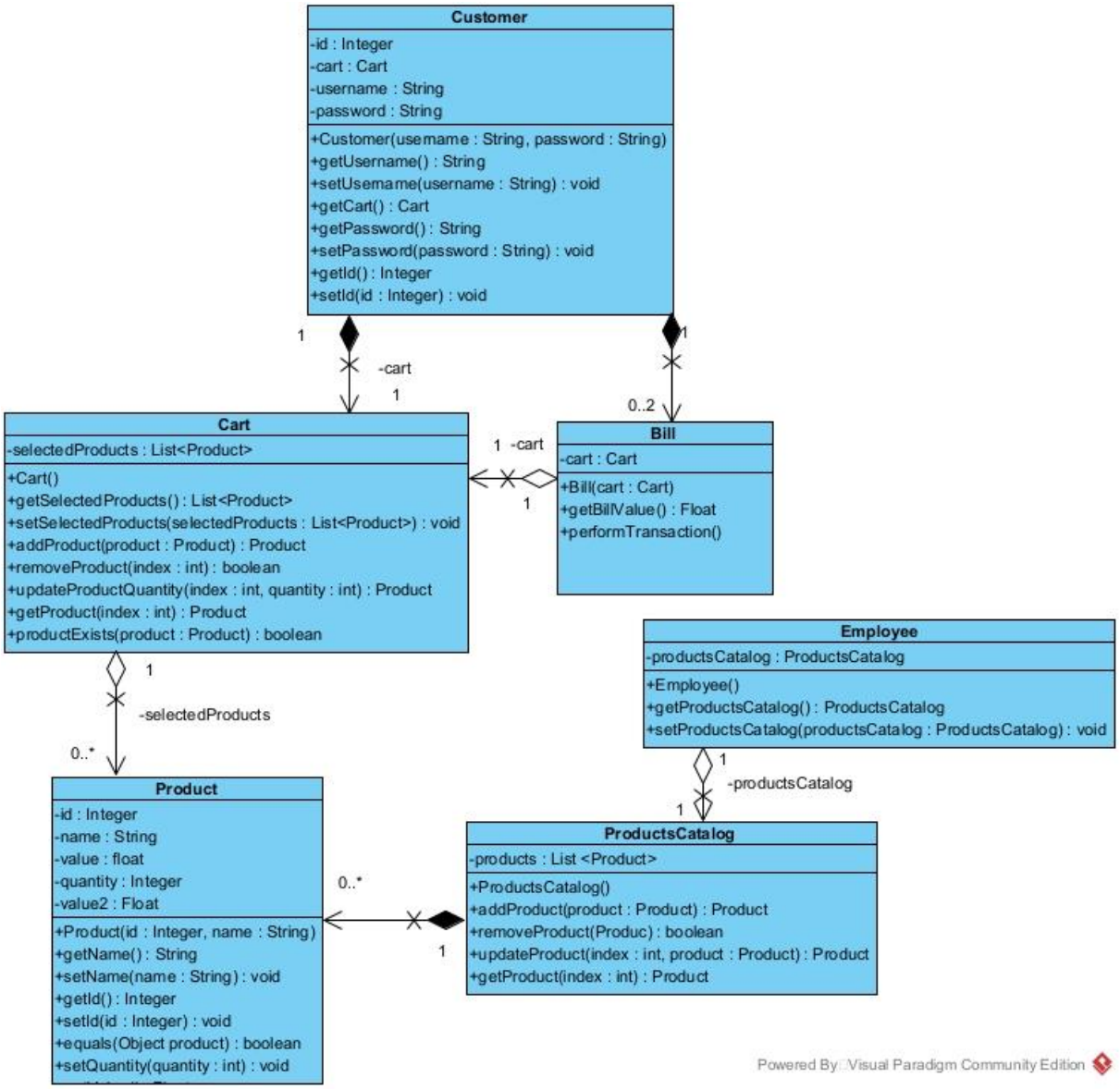
Podczas zajęć udało nam się zrealizować wszystkie wymagane zadania. Dodatkowo stworzyliśmy diagramy dla dwóch dodatkowych PU. Podczas tworzenia diagramów napotkaliśmy na pewne problemy konceptowe, które udało się rozwiązać. Zadanie pozwoliło nam lepiej zapoznać się z technikami i metodami modelowania UML, oraz rozwinąć projekt.

Siódme zajęcia laboratoryjne

1. Zadanie laboratoryjne:

Celem laboratorium było stworzenie modelu projektowanego programowania opartego o identyfikację klas, reprezentujących logikę biznesową projektowanego systemu. Należało wykonać wstępny diagram klas oraz szkielet kodu źródłowego z niego wynikającego.

2. Diagram klas



Rysunek 9: Wstępny diagram klas

2. Szkielet programu oparty na diagramie klas

```
public class Bill {

	private Cart cart;

	public Bill(Cart cart) {

		this.cart = cart;

	}

	public float getBillValue() {

		float billValue = 0;

		for(Product product : cart.getSelectedProducts()) {

			billValue += product.getValue();
		}

		return billValue;
	}

	public boolean performTransaction() {

		return getBillValue() > -1;
	}
}
```

Listing 1: szkielet klasy Bill

```
public class Employee {

	private ProductsCatalog productsCatalog;

	public Employee() {

	}

	public ProductsCatalog getProductsCatalog() {
		return productsCatalog;
	}

	public void setProductsCatalog(ProductsCatalog productsCatalog) {
		this.productsCatalog = productsCatalog;
	}
}
```

Listing 2: Szkielet klasy Employee

```
import java.util.ArrayList;
import java.util.List;

public class Cart {

	private List <Product> selectedProducts;

	public Cart() {

		selectedProducts = new ArrayList<>();
	}

	public List<Product> getSelectedProducts() {
		return selectedProducts;
	}

	public void setSelectedProducts(List<Product> selectedProducts) {
		this.selectedProducts = selectedProducts;
	}

	public Product addProduct(Product product) {

		if(productExists(product)) {

			selectedProducts.add(product);

			return product;
		}

		return null;
	}

	public boolean removeProduct(int index) {

		return selectedProducts.remove(index) != null;
	}

	public Product updateProduct(int index, Product product) {

		return selectedProducts.set(index, product);
	}

	public boolean productExists(Product product) {

		return selectedProducts.contains(product);
	}

	public Product getProduct(int index) {

		return selectedProducts.get(index);
	}
```

Listing 3: Szkielet klasy Cart

```
public class Customer {

	private Integer id;
	private Cart cart;
	private String username;
	private String password;

	public Customer(String username, String password) {

		this.username = username;
		this.password = password;
	}

	public Integer getId() {
		return id;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public Cart getCart() {
		return cart;
	}
}
```

Listing 4: Szkielet klasy Customer

```
public class Product {

	private Integer id;
	private String name;
	private float value;

	public Product(Integer id, String name) {

		this.id = id;
		this.name = name;
	}

	public Integer getId() {
		return id;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public float getValue() {
		return value;
	}

	public void setValue(float value) {
		this.value = value;
	}
}
```

Listing 5: Szkielet klasy Product

```
import java.util.ArrayList;
import java.util.List;

public class ProductsCatalog {

	private List<Product> products;

	public ProductsCatalog() {

		this.products = new ArrayList<>();
	}

	public Product addProduct(Product product) {

		this.products.add(product);

		return product;
	}

	public boolean removeProduct(Product product) {

		return this.products.remove(product);
	}

	public Product updateProduct(int index, Product product) {

		return products.set(index, product);
	}

	public Product getProduct(int index) {

		return products.get(index);
	}
}
```

Listing 6: Szkielet klasy ProductsCatalog

4. Wnioski

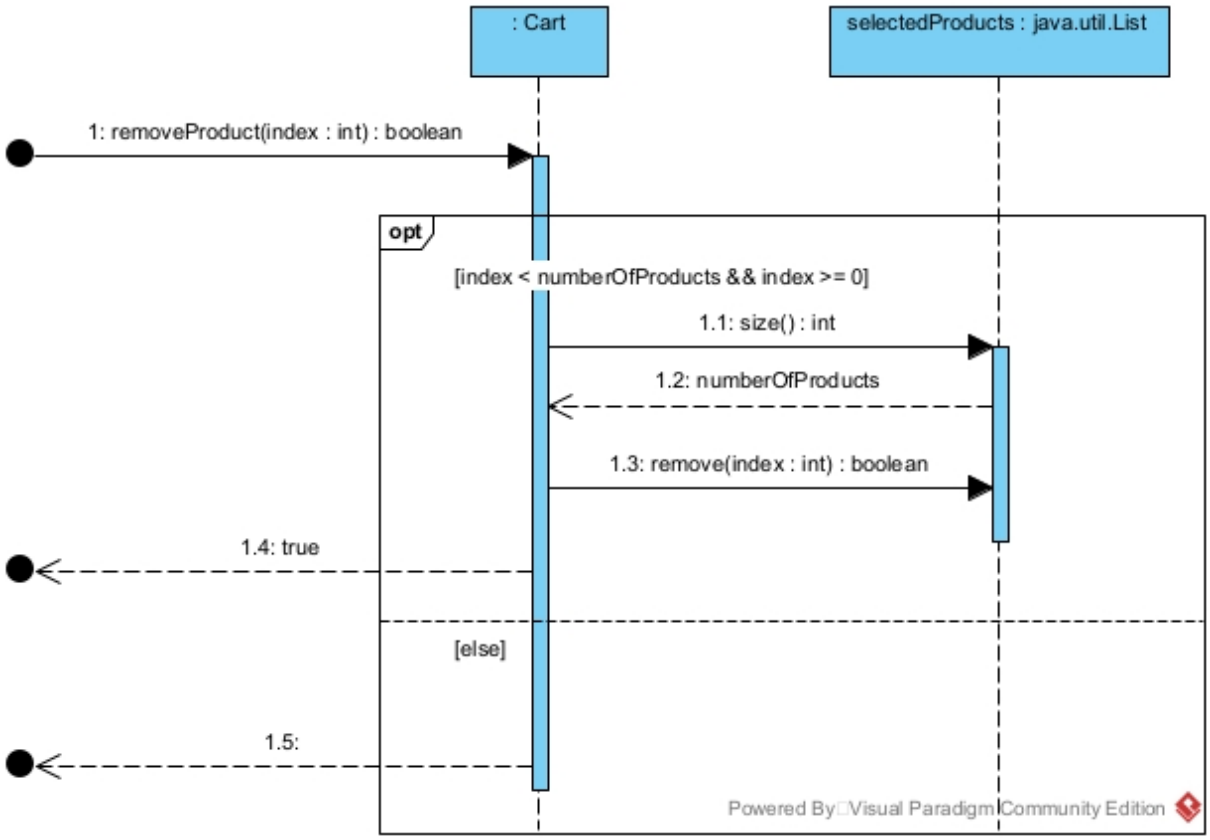
Podczas zajęć udało nam się zrealizować wszystkie wymagane zadania. Stworzyliśmy pełen diagram klas oraz szkielety kodu w języku java. Zadanie pozwoliło nam lepiej zapoznać się z technikami i metodami modelowania diagramów klas, oraz rozwinąć projekt.

Ósme zajęcia laboratoryjne

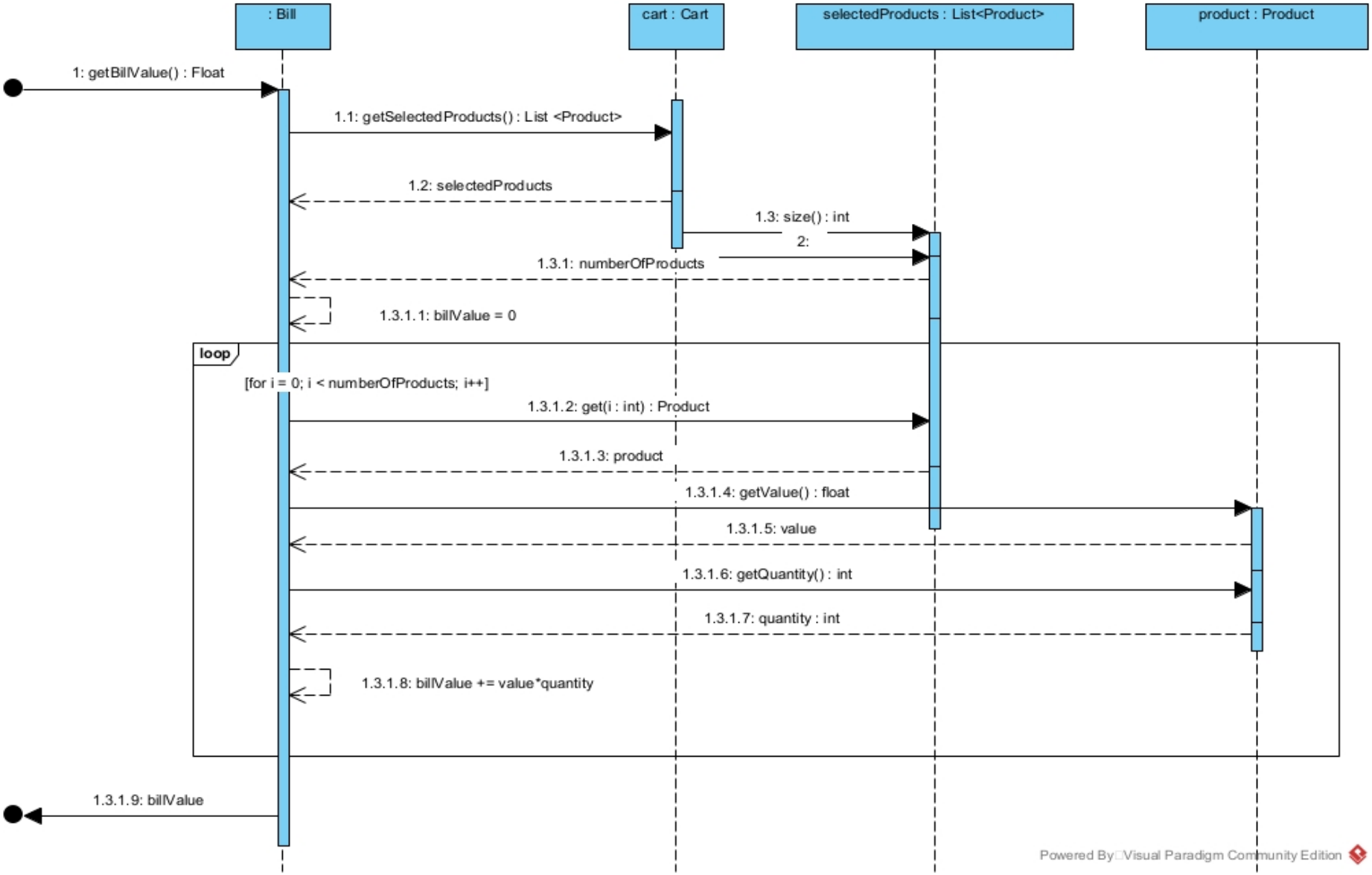
1. Zadanie laboratoryjne

Celem laboratorium było definiowanie modelu projektowego programowania opartego na modelowaniu logiki biznesowej za pomocą wstępnych diagramów sekwencji.

2. Diagramy sekwencji



Rysunek 10: Diagram sekwencji dla PU Usuń Produkt



Rysunek 11: Diagram sekwencji dla PU Oblicz Wartość Rachunku

4. Wnioski

Podczas zajęć udało nam się oba zlecone zadania. Stworzyliśmy dwa rozbudowane diagramy sekwencji reprezentujące Przypadki Użycia. Zadanie pozwoliło nam zapoznać się z tworzeniem diagramów sekwencji i pozwoliło lepiej zrozumieć ideę modelowania UML.