

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

# StoreX

## Spis treści

1.	Wizja.....	4
1.1	Wprowadzenie .....	4
1.2	Pozycjonowanie .....	4
1.2.1	Sformułowanie problemu .....	4
1.2.2	Opis pozycji produktu.....	5
1.3	Opis udziałowców i użytkowników .....	5
1.3.1	Podsumowanie udziałowców.....	5
1.3.2	Podsumowanie użytkowników .....	5
2.	Opis produktu .....	6
2.1	Potrzeby i cechy .....	6
2.2	Inne wymagania produktowe .....	6
3.	Słownik.....	7
4.	Model domenowy .....	9
5.	Reguły biznesowe.....	10
5.1	Bilans .....	10
5.1.1	Słownie .....	10
5.1.2	OCL .....	10
5.2	Towar (wraz z pozycjami bilansu) .....	11
5.3	Lokalizacja.....	11
5.4	Zamówienie.....	12
5.5	Pozycja zamówienia.....	12
5.6	Zamówienie zakupu .....	12
5.7	Przyjęcie zamówienia.....	12
5.8	Pozycja przyjęcia .....	12
5.9	Zamówienie dostawy .....	12
5.10	Wydanie zamówienia .....	12
5.11	Pozycja wydania .....	12
5.12	Klient.....	13
5.13	Dostawca.....	13
5.14	Regał .....	13
5.15	Przestrzeń magazynowa .....	13
5.16	Inne .....	13
6.	Specyfikacja wymagań .....	14
7.	Przypadki użycia.....	15

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

7.1	Diagram przypadków użycia.....	15
7.2	Opisy streszczające .....	16
7.2.1	Pobranie stanu magazynu : .....	16
7.2.2	Generowanie powiadomienia o zalegającym towarze: .....	16
7.2.3	Pobranie zamówień:.....	16
7.2.4	Edycja bilansu:.....	16
7.2.5	Przeglądanie bilansu: .....	16
7.2.6	Przyjmowanie zamówienia na stan: .....	16
7.2.7	Rozładunek zamówienia na magazyn: .....	16
7.2.8	Zmiana lokalizacji towaru: .....	16
7.2.9	Edytowanie przestrzeni magazynowej:.....	16
7.2.10	Zmiana lokalizacji regałów: .....	16
7.2.11	Zmiana właściwości regału: .....	16
7.2.12	Przeglądanie mapy produktów: .....	16
7.2.13	Wydanie zamówienia: .....	17
7.3	Specyfikacja wybranych przypadków użycia .....	18
7.3.1	Generowanie bilansu.....	18
7.3.2	Kompletowanie zamówienia.....	19
7.4	Macierz śladowania.....	20
8.	Model informacyjny .....	21
9.	Mockupy a realizacja – dokumentacja rozbieżności.....	22
9.1	Podsumowanie .....	38
10.	Architektura systemu.....	39
10.1	Architektura logiczna (diagram pakietów) .....	39
10.2	Architektura fizyczna (diagram rozmieszczenia) .....	40
11.	Projekt bazy danych .....	41
11.1	Diagram ERD.....	41
11.2	Wygenerowany skrypt .....	42
11.2.1	Tworzenie tabel .....	42
11.2.2	Dodawnie ograniczeń (constraint) .....	43
12.	Realizacja przypadków użycia .....	45
12.1	Bilansowanie .....	45
12.2	Kompletowanie zamówienia .....	47
13.	Implementacja .....	54
13.1	Kod.....	54
13.2	Wygenerowane fragmenty kodu .....	54
13.3	Dokumentacja .....	56
13.4	Wykorzystane wzorce projektowe .....	57

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

14.	Testy .....	60
14.1	Testy jednostkowe.....	60
14.2	Przypadki testowe .....	62
14.2.1	Kompletowanie zamówienia .....	62
14.2.1.3	003 Kompletowanie zamówienia - wybrano więcej niż w lokalizacji .....	64
14.2.2	Bilansowanie .....	82
14.3	Automatyzacja testów funkcjonalnych .....	91
15.	Badanie jakości projektu (JDepend).....	93
16.	Podsumowanie.....	107

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 1. Wizja

### 1.1 Wprowadzenie

Magazyn towarów obsługuje takie operacje magazynowe jak przyjęcie i wydawanie towaru oraz udostępnianie bieżących stanów magazynowych. Prowadzona jest ewidencja magazynowa towarów posiadających unikatową nazwę i kod oraz jednostkę miary. Ewidencja prowadzona jest poprzez dokumentowanie przyjmowanych i wydawanych towarów. Sporządzane są comiesięczne bilanse w celu ułatwienia odczytu stanu magazynu w podanym punkcie czasu w przeszłości. System generuje powiadomienia o towarach zalegających oraz o kończących się. System zapewnia również optymalizację rozmieszczenia towarów w magazynie oraz ułatwia zarządzanie przestrzenią magazynową poprzez jej dogodną aranżację oraz modyfikację. Towary i zamówienia są identyfikowane poprzez kody kreskowe.

### 1.2 Pozycjonowanie

#### 1.2.1 Sformułowanie problemu

Problem	Prowadzenie ewidencji poprzez dokumentację na papierze oraz nieoptymalne rozmieszczenie towarów w magazynie powoduje trudności w organizacji i znaczne spowolnienie pracy
Dotyczy	Pracownicy magazynu, dział sprzedaży, dział zaopatrzenia
Wpływ problemu	Trudności w organizacji oraz spowolnienie pracy przekłada się na spadek zadowolenia klientów oraz ograniczenie obrotu.
Pomyślne rozwiązanie	Optymalne rozmieszczenie towarów w magazynie, poprzez wykorzystanie podpowiedzi systemu, przekładające się na skrócenie czasu rozładunku oraz przygotowania zamówień  Łatwy dostęp do stanu magazynowego towarów w dowolnym punkcie w czasie  Przejrzysta kartoteka magazynowa

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 1.2.2 Opis pozycji produktu

Dla	Hurtowania HurteX
Który	Chce zwiększyć obrót poprzez usprawnienie działania magazynu
StoreX	Aplikacja Klient-Serwer
Który	Ułatwia dokumentowanie obrotu oraz przyśpiesza pracę w magazynie
Inaczej niż	Inne produkty do wspomagania obsługi i zarządzania magazynem
Nasz produkt	Zapewnia inteligentne funkcje optymalizacji rozmieszczenia oraz przyjazny, łatwy w użyciu interfejs użytkownika

### 1.3 Opis udziałowców i użytkowników

#### 1.3.1 Podsumowanie udziałowców

Nazwa	Opis	Odpowiedzialności
<b>Właściciel hurtowni</b>	<b>Osoba posiadająca większość udziałów</b>	<b>Zarządzanie hurtownią</b>

#### 1.3.2 Podsumowanie użytkowników

Nazwa	Opis	Odpowiedzialności	Dodatkowe informacje
Pracownik magazynu	Pracownik fizyczny	Odpowiedzialny umieszczenie towaru w magazynie oraz wydanie towaru, wprowadzanie do systemu danych dotyczących rozmieszczenia towaru	
Pracownik działu sprzedaży	Osoba zajmująca się sprzedażą towarów	Odpowiedzialny za sprzedaż towarów	Odbiera powiadomienia na temat zaledającego, brakującego towaru, otrzymuje informacje na temat stanu magazynu
Pracownik działu zaopatrzenia	Osoba zajmująca się zaopatrzeniem	Odpowiedzialny za zamawianie towarów do magazynu	Odbiera powiadomienia na temat zaledującego, brakującego towaru, otrzymuje informacje na temat stanu magazynu
Księgowy	Osoba zajmująca się sprawami finansowymi hurtowni	Odpowiedzialny za sprawy finansowe firmy	Odczytuje informacje na temat stanu magazynu

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 2. Opis produktu

### 2.1 Potrzeby i cechy

Potrzeba	Priorytet	Cechy	Planowane wydanie
Aktualizacja ilości towaru	Wysoki	Zarządzanie stanem	Pierwsze
Podpowiedź rozmieszczenia towaru	Wysoki	Lokacja produktu	Pierwsze
Przeglądanie mapy lokalizacji towarów	Wysoki	Lokacja produktu	Pierwsze
Dokumentacja przyjęcia towaru	Wysoki	Rozliczenie finansowe	Pierwsze
Dokumentacja wydania towaru	Wysoki	Rozliczenie finansowe	Pierwsze
Aranżacja przestrzeni magazynowej	Wysoki	Przestrzeń magazynowa	Pierwsze
Sporządzanie bilansu	Średni	Rozliczenie finansowe	Drugie
Udostępnianie stanów magazynowych	Wysoki	Zarządzanie stamen magazynu	Pierwsze
Powiadomienie o zalegającym towarze	Średni	Zarządzanie stamen magazynu	Drugie
Powiadomienie o brakującym towarze	Średni	Zarządzanie stamen magazynu	Drugie
Optymalizacja rozmieszczenia towarów	Średni	Lokacja produktu	Drugie

### 2.2 Inne wymagania produktowe

Wymaganie	Priorytet	Planowane wydanie
Analiza kosztów	Wysoki	Pierwsze
Aktualizacja system operacyjnego	Wysoki	Pierwsze

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 3. Słownik

ID	Nazwa	Opis
3	Zamówienie zakupu	operacja magazynowa polegająca na zmianie stanu ilościowego magazynu. Ilostan odpowiednich towarów jest dekrementowany w ilości i jednostce odpowiadającej ilostanowi towarów przeznaczonych do wydania do sprzedaży.
4	Zamówienie dostawy	operacja magazynowa polegająca na zmianie stanu ilościowego magazynu. Ilostan odpowiednich towarów jest inkrementowany w ilości i jednostce odpowiadającej ilostanowi nowo przybyłych do magazynu towarów.
5	Zamówienie	Zamówienie jest to polecenie dostarczenia lub wydania towaru. W systemie identyfikowane jest przez unikalny kod
6	Wydanie Zamówienia	Realizacja zamówienia zakupu
7	Umieszczenie	ziązek towaru z lokalizacją, posiada informację o ilości towaru w lokalizacji
8	Udostępnienie bieżących stanów magazynowych	operacja polegająca na przekazaniu aktualnego stanu ilościowego poszczególnych towarów ewidencjonowanych w magazynie. Operacja wykonywana jest na żądanie przychodzące z działu zaopatrzenia lub z działu sprzedaży.
9	Towar zalegający	twarzostaje uznany za zalegający jeśli znajduje się w przestrzeni magazynowej dłużej niż maksymalny czas dla niego określony oraz nie została dostarczona żadna nowa dostawa tego towaru.
10	Towar zablokowany	twarzostaje zablokowany jeśli zamówienie które wymaga danego towaru zostaje potwierdzone w systemie
11	Towar kończący się	jest to towar którego ilość jest poniżej ilości minimalnej
12	Towar	dobra materialne przeznaczone na sprzedaż. Towar posiada unikatową nazwę oraz kod. Jest mierzalny w odpowiedniej dla niego jednostce miary, może być również identyfikowany poprzez odpowiednią kategorię (grupę branżową).
13	Stan magazynu	stan magazynu czyli ilostan wszystkich produktów w danych lokalizacjach w określonym punkcie czasu
14	Sektor	pionowa jednostka podziału regałów
15	Regał	miejsce przechowywania towarów, posiada półki oraz sektory. Do niego odnoszą się lokalizacje
16	Półka	pozioma jednostka podziału regałów
17	Przyjęcie Zamówienia	Realizacja zamówienia dostawy
18	Przyjęcie na stan	operacja wpisania Towaru na stan magazynu, faktyczna zmiana ilostanu dostępnego produktu oraz przypisanie nowej lokalizacji.
19	Pracownik magazynu	osoba zatrudniona w magazynie, odpowiada za obsługę systemu, przyjmowanie i wydawanie zamówień oraz zarządzaniem dostępnym zasobem towarów
20	PozycjaZamówienia	niepodzielny element, najmniejsza część składowa zamówienia, dotyczy towaru
21	PozycjaBilansu	niepodzielny element, najmniejsza część składowa bilansu, dotyczy towaru
22	Pozycja Wydania	niepodzielny element, najmniejsza część składowa wydania zamówienia, dotyczy towaru
23	Pozycja Przyjęcia	niepodzielny element, najmniejsza część składowa przyjęcia zamówienia, dotyczy towaru
24	Powierzchnia (magazynowa)	w systemie, przechowuje fizyczną reprezentację położenia wszystkich regałów.
25	Powiadomienie	wiadomość w formie pop-up wymagająca podjęcia akcji
26	Operacja	ogół czynności wynikających z działalności magazynu. Operacje mogą być wykonywane przez żądanie ze strony pracowników, a także ze strony samego systemu.

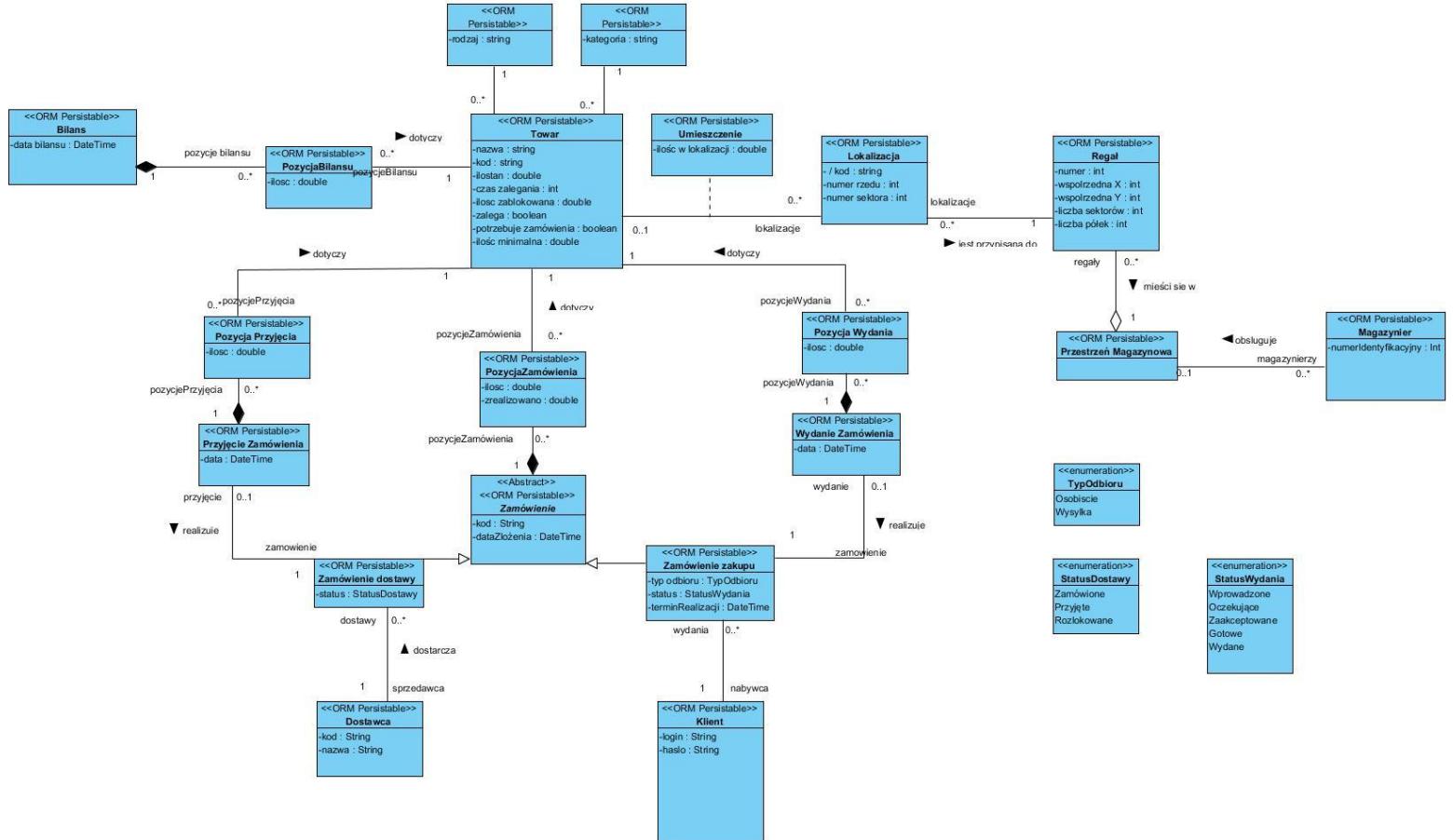
<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

27	Okresowa kartoteka magazynowa	obejmuje wszystkie dowody dotyczące danego towaru. Uwidacznia ilość (i wartość) przychodu, rozchodu i stanu ilościowego po każdej wykonanej operacji w układzie chronologicznym. Wyróżniamy kartoteki w układzie ilościowym lub ilościowo-wartościowym.
28	Obrót magazynowy	wyróżniamy dwa podstawowe pojęcia: Pz (przychód zewnętrzny), Wz (wydanie towaru na zewnątrz) oraz inne (np. protokoły przychodów i rozchodów nadzwyczajnych, protokoły nadwyżek i niedoborów inwentaryzacyjnych, itp.). Podstawową perspektywą wyglądu w obrót danym towarem jest okresowa kartoteka magazynowa.
29	Obrót	różnica między ilością towaru wydanego a dostarczonego
30	Magazyn	miejsce gdzie znajduje się dostępna powierzchnia magazynowa
31	Lokalizacja	unikalna w systemie, posiada numer regału, półkę oraz sektor, np. 15.8C
32	Kod (kreskowy)	unikalny w kontekście całego systemu, może odnosić się zarówno do Towaru jak i Zamówienia.
33	Klient	reprezentacja osoby korzystającej z usług magazynu, dokonuje zamówień
34	Ilość minimalna	określona z góry ilość towaru może zostać uznana za minimalną. minimalna ilość wymaga podjęcia decyzji przez dział dostaw.
35	Ewidencja magazynowa	wykaz, spis dotyczący obrotów w przedsiębiorstwie. Wyróżnionymi elementami ewidencji są: dokumenty (dowody) obrotu magazynowego.
36	Dział zamówień	dodaje nowe zamówienia przychodzące do systemu. Odpowiada za zamawianie towaru do magazynu.
37	Dział sprzedawy	odpowiada za obsługę klienta, z niego przychodzą żądania zamówień wychodzących
38	Dostawca	reprezentacja firmy dostarczającej zamówienia na stan magazynu.
39	Data wykonania	Data, w której został wykonany bilans.
40	Data bilansu	Data zawierająca miesiąc, dla którego sporządzany jest bilans.
41	Bilans	raport przychodów i rozchodów wykonywany co miesiąc

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 4. Model domenowy

Vista [mlynarczyk.institute of informatics](http://mlynarczyk.institute of informatics)



<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 5. Reguły biznesowe

### 5.1 Bilans

#### 5.1.1 Słownie

Bilans dla danego miesiąca może być wykonany dopiero, kiedy dany miesiąc się skończył  
Bilans musi być sporządzony raz dla każdego miesiąca..

Bilans musi mieć określona datę wykonania.

Bilans musi mieć określona datę bilansowanego miesiąca.

Bilans może zawierać wiele pozycji bilansu.

Pozycja bilansu należy do jednego bilansu.

Pozycja bilansu dotyczy jednego towaru.

Pozycja bilansu musi mieć określoną ilość.

Ilość w pozycji bilansu musi być obliczona jako suma ilości danego towaru w pozycji bilansu z poprzedniego bilansu oraz obrotu danym towarem od czasu poprzedniego bilansu.

Obrót towarem jest obliczany na podstawie przyjęć zamówień oraz wydań zamówień.

Jeżeli towar nie wystąpił w pozycji bilansu w poprzednim bilansie, to jego ilość w poprzednim bilansie przyjmowana jest jako 0.

Jeżeli tworzony jest pierwszy bilans, to ilość towaru w pozycji bilansu w poprzednim bilansie jest przyjmowana jako 0.

#### 5.1.2 OCL

context Bilans

pre:

Bilans.allInstances().select(el|el.dataBilansu.getMonth() == self.dataBilansu.getMonth() and el.dataBilansu.getYear() == self.dataBilansu.getYear()).count() == 0

inv:

self.dataWykonania <> null

self.dataBilansu <> null

self.dataBilansu.getMonth() < DateTime.Now.getMonth();

context PozycjaBilansu inv:

self.bilans <> null

self.twarz <> null

self.ilosc <> null

derive self.ilosc :double

```
let iloscWPoprzednimBilansie : double = Bilans.allInstances().select(el|el.dataBilansu <
self.bilans.dataBilansu).sortedBy(dataBilansu).collectNested(pozycjeBilansu).flatten().select(el|
el.twarz.nazwa ==self.twarz.nazwa).first().twarz.ilosc,
```

```
let nowyMiesiac: int = Bilans.allInstances().sortedBy(dataBilansu).last().data.getMonth()
```

```
let iloscWPrzyjeciachZamowien : double = PrzyjecieZamowienia.allInstances.select(el|el.data.getMonth() ==
nowyMiesiac).collectNested(pozycjeWydania).flatten().select(el|el.twarz.nazwa ==
self.twarz.nazwa).collectNested(ilosc).flatten().sum(),
```

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

```

let iloscWWydaniachZamowien: double = WydanieZamowienia.allInstances.select(el|el.data.getMonth()
== nowyMiesiac).collectNested(pozycjeWydania).flatten().select(el|el.towar.nazwa ==
self.towar.nazwa).collectNested(ilosc).flatten().sum()

in
if (Bilans.allInstances().select(el|el.dataBilansu < self.bilans.dataBilansu).isEmpty) ||
Bilans.allInstances().select(el|el.dataBilansu <
self.bilans.dataBilansu).sortedBy(dataBilansu).collectNested(pozycjeBilansu).flatten().select(el|
el.towar.nazwa ==self.towar.nazwa).isEmpty)
then
iloscwPrzyjaciachZamowien - iloscWWydaniachZamowien
else
iloscwPrzyjaciachZamowien - iloscWWydaniachZamowien + iloscWPoprzednimBilansie

```

## 5.2 Towar (wraz z pozycjami bilansu)

Nazwa towaru musi być unikalna.

Kod towaru musi być unikalny.

Każdy towar musi posiadać aktualny ilostan.

Każdy towar musi mieć określona jednostkę miary.

Każdy towar musi mieć określony czas zalegania.

Każdy towar musi mieć określona kategorię.

Każdy towar musi mieć określona ilość minimalną.

Każdy towar musi mieć określona ilość zablokowaną.

Ilość zablokowana musi być mniejsza lub równa ilostanowi.

Ilostan zawiera w sobie ilość zablokowaną.

Ilość zablokowana jest obliczana na podstawie różnicy ilości towaru w pozycjach zamówień zakupu o statusie Zaakceptowane, Przygotowywane, Gotowe oraz ilości towaru w pozycjach wydania.

Jeżeli różnica ilości towaru w pozycjach zamówień zakupu o statusie Zaakceptowane, Przygotowane, Gotowe oraz ilości towaru w pozycjach wydania jest większa od ilostanu to ilość zablokowana jest równa ilostanowi, a towar musi być uznany za potrzebujący uzupełnienia.

Towar może występować w wielu pozycjach bilansu.

W jednym bilansie towar może wystąpić tylko w jednej pozycji bilansu.

Towar może występować w wielu lokalizacjach.

Towar może występować w wielu pozycjach zamówienia.

Towar może występować w wielu pozycjach przyjęcia.

Towar może występować w wielu pozycjach wydania.

Dla jednego zamówienia towar może wystąpić tylko w jednej pozycji zamówienia.

Towar musi być uznany za zalegający, jeśli nie był wydawany przez określoną wcześniej ilość dni.

Towar musi być uznany za potrzebujący uzupełnienia, jeżeli różnica jego ilostanu i ilości zablokowanej jest mniejsza niż ilość minimalna.

Ilość towaru w podanym punkcie czasu jest obliczona jako suma ilości w pozycji bilansu w najpóźniejszym bilansie przed podanym punktem czasu oraz obrotu danym towarem od czasu tego bilansu.

Jeżeli podany punkt czasu występuje przed datą pierwszego bilansu, to ilość w pozycji bilansu w poprzednim bilansie jest przyjmowana jako 0.

## 5.3 Lokalizacja

Kod lokalizacji musi być unikalny.

Kod lokalizacji jest wyliczany na podstawie numeru odpowiedniego regału, półki oraz sektora.

Lokalizacja może być przypisana do maksymalnie jednego regału.

Lokalizacja musi być przypisana do regału.

Lokalizacja może mieć tylko jeden towar.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

Lokalizacja nie musi mieć przypisanego towaru.

Lokalizacja musi przechowywać dane o swojej zajętości .

Zajętość lokalizacji oblicza się na postawie tego, czy do lokalizacji przypisany jest towar.

#### **5.4 Zamówienie**

Zamówienie może składać się z wielu pozycji zamówienia.

Zamówienie musi mieć co najmniej jedną pozycję zamówienia.

Zamówienie musi mieć określona datę złożenia

Kod zamówienia musi być unikalny.

#### **5.5 Pozycja zamówienia**

Pozycja zamówienia należy do jednego bilansu.

Pozycja zamówienia dotyczy jednego towaru.

Pozycja zamówienia musi mieć określoną ilość.

#### **5.6 Zamówienie zakupu**

Zamówienie zakupu towaru musi mieć określony typ odbioru.

Zamówienie zakupu może być odebrane tylko przez jednego klienta.

Zamówienie zakupu musi mieć przypisanego klienta.

Zamówienie zakupu musi posiadać aktualny status.

Zamówienie zakupu może być realizowane przez jedno wydanie zamówienia.

#### **5.7 Przyjęcie zamówienia**

Przyjęcie zamówienia musi realizować jedno zamówienie dostawy.

Przyjęcie zamówienia może zawierać wiele pozycji przyjęcia.

Przyjęcie zamówienia musi mieć określoną datę.

#### **5.8 Pozycja przyjęcia**

Pozycja przyjęcia musi dotyczyć jednego przyjęcia zamówienia.

Pozycja przyjęcia może dotyczyć wielu towarów.

Pozycja przyjęcia musi mieć określoną ilość.

#### **5.9 Zamówienie dostawy**

Zamówienie dostawy musi pochodzić od jednego dostawcy, w przeciwnym razie musi zostać wygenerowane osobny kod.

Zamówienie dostawy musi posiadać aktualny status.

Zamówienie dostawy może być realizowane przez jedno przyjęcie zamówienia.

#### **5.10 Wydanie zamówienia**

Wydanie zamówienia musi realizować jedno zamówienie zakupu.

Wydanie zamówienia może zawierać wiele pozycji wydania.

Wydanie zamówienia musi mieć określoną datę.

#### **5.11 Pozycja wydania**

Pozycja wydania musi dotyczyć jednego wydania zamówienia.

Pozycja wydania może dotyczyć wielu towarów.

Pozycja wydania musi mieć określoną ilość.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## **5.12 Klient**

Kod klienta musi być unikalny.

Klient może nie odebrać żadnego zamówienia.

Klient może odbierać wiele zamówień.

## **5.13 Dostawca**

Kod dostawcy musi być unikalny.

Dostawca może dostarczyć wiele zamówień.

Dostawca nie musi dostarczyć żadnego zamówienia.

## **5.14 Regał**

Regał nie może zostać usunięty dopóki jest na nim przechowywany towar.

Regał może mieścić się tylko w jednej przestrzeni magazynowej.

Regał musi mieścić się w przestrzeni magazynowej.

Regał może mieć przypisanych wiele lokalizacji.

Regał może nie mieć przypisanych lokalizacji.

Regał musi mieć co najmniej jeden sektor.

Regał musi mieć co najmniej jedną półkę.

Regał musi posiadać współrzędne X i Y.

Regał musi posiadać unikalny w przestrzeni magazynowej numer.

## **5.15 Przestrzeń magazynowa**

Przestrzeń magazynowa może posiadać wiele regałów.

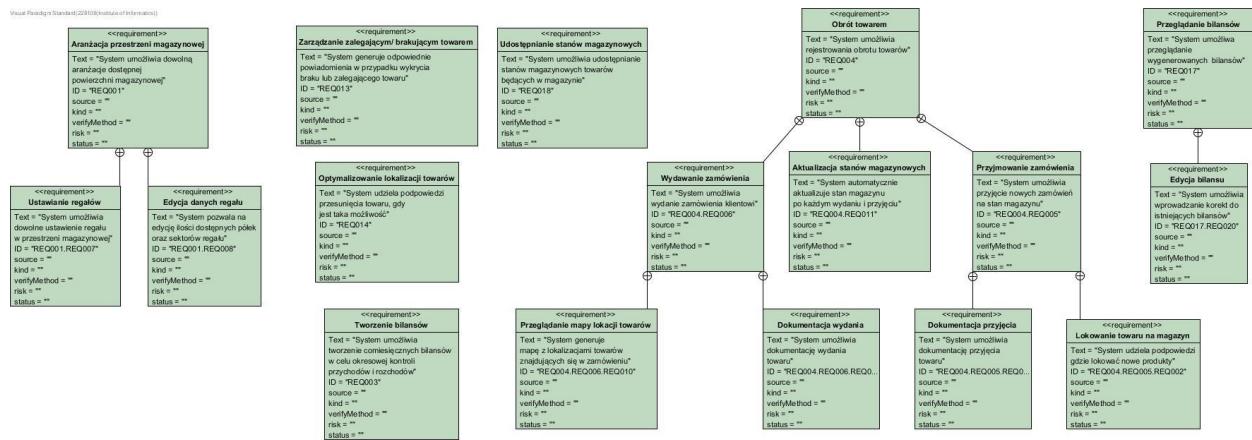
## **5.16 Inne**

Stan magazynu jest na bieżąco aktualizowany, tj. po każdym zakończonym rozpakowaniu oraz wydaniu towaru.

Ilość zablokowana jest aktualizowana po wprowadzeniu zamówienia zakupu oraz po wydaniu towaru.

<b>Storex</b> Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>
---	--------------------

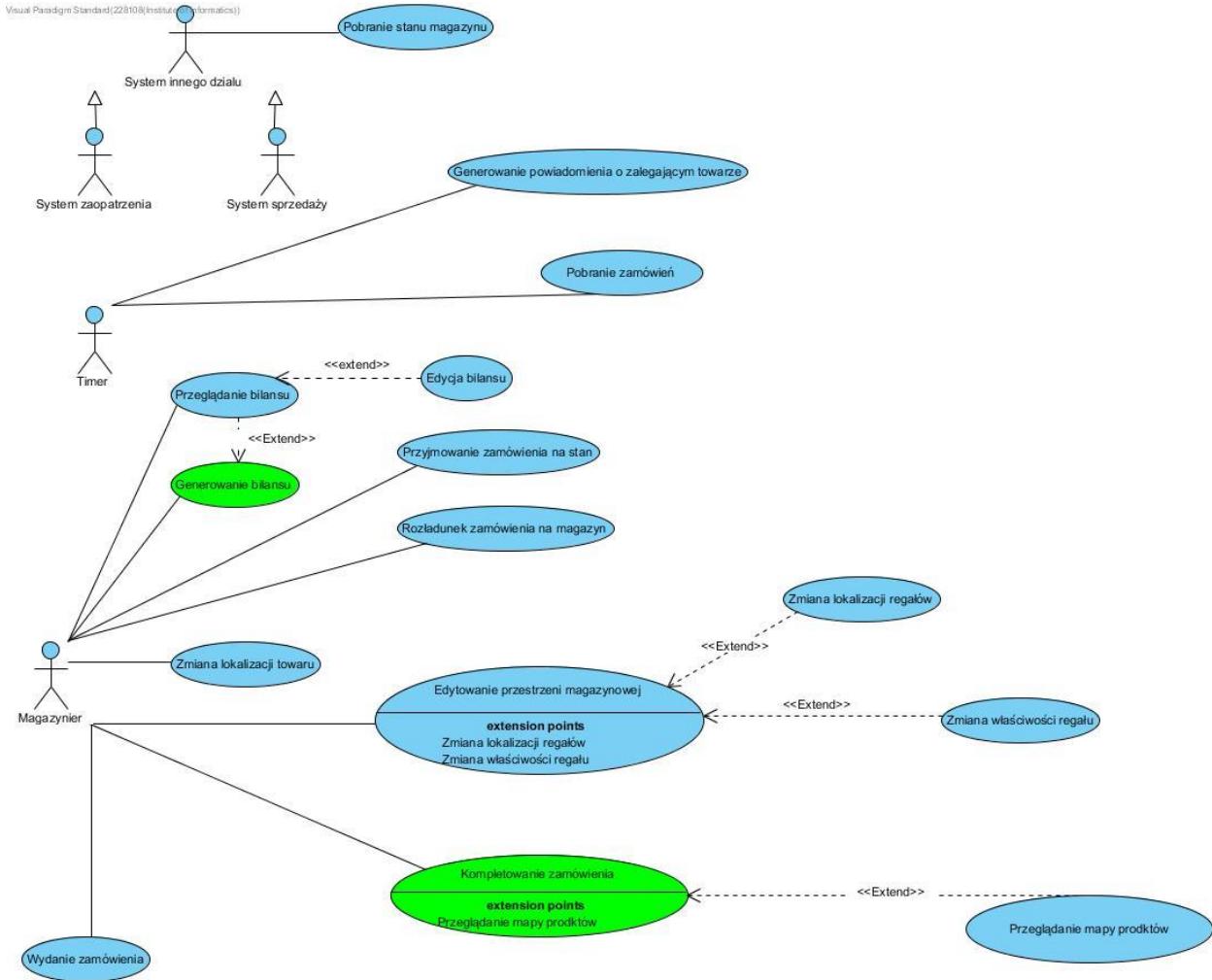
## 6. Specyfikacja wymagań



<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 7. Przypadki użycia

### 7.1 Diagram przypadków użycia



<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 7.2 Opisy streszczające

### 7.2.1 *Pobranie stanu magazynu :*

Odczytanie informacji na temat ilostanu oraz ilości zablokowanej podanego przez użytkownika towaru lub wszystkich towarów.

### 7.2.2 *Generowanie powiadomienia o zalegającym towarze:*

Generowanie przez system powiadomień o zalegającym towarze na podstawie sprawdzania daty ostatniego zamówienia dla każdego towaru. Sprawdzanie wykonywane jest co określony przedział czasu. Powiadomienie jest wysyłane do systemu zewnętrznego.

### 7.2.3 *Pobranie zamówień:*

Pobieranie przez system nowych zamówień. Pobranie zamówień powoduje aktualizację ilości zablokowanej towarów, które znajdują się w zamówieniach. W przypadku, gdy ilość towaru niezablokowanego będzie za mała, generowane jest powiadomienie o brakującym towarze.

### 7.2.4 *Edycja bilansu:*

Wprowadzanie korekt do wygenerowanego wcześniej bilansu.

### 7.2.5 *Przeglądanie bilansu:*

Przeglądanie danych dotyczące bilansu wygenerowanego przez system.

### 7.2.6 *Przyjmowanie zamówienia na stan:*

Przyjęcie dostawy towarów do magazynu. Zatwierdzenie operacji przez magazyniera powoduje wygenerowanie nowego przyjęcia zamówienia oraz zmianę statusu zamówienia na "Przyjęte"

### 7.2.7 *Rozładunek zamówienia na magazyn:*

Umieszczanie towaru w odpowiednich lokalizacjach na podstawie podpowiedzi systemu lub samodzielnnej decyzji magazyniera. Umieszczenie towaru oraz zatwierdzenie czynności przez magazyniera powoduje aktualizację danych na temat lokalizacji towaru oraz zajętości lokalizacji. Po zakończonym rozładunku zamówienie zmienia stan na "Rozlokowane"

### 7.2.8 *Zmiana lokalizacji towaru:*

Zmiana lokalizacji towaru przez magazyniera na podstawie podpowiedzi systemu lub decyzji magazyniera za to odpowiedzialnego. Potwierdzenie przez magazyniera zmiany lokalizacji powoduje aktualizację danych na temat lokalizacji towaru oraz zajętości lokalizacji.

### 7.2.9 *Edytowanie przestrzeni magazynowej:*

System wyświetla listę dostępnych regałów i pozwala na ich edycję

### 7.2.10 *Zmiana lokalizacji regałów:*

Zmiana lokalizacji regałów przez magazynierów. Zatwierdzenie przez magazyniera operacji powoduje aktualizację danych dotyczących współrzędnych regałów.

### 7.2.11 *Zmiana właściwości regalu:*

Zmiana liczby półek oraz sektorów w regale samodzielnej decyzji magazyniera za to odpowiedzianego. Dane są podawane przez magazyniera. Zatwierdzenie danych powoduje aktualizacje na temat liczby półek oraz sektorów w danym regale.

### 7.2.12 *Przeglądanie mapy produktów:*

System umożliwia przeglądanie mapy z zaznaczonymi na niej produktami potrzebnymi do skompletowania zamówienia przez magazyniera.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

**7.2.13 Wydanie zamówienia:**

Wydanie przez magazyniera skompletowanego zamówienia klientowi lub przygotowanie oraz wysłanie paczki, w przypadku wybrania przez klienta dostarczenia zamówienia poprzez wysyłkę. Zatwierdzenie operacji przez magazyniera powoduje aktualizację danych na temat ilostanu oraz ilości zablokowanej towarów w magazynie. zamówienie zmienia stan na Wydane

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 7.3 Specyfikacja wybranych przypadków użycia

#### 7.3.1 Generowanie bilansu

❖ Use Case Specification

General Extension Points Stereotypes Tagged Values

Name: Generowanie bilansu  
ID: UC14  
Rank: Unspecified  
Description:

B F E + Q

Nazwa: Generowanie Bilansu  
Aktorzy: Magazynier  
Wyzwalańce: Magazynier  
Warunki wejściowe: Bilans nie został wykonany w miesiącu zawartym w dniu podanym przez magazyniera.  
Warunki wyjściowe: Bilans został zapisany w systemie.

Scenariusz główny:  
1. Magazynier wybiera datę bilansu  
2. System stwierdza, że data jest poprawna  
3. System zapisuje bilans oraz udostępnia opcję przeglądania.

Scenariusz alternatywny:  
2a. System stwierdza, że data jest niepoprawna.  
Powrót do kroku 1

Abstract  Leaf  Root  Business model

Reset OK Cancel Apply Help

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 7.3.2 Kompletowanie zamówienia

❖ Use Case Specification

General Extension Points Stereotypes Tagged Values

Name: Kompletowanie zamówienia.

ID: UC08

Rank: Unspecified

Description:

Aktor: Magazynier

Warunki pre: Zamówienie istnieje w systemie

Warunki post: Zamówienie zmienia status na gotowe (czekuje na odbiór lub wysyłkę)

Scenariusz główny:

1. Magazynier rozpoczyna kompletowanie zamówienia.
2. System wyświetla liste zamówień oczekujących na skompletowanie.
3. Magazynier wybiera zamówienie.
4. System wyświetla numer zamówienia, liste produktów w nim zawartych wraz z ich ilościami w zamówieniu oraz ilościami zrealizowanymi oraz udostępnia przeglądanie mapy lokalizacji.
5. Magazynier wybiera produkt.
6. System wyświetla posortowaną względem odległości listę lokalizacji wraz z ilością towaru dostępna w danej lokalizacji.
7. Magazynier wybiera lokalizację i podaje pobrana ilość.
8. System stwierdza, że podana ilość jest prawidłowa i zmienia ilość danego towaru w lokalizacji oraz ilość zrealizowaną.
- Sekwencja kroków 4-8 może być wykonywana wielokrotnie przed przejściem do kroku 9.
9. Po ukończeniu kompletowania magazynier przekazuje zamówienie do wydania.
10. Magazynier zatwierdza skompletowanie zamówienia.
11. System wyświetla informacje o danych do nadania paczki.

Scenariusz alternatywny:

- 2A. Jeżeli nie ma żadnych oczekujących zamówień system wyświetla monit o braku oczekujących zamówień.
- 6A. Jeżeli produktu nie ma na stanie magazynu system wyświetla monit o braku danego towaru.
- 8A. Jeżeli wpisana ilość produktu jest większa system informuje o błędzie i wraca do kroku 6.
- 10A. Magazynier anuluje przekazanie do wydania  
Powrót do kroku 4
- 11A. Jeżeli zamówienie jest oznaczone jako zamówienie do odbioru osobistego wysyłany jest e-mail do klienta informujący o możliwości odbioru.

Abstract  Leaf  Root  Business model

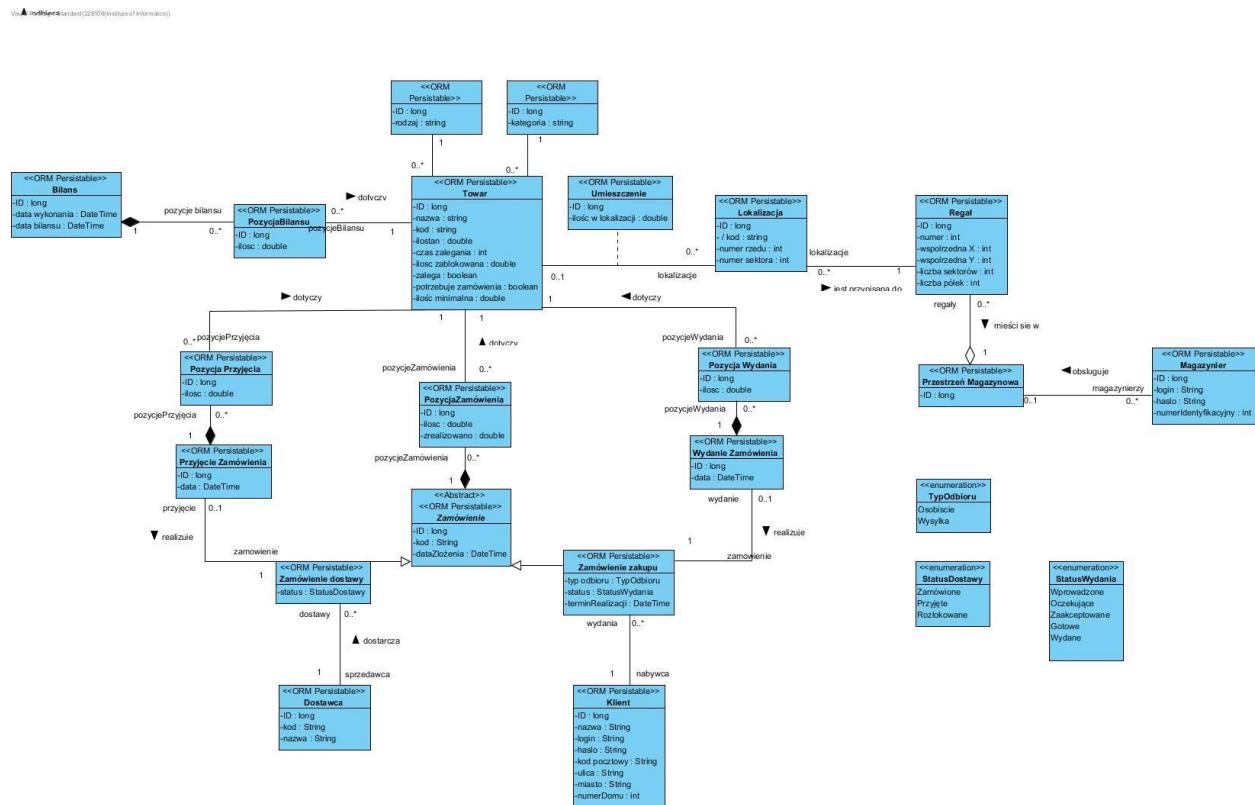
Reset

## 7.4 Macierz śladowania

		(15) Use Case														
		Edycja bilansu	Generowanie bilansu	Pobranie zamówień	Przeglądanie bilansu	Zmiana właściwości regału	Zmiana lokalizacji regałów	Przeglądanie mapy produktów	Wydanie zamówienia	Kompletowanie zamówienia	Rozładunek zamówienia na magazyn	Przyjmowanie zamówienia na stan	Generowanie powiadomienia o zalegającym towarze	Pobranie stanu magazynu		
		(17) Dokumentacja przyjęcia, Dokumentacja wydani...														
	Dokumentacja przyjęcia														R	
	Dokumentacja wydania														R	
	Edycja bilansu	R													R	
	Optymalizowanie lokalizacji towarów				R										R	
	Przeglądanie bilansów					R									R	
	Udostępnianie stanów magazynowych						R								R	
	Zarządzanie zalegającym/ brakującym towarem							R							R	
	Aktualizacja stanów magazynowych								R						R	
	Przeglądanie mapy lokacji towarów								R						R	
	Edycja danych regału									R					R	
	Ustawianie regałów										R				R	
	Wydawanie zamówienia											R			R	
	Przyjmowanie zamówienia												R		R	
	Obrót towarem														R	R
	Tworzenie bilansów														R	
	Lokowanie towaru na magazyn														R	
	Aranżacja przestrzeni magazynowej															

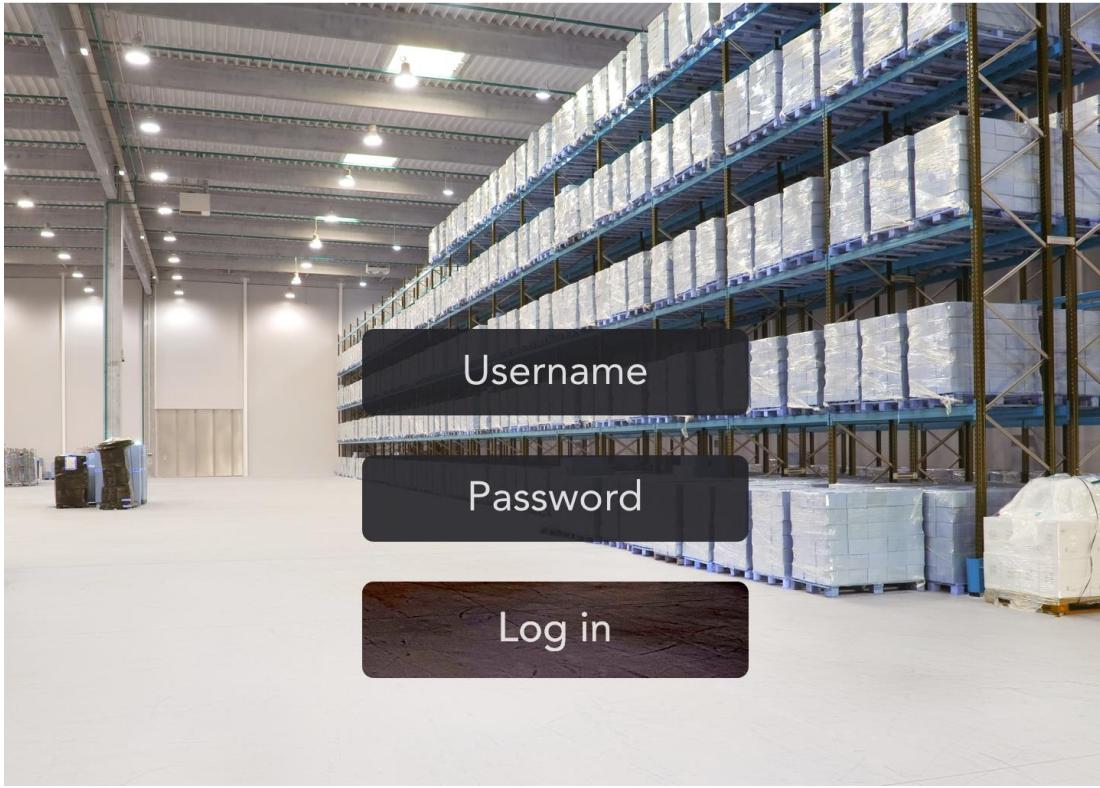
Macierz śladowania

## 8. Model informacyjny

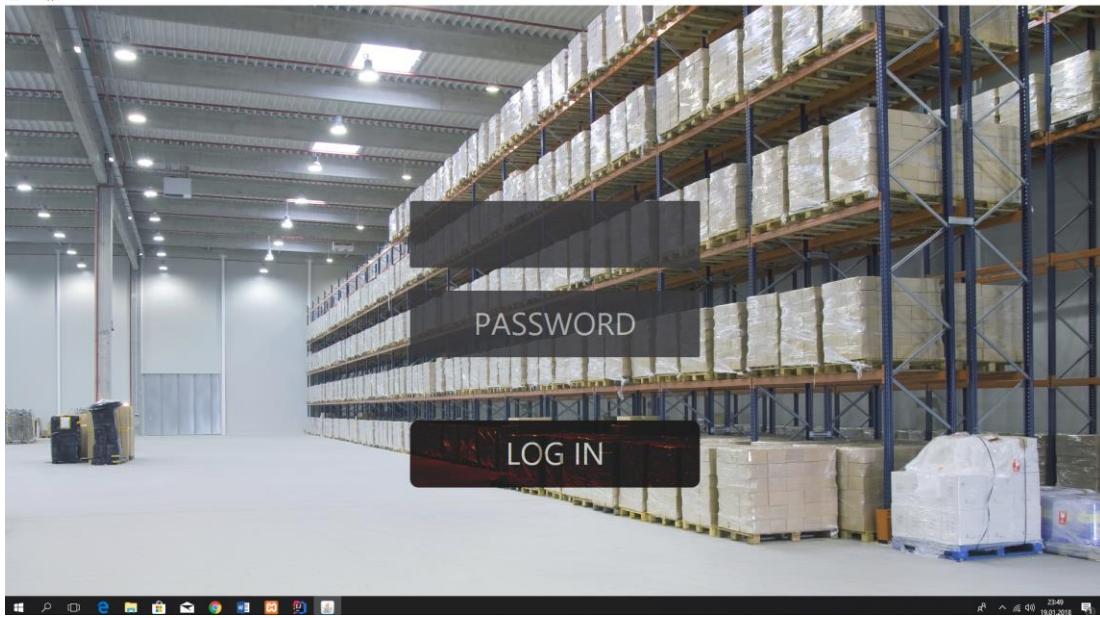


<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 9. Mockupy a realizacja – dokumentacja rozbieżności



*Ekran logowania*



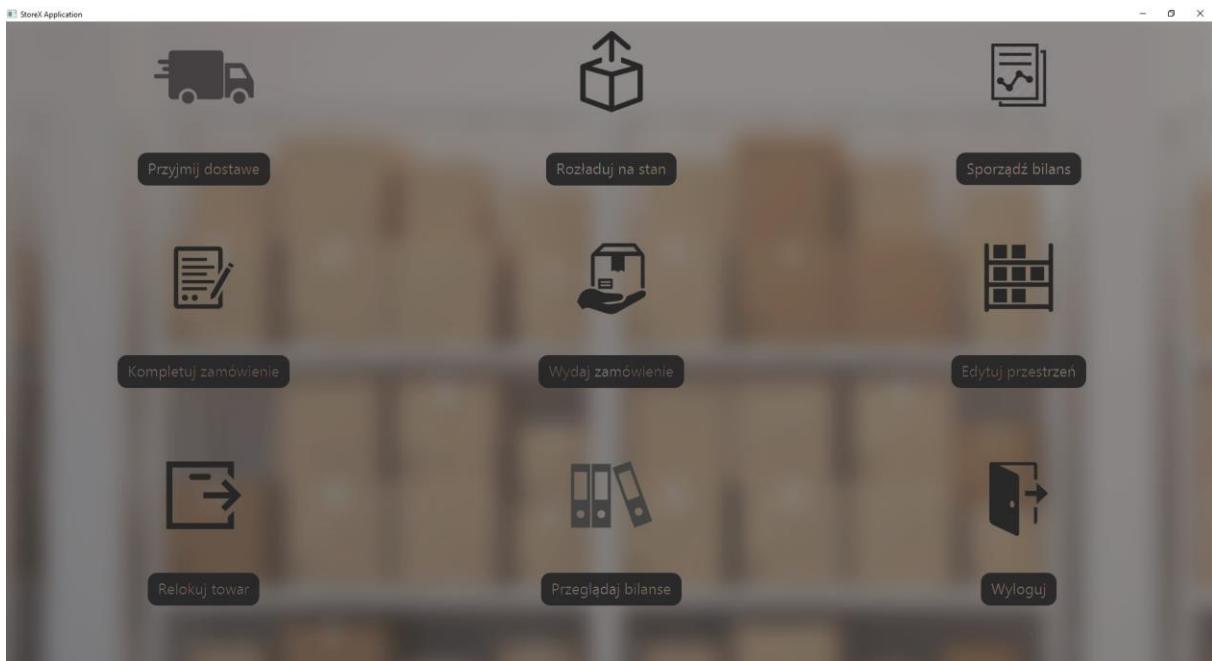
*Ekran logowania - implementacja*

Nie zanotowano większych różnic w implementacji ekranu logowania. Odcień tła cieplejszy na mockupie.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



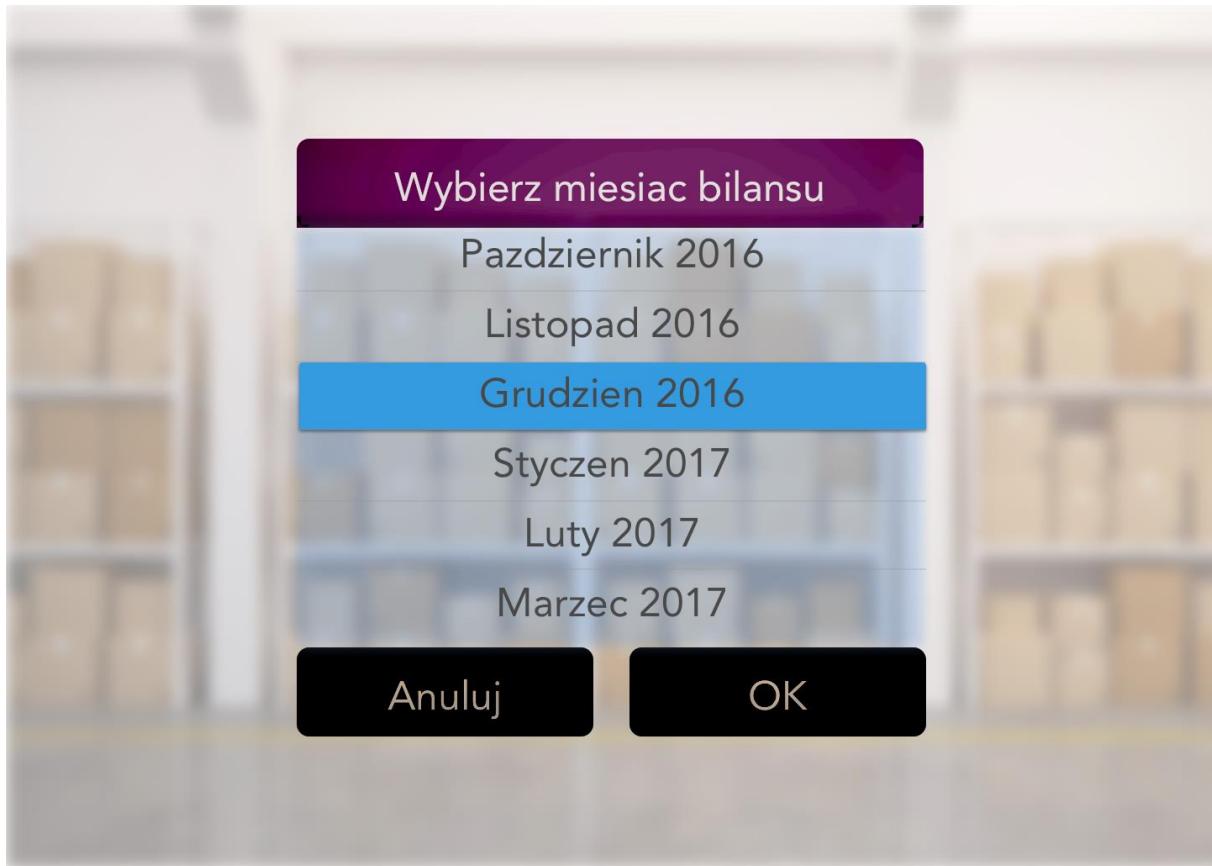
Ekran główny



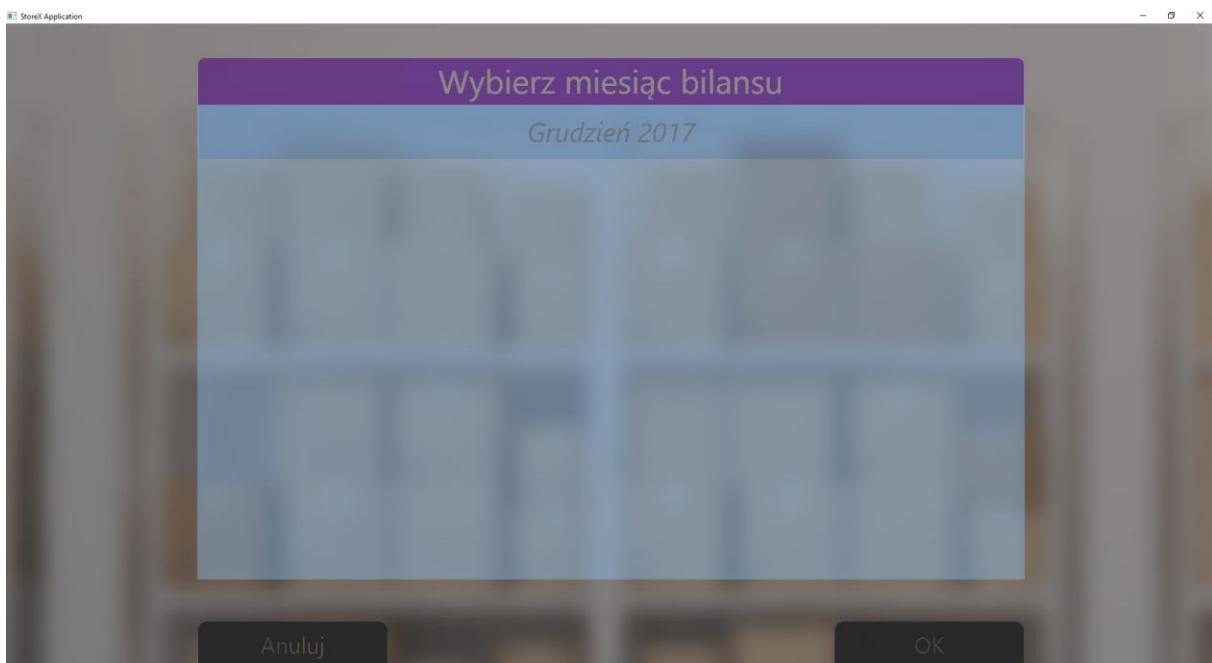
Ekran główny - implementacja

Nie zanotowano większych różnic w implementacji ekranu głównego. Ikony zostały zmniejszone, tło przyciemnione.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



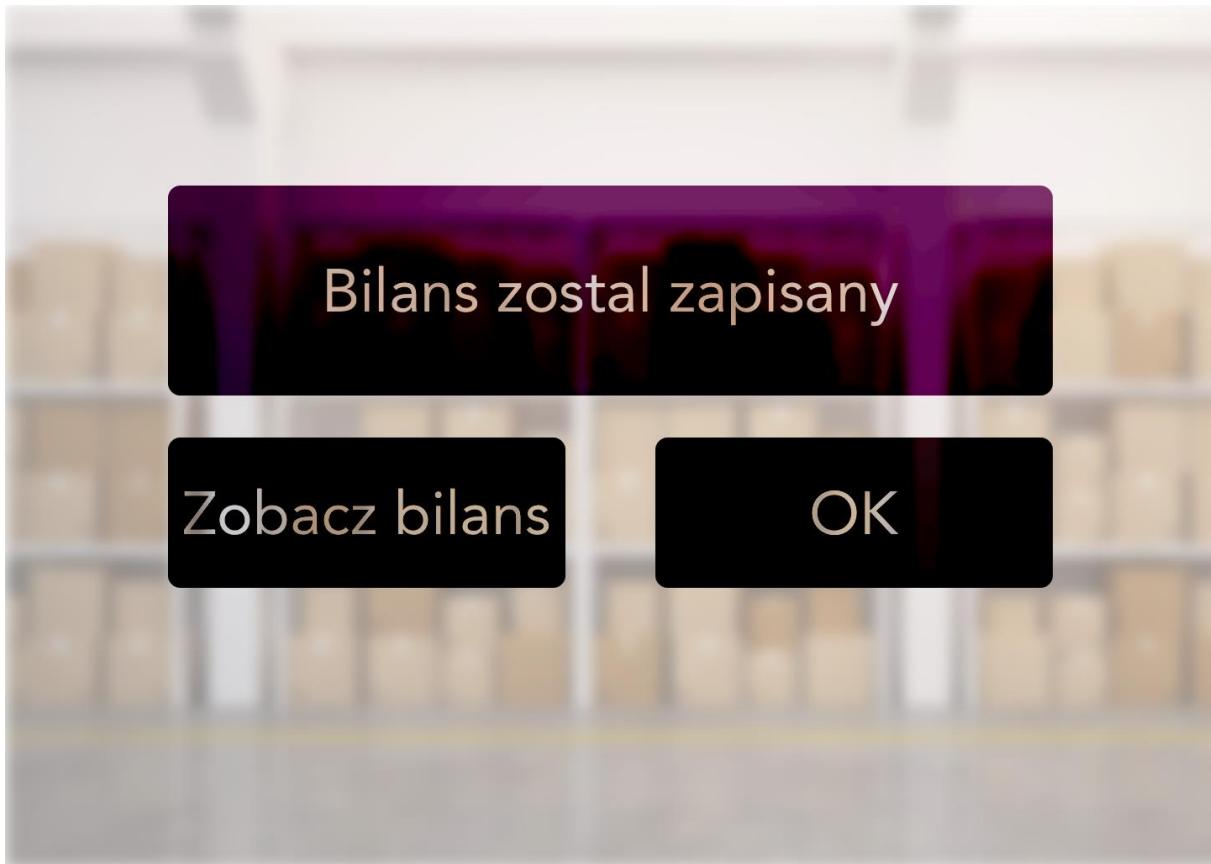
*Ekran wyboru daty bilansowania*



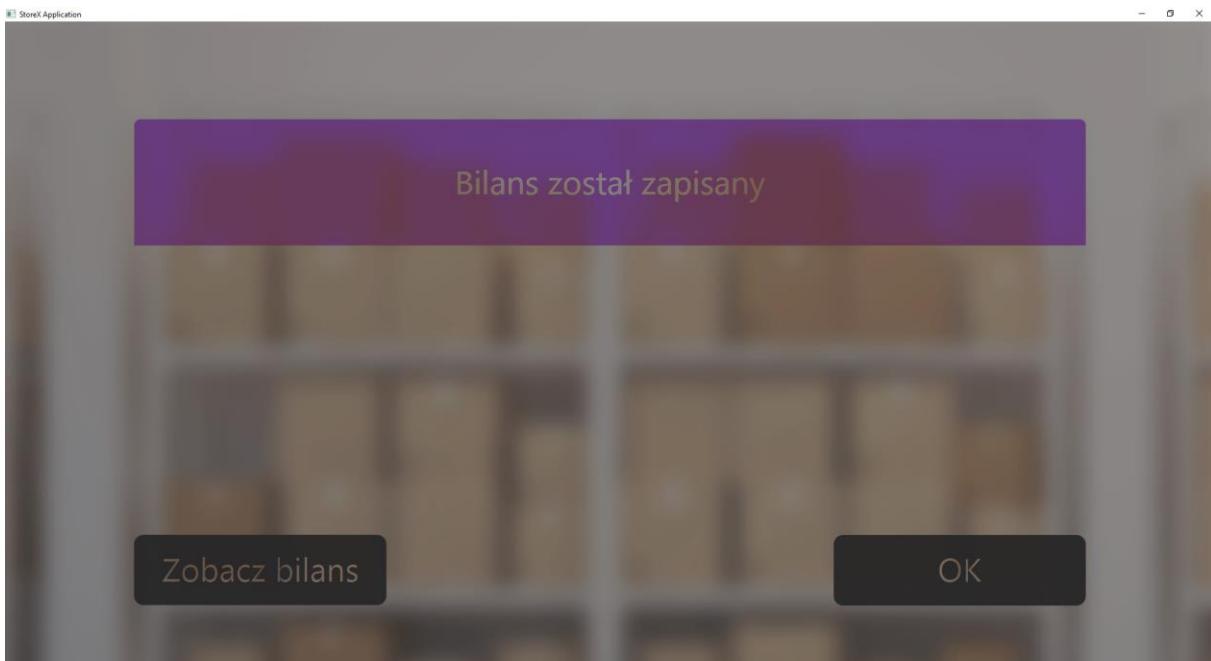
*Ekran wyboru daty bilansowania - implementacja*

Nie zanotowano większych różnic w implementacji ekranu wyboru daty. Tabela zajmuje większą część ekranu. Barwy elementów interfejsu miej kontrastowe.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



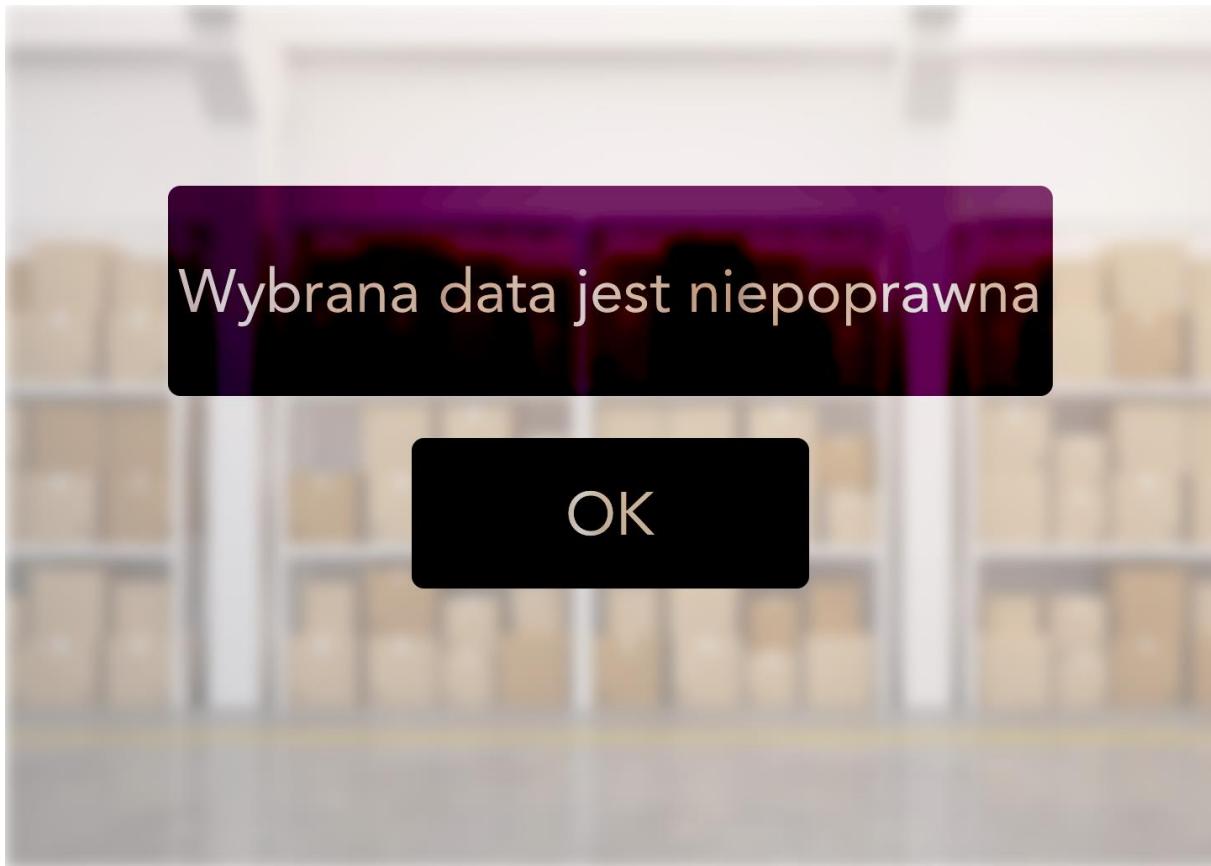
*Ekran potwierdzenia zapisu bilansu*



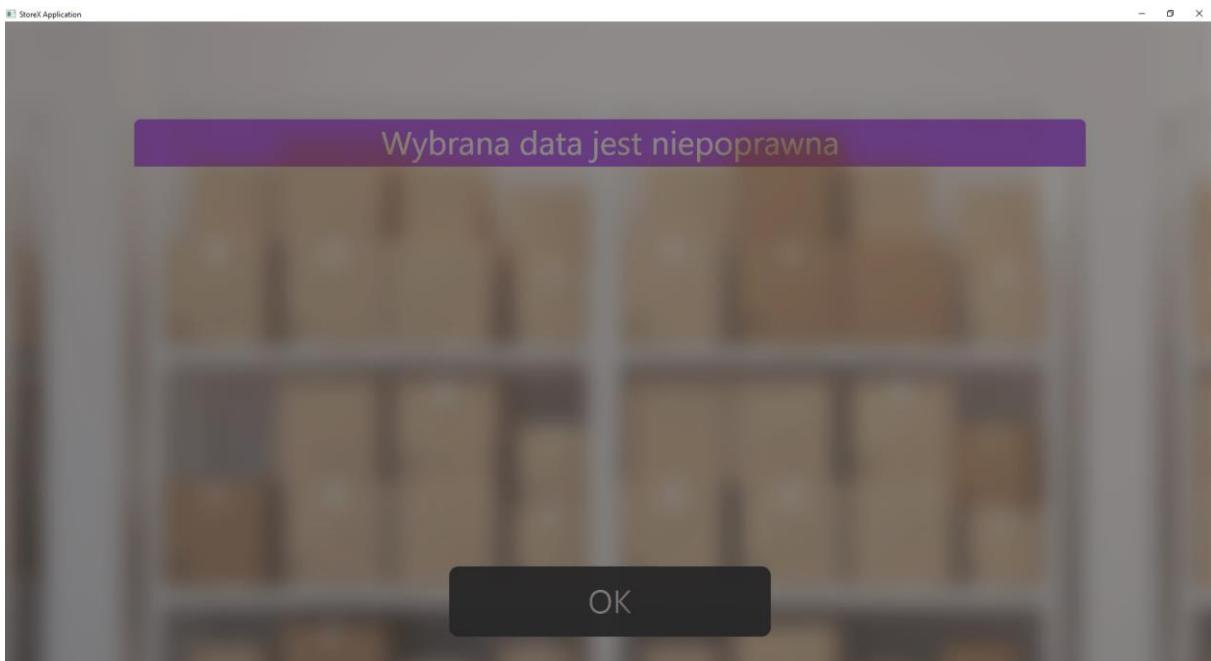
*Ekran potwierdzenia zapisu bilansu - implementacja*

Nie zanotowano większych różnic w implementacji ekranu akceptacji bilansu. Elementy tekstowe zmniejszone. Barwy mniej kontrastowe.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



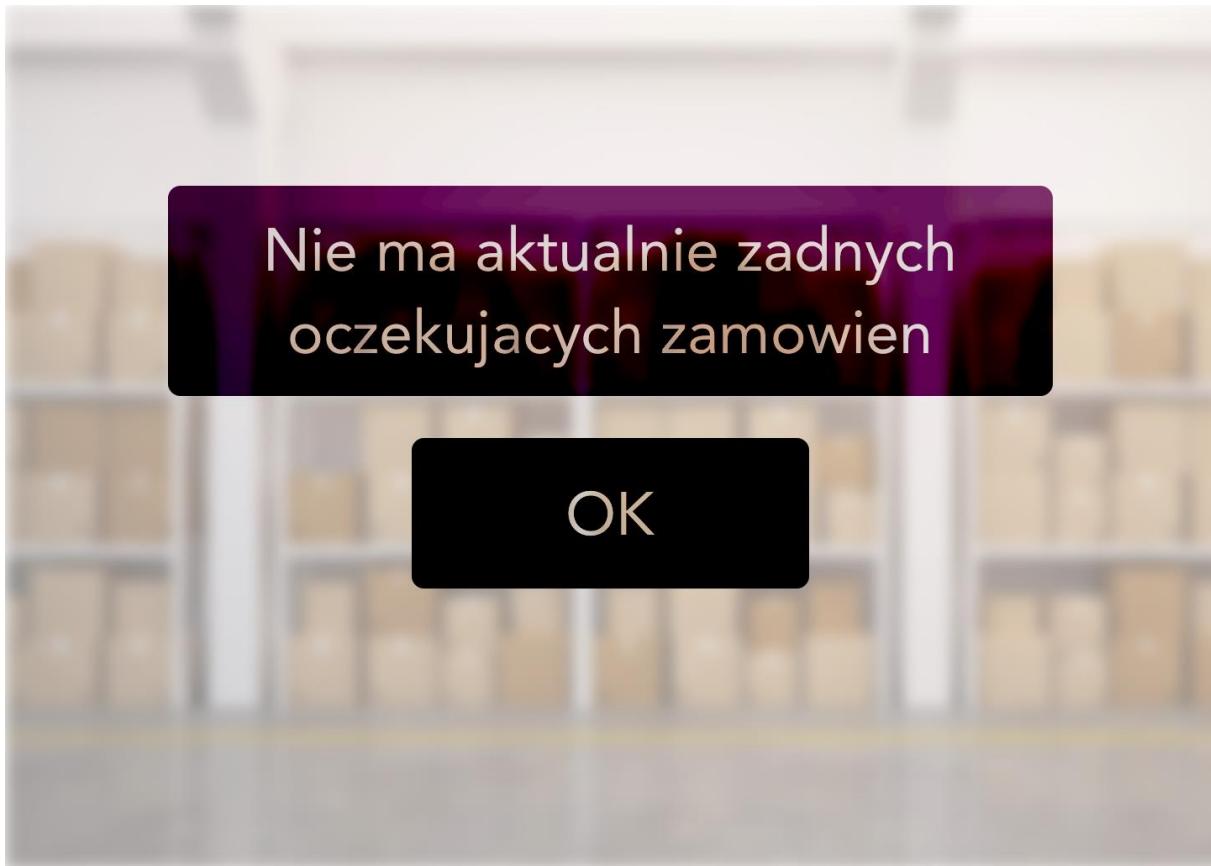
*Ekran informujący o zlej dacie*



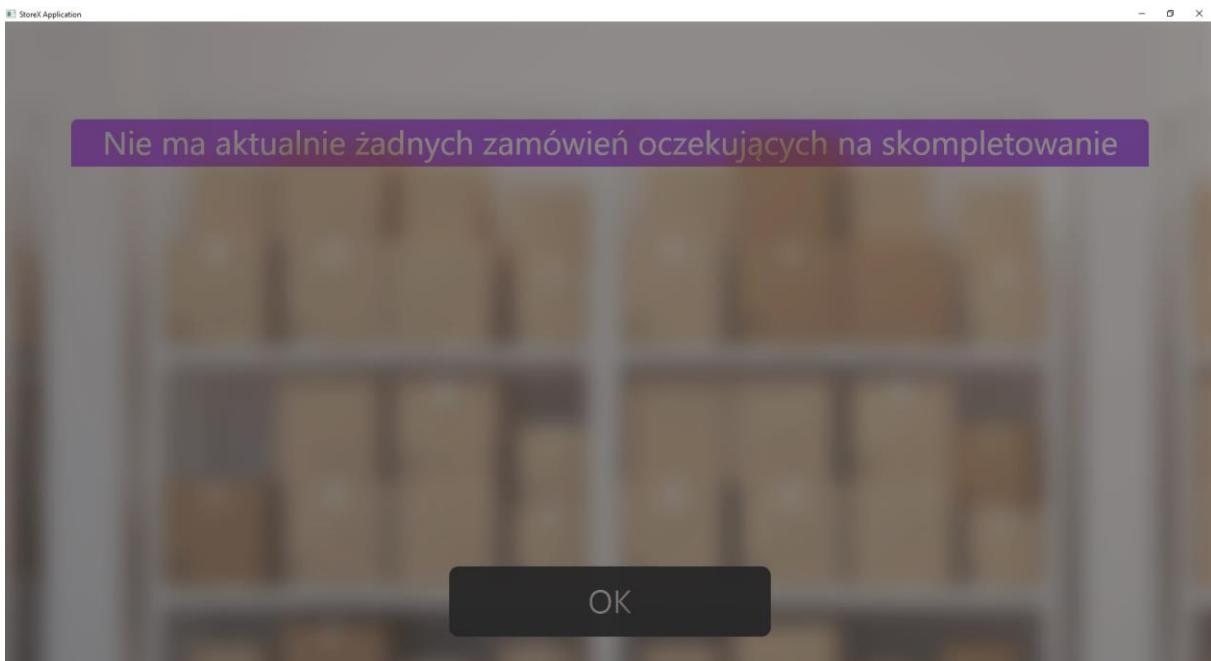
*Ekran informujący o zlej dacie – implementacja*

Nie zanotowano większych różnic w implementacji ekranu akceptacji błędnej daty. Elementy tekstowe zmniejszone. Barwy mniej kontrastowe.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



*Ekran informujący o braku oczekujących zamówień*



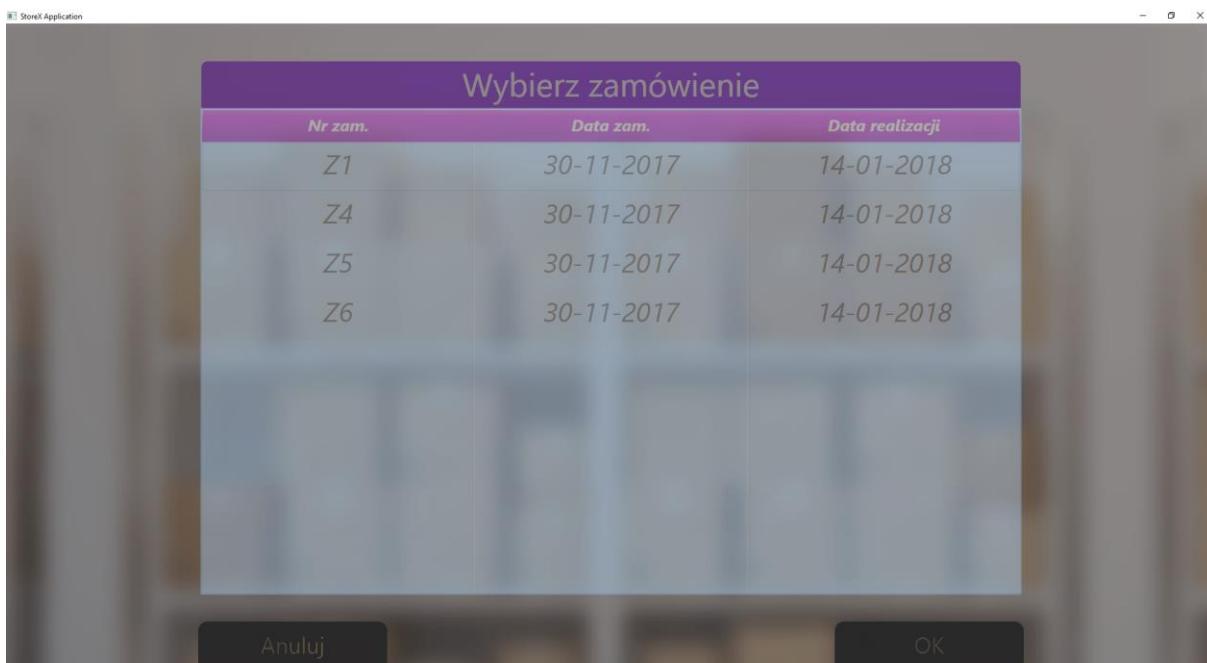
*Ekran informujący o braku oczekujących zamówień - implementacja*

Nie zanotowano większych różnic w implementacji ekranu braku oczekujących zamówień. Elementy tekstowe zmniejszone. Barwy mniej kontrastowe.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



*Ecran wyboru zamówienia*



*Ecran wyboru zamówienia - implementacja*

Nie zanotowano większych różnic w implementacji ekranu wyboru zamówienia. Barwy mniej kontrastowe.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

Zamowienie: FE4532 Termin realizacji: 15-12-2017

Wybierz produkt

Kod towaru	Nazwa Towaru	Ilosc w zam.	Zrealizowano
14134	Kosz Midi2	4	1
75431	Podstawka Rama	2	1
40610	Stol Firx	1	1
48663	Kaloryfer Z45	4	0
57914	Rama Obraz 45''	2	2

Przekaz do wydania      Wróć      OK

*Ekran wyboru produktu*

Zamówienie: Z1 Termin realizacji: 14-01-2018

Wybierz produkt

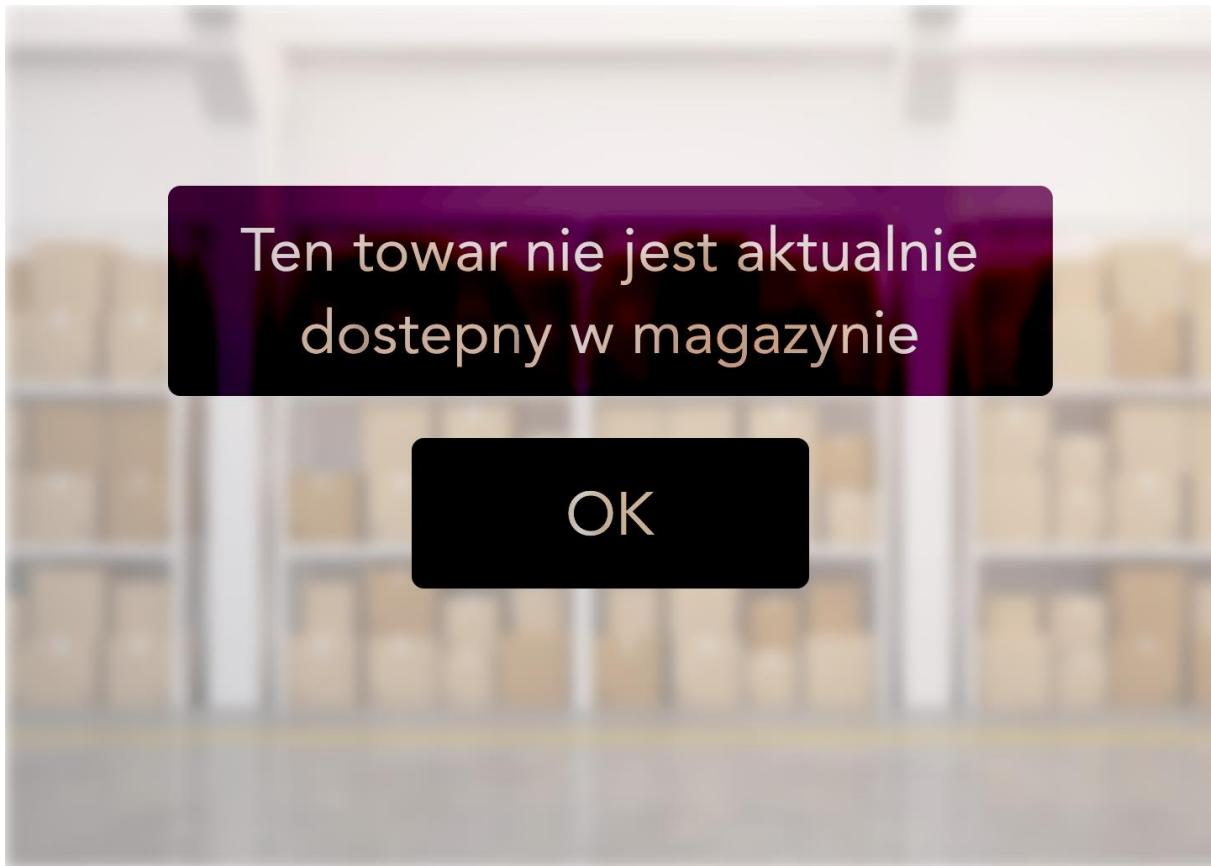
Kod towaru	Nazwa	Ilość w zam.	Zrealizowano
t1A48	Rama (sztuki)	20.0	0.0
t2B28	Kolo (sztuki)	40.0	0.0
t3A38	Dzwonek (sztuki)	10.0	0.0
t4A47	Korba (sztuki)	10.0	0.0
t5U48	Lancuch (sztuki)	20.0	0.0
t6J48	Pompka (sztuki)	10.0	0.0
t7A49	Siodelko (sztuki)	20.0	0.0
t8A50	Fotel (sztuki)	10.0	0.0
t9A48	Kierownica (sztuki)	10.0	0.0

Przekaz do wydania      Wróć      OK

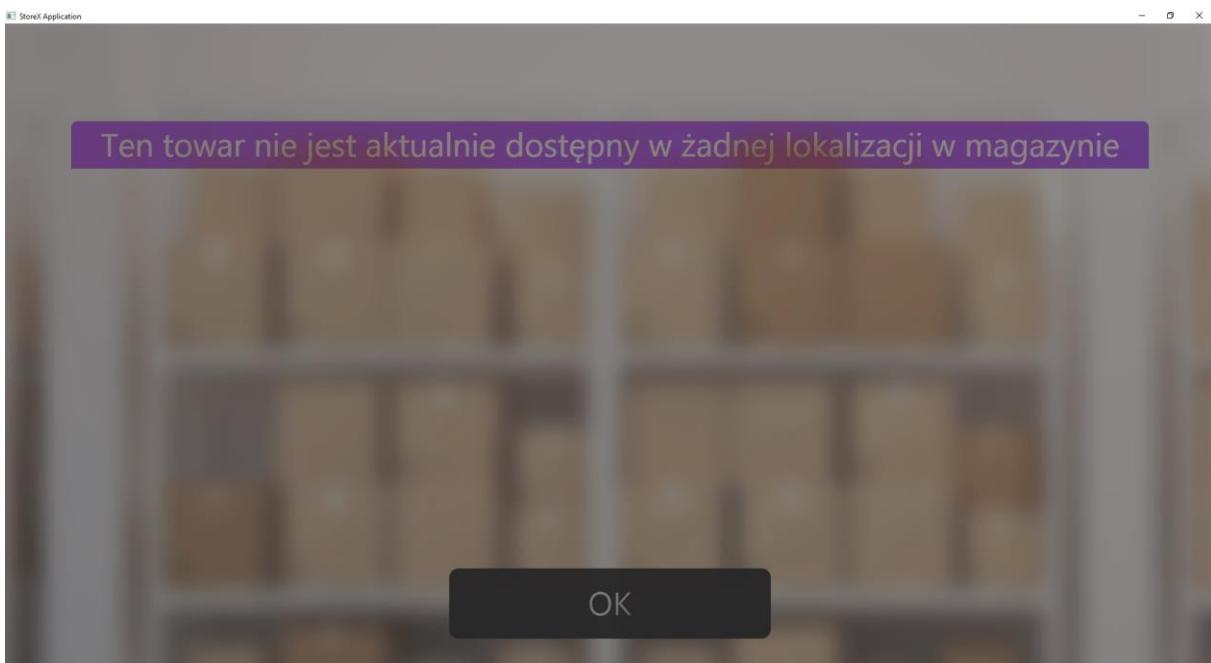
*Ekran wyboru produktu - implementacja*

Nie zanotowano większych różnic w implementacji ekranu wyboru produktu. Barwy mniej kontrastowe. Zlagodzono kontrastujący kolor czerwony w górnej części ekranu. Odciągał on wzrok i działał niekorzystnie na odbiór interfejsu. Ikona mapy została usunięta, gdyż funkcjonalność z nią związana nie została zaimplementowana.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



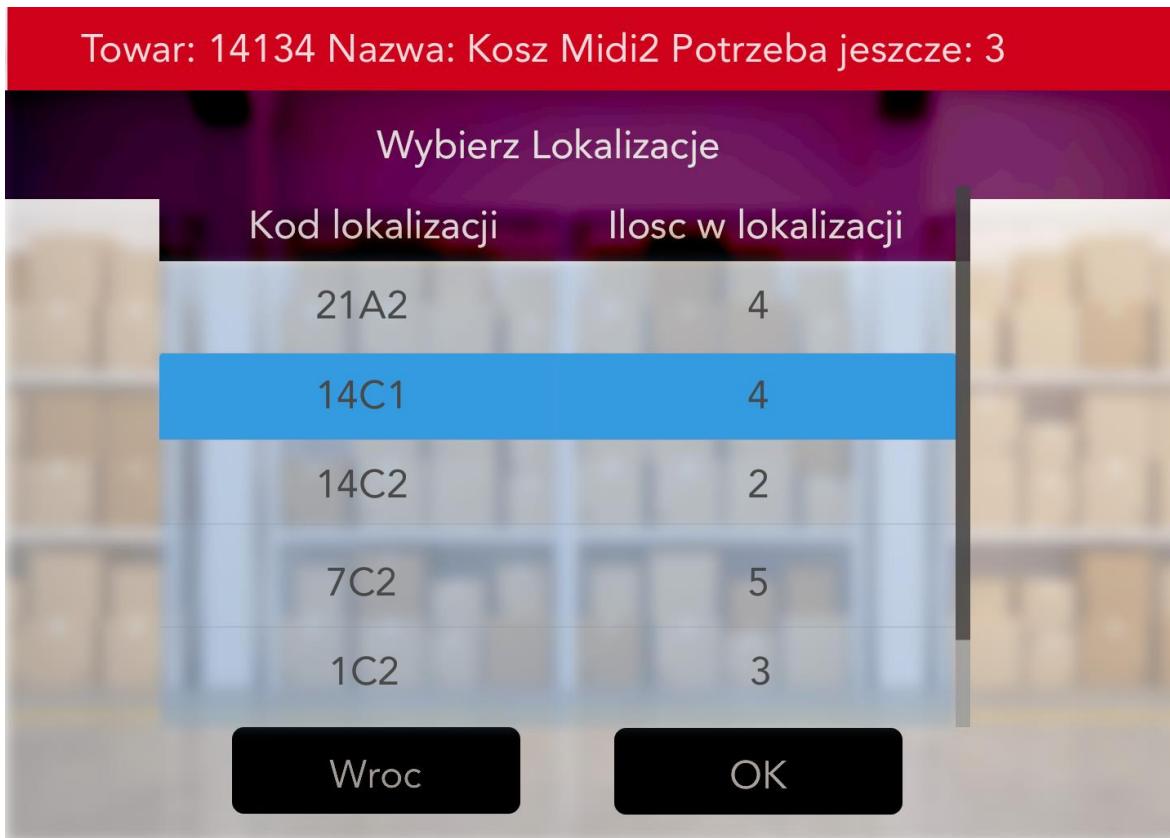
*Ekran informujący o braku danego towaru na stanie magazynu*



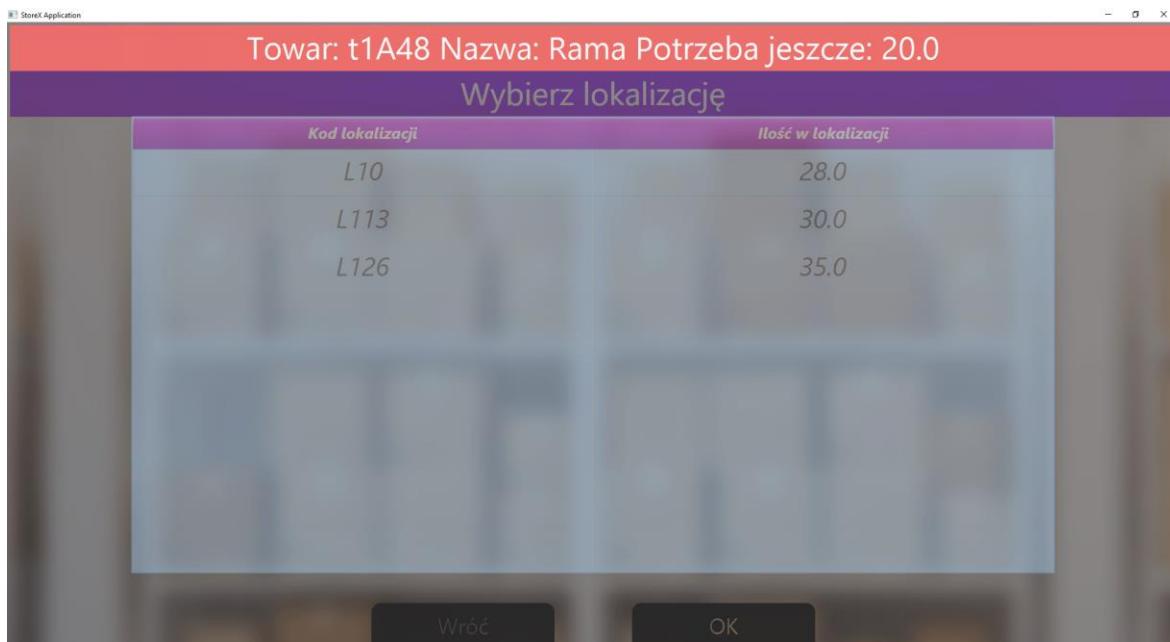
*Ekran informujący o braku danego towaru na stanie magazynu - implementacja*

Nie zanotowano większych różnic w implementacji ekranu wyboru produktu. Barwy mniej kontrastowe. Elementy tekstowe wyraźnie zmniejszone.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



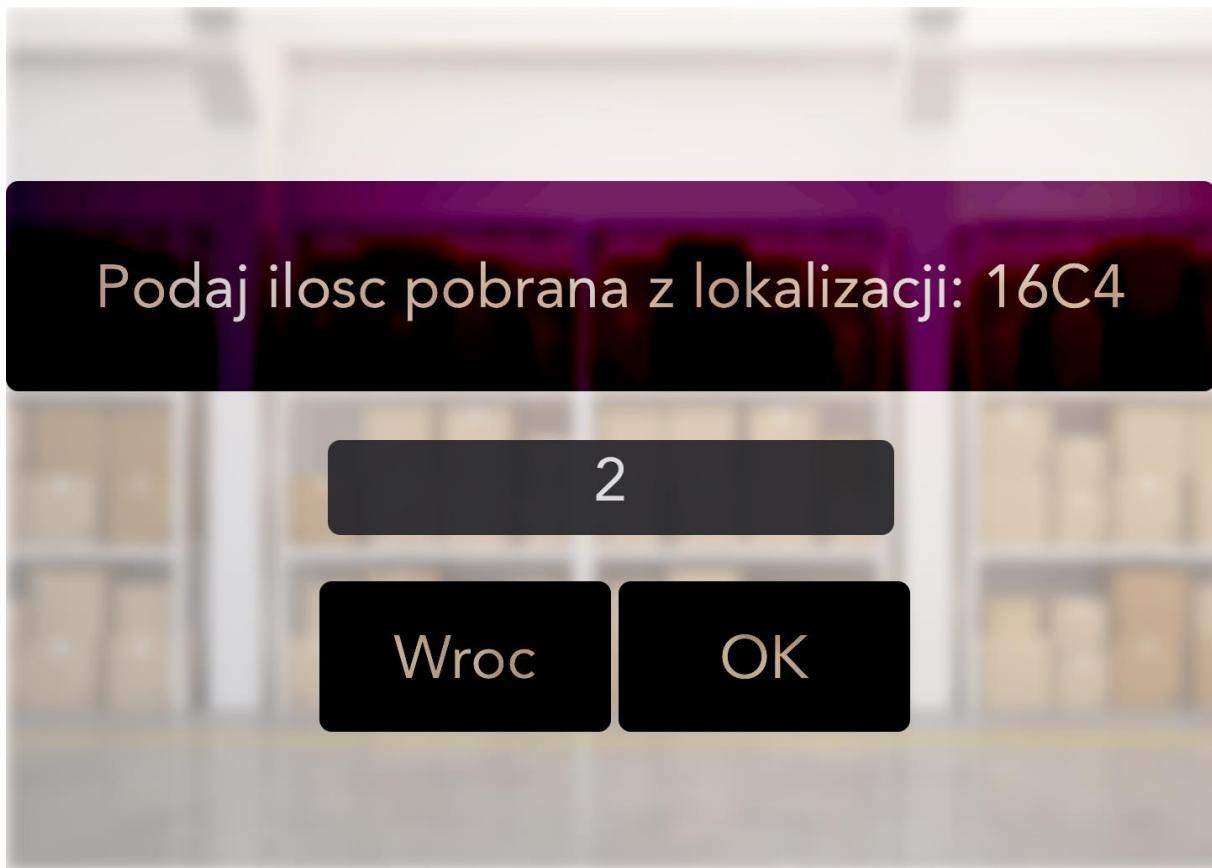
*Ekran wyboru lokalizacji*



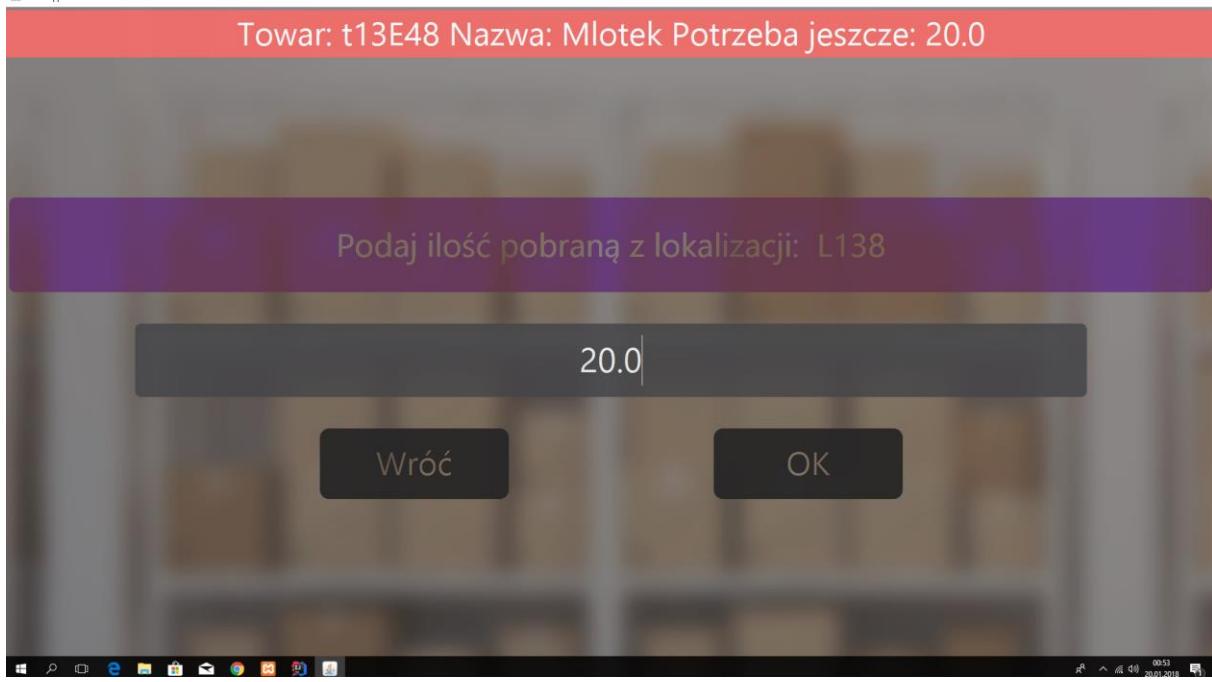
*Ekran wyboru lokalizacji - implementacja*

Nie zanotowano większych różnic w implementacji ekranu wyboru lokalizacji. Barwy mniej kontrastowe. Złagodzono kontrastujący kolor czerwony w górnej części ekranu. Odciągał on wzrok i działał niekorzystnie na odbiór interfejsu. Ikona mapy została usunięta, gdyż funkcjonalność z nią związana nie została zaimplementowana.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



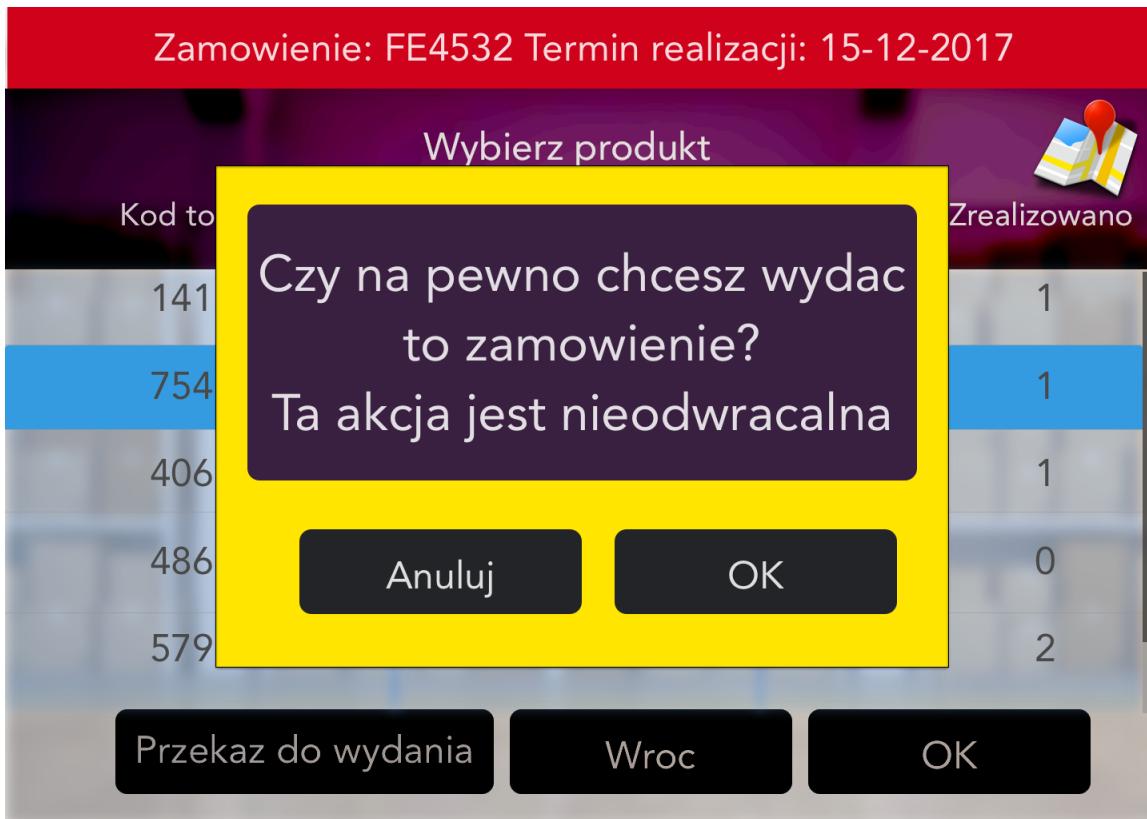
*Ekran wyboru pobranej ilości*



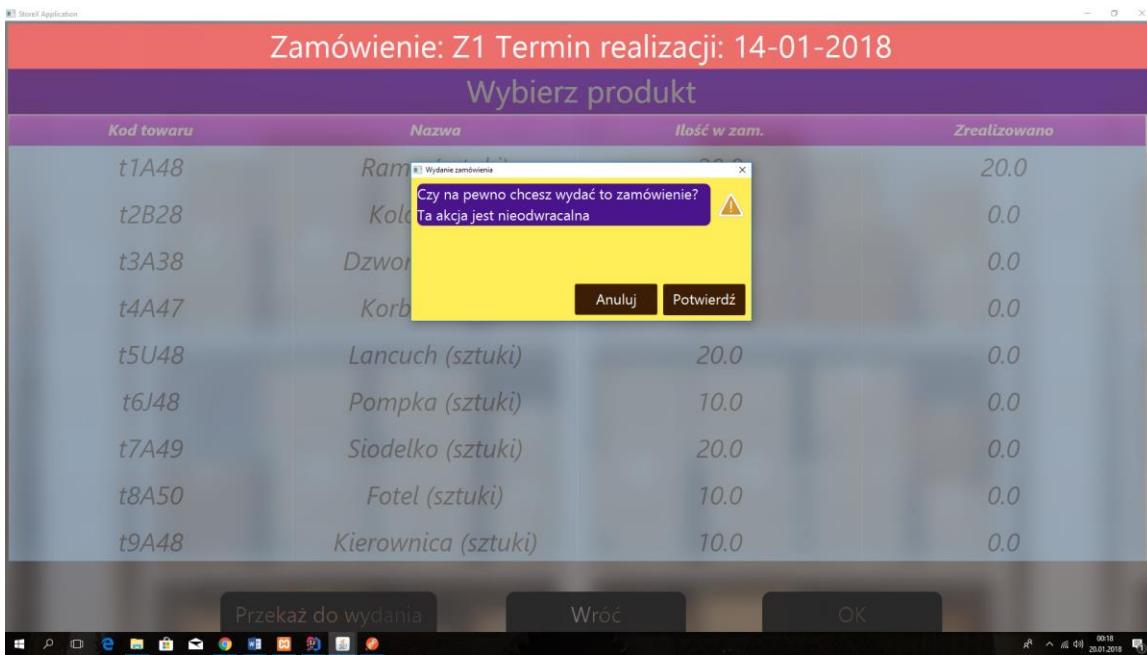
*Ekran wyboru pobranej ilości - implementacja*

Dodano dialog w górnej części ekranu. Potrzeba motywowana jest lepszym poinformowaniem użytkownika o aktualnym stanie realizowanego zamówienia.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



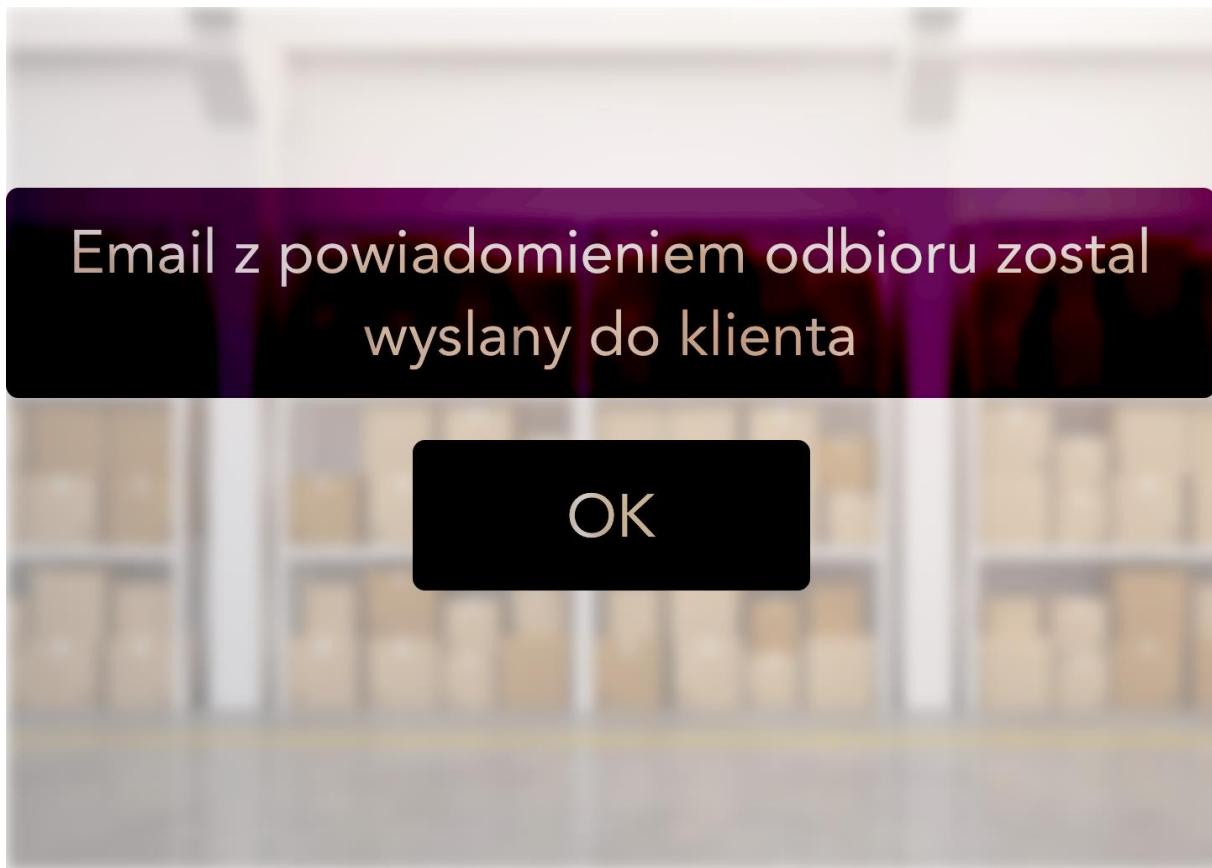
Alert wydania zamówienia



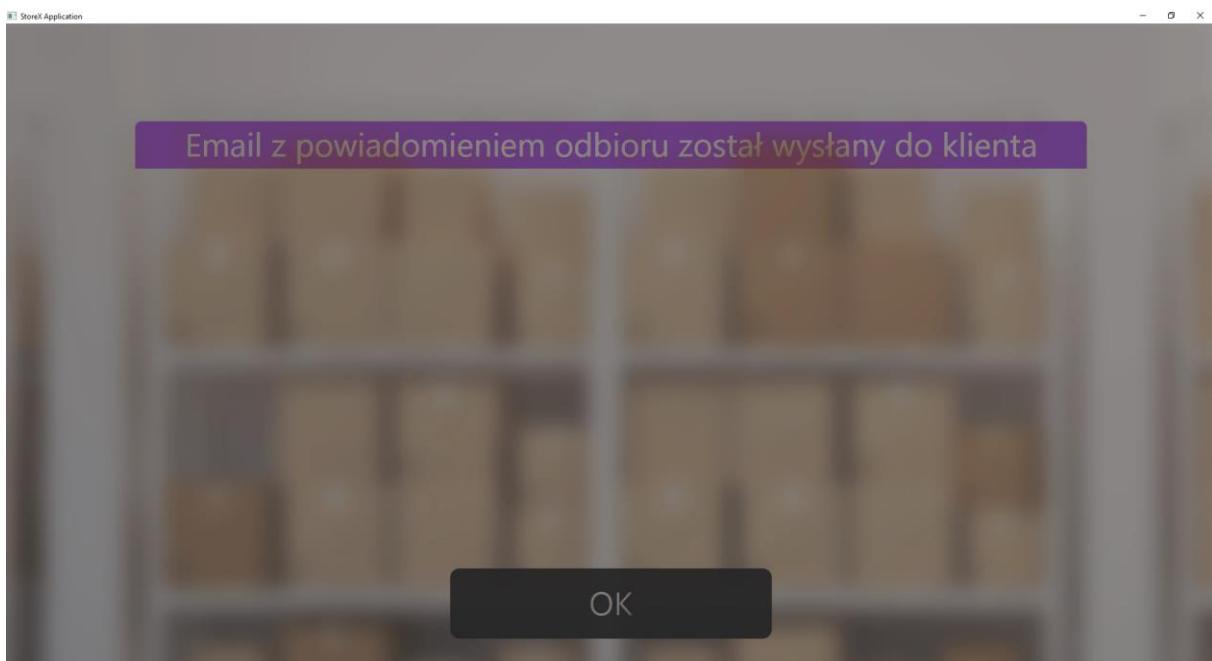
Alert wydania zamówienia - implementacja

Nie zanotowano większych różnic w implementacji alertu wydania. Barwy mniej kontrastowe. Zlagodzono kontrast w celu lepszej adaptacji wzroku do wyskakującego powiadomienia. Układ wyświetlanych elementów nieznacznie zmieniony. Sam alert – zajmuje znacznie mniejszą część ekranu.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



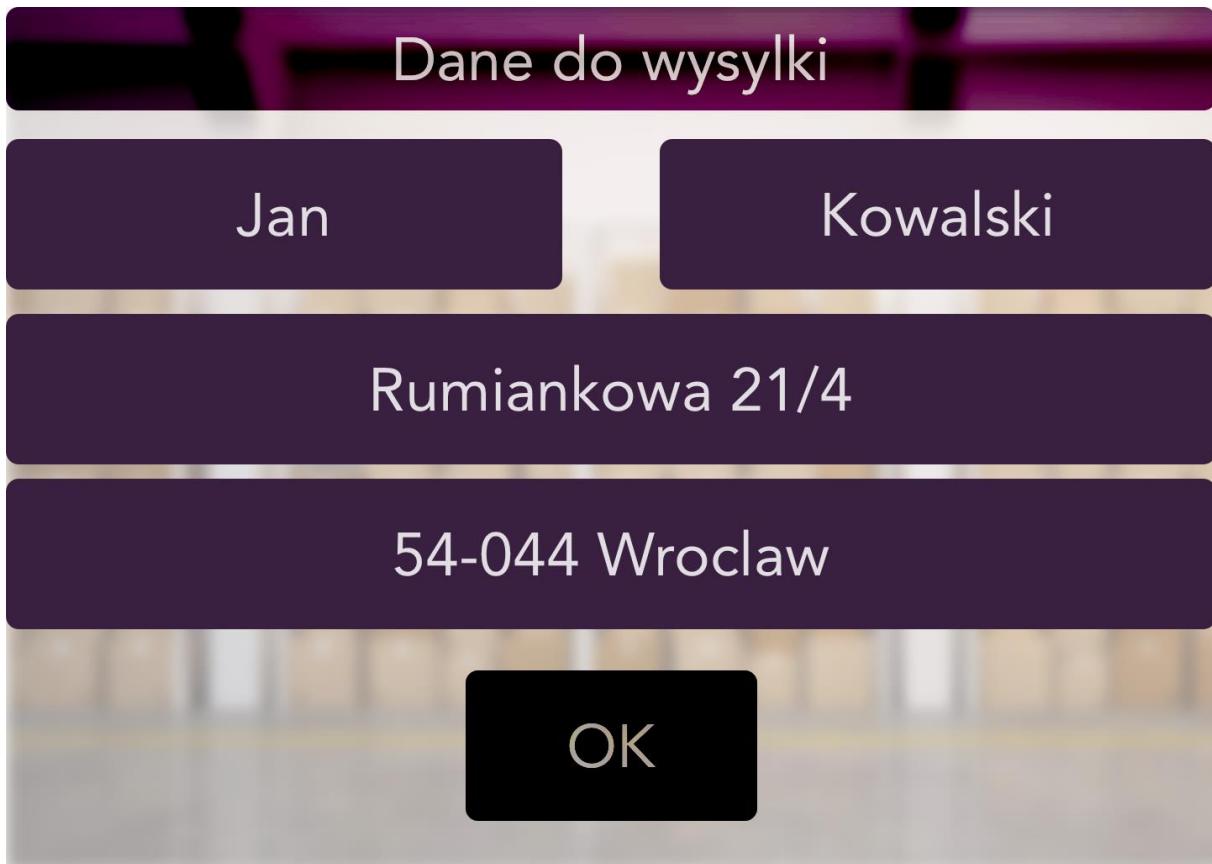
*Ekran powiadomienia o wysłanym emailu*



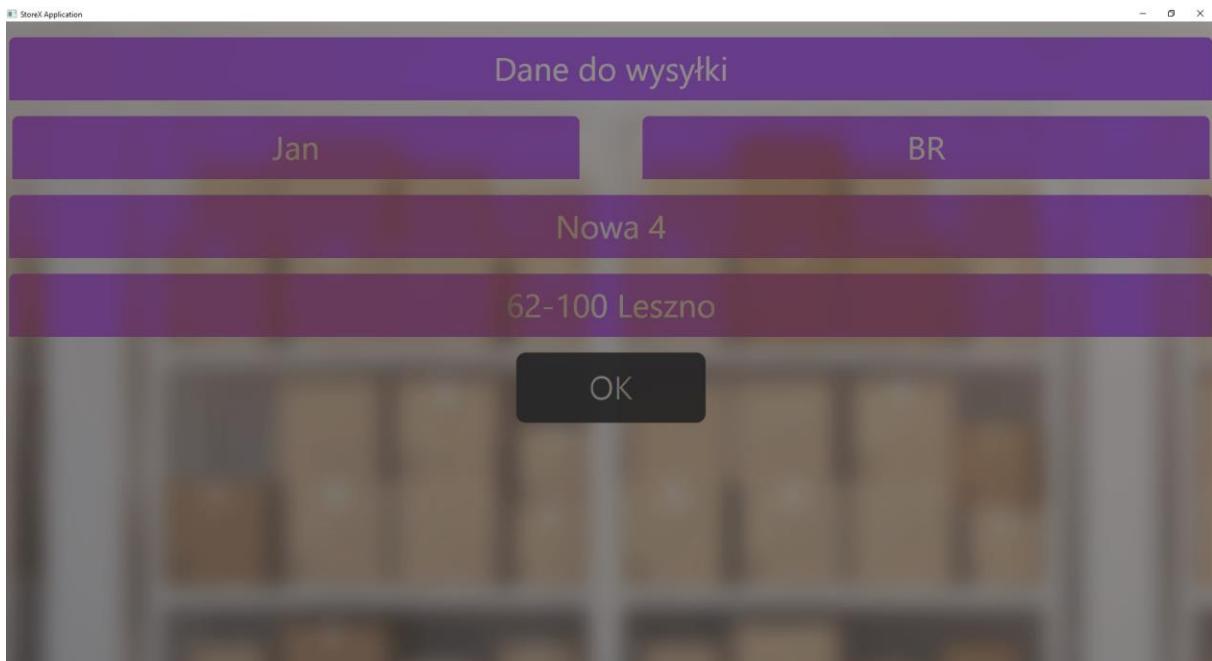
*Ekran powiadomienia o wysłanym emailu - implementacja*

Nie zanotowano większych różnic w implementacji ekranu braku oczekujących zamówień. Elementy tekstowe zmniejszone. Barwy mniej kontrastowe.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



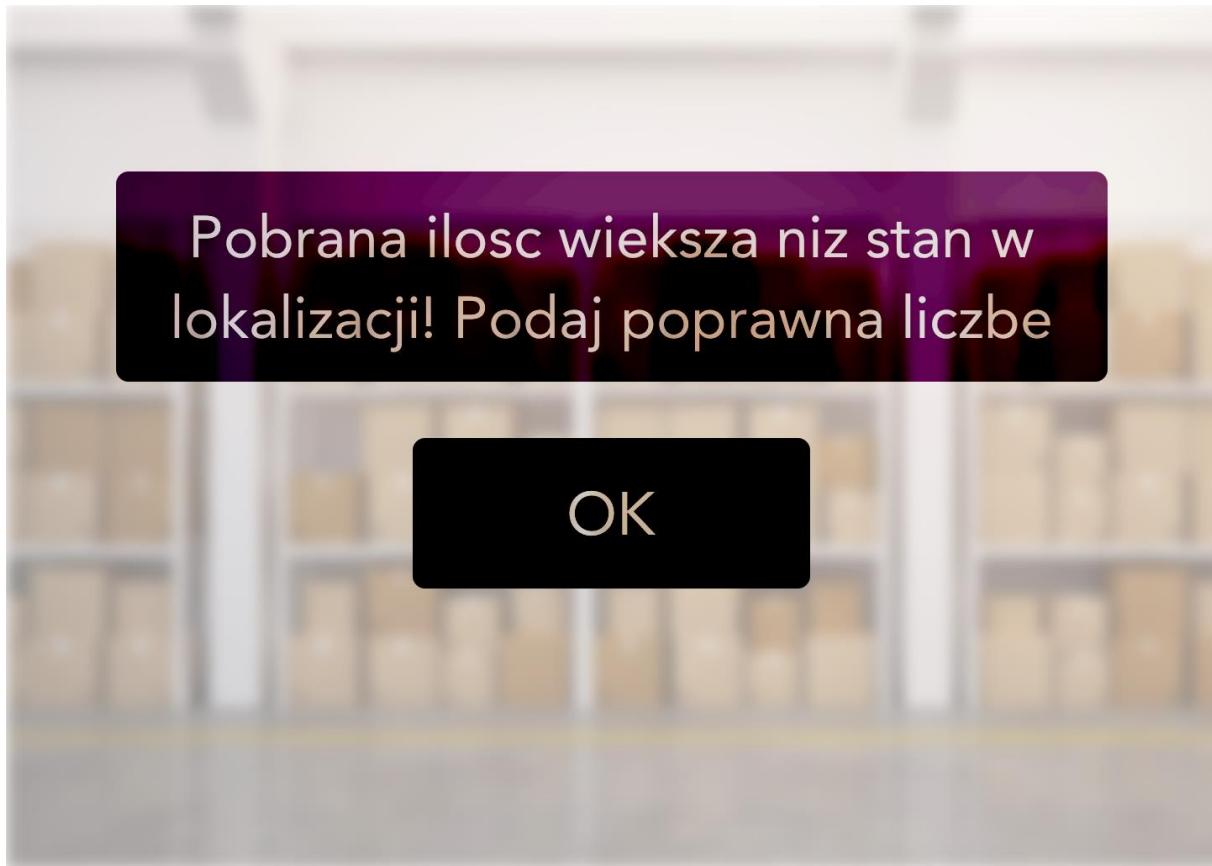
*Ekran z danymi do wysyłki*



*Ekran z danymi do wysyłki - implementacja*

Nie zanotowano większych różnic w implementacji ekranu braku oczekujących zamówień. Elementy tekstowe zmniejszone. Barwy mniej kontrastowe.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



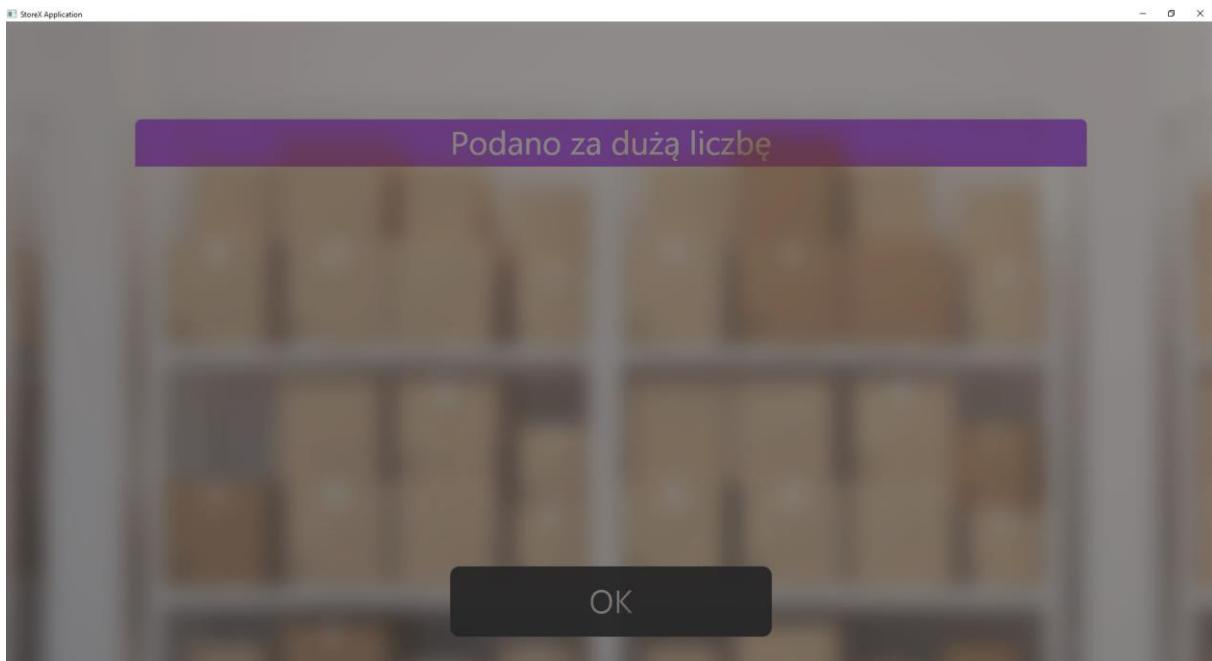
*Ekran błędu ilości*

Ekran wyboru niepoprawnej ilości został rozdzielony na ekran z różnymi komunikatami, tak aby informować użytkownika co dokładnie było przyczyną przerwania akcji kompletowania.

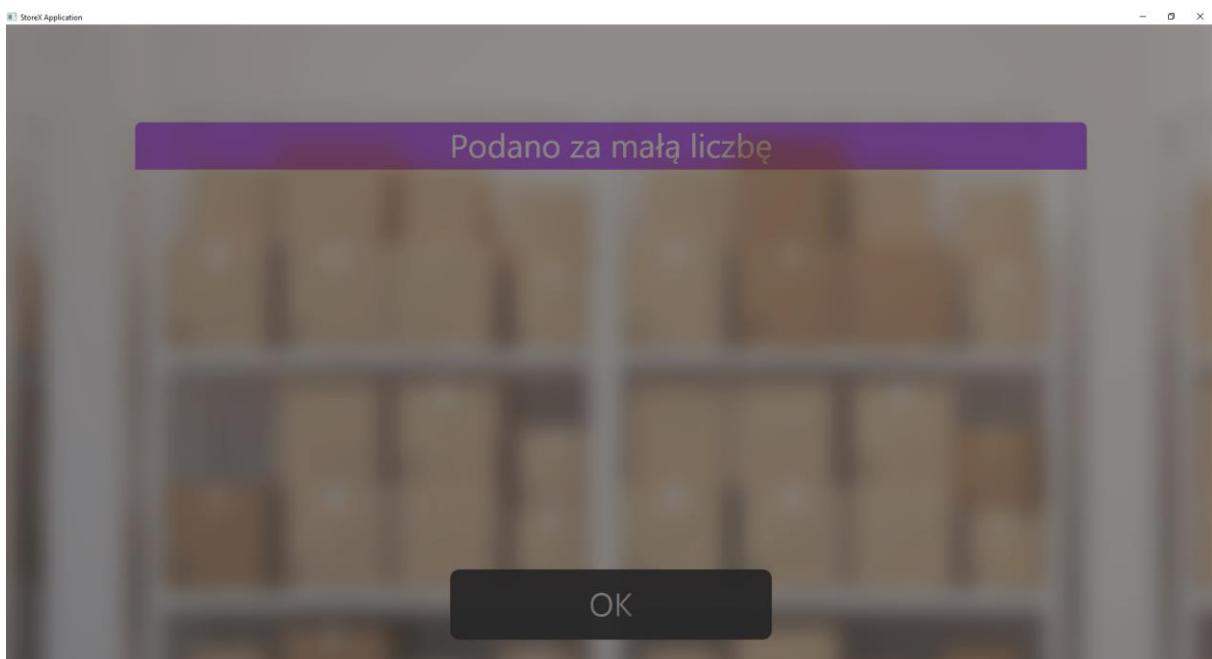
Wyróżniono następujące komunikaty:

- Podano za dużą ilość, komunikat odnoszący się do sytuacji w której pobrana ilość jest większa od ilości w danej lokalizacji lub większa od potrzebnej do zrealizowania.
- Podano za małą ilość, gdy podano liczbę mniejszą lub równą zero
- Podaj poprawną liczbę, gdy w pole tekstowe zostanie wprowadzony ciąg znaków nie będących liczbami.

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

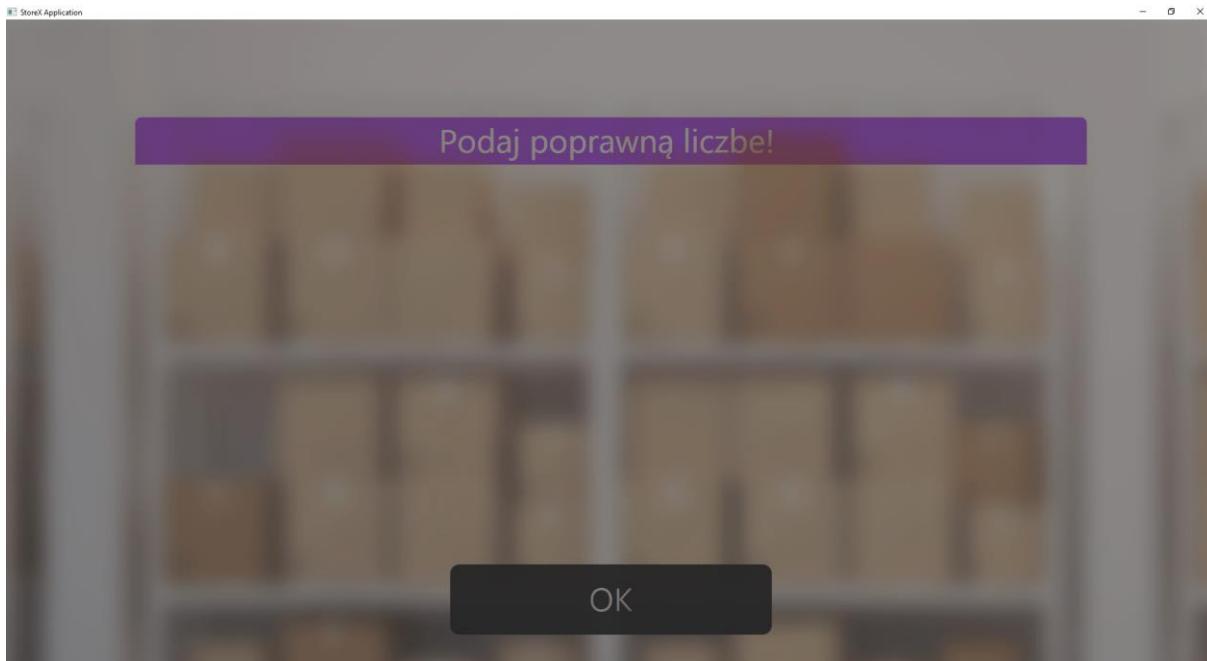


*Ekran informujący o zbyt dużej ilości pobranej - implementacja*



*Ekran informujący o za małej ilości pobranej - implementacja*

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



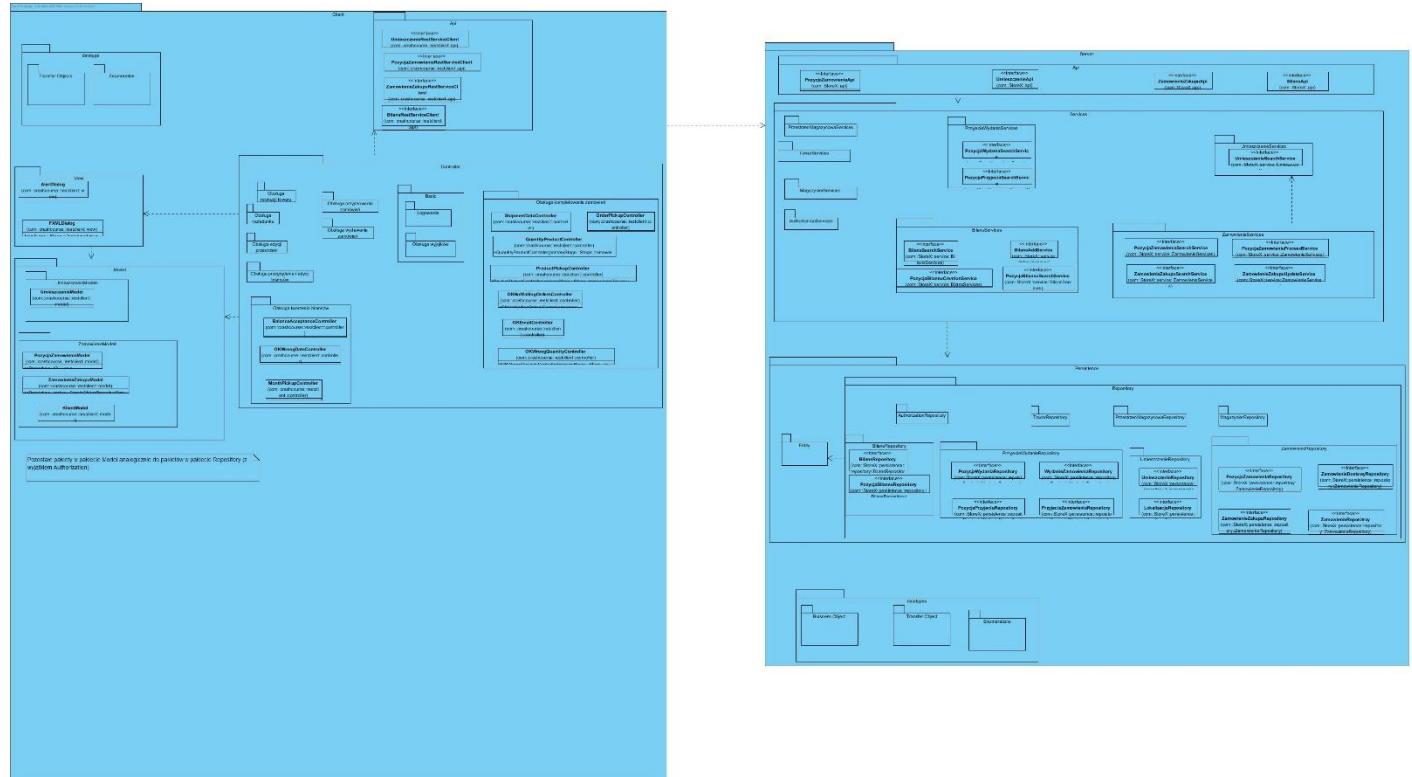
*Ekran informujący o niepoprawnie podanej liczbie - implementacja*

### 9.1 Podsumowanie

Ogólna funkcjonalność interfejsu została spełniona, należy jednak zastanowić się nad bardziej kontrastową szatą graficzną. Część poczynionych zmian nie wniosła oczekiwanej poprawy jakości interfejsu, niektóre z nich doprowadziły nawet do wizualnego pogorszenia odbioru oprogramowania. Duże części ekranu pozostają niewykorzystane – sprawiają, że interfejs wygląda na niekompletny i niedokończony. Ogólna ocena funkcjonalna jest pozytywna, jednak w kategoriach estetycznych nie zostały spełnione stawiane cele.

## 10. Architektura systemu

### 10.1 Architektura logiczna (diagram pakietów)



Aplikacja jest podzielona na instancję klienta i serwera. Jest to najczęściej stosowany wzorzec architektoniczny. Po stronie serwera realizowane jest połączenie z bazą danych oraz logika biznesowa. Po stronie klienta natomiast następuje interakcja z użytkownikiem oraz prezentacja danych.

Obiekty danych zostały podzielone na następujące: encje, obiekty transferowe, obiekty biznesowe oraz obiekty modelowe. Obiekty encji są bezpośrednią reprezentacją bazy, dostęp do nich zapewniony jest przez warstwę dostępu do danych – repository. Obiekty biznesowe – odpowiedzialne za logikę biznesową aplikacji, wszystkie operacje w serwisach są wykonywane na tych obiektach. Obiekty transferowe – używane do komunikacji między klientem a serwerem. Obiekty modelowe, używane bezpośrednio podczas prezentacji danych w kliencie.

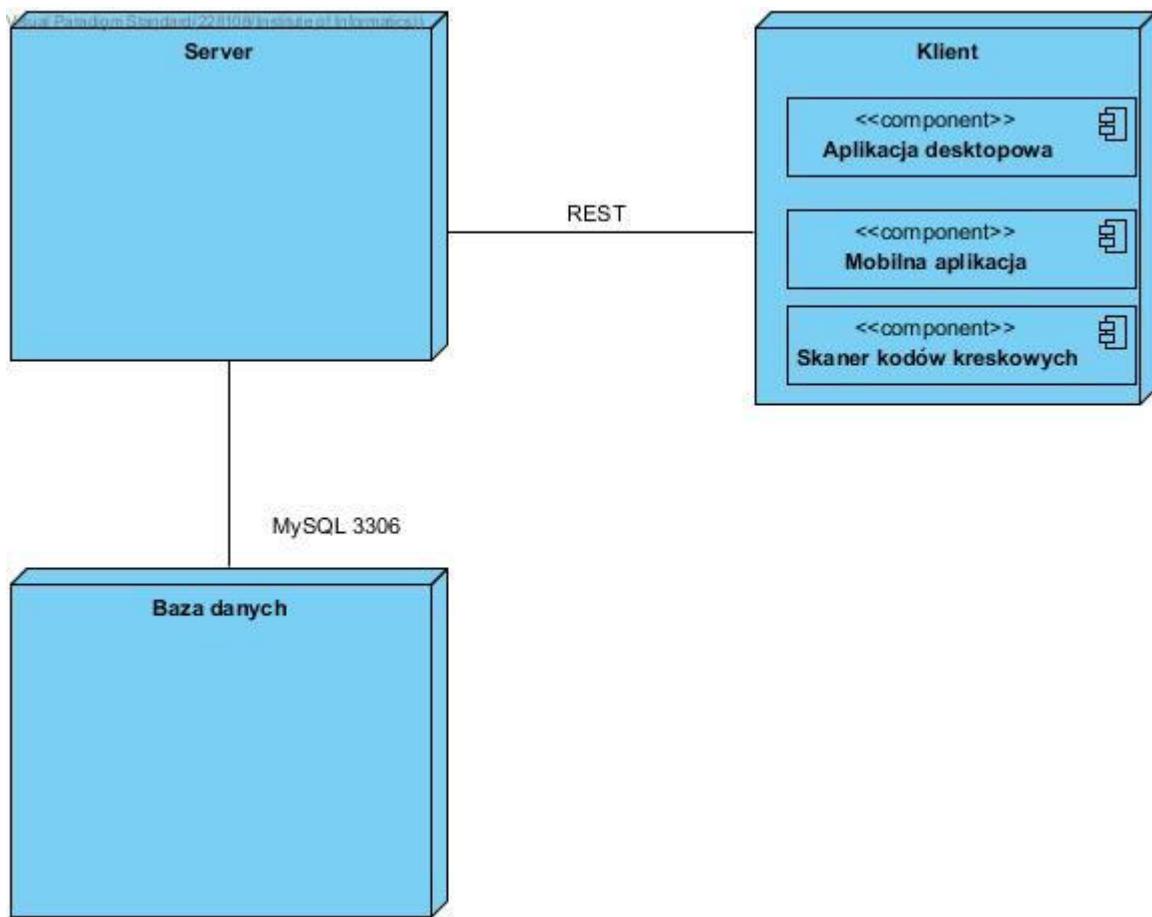
Zapytania REST korzystają z API Sewera, które po swojej stronie ma odwzorowanie w klienckich metodach API.

Po stronie klienta realizowany jest wzorzec architektoniczny MVC, separujący klasy widoku, modelu oraz wyróżniający klasę kontrolera tychże.

Warto nadmienić, że po stronie serwera została użycia architektura mikroserwisów. Realizująca atomowe i niepodzielne zadania.

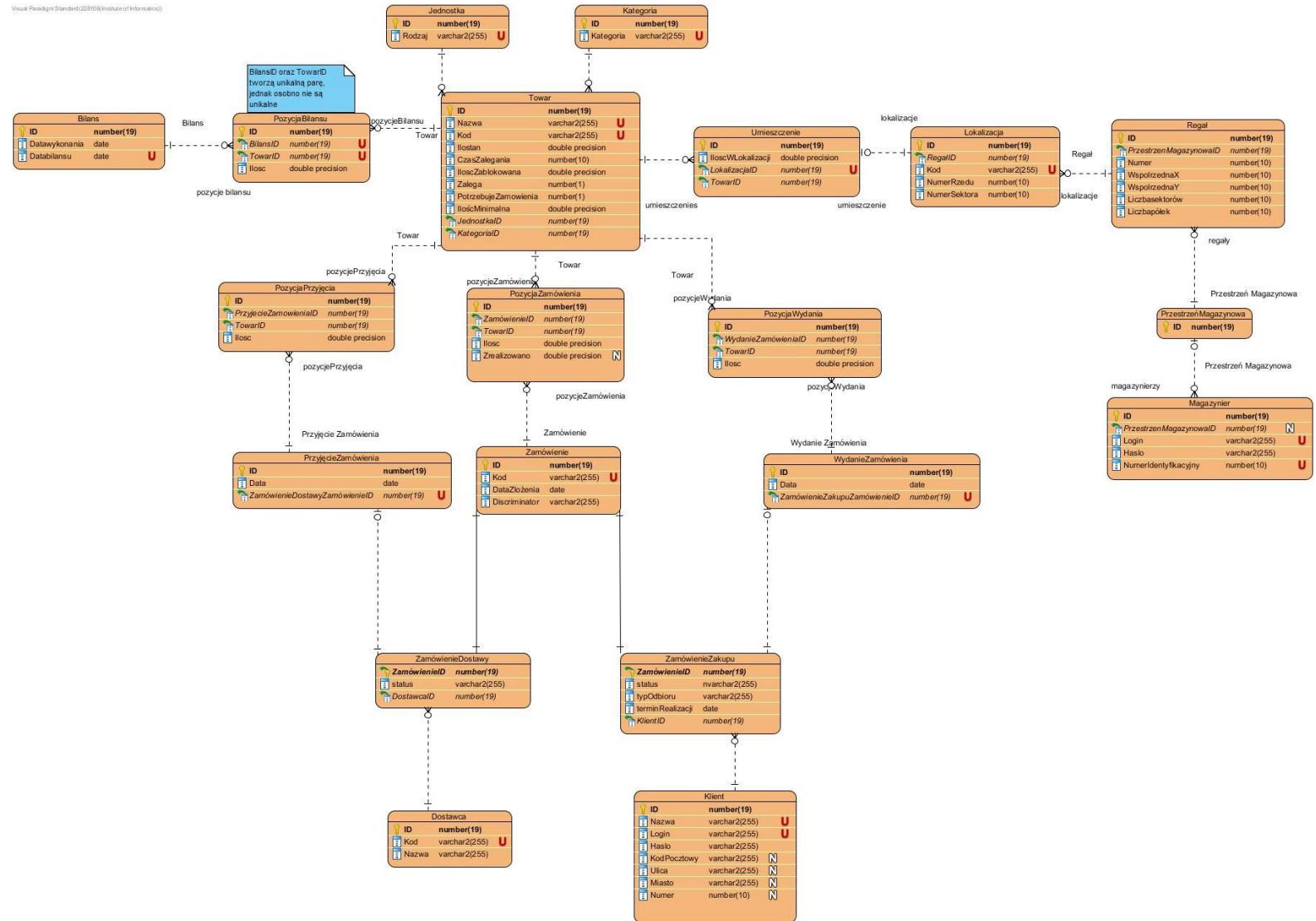
<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 10.2 Architektura fizyczna (diagram rozmieszczenia)



## 11. Projekt bazy danych

### 11.1 Diagram Erd



<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 11.2 Wygenerowany skrypt

### Database.ddl

#### 11.2.1 Tworzenie tabel

```
CREATE TABLE Towar (ID bigint(19) AUTO_INCREMENT NOT NULL, Nazwa varchar(255) NOT NULL UNIQUE, Kod varchar(255) NOT NULL UNIQUE, Ilostan double NOT NULL, CzasZalegania int(10) NOT NULL, IloscZablokowana double NOT NULL, Zalega bit(1) NOT NULL, PotrzebujeZamowienia bit(1) NOT NULL, IloscMinimalna double NOT NULL, JednostkaID bigint(19) NOT NULL, KategoriaID bigint(19) NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE Lokalizacja (ID bigint(19) AUTO_INCREMENT NOT NULL, RegalID bigint(19) NOT NULL, Kod varchar(255) NOT NULL UNIQUE, NumerRzedu int(10) NOT NULL, NumerSektora int(10) NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE Dostawca (ID bigint(19) AUTO_INCREMENT NOT NULL, Kod varchar(255) NOT NULL UNIQUE, Nazwa varchar(255) NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE Klient (ID bigint(19) AUTO_INCREMENT NOT NULL, Nazwa varchar(255) NOT NULL UNIQUE, Login varchar(255) NOT NULL UNIQUE, Haslo varchar(255) NOT NULL, KodPoczty varchar(255), Ulica varchar(255), Miasto varchar(255), Numer int(10), PRIMARY KEY (ID));
```

```
CREATE TABLE Umieszczenie (ID bigint(19) AUTO_INCREMENT NOT NULL, IloscWLocalizacji double NOT NULL, LocalizacjaID bigint(19) NOT NULL UNIQUE, TowarID bigint(19) NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE Bilans (ID bigint(19) AUTO_INCREMENT NOT NULL, DataWykonania date NOT NULL, DataBilansu date NOT NULL UNIQUE, PRIMARY KEY (ID));
```

```
CREATE TABLE Regal (ID bigint(19) AUTO_INCREMENT NOT NULL, PrzestrzenMagazynowaID bigint(19) NOT NULL, Numer int(10) NOT NULL, WspolrzednaX int(10) NOT NULL, WspolrzednaY int(10) NOT NULL, LiczbaSektorow int(10) NOT NULL, LiczbaPolek int(10) NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE PrzestrzenMagazynowa (ID bigint(19) AUTO_INCREMENT NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE PozycjaBilansu (ID bigint(19) AUTO_INCREMENT NOT NULL, BilansID bigint(19) NOT NULL, TowarID bigint(19) NOT NULL, Ilosc double NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE PozycjaZamowienia (ID bigint(19) AUTO_INCREMENT NOT NULL, ZamowienieID bigint(19) NOT NULL, TowarID bigint(19) NOT NULL, Ilosc double NOT NULL, Zrealizowano double, PRIMARY KEY (ID));
```

```
CREATE TABLE WydanieZamowienia (ID bigint(19) AUTO_INCREMENT NOT NULL, Data date NOT NULL, ZamowienieZakupuZamowienieID bigint(19) UNIQUE, PRIMARY KEY (ID));
```

```
CREATE TABLE PozycjaPrzyjecia (ID bigint(19) AUTO_INCREMENT NOT NULL, PrzyjecieZamowieniaID bigint(19) NOT NULL, TowarID bigint(19) NOT NULL, Ilosc double NOT NULL, PRIMARY KEY (ID));
```

```
CREATE TABLE PozycjaWydania (ID bigint(19) AUTO_INCREMENT NOT NULL, WydanieZamowieniaID bigint(19) NOT NULL, TowarID bigint(19) NOT NULL, Ilosc double NOT NULL, PRIMARY KEY (ID));
```

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

CREATE TABLE Magazynier (ID bigint(19) AUTO\_INCREMENT NOT NULL, PrzestrzenMagazynowaID bigint(19), Login varchar(255) NOT NULL UNIQUE, Haslo varchar(255) NOT NULL, NumerIdentyfikacyjny int(10) NOT NULL UNIQUE, PRIMARY KEY (ID));

CREATE TABLE ZamowienieDostawy (ZamowienieID bigint(19) NOT NULL, DostawcaID bigint(19) NOT NULL, status varchar(255) NOT NULL, PRIMARY KEY (ZamowienieID));

CREATE TABLE ZamowienieZakupu (ZamowienieID bigint(19) NOT NULL, KlientID bigint(19) NOT NULL, status varchar(255) NOT NULL, typOdbioru varchar(255) NOT NULL, terminRealizacji date NOT NULL, PRIMARY KEY (ZamowienieID));

CREATE TABLE Jednostka (ID bigint(19) AUTO\_INCREMENT NOT NULL, Rodzaj varchar(255) NOT NULL UNIQUE, PRIMARY KEY (ID));

CREATE TABLE Kategoria (ID bigint(19) AUTO\_INCREMENT NOT NULL, Kategoria varchar(255) NOT NULL UNIQUE, PRIMARY KEY (ID));

CREATE TABLE Zamowienie (ID bigint(19) AUTO\_INCREMENT NOT NULL, Kod varchar(255) NOT NULL UNIQUE, DataZlozenia date NOT NULL, Discriminator varchar(255) NOT NULL, PRIMARY KEY (ID));

CREATE TABLE PrzyjcieZamowienia (ID bigint(19) AUTO\_INCREMENT NOT NULL, Data date NOT NULL, ZamowienieDostawyZamowienieID bigint(19) UNIQUE, PRIMARY KEY (ID));

#### 11.2.2 Dodawnie ograniczeń (constraints)

ALTER TABLE WydanieZamowienia ADD CONSTRAINT WZ\_FK\_ZZZ FOREIGN KEY (ZamowienieZakupuZamowienieID) REFERENCES ZamowienieZakupu (ZamowienieID);

ALTER TABLE PrzyjcieZamowienia ADD CONSTRAINT PZ\_FK\_ZDZ FOREIGN KEY (ZamowienieDostawyZamowienieID) REFERENCES ZamowienieDostawy (ZamowienieID);

ALTER TABLE Bilans ADD CONSTRAINT BI\_CH\_DWY\_DB1 CHECK (DataWykonania > DataBilansu);

ALTER TABLE Klient ADD CONSTRAINT KL\_CH\_NUM CHECK (Numer >= 0);

ALTER TABLE Magazynier ADD CONSTRAINT MA\_FK\_PRZ FOREIGN KEY (PrzestrzenMagazynowaID) REFERENCES PrzestrzenMagazynowa (ID);

ALTER TABLE Magazynier ADD CONSTRAINT MA\_CH\_NUM CHECK (NumerIdentyfikacyjny >= 0);

ALTER TABLE ZamowienieZakupu ADD CONSTRAINT ZZ\_FK\_ZAM FOREIGN KEY (ZamowienieID) REFERENCES Zamowienie (ID);

ALTER TABLE ZamowienieZakupu ADD CONSTRAINT ZZ\_FK\_KLI FOREIGN KEY (KlientID) REFERENCES Klient (ID);

ALTER TABLE ZamowienieDostawy ADD CONSTRAINT ZD\_FK\_ZAM FOREIGN KEY (ZamowienieID) REFERENCES Zamowienie (ID);

ALTER TABLE ZamowienieDostawy ADD CONSTRAINT ZD\_FK\_DOS FOREIGN KEY (DostawcaID) REFERENCES Dostawca (ID);

ALTER TABLE PozycjaZamowienia ADD CONSTRAINT PZ\_FK\_TOW FOREIGN KEY (TowarID) REFERENCES Towar (ID);

ALTER TABLE PozycjaZamowienia ADD CONSTRAINT PZ\_FK\_ZAM FOREIGN KEY (ZamowienieID) REFERENCES Zamowienie (ID);

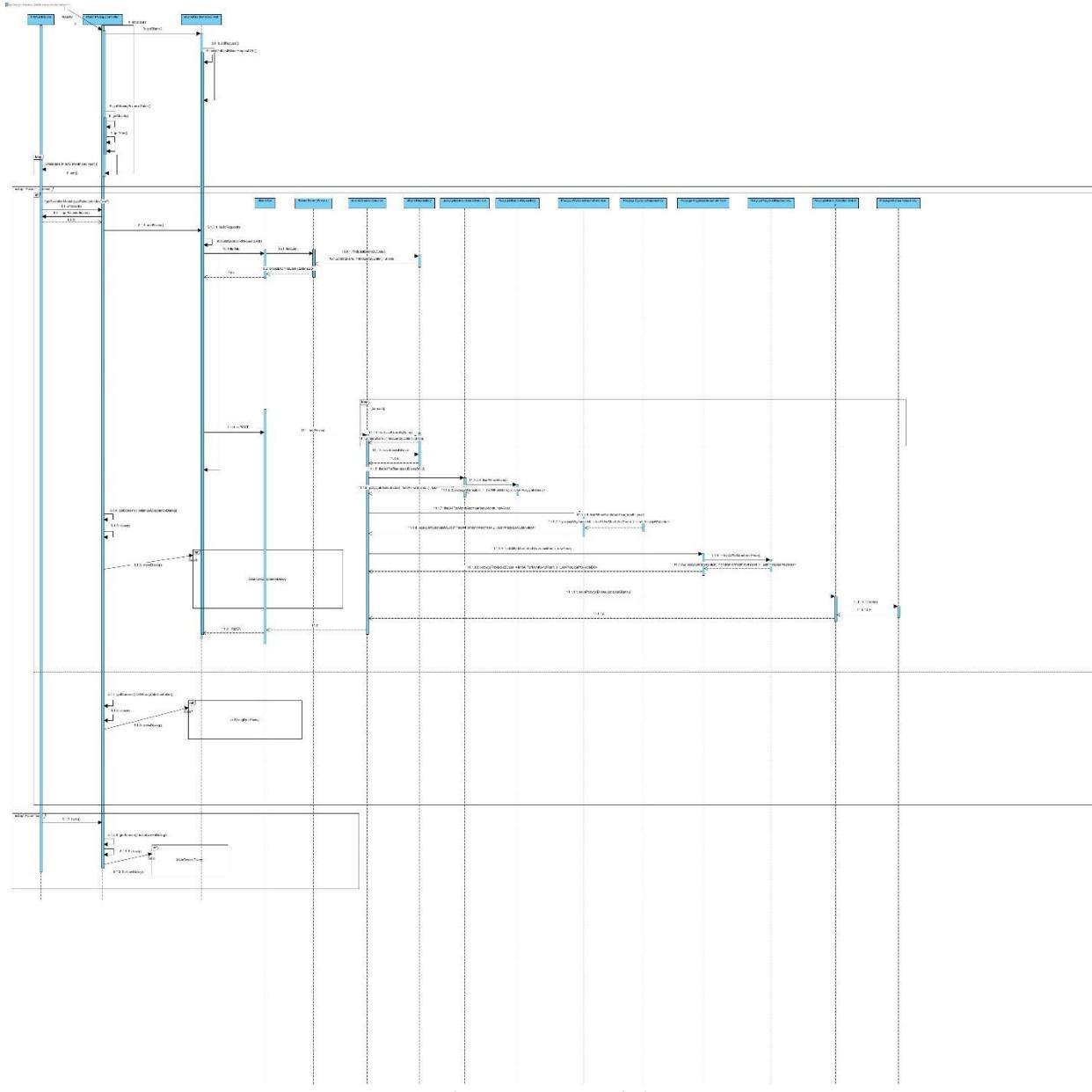
<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ALTER TABLE PozycjaZamowienia ADD CONSTRAINT PZ\_CH\_ILO CHECK (Ilosc > 0);  
 ALTER TABLE PozycjaZamowienia ADD CONSTRAINT PZ\_CH\_ZRE CHECK (Zrealizowano >= 0);  
 ALTER TABLE PozycjaZamowienia ADD CONSTRAINT PZ\_CH\_ZRE\_ILO CHECK (Zrealizowano <= Ilosc);  
  
 ALTER TABLE PozycjaWydania ADD CONSTRAINT PW\_FK\_TOW FOREIGN KEY (TowarID)  
 REFERENCES Towar (ID);  
 ALTER TABLE PozycjaWydania ADD CONSTRAINT PW\_FK\_WZA FOREIGN KEY (WydanieZamowieniaID)  
 REFERENCES WydanieZamowienia (ID);  
 ALTER TABLE PozycjaWydania ADD CONSTRAINT PW\_CH\_ILO CHECK (Ilosc > 0);  
  
 ALTER TABLE PozycjaPrzyjecia ADD CONSTRAINT PP\_FK\_TOW FOREIGN KEY (TowarID) REFERENCES  
 Towar (ID);  
 ALTER TABLE PozycjaPrzyjecia ADD CONSTRAINT PP\_FK\_PZA FOREIGN KEY (PrzyjcieZamowieniaID)  
 REFERENCES PrzyjcieZamowienia (ID);  
 ALTER TABLE PozycjaPrzyjecia ADD CONSTRAINT PP\_CH\_ILO CHECK (Ilosc > 0);  
  
 ALTER TABLE PozycjaBilansu ADD CONSTRAINT PB\_FK\_TOW FOREIGN KEY (TowarID) REFERENCES  
 Towar (ID);  
 ALTER TABLE PozycjaBilansu ADD CONSTRAINT PB\_FK\_BIL FOREIGN KEY (BilansID) REFERENCES  
 Bilans (ID);  
 ALTER TABLE PozycjaBilansu ADD CONSTRAINT PB\_CH\_ILO CHECK (Ilosc >= 0);  
 ALTER TABLE PozycjaBilansu ADD CONSTRAINT PB\_UN\_BIL\_TOW UNIQUE (BilansID, TowarID);  
  
 ALTER TABLE Umieszczenie ADD CONSTRAINT UM\_FK\_LOK FOREIGN KEY (LokalizacjaID)  
 REFERENCES Lokalizacja (ID);  
 ALTER TABLE Umieszczenie ADD CONSTRAINT UM\_FK\_TOW FOREIGN KEY (TowarID) REFERENCES  
 Towar (ID);  
 ALTER TABLE Umieszczenie ADD CONSTRAINT UM\_CH\_ILO CHECK (IloscWLokalizacji >= 0);  
  
 ALTER TABLE Towar ADD CONSTRAINT TO\_FK\_JED FOREIGN KEY (JednostkaID) REFERENCES  
 Jednostka (ID);  
 ALTER TABLE Towar ADD CONSTRAINT TO\_FK\_KAT FOREIGN KEY (KategoriaID) REFERENCES  
 Kategoria (ID);  
 ALTER TABLE Towar ADD CONSTRAINT TO\_CH\_ILO CHECK (Ilostan >= 0);  
 ALTER TABLE Towar ADD CONSTRAINT TO\_CH\_ILO\_IZA CHECK (Ilostan >= IloscZablokowana);  
 ALTER TABLE Towar ADD CONSTRAINT TO\_CH\_CZA CHECK (CzasZalegania > 0);  
 ALTER TABLE Towar ADD CONSTRAINT TO\_CH\_MIN CHECK (IloscMinimalna >= 0);  
  
 ALTER TABLE Regal ADD CONSTRAINT RE\_FK\_PRZ FOREIGN KEY (PrzestrzenMagazynowaID)  
 REFERENCES PrzestrzenMagazynowa (ID);  
 ALTER TABLE Regal ADD CONSTRAINT RE\_CH\_LSE CHECK (LiczbasSektorow > 0);  
 ALTER TABLE Regal ADD CONSTRAINT RE\_CH\_LPO CHECK (LiczbaPolek > 0);  
 ALTER TABLE Regal ADD CONSTRAINT RE\_CH\_NUM CHECK (Numer > 0);  
  
 ALTER TABLE Lokalizacja ADD CONSTRAINT LO\_FK\_REG FOREIGN KEY (RegalID) REFERENCES Regal  
 (ID);  
 ALTER TABLE Lokalizacja ADD CONSTRAINT LO\_CH\_NSE CHECK (NumerSektora > 0);  
 ALTER TABLE Lokalizacja ADD CONSTRAINT LO\_CH\_NRZ CHECK (NumerRzedu > 0);

Storex	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

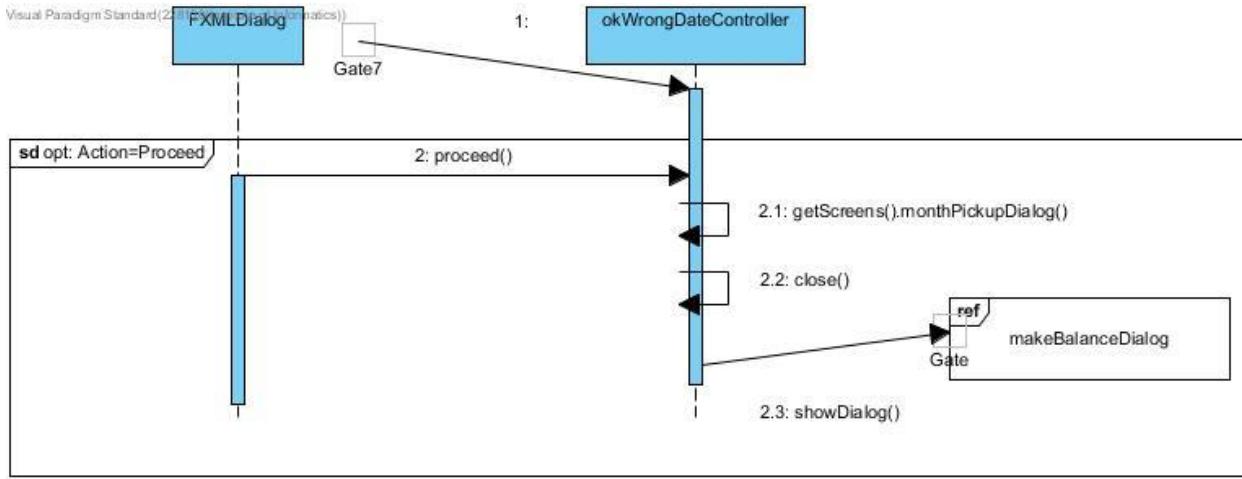
## 12. Realizacja przypadków użycia

### 12.1 Bilansowanie



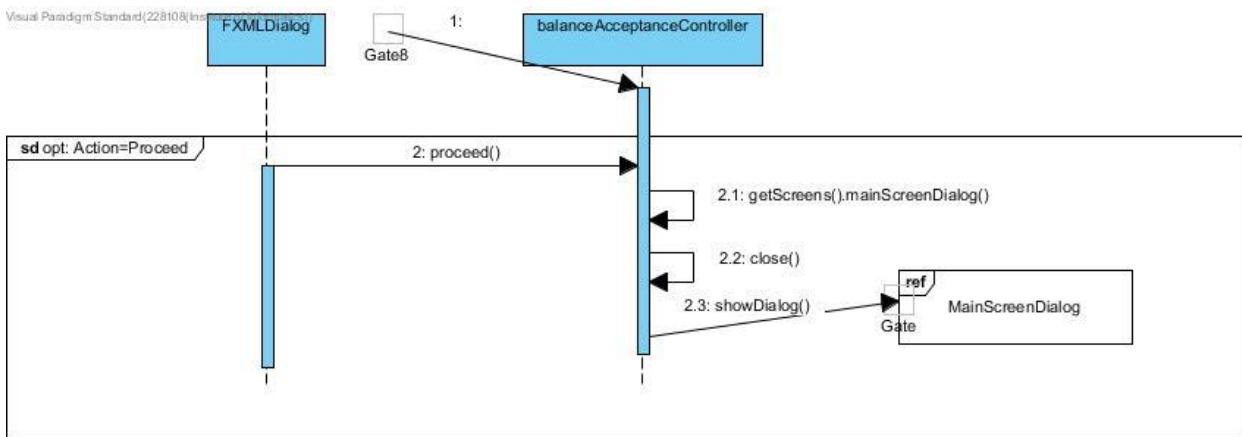
Przy inicjalizacji dialogu wywoływanie jest żądanie do serwera. Pobierany jest ostatni bilans z bazy. Następnie kontroler oblicza brakujące miesiące bilansów i wyświetla ich daty na ekranie. Przy kliknięciu anuluj, dialog jest zamknięty i następuje powrót do ekranu głównego. W przypadku wybrania przycisku OK następuje procedura sprawdzania poprawności daty. Jeśli data nie jest pierwszą możliwą bilansowaną, następuje wyświetlenie dialogu informującego o złej dacie. Natomiast jeśli wybrana data jest poprawna następuje przekazanie jej do serwera. Na serwerze następuje wyszukanie wszystkich pozycji poprzedniego bilansu oraz wszystkich pozycji wydań i przyjęć w okresie do zbilansowania. Obliczane są nowe pozycje dla bilansu - jeśli nie było ich w poprzednim bilansie to wyrażone jest to wzorem (PP-PW) w przeciwnym wypadku (PPB+PP-PW), gdzie PW – pozycja wydania, PP – pozycja przyjęcia, PPB – pozycja poprzedniego bilansu. Bilans i jego pozycje są zapisywane w bazie i zwracany jest komunikat HTTP:OK.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



*Ecran akceptacji zlej daty*

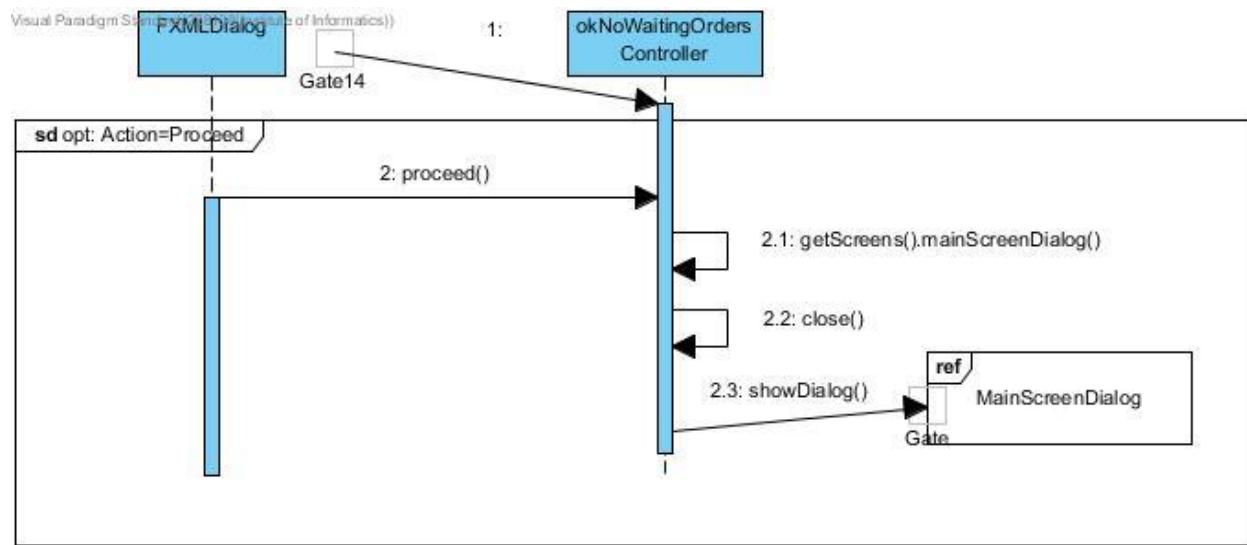
Przy kliknięciu przycisku na ekranie wywoływana jest metoda `proceed()`, zamyka ona aktualny dialog i przechodzi do ekranu wyboru daty dialogu.



*Ecran potwierdzenia zapisu bilansu*

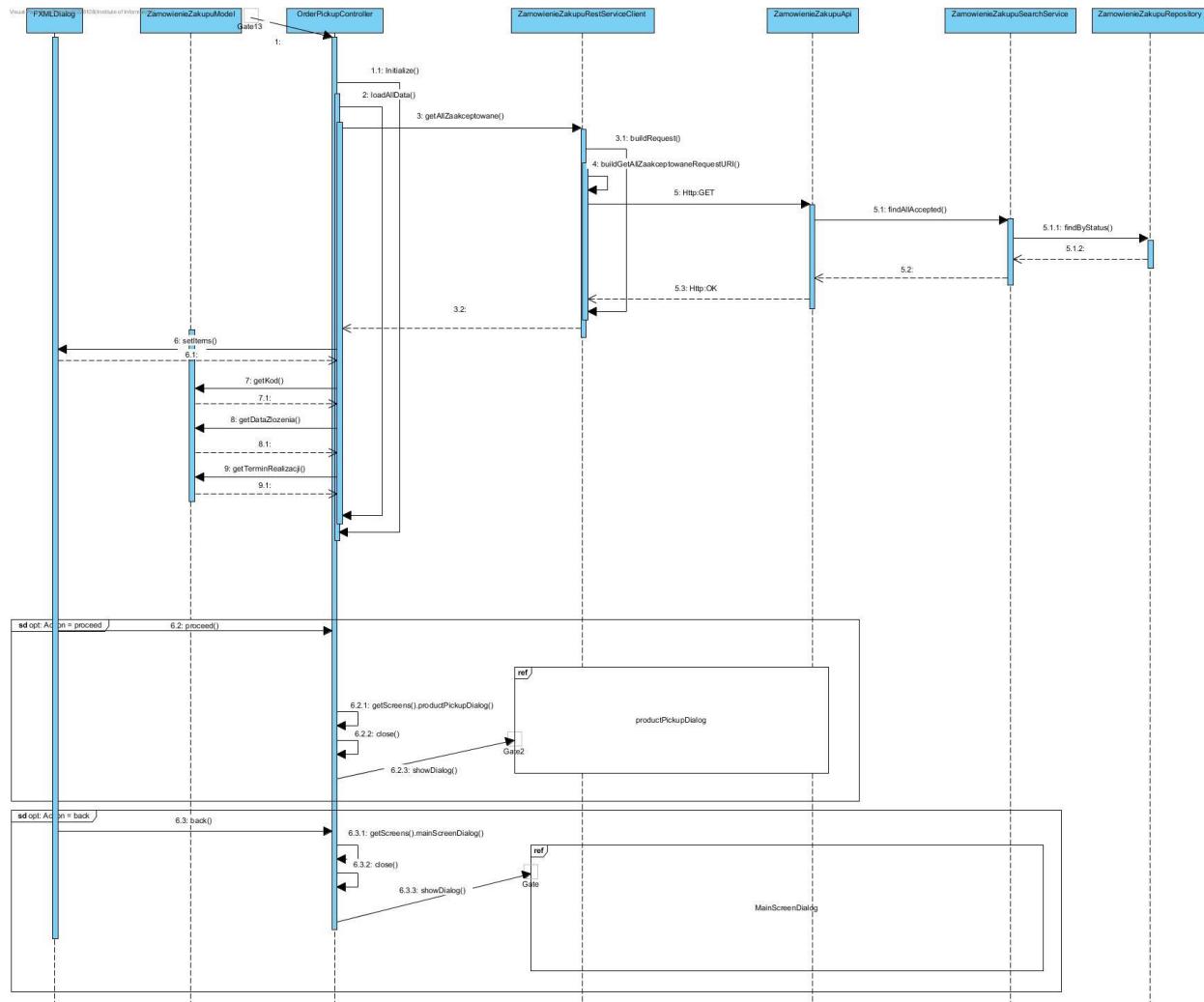
Przy kliknięciu przycisku na ekranie wywoływana jest metoda `proceed()`, zamyka ona aktualny dialog i przechodzi do ekranu głównego.

## 12.2 Kompletowanie zamówienia



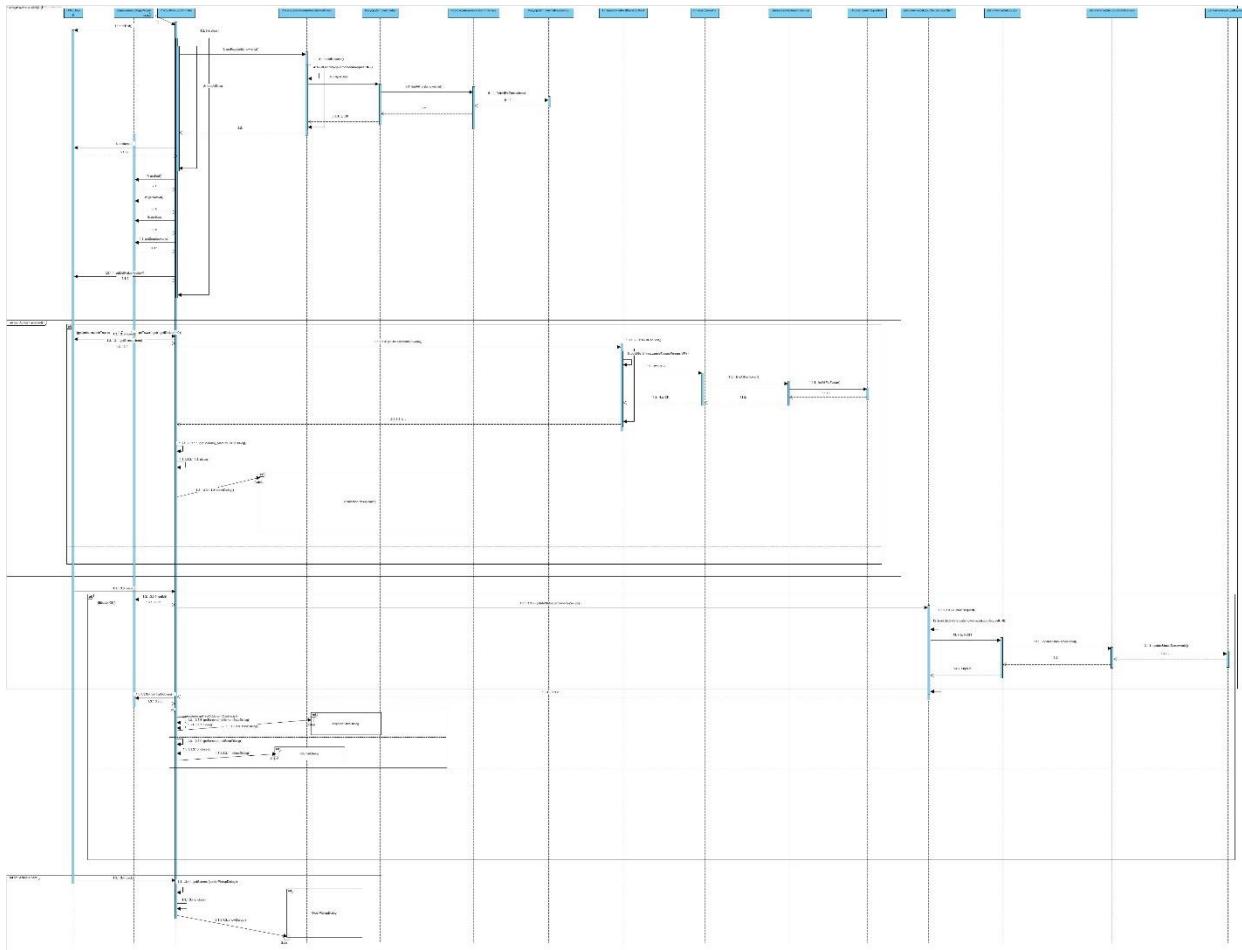
Przy kliknięciu przycisku na ekranie wywoływana jest metoda proceed(), zamyka ona aktualny dialog i przechodzi do ekranu głównego.

<b>Storex</b> Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>
---	--------------------



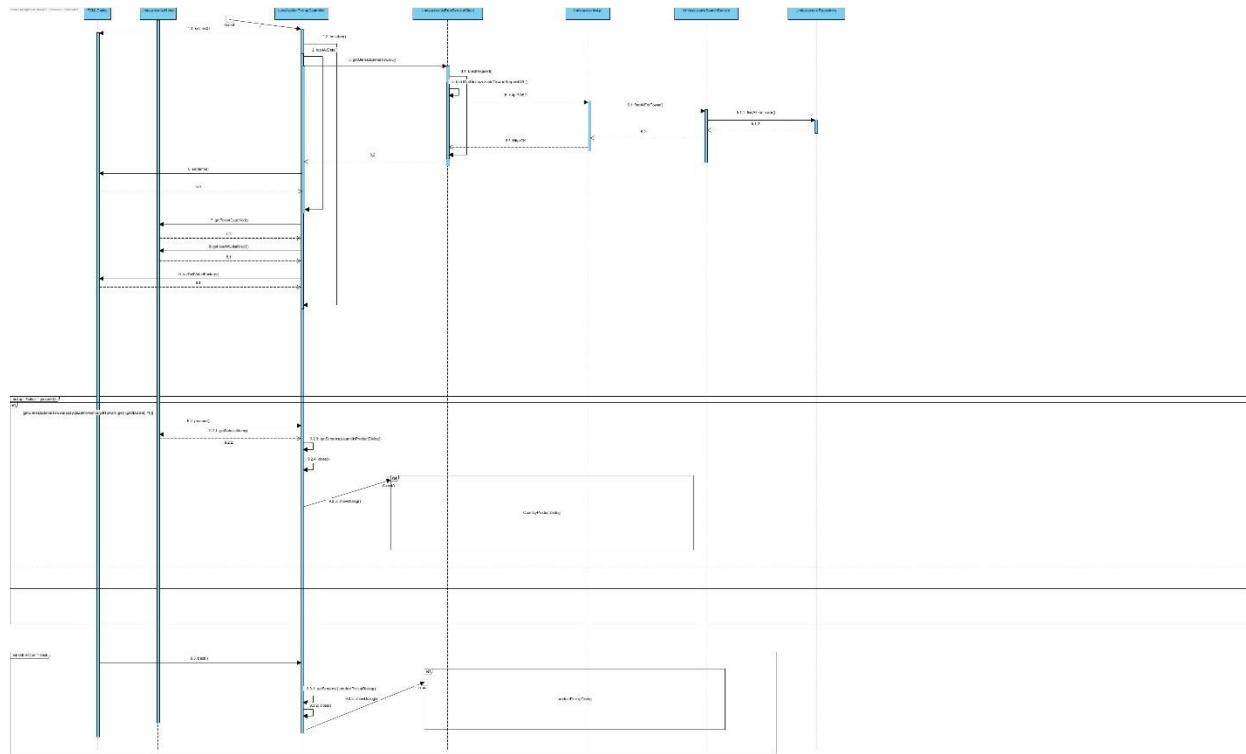
Dialog jest inicjowany, wysyła żądanie pobrania wszystkich zamówień o statusie zaakceptowane do serwera. Zwracana jest lista, oraz status HTTP:OK. Następnie pobierany jest kod, data złożenia, i termin realizacji. Wartości w tabeli wypełniane są podanymi danymi.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



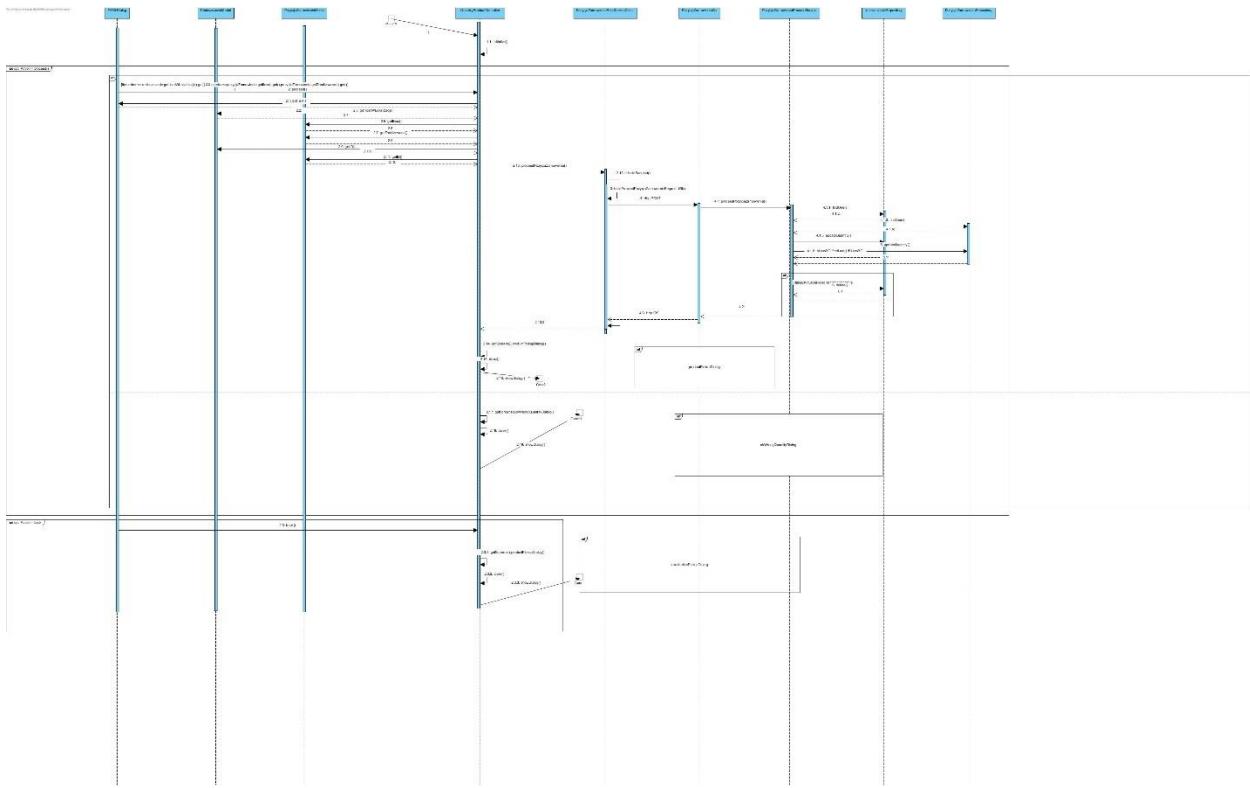
Następuje inicjacja dialogu, wysyłane jest żądanie pobrania wszystkich pozycji zamówienia wybranego na poprzednim ekranie. Serwer odpowiada listą pozycji zamówienia o podanym ID oraz komunikatem HTTP:OK. Pobierany jest kod, ilość do zrealizowania, nazwa i ilość zrealizowana. Wartości te są ustawiane w tabeli dla każdej pozycji. Akcja powrotu pozwala na ukazanie ekranu wyboru zamówienia. Akcja przycisku OK wywołuje metodę proceed(). Najpierw pobierana jest lista umieszczeń wybranego towaru z bazy danych za pomocą żądania wysłanego do serwera. Jeśli ilość ta jest większa od zera ukazywany jest ekran wyboru lokalizacji. W przeciwnym wypadku wyświetlany jest komunikat o chwilowym braku danego produktu na stanie. Jeśli żądaniem jest przycisk wydania zamówienia, wywoływana jest metoda issue(). Wyświetlany jest alert z potwierdzeniem akcji, następnie wysyłane jest żądanie zmiany statusu zamówienia o podanym ID na gotowe. Serwer zwraca status HTTP:OK w przypadku powodzenia operacji, a zamówienie zmienia status w bazie. Następnie wyświetlany jest ekran z danymi do wysyłki lub ekran powiadamiający o wysłanym emailu, zależnie od typu wydanego zamówienia.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



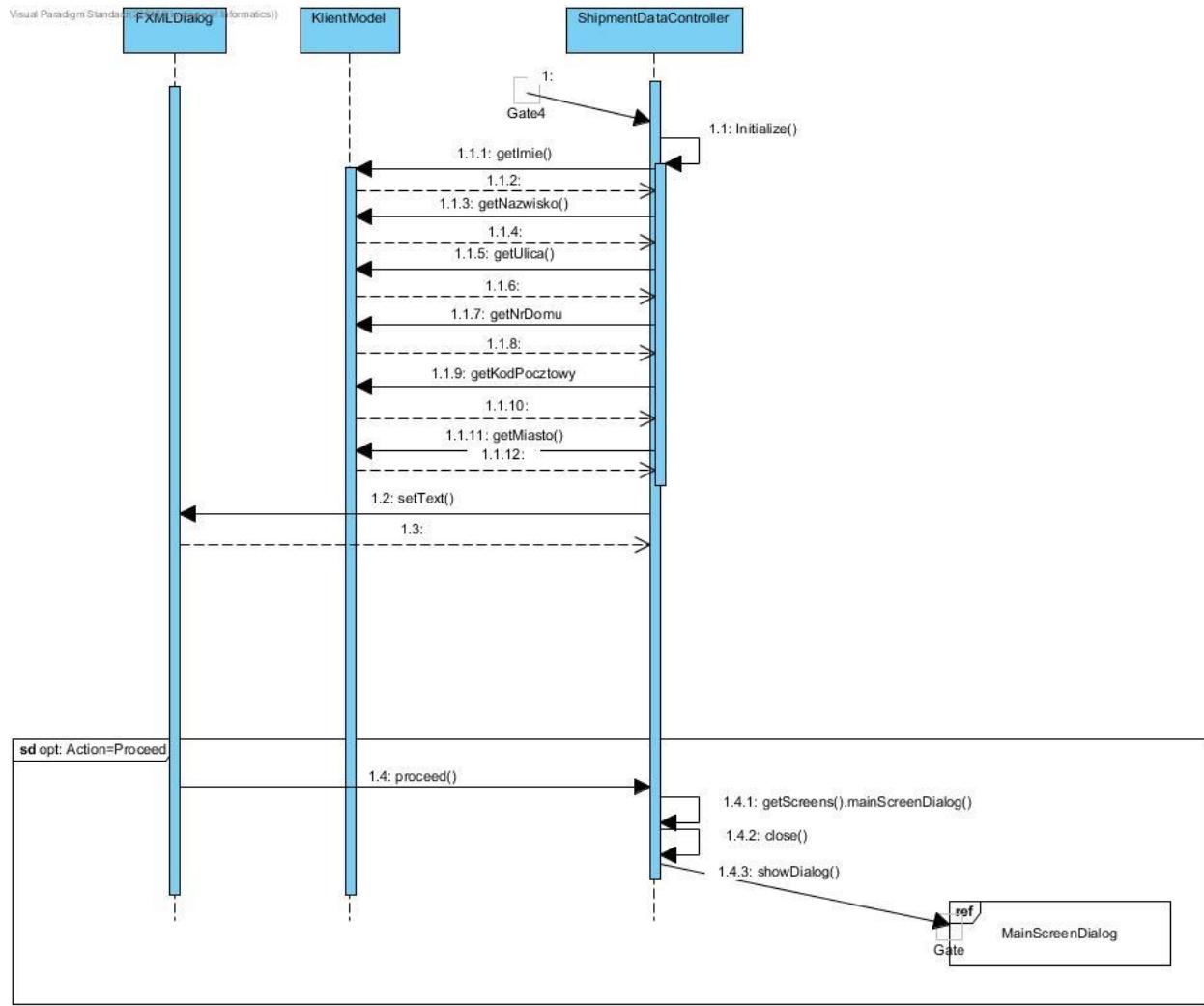
Dialog jest inicjowany, wysłane zostaje żądanie dotyczące umieszczeń towaru o podanym ID. Serwer odpowiada listą umieszczeń danego towaru. Pobierana jest ilość w lokalizacji oraz kod danej lokalizacji, wartości te są ustawiane w tabeli. Akcja powrotu umożliwia ukazanie ekranu wyboru produktu. Akcja przycisku OK wywołuje metodę proceed() i ukazuje ekran wyboru ilości pobranej.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



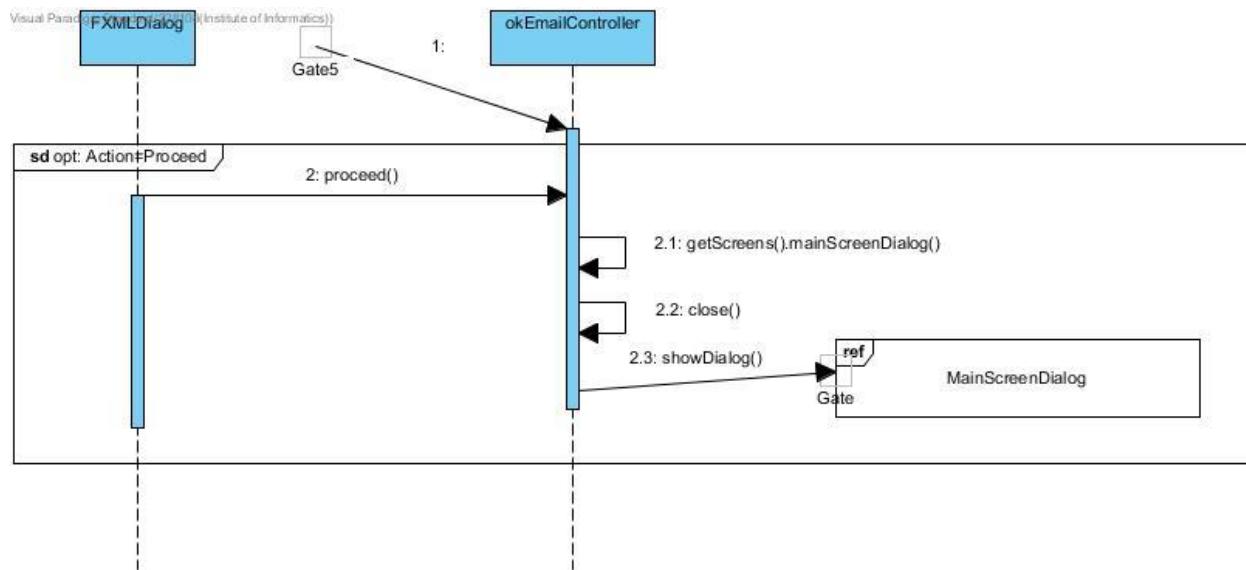
Jeśli wybrano akcję przycisku powrotu, ukazywany jest ekran z wyborem lokalizacji. W przeciwnym wypadku sprawdzany jest tekst podany przez użytkownika. Jeśli podano tekst zgodny z formatem liczbowym wysyłane jest żądanie aktualizacji umieszczenia oraz pozycji zamówienia o podanych ID o ilość wybraną przez użytkownika. Serwer zwraca komunikat HTTP:OK w przypadku powodzenia operacji. Następuje powrót do wyboru produktu. W przeciwnym wypadku, jeśli wybrano ilość mniejszą bądź równą zero, wyświetlany jest ekran informujący o za małej ilości wprowadzonej. Jeśli wprowadzono ilość większą niż dostępna w lokalizacji lub większa niż potrzebna do zrealizowania zamówienia to wyświetlony zostanie komunikat o zbyt dużej ilości pobranej. Jeśli podano ciąg znaków nie składający się tylko z liczb to następuje wyświetlenie komunikatu o niepoprawnie wprowadzonych danych.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



Dialog jest inicjowany danymi wybranymi podczas kompletowania. Wyświetlane są informacje o kliencie i danych dostawy. W przypadku przycisku akcji ukazywany jest ekran główny.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



Na ekranie wyświetlana jest informacja o wysłanym powiadomieniu email. W przypadku przycisku akcji ukazywany jest ekran główny.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## **13. Implementacja**

### **13.1 Kod**

Cały kod implementacji projektu: [Implementacja](#)

### **13.2 Wygenerowane fragmenty kodu**

Fragmenty kodu zostały wygenerowane z diagramu klas modelu informacyjnego oraz diagramu klas dla realizacji PU przy użyciu Visual Paradigm. Ustawienia generatora przedstawione zostały na zdjęciu poniżej. Typ generowanych kolekcji to ArrayList, domyślny typ parametrów oraz atrybutów to Object, a domyślny typ zwracany dla metod to void dodatkowo wygenerowane zostały informacje o importowanych pakietach

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

**Advanced Options for Java Code Generation** X

**Encoding**

Default (windows-1252)  
 Other:

**Language**

Follow Convention

Attribute prefix:

Parameter prefix:

Include referenced projects

Indentation:

Generate unnamed attribute

Unnamed attribute:

Invalid char replacement:

Default attribute type: Object

Default parameter type: Object

Default operation return type: void

Import Fully Qualified

Generate <<import>> dependencies

Generate hashCode and equals operations

Generate Ant build file

Implement abstract operations

Generate association operations

Generate simple collection operations

Generate additional collection operations

Local variable prefix:

Association implementation:

JDK Version:

Pre/Post Condition generation:

**Version Details**

Generics (Template)

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

Wygenerowane fragmenty kodu: [Fragmenty kodu](#)

Przykładowe wygenerowane fragmenty kodu:

```
import java.util.ArrayList;

public class Towar {
    private long _id;
    private String _nazwa;
    private String _kod;
    private double _ilostan;
    private int _czas_zalegania;
    private double _ilosc_zablokowana;
    private boolean _zalega;
    private boolean _potrzebuje_zamówienia;
    private double _ilość_minimalna;
    public Jednostka _unnamed_Jednostka_;
    public Kategoria _unnamed_Kategoria_;
    public ArrayList<Umieszczenie> _unnamed_Umieszczenie_ = new ArrayList<Umieszczenie>();
    public ArrayList<PozycjaBilansu> _pozycjeBilansu = new ArrayList<PozycjaBilansu>();
    public ArrayList<PozycjaZamówienia> _pozycjeZamówienia = new ArrayList<PozycjaZamówienia>();
    public ArrayList<Pozycja_Wydania> _pozycjeWydania = new ArrayList<Pozycja_Wydania>();
    public ArrayList<Pozycja_Przyjcia> _pozycjePrzyjcia = new ArrayList<Pozycja_Przyjcia>();
}
```

Klasa modelu informacyjnego Towar. Importuje pakiet `java.util.ArrayList`, jest klasą publiczną, asocjacje dla liczności maksymalnie jeden po stronie klasy Towar i liczności 'wiele' dla innej klasy zostały wygenerowane jako kolekcja `ArrayList`. Natomiast asocjacje dla liczności 'wiele' po stronie klasy Towar i liczności 'jeden' dla innej klasy zostały wygenerowane jako pole typu danej klasy, do której odnosi się asocjacja.

```
public class Zamówienie_zakupu extends Zamówienie {
    private TypOdbioru _typ_odbioru;
    private StatusWydania _status;
    private DateTime _terminRealizacji;
    public Wydanie_Zamówienia _wydanie;
    public Klient _nabywca;
}
```

Klasa modelu informacyjnego Zamówienie\_zakupu. Jest to klasa dziedzicząca po klasie Zamówienie – 'extends Zamówienie'

```
package com.StoreX.service.PozycjeWydanychServices;
public interface PozycjaWydaniaSearchService {
    /**
     * @param Month
     * @param Year
     */
    java.util.List<com.StoreX.common.datatypes.bo.PozycjaWydaniaBO> findAllForMonthAndYear(int Month, int Year) throws javax.naming.AuthenticationException;
}
```

Interfejs serwisu PozycjaWydaniaSearchService – zawiera informacje o możliwym rzucanym wyjątku

### 13.3 Dokumentacja

Dokumentacja została wykonana przy użyciu JavaDoc. Należało dodać komentarz do klas, a następnie wygenerować dokumentację w postaci plików z rozszerzeniem html za pomocą JavaDoc.

Dokumentacja serwer: [DocumentationServer\overview-summary.html](#)

Dokumentacja klient: [DocumentationClient\index.html](#)

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 13.4 Wykorzystane wzorce projektowe

W projekcie użyto następujących wzorców projektowych:

- Fabryka

```
codeColumn.setCellValueFactory(celldata -> celldata.getValue().getTowar().get().getKod());
```

Fabryki zostały użyte do ustawienia każdej z wartości tabel i list

```
loader.setControllerFactory(aClass -> controller);
```

Fabryka została również użyta do ustawienia kontrolera dialogu

- Singleton

Singleton został użyty jako sposób dostępu do alertów w systemie.

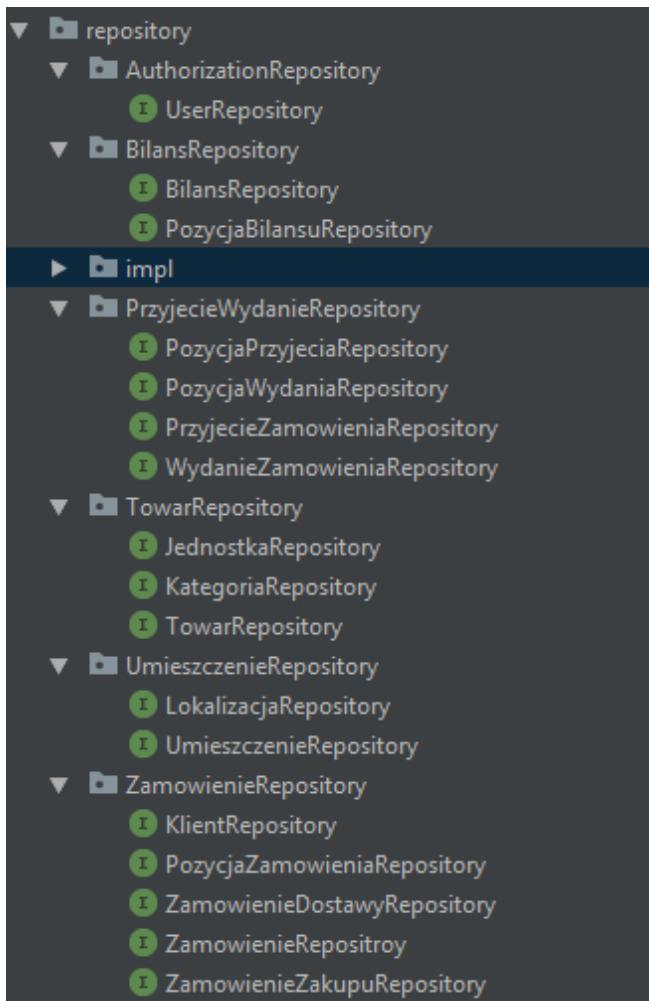
```
private AlertDialog(){
}

/**
 * metoda dostępu do podstawowego alertu
 * @return alert
 */
public Alert getAlert(){
    if (alert==null)
        alert = new Alert(Alert.AlertType.WARNING);
```

- Wzorzec repozytorium

Funkcjonuje w całym projekcie jako warstwa dostępu do danych.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



```

/*
 * Repozytorium używane do akcji z bazą danych oraz Bilans Entity.
 */
public interface BilansRepository extends JpaRepository<Bilans, Long> {

    /**
     * pobiera z bazy bilans z najnowszą datą bilansowania
     * @return bilans z najnowszą datą bilansowania
     */
    @Query(value = "select * from Bilans order by data_bilansu desc limit 1", nativeQuery = true)
    Bilans findLastBilansByDate();

    /**
     * pobiera liczbę bilansów dla konkretnej daty bilansowania
     * @param month miesiąc bilansowania
     * @param year rok bilansowania
     * @return liczba bilansów dla wskazanej daty bilansowania
     */
    @Query(value = "select count(*) from bilans where year(data_bilansu) = ?2 and month(data_bilansu) = ?1", nativeQuery = true)
    int findBilansForMonthAndYear(int month, int year);
}
    
```

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

- Wzorzec fasady

W kodzie użyto także wzorca fasady, np. w serwisie odpowiedzialnym za aktualizację ilości w umieszczeniu i ilości zrealizowanej w pozycji zamówienia. Metody serwisu udostępniają działania z repozytoriów zarówno umieszczeń jak i pozycji zamówień. Podobnie wygląda implementacja po stronie API serwerowego.

```

@Override
@Transactional(propagation = Propagation.REQUIRED, rollbackFor = Exception.class, readOnly = false)
public boolean ProcedePozycjaZamowienia(String sessionId, Long idPozycji, Long idUmieszczenia, double iloscRealizowana) throws A
{
    if(!authorizationService.isUserAuthorized(UUID.fromString(sessionId))) {
        throw new AuthenticationException();
    }

    PozycjaZamowienia pozycjaZamowienia = getPozycjaZamowieniaRepository().findOne(idPozycji);
    Umieszczenie umieszczenie = getUmieszczenieRepository().findOne(idUmieszczenia);
    double iloscJuzZrealizowana = pozycjaZamowienia.getZrealizowano();
    double iloscAktualnaUmieszczenia = umieszczenie.getIloscWLokalizacji();
    double iloscCalkowita = pozycjaZamowienia.getIlosc();

    /*
    Scenariusz alternatywny
    */
    if(iloscJuzZrealizowana + iloscRealizowana > iloscCalkowita)
        throw new Exception("Podano za dużą ilość");
    if(iloscAktualnaUmieszczenia - iloscRealizowana < 0)
        throw new Exception("Podano za dużą ilość Nie ma tyle w lokalizacji");
    if(iloscRealizowana < 0)
        throw new Exception("Podana ilość jest mniejsza od zera");

    getPozycjaZamowieniaRepository().realizacjaPozycjiZmowienia(idPozycji, ilosc iloscJuzZrealizowana + iloscRealizowana);
    getUmieszczenieRepository().realizacjaPozycjiZmowienia(idUmieszczenia, ilosc iloscAktualnaUmieszczenia - iloscRealizowana);

    if(iloscAktualnaUmieszczenia - iloscRealizowana == 0)
    {
        getUmieszczenieRepository().delete(idUmieszczenia);
    }

    return true;
}

```

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 14. Testy

### 14.1 Testy jednostkowe

Testy jednostkowe zostały wykonane dla metod klas serwisowych

PozycjaZamowieniaProceedService -

[Implementacja\server\src\main\java\com\StoreX\service\impl\ZamowienieServicesImpl\PozycjaZamowieniaProceedServiceImpl.java](#)

Testy wykonano dla metody:

Boolean ProceedPozycjaZamowienia(string sessionId, Long idPozycji, Long idUmieszczenia, double ilość.

Kompletny kod testów:

[Implementacja\server\src\test\java\com\StoreX\service\impl\ZamowienieServicesImpl\PozycjaZamowieniaProceedServiceImplTest.java](#)

Przykładowy test:

```
/**
 * WARUNEK BRZEGOWY
 * Ilosc podana w parametrze metody jest równa ilosci w Umieszczeniu
 * Operacja zostanie wykonana poprawnie, zwróci true
 */
@Test
public void proceedPozycjaZamowienia_IloscRownaIlosciWUmieszczeniu_IsTrue() {
    //arrange
    long idUmieszczenia = 2L;
    long idPozycji = 1L;
    String sessionId = "53b37a38-7bf1-48dd-9b92-f14e1b691adf";
    int iloscRealizowana = 10;
    boolean sukces = false;

    //act
    try {
        sukces = pozycjaZamowieniaProceedService.ProceedPozycjaZamowienia(sessionId, idPozycji, idUmieszczenia, iloscRealizowana);
    } catch (Exception e) {
        e.printStackTrace();
    }

    //assert
    Assert.assertTrue(sukces);
}
```

BilansCreationService -

[Implementacja\server\src\main\java\com\StoreX\service\BilansServices\PozycjaBilansuCreationService.java](#)

Testy wykonano dla metody:

Boolean addBilans(String sessionId, Date dataBilansowana)

Kompletny kod testów:

[Implementacja\server\src\test\java\com\StoreX\service\impl\BilansServicesImpl\BilansCreationServiceImplTest.java](#)

Przykładowy test:

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

```

/**
 * Jeżeli próbujemy dodać bilans dla już zbilansowanego miesiąca, operacja nie zostaje wykonana i zostaje rzucony wyjątek
 */
@Test
public void addBilans_BilansAlreadyExist_AreEqual() {
    //arrange
    //+1, ze wzgledu na indeksowanie od 0 przy ustalaniu daty
    Mockito.when(bilansRepository.findBilansForMonthAndYear( month: 9 + 1, year: 2017)).thenReturn(1);
    Calendar dataBilansowana = Calendar.getInstance();
    dataBilansowana.set( year: 2017, month: 9, date: 10 );
    String errorMessage = "";

    //act
    try {
        bilansCreationService.addBilans( sessionId: "53b37a38-7bf1-48dd-9b92-f14e1b691adf", dataBilansowana.getTime() );
    } catch (Exception e) {
        errorMessage = e.getMessage();
    }

    //assert
    Assert.assertEquals( expected: "Bilans już istnieje", errorMessage );
}

}

```

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 14.2 Przypadki testowe

### 14.2.1 Kompletowanie zamówienia

ID	14.2.1.1 001 Kompletowanie zamówienia - brak zamówień
Opis	Kompletowanie zamówienia - brak zamówień
Warunki wstępne	Użytkownik jest zalogowany. Aplikacja otwarta na ekranie głównym. W bazie nie ma zamówień zakupu o statusie zaakceptowane
Oczekiwany rezultat	System wyświetla informację "Nie ma aktualnie żadnych oczekujących na skompletowanie zamówień". Brak zmian w bazie danych

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawił się komunikat "Nie ma aktualnie żadnych zamówień oczekujących na skompletowanie"

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.2 002 Kompletowanie zamówienia - brak lokalizacji zawierającej wybrany towar
Opis	Kompletowanie zamówienia - brak lokalizacji zawierającej wybrany towar
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <p>-Zamówienie zakupu:  Id: 1  Status: 'Zaakceptowane'  Data złożenia: 2017-12-01  Termin realizacji: 2018-02-02  Typ odbioru: 'Wysyłka'  Klient Id: 1</p> <p>-Pozycja zamówienia:  Id: 1  Ilość: 10  Zrealizowano: 2  Towar Id: 1  Zamówienie Id: 1</p> <p>-Towar:  Id: 1  Ilosan: 20  Kod: t01  Nazwa: Rower  Jednostka: Sztuka  Kategoria: Sport</p> <p>W bazie nie istnieje umieszczenie dla towaru o Id 1</p>
Oczekiwany rezultat	System wyświetla informację "Ten towar nie jest aktualnie dostępny w żadnej lokalizacji w magazynie". Kliknięcie OK powoduje powrót do listy produktów w zamówieniu. Brak zmian w bazie danych

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK
4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawił się komunikat "Ten towar nie jest aktualnie dostępny w żadnej lokalizacji w magazynie"
7. Kliknij OK
8. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.3 003 Kompletowanie zamówienia - wybrano więcej niż w lokalizacji.
Opis	Kompletowanie zamówienia - wybrano więcej niż w lokalizacji. Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik podaje liczbę większą, niż ilość towaru w wybranej lokalizacji
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <p>-Zamówienie zakupu:  Id: 1  Status: 'Zaakceptowane'  Data złożenia: 2017-12-01  Termin realizacji: 2018-02-02  Typ odbioru: 'Wysyłka'  Klient Id: 1</p> <p>-Pozycja zamówienia:  Id: 1  Ilość: 10  Zrealizowano: 2  Towar Id: 1  Zamówienie Id: 1</p> <p>-Towar:  Id: 1  Ilosan: 20  Kod: t01  Nazwa: Rower  Jednostka: Sztuka  Kategoria: Sport</p> <p>-Lokalizacja:  Id: 1  Kod: L01</p> <p>-Umieszczenie:  Id: 1  Ilość w lokalizacji: 5  Lokalizacja Id: 1  Towar Id: 1</p>
Oczekiwany rezultat	System wyświetla informację "Podano za dużą liczbę". Kliknięcie OK powoduje powrót do ekranu proszącego o podanie ilości pobranej z lokalizacji. Brak zmian w bazie danych

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

3. Wybierz zamówienie o numerze z01 i kliknij OK
4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 5.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 5.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Zmień wartość 5.0 na 7.0
10. Kliknij OK
11. Sprawdź, czy na ekranie pojawił się komunikat „Podano za dużą liczbę”
12. Kliknij OK
13. Sprawdź, czy na ekranie pojawiły się informację jak w punkcie 8.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.4 004 Kompletowanie zamówienia – wybrano więcej niż w pozycji zamówienia
Opis	Kompletowanie zamówienia – wybrano więcej niż w pozycji zamówienia: Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik podaje liczbę większą, niż ilość towaru, która została do realizacji w wybranej pozycji zamówienia
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <p>-Zamówienie zakupu:  Id: 1  Status: 'Zaakceptowane'  Data złożenia: 2017-12-01  Termin realizacji: 2018-02-02  Typ odbioru: 'Wysyłka'  Klient Id: 1</p> <p>-Pozycja zamówienia:  Id: 1  Ilość: 10  Zrealizowano: 2  Towar Id: 1  Zamówienie Id: 1</p> <p>-Towar:  Id: 1  Ilostan: 20  Kod: t01  Nazwa: Rower  Jednostka: Sztuka  Kategoria: Sport</p> <p>-Lokalizacja:  Id: 1  Kod: L01</p> <p>-Umieszczenie:  Id: 1  Ilość w lokalizacji: 20  Lokalizacja Id: 1  Towar Id: 1</p>
Oczekiwany rezultat	System wyświetla informację "Podano za dużą liczbę". Kliknięcie OK powoduje powrót do ekranu proszącego o podanie ilości pobranej z lokalizacji. Brak zmian w bazie danych.

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 20.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 8.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Zmień wartość 8.0 na 10.0
10. Kliknij OK
11. Sprawdź, czy na ekranie pojawił się komunikat „Podano za dużą liczbę”
12. Kliknij OK
13. Sprawdź, czy na ekranie pojawiły się informacje jak w punkcie 8.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.5 005 Kompletowanie zamówienia – wybrano 0
Opis	Kompletowanie zamówienia – wybrano 0: Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik podaje liczbę 0
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Status: 'Zaakceptowane'</li> <li>Data złożenia: 2017-12-01</li> <li>Termin realizacji: 2018-02-02</li> <li>Typ odbioru: 'Wysyłka'</li> <li>Klient Id: 1</li> </ul> </li> <li>-Pozycja zamówienia: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość: 10</li> <li>Zrealizowano: 2</li> <li>Towar Id: 1</li> <li>Zamówienie Id: 1</li> </ul> </li> <li>-Towar: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilostan: 20</li> <li>Kod: t01</li> <li>Nazwa: Rower</li> <li>Jednostka: Sztuka</li> <li>Kategoria: Sport</li> </ul> </li> <li>-Lokalizacja: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod: L01</li> </ul> </li> <li>-Umieszczenie: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość w lokalizacji: 20</li> <li>Lokalizacja Id: 1</li> <li>Towar Id: 1</li> </ul> </li> </ul>
Oczekiwany rezultat	System wyświetla informację "Podano za małą liczbę". Kliknięcie OK powoduje powrót do ekranu proszącego o podanie ilości pobranej z lokalizacji. Brak zmian w bazie danych.

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 20.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 8.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Zmień wartość 8.0 na 0.0
10. Kliknij OK
11. Sprawdź, czy na ekranie pojawił się komunikat „Podano za małą liczbę”
12. Kliknij OK
13. Sprawdź, czy na ekranie pojawiły się informacje jak w punkcie 8.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.6 006 Kompletowanie zamówienia – podano liczbę ujemną
Opis	Kompletowanie zamówienia – podano liczbę ujemną: Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik podaje liczbę ujemną
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu:  Id: 1  Status: 'Zaakceptowane'  Data złożenia: 2017-12-01  Termin realizacji: 2018-02-02  Typ odbioru: 'Wysyłka'  Klient Id: 1</li> <li>-Pozycja zamówienia:  Id: 1  Ilość: 10  Zrealizowano: 2  Towar Id: 1  Zamówienie Id: 1</li> <li>-Towar:  Id: 1  Ilostan: 20  Kod: t01  Nazwa: Rower  Jednostka: Sztuka  Kategoria: Sport</li> <li>-Lokalizacja:  Id: 1  Kod: L01</li> <li>-Umieszczenie:  Id: 1  Ilość w lokalizacji: 20  Lokalizacja Id: 1  Towar Id: 1</li> </ul>
Oczekiwany rezultat	System wyświetla informację "Podano za małą liczbę". Kliknięcie OK powoduje powrót do ekranu proszącego o podanie ilości pobranej z lokalizacji. Brak zmian w bazie danych.

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 20.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 8.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Zmień wartość 8.0 na -1
10. Kliknij OK
11. Sprawdź, czy na ekranie pojawił się komunikat „Podano za małą liczbę”
12. Kliknij OK
13. Sprawdź, czy na ekranie pojawiły się informacje jak w punkcie 8.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.7 007 Kompletowanie zamówienia – niepoprawny format danych
Opis	Kompletowanie zamówienia – niepoprawny format danych Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik podaje tekst 'test1'
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu:  Id: 1  Status: 'Zaakceptowane'  Data złożenia: 2017-12-01  Termin realizacji: 2018-02-02  Typ odbioru: 'Wysyłka'  Klient Id: 1</li> <li>-Pozycja zamówienia:  Id: 1  Ilość: 10  Zrealizowano: 2  Towar Id: 1  Zamówienie Id: 1</li> <li>-Towar:  Id: 1  Ilostan: 20  Kod: t01  Nazwa: Rower  Jednostka: Sztuka  Kategoria: Sport</li> <li>-Lokalizacja:  Id: 1  Kod: L01</li> <li>-Umieszczenie:  Id: 1  Ilość w lokalizacji: 20  Lokalizacja Id: 1  Towar Id: 1</li> </ul>
Oczekiwany rezultat	System wyświetla informację "Podaj poprawną liczbę". Kliknięcie OK powoduje powrót do ekranu proszącego o podanie ilości pobranej z lokalizacji. Brak zmian w bazie danych.

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

3. Wybierz zamówienie o numerze z01 i kliknij OK
4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz towar o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 20.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 8.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Zmień wartość 8.0 na ‘test1’
10. Kliknij OK
11. Sprawdź, czy na ekranie pojawił się komunikat „Podaj poprawną liczę”
12. Kliknij OK
13. Sprawdź, czy na ekranie pojawiły się informację jak w punkcie 8.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.8 008 Kompletowanie zamówienia – pobrano ilość do zrealizowania
Opis	Kompletowanie zamówienia – pobrano ilość do zrealizowania: Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik pozostawia pole wypełnione odpowiedzią systemu.
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Status: 'Zaakceptowane'</li> <li>Data złożenia: 2017-12-01</li> <li>Termin realizacji: 2018-02-02</li> <li>Typ odbioru: 'Wysyłka'</li> <li>Klient Id: 1</li> </ul> </li> <li>-Pozycja zamówienia: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość: 10</li> <li>Zrealizowano: 2</li> <li>Towar Id: 1</li> <li>Zamówienie Id: 1</li> </ul> </li> <li>-Towar: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilostan: 20</li> <li>Kod: t01</li> <li>Nazwa: Rower</li> <li>Jednostka: Sztuka</li> <li>Kategoria: Sport</li> </ul> </li> <li>-Lokalizacja: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod: L01</li> </ul> </li> <li>-Umieszczenie: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość w lokalizacji: 20</li> <li>Lokalizacja Id: 1</li> <li>Towar Id: 1</li> </ul> </li> </ul>
Oczekiwany rezultat	Na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 10.0

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>
Zmiany w bazie danych: Ilość zrealizowana towaru w pozycji zamówienia o id: 1 została zwiększena do 10, Ilość w umieszczeniu o Id: 1 została zmniejszona do 12	

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK
4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz товар o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 20.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 8.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Nie zmieniaj wartości 8.0, kliknij OK
10. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 i ilością zrealizowaną równą 10.0
11. W bazie danych sprawdź, czy Ilość zrealizowana towaru w pozycji zamówienia o id: 1 została zwiększena do 10, Ilość w umieszczeniu o Id: 1 została zmniejszona do 12

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.9 009 Kompletowanie zamówienia – usunięcie umieszczenia
Nazwa	Kompletowanie zamówienia – usunięcie umieszczenia: Kiedy system prosi o podanie ilości towaru pobranej z lokalizacji, użytkownik pozostawia pole wypełnione podpowiedzią systemu z wartością 5.0. Ilość ta jest równa ilości towaru w umieszczeniu, umieszczenie zostanie usunięte
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Status: 'Zaakceptowane'</li> <li>Data złożenia: 2017-12-01</li> <li>Termin realizacji: 2018-02-02</li> <li>Typ odbioru: 'Wysyłka'</li> <li>Klient Id: 1</li> </ul> </li> <li>-Pozycja zamówienia: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość: 10</li> <li>Zrealizowano: 2</li> <li>Towar Id: 1</li> <li>Zamówienie Id: 1</li> </ul> </li> <li>-Towar: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilosstan: 20</li> <li>Kod: t01</li> <li>Nazwa: Rower</li> <li>Jednostka: Sztuka</li> <li>Kategoria: Sport</li> </ul> </li> <li>-Lokalizacja: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod: L01</li> </ul> </li> <li>-Umieszczenie: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość w lokalizacji: 5</li> <li>Lokalizacja Id: 1</li> <li>Towar Id: 1</li> </ul> </li> </ul>
Oczekiwany rezultat	Po drugiej próbie realizacji pozycji zamówienia zostanie wyświetlony komunikat: „Ten towar nie jest aktualnie dostępny w żadnej lokalizacji w magazynie”

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

Zmiany w bazie danych: Umieszczenie o id: 1 zostanie usunięte. Zrealizowana ilość towaru pozycji zamówienia o id: 1 została zmieniona z 2.0 na 7.0
--

1. Kliknij "Kompletuj zamówienie"
2. Sprawdź, czy na ekranie pojawiła się lista zamówień z zamówieniem o numerze z01.
3. Wybierz zamówienie o numerze z01 i kliknij OK
4. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 oraz ilością zrealizowaną równą 2.0
5. Wybierz товар o numerze t01 i kliknij OK
6. Sprawdź, czy na ekranie pojawiła się lista lokalizacja z lokalizacją o kodzie L01 i ilością w lokalizacji 5.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
7. Wybierz lokalizację o kodzie L01 i kliknij OK
8. Sprawdź, czy na ekranie pojawiła się prośba o podanie ilości pobranej z lokalizacji oraz automatycznie uzupełnione pole z wartością 5.0 oraz czy na górze ekranu pojawiła się informacja o kodzie i nazwie towaru oraz potrzebnej ilości równej 8.0
9. Nie zmieniaj wartości 8.0, kliknij OK
10. Sprawdź, czy na ekranie pojawiła się lista towarów z towarem o numerze t01 z ilością w zamówieniu równą 10.0 i ilością zrealizowaną równą 7.0
11. Wybierz товар o numerze t01 i kliknij OK
12. Sprawdź, czy na ekranie pojawił się komunikat „Ten товар не jest aktualnie dostępny w żadnej lokalizacji w magazynie”.
13. Sprawdź, czy w bazie danych zostało usunięte umieszczenie o id: 1 oraz czy zrealizowana ilość towaru w pozycji zamówienia o id: 1 została zmieniona z 2.0 na 7.0

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.10 010 Wydanie zamówienia – Typ odbioru: wysyłka
Nazwa	Wydanie zamówienia – Typ odbioru: wysyłka
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie z listą zrealizowanych pozycji zamówienia.</p> <p>W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Status: 'Zaakceptowane'</li> <li>Data złożenia: 2017-12-01</li> <li>Termin realizacji: 2018-02-02</li> <li>Typ odbioru: 'Wysyłka'</li> <li>Klient Id: 1</li> </ul> </li> <li>-Pozycja zamówienia: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość: 10</li> <li>Zrealizowano: 2</li> <li>Towar Id: 1</li> <li>Zamówienie Id: 1</li> </ul> </li> <li>-Towar: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilostan: 20</li> <li>Kod: t01</li> <li>Nazwa: Rower</li> <li>Jednostka: Sztuka</li> <li>Kategoria: Sport</li> </ul> </li> <li>-Lokalizacja: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod: L01</li> </ul> </li> <li>-Umieszczenie: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość w lokalizacji: 20</li> <li>Lokalizacja Id: 1</li> <li>Towar Id: 1</li> </ul> </li> <li>-Klient: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Email: 'jan.b@live.com'</li> <li>Imię: 'Jan'</li> <li>Nazwisko: 'Borowiak'</li> <li>Login: 'janbo'</li> <li>Kod pocztowy: '42-110'</li> <li>Miasto: 'Leszno'</li> <li>Numer domu: '1'</li> <li>Ulica: 'Polna'</li> </ul> </li> </ul>

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>
Oczekiwany rezultat	Na ekranie pojawiła się informacja o danych do wysyłki. Zmiany w bazie danych: Zamówienie o Id: 1 zmienia stan z ‘Zaakceptowane’ na ‘Gotowe’

1. Kliknij „Przekaż do wydania”
2. Sprawdź, czy na ekranie pojawiło się pytanie o potwierdzenie wykonania akcji.
3. Kliknij OK.
4. Sprawdź, czy na ekranie pojawiła się informacja o danych do wysyłki zgodnych z danymi klienta o ID 1
5. Sprawdź w bazie danych, czy Zamówienie o Id: 1 zmieniło stan z ‘Zaakceptowane’ na ‘Gotowe’

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.1.11 011 Wydanie zamówienia – Typ odbioru: Osobiście
Nazwa	Wydanie zamówienia – Typ odbioru: Osobiście
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie z listą zrealizowanych pozycji zamówienia.</p> <p>W bazie istnieją:</p> <ul style="list-style-type: none"> <li>-Zamówienie zakupu: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Status: 'Zaakceptowane'</li> <li>Data złożenia: 2017-12-01</li> <li>Termin realizacji: 2018-02-02</li> <li>Typ odbioru: 'Osobiście'</li> <li>Klient Id: 1</li> </ul> </li> <li>-Pozycja zamówienia: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość: 10</li> <li>Zrealizowano: 2</li> <li>Towar Id: 1</li> <li>Zamówienie Id: 1</li> </ul> </li> <li>-Towar: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilostan: 20</li> <li>Kod: t01</li> <li>Nazwa: Rower</li> <li>Jednostka: Sztuka</li> <li>Kategoria: Sport</li> </ul> </li> <li>-Lokalizacja: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod: L01</li> </ul> </li> <li>-Umieszczenie: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Ilość w lokalizacji: 20</li> <li>Lokalizacja Id: 1</li> <li>Towar Id: 1</li> </ul> </li> <li>-Klient: <ul style="list-style-type: none"> <li>Id: 1</li> <li>Email: 'jan.b@live.com'</li> <li>Imię: 'Jan'</li> <li>Nazwisko: 'Borowiak'</li> <li>Login: 'janbo'</li> <li>Kod pocztowy: '42-110'</li> <li>Miasto: 'Leszno'</li> <li>Numer domu: '1'</li> <li>Ulica: 'Polna'</li> </ul> </li> </ul>

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>
Oczekiwany rezultat	<p>Na ekranie pojawił się komunikat „Email z powiadomieniem odbioru został wysłany do klienta”</p> <p>Zmiany w bazie danych: Zamówienie o Id: 1 zmienia stan z ‘Zaakceptowane’ na ‘Gotowe’</p>

1. Kliknij „Przekaż do wydania”
2. Sprawdź, czy na ekranie pojawiło się pytanie o potwierdzenie wykonania akcji.
3. Kliknij OK.
4. Sprawdź, czy na ekranie pojawił się komunikat „Email z powiadomieniem odbioru został wysłany do klienta”
5. Sprawdź w bazie danych, czy Zamówienie o Id: 1 zmieniło stan z ‘Zaakceptowane’ na ‘Gotowe’

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

#### 14.2.2 Bilansowanie

ID	14.2.2.1 011 Bilansowanie – generowanie pierwszego bilansu w systemie
Nazwa	Bilansowanie – generowanie pierwszego bilansu w systemie
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  Aktualny miesiąc: 01/2018  W bazie nie istnieje żaden bilans  W bazie istnieją następujące dane:</p> <ul style="list-style-type: none"> <li>- Towar Id: 1 Kod:t01 Nazwa: Rower</li> <li>-Towar Id: 2 Kod:t02 Nazwa: Sztuki</li> <li>-Przyjęcie zamówienia: Id: 1; Data: 2017-12-12</li> <li>-Pozycja przyjęcia: Id:1 Ilość: 20 Przyjęcie zamówienia Id:1 Towar Id: 1</li> <li>-Pozycja przyjęcia: Id:2 Ilość: 30 Przyjęcie zamówienia Id:1 Towar Id: 2</li> <li>-Wydanie zamówienia: Id: 1; Data: 2017-12-14</li> <li>-Pozycja wydania: Id:1 Ilość: 10 Wydanie zamówienia Id:1 Towar Id: 1</li> <li>-Pozycja Wydania Id:2</li> </ul>

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

	Ilość: 10 Wydanie zamówienia Id:1 Towar Id: 1
Oczekiwany rezultat	<p>Na ekranie pojawił się komunikat ‘Bilans został zapisany’</p> <p>Zmiany w bazie danych: Zapisany został Bilans:</p> <ul style="list-style-type: none"> <li>- id: 1, ilość: 10, bilans Id: 1, towar Id : 1</li> <li>- id: 2, ilość: 20, bilans Id: 1, towar Id : 2</li> </ul>

1. Kliknij ‘Sporządź bilans’
2. Sprawdź, czy na ekranie pojawiła się lista miesięcy z wierszem ‘Grudzień 2017’
3. Wybierz ‘Grudzień 2017’ i kliknij OK.
4. Sprawdź, czy na ekranie pojawił się komunikat ‘Bilans został zapisany’.
5. Sprawdź w bazie danych, czy istnieje bilans o następujących danych:
  - id: 1
  - data bilansu: 2017-12-01
  - data wykonania: obecna data
 Oraz pozycje bilansu o następujących danych:
  - id: 1, ilość: 10, bilans Id: 1, towar Id : 1
  - id: 2, ilość: 20, bilans Id: 1, towar Id : 2

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.2.2 012 Bilansowanie – wybór niepoprawnego miesiąca
Nazwa	Bilansowanie – wybór niepoprawnego miesiąca
Warunki wstępne	<p>Użytkownik jest zalogowany.  Aplikacja otwarta na ekranie głównym  Aktualny miesiąc: 01/2018  W bazie istnieją:</p> <p>-Bilans  Id: 1  Data bilansu: 01-10-2017  Data wykonania bilansu</p>
Oczekiwany rezultat	<p>Na ekranie pojawił się komunikat ‘Wybrana data jest niepoprawna’  Brak zmian w bazie danych</p>

1. Kliknij ‘Sporządź bilans’
2. Sprawdź, czy na ekranie pojawiła się lista miesięcy z wierszami ‘Listopad 2017’ oraz ‘Grudzień 2017’
3. Wybierz ‘Grudzień 2017’ i kliknij OK
4. Sprawdź, czy na ekranie pojawił się komunikat ‘Wybrana data jest niepoprawna’

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.2.3 013 Bilansowanie – pozycje bilansu, wydań i przyjęć
Nazwa	Bilansowanie – pozycje przyjęć, wydań i przyjęć. Dla bilansowanego miesiąca istnieją pozycje wydań i pozycje przyjęć oraz pozycje bilansu dla bilansu dla poprzedzającego miesiąca. Wszystkie wymienione elementy powinny zostać uwzględnione w generowanym bilansie.
Warunki wstępne	<p>Użytkownik jest zalogowany. Aplikacja otwarta na ekranie głównym Aktualny miesiąc: 01/2018 W bazie istnieją następujące dane:</p> <ul style="list-style-type: none"> <li>-Bilans           <ul style="list-style-type: none"> <li>Id:1</li> <li>Data bilansu: 2017-10-01</li> <li>Data wykonania bilansu: 2017-11-14</li> </ul> </li> <li>-Pozycja bilansu:           <ul style="list-style-type: none"> <li>id: 1</li> <li>ilość: 10</li> <li>bilans Id: 1</li> <li>towar Id : 1</li> </ul> </li> <li>-Pozycja bilansu:           <ul style="list-style-type: none"> <li>id: 2</li> <li>ilość: 20</li> <li>bilans Id: 1</li> <li>towar Id : 2</li> </ul> </li> <li>- Towar           <ul style="list-style-type: none"> <li>Id: 1</li> <li>Kod:t01</li> <li>Nazwa: Rower</li> </ul> </li> <li>-Towar           <ul style="list-style-type: none"> <li>Id: 2</li> <li>Kod:t02</li> <li>Nazwa: Sztuki</li> </ul> </li> <li>-Przyjęcie zamówienia:           <ul style="list-style-type: none"> <li>Id: 1;</li> <li>Data: 2017-11-12</li> </ul> </li> <li>-Pozycja przyjęcia:           <ul style="list-style-type: none"> <li>Id:1</li> <li>Ilość: 20</li> <li>Przyjęcie zamówienia Id:1</li> <li>Towar Id: 1</li> </ul> </li> <li>-Pozycja przyjęcia:</li> </ul>

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

	<p>Id:2 Ilość: 30 Przyjęcie zamówienia Id:1 Towar Id: 2</p> <p>-Wydanie zamówienia: Id: 1; Data: 2017-11-14</p> <p>-Pozycja wydania: Id:1 Ilość: 10 Wydanie zamówienia Id:1 Towar Id: 1</p> <p>-Pozycja Wydania Id:2 Ilość: 10 Wydanie zamówienia Id:1 Towar Id: 1</p>
Oczekiwany rezultat	<p>Na ekranie pojawił się komunikat 'Bilans został zapisany' Zmiany w bazie danych: Zapisany został Bilans: Id:2 data bilansu: 2017-11-01 data wykonania: obecna data</p> <p>Pozycje bilansu:</p> <ul style="list-style-type: none"> <li>- id: 3, ilość: 20, bilans Id: 2, towar Id : 1</li> <li>- id: 4, ilość: 40, bilans Id: 2, towar Id : 2</li> </ul>

1. Kliknij 'Sporządź bilans'
2. Sprawdź, czy na ekranie pojawiła się lista miesięcy z wierszami 'Listopad 2017' oraz 'Grudzień 2017'
3. Wybierz 'Listopad 2017' i kliknij OK
4. Sprawdź, czy na ekranie pojawił się komunikat 'Bilans został zapisany'.
5. Sprawdź w bazie danych, czy istnieje bilans o następujących danych:
  - id: 2
  - data bilansu: 2017-11-01
  - data wykonania: obecna data
 Oraz pozycje bilansu o następujących danych:
  - id: 1, ilość: 10, bilans Id: 1, towar Id : 1
  - id: 2, ilość: 20, bilans Id: 1, towar Id : 2

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.2.4 014 Bilansowanie – różne towary w pozycjach bilansu, wydań i przyjęć
Nazwa	Bilansowanie – pozycje przyjęć, wydań i przyjęć. Dla bilansowanego miesiąca istnieje pozycja wydania i pozycja przyjęcia oraz pozycja bilansu dla bilansu dla poprzedzającego miesiąca. Każda pozycja zawiera inny towar. Wszystkie wymienione elementy powinny zostać uwzględnione w generowanym bilansie.
Warunki wstępne	<p>Użytkownik jest zalogowany. Aplikacja otwarta na ekranie głównym Aktualny miesiąc: 01/2018 W bazie istnieją następujące dane:</p> <ul style="list-style-type: none"> <li>-Bilans Id:1 Data bilansu: 2017-10-01 Data wykonania bilansu: 2017-11-14</li> <li>-Pozycja bilansu: id: 1 ilość: 10 bilans Id: 1 towar Id : 1</li> <li>- Towar Id: 1 Kod:t01 Nazwa: Rower</li> <li>-Towar Id: 2 Kod:t02 Nazwa: Sztuki</li> <li>Towar Id: 3 Kod:t03 Nazwa: Sztuki</li> <li>-Przyjęcie zamówienia: Id: 1; Data: 2017-11-12</li> <li>-Pozycja przyjęcia: Id:1 Ilość: 20 Przyjęcie zamówienia Id:1 Towar Id: 2</li> </ul>

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

	<p>-Wydanie zamówienia: Id: 1; Data: 2017-11-14</p> <p>-Pozycja wydania: Id:1 Ilość: 30 Wydanie zamówienia Id:1 Towar Id: 3</p>
Oczekiwany rezultat	<p>Na ekranie pojawił się komunikat ‘Bilans został zapisany’ Zmiany w bazie danych: Zapisany został Bilans: Id:2 data bilansu: 2017-11-01 data wykonania: obecna data Pozycje bilansu: - id: 2, ilość: 10, bilans Id: 2, towar Id : 1 - id: 3, ilość: 20, bilans Id: 2, towar Id : 2 - id: 4, ilość: 30, bilans Id: 2, towar Id : 3</p>

1. Kliknij ‘Sporządź bilans’
2. Sprawdź, czy na ekranie pojawiła się lista miesięcy z wierszami ‘Listopad 2017’ oraz ‘Grudzień 2017’
3. Wybierz ‘Listopad 2017’ i kliknij OK
4. Sprawdź, czy na ekranie pojawił się komunikat ‘Bilans został zapisany’.
5. Sprawdź w bazie danych, czy istnieje bilans o następujących danych:  
 - id: 2  
 - data bilansu: 2017-11-01  
 - data wykonania: obecna data  
 Oraz pozycje bilansu o następujących danych:  
 - id: 2, ilość: 10, bilans Id: 2, towar Id : 1  
 - id: 3, ilość: 20, bilans Id: 2, towar Id : 2  
 - id: 4, ilość: 30, bilans Id: 2, towar Id : 3

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

ID	14.2.2.5 015 Bilansowanie – brak pozycji wydań i przyjęć
Nazwa	Bilansowanie – brak pozycji wydań i przyjęć Dla bilansu dla miesiąca poprzedzającego bilansowany miesiąc istnieje pozycja bilansu, dla bilansowanego miesiąca nie istnieją pozycje wydań i zamówień. Nowy bilans powinien zawierać jedną pozycję bilansu z towarem oraz ilością takimi jak w poprzednim bilansie.
Warunki wstępne	Użytkownik jest zalogowany. Aplikacja otwarta na ekranie głównym Aktualny miesiąc: 01/2018 W bazie istnieją następujące dane:  -Bilans Id:1 Data bilansu: 2017-10-01 Data wykonania bilansu: 2017-11-14  -Pozycja bilansu: id: 1 ilość: 10 bilans Id: 1 towar Id : 1  - Towar Id: 1 Kod:t01 Nazwa: Rower
Oczekiwany rezultat	Na ekranie pojawił się komunikat ‘Bilans został zapisany’ Zmiany w bazie danych: Zapisany został Bilans: Id:2 data bilansu: 2017-11-01 data wykonania: obecna data Pozycje bilansu: - id: 2, ilość: 10, bilans Id: 2, towar Id : 1

1. Kliknij ‘Sporządź bilans’
2. Sprawdź, czy na ekranie pojawiła się lista miesięcy z wierszami ‘Listopad 2017’ oraz ‘Grudzień 2017’
3. Wybierz ‘Listopad 2017’ i kliknij OK
4. Sprawdź, czy na ekranie pojawił się komunikat ‘Bilans został zapisany’.
5. Sprawdź w bazie danych, czy istnieje bilans o następujących danych:  
- id: 2

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

- data bilansu: 2017-11-01
  - data wykonania: obecna data
- Oraz pozycje bilansu o następujących danych:
- id: 2, ilość: 10, bilans Id: 2, towar Id : 1

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### 14.3 Automatyzacja testów funkcjonalnych

```

    @Test
    public void testWrongQuantitiesAmountBiggerThanLocated() {
        testGeneratorClient.add003_9();
        //logowanie
        clickOn( query: "#username");
        sleep(1000);
        write("admin");
        sleep(1000);
        clickOn( query: "#password");
        sleep(1000);
        write("topsecret");
        sleep(1000);
        clickOn( query: "#login");

        //kompletowanie
        sleep(1000);
        clickOn( query: "#completeOrder");

        //wybor zamówienia
        sleep(1000);
        clickOn((Node) lookup( query: ".table-cell").nth(2).query());
        sleep(3000);
        clickOn( query: "#OK");

        //wybor produktu
        sleep(1000);
        clickOn((Node) lookup( query: ".table-cell").nth(3).query());
        sleep(3000);
        clickOn( query: "#OK");

        //wybor lokalizacji
        sleep(1000);
        clickOn((Node) lookup( query: ".table-cell").nth(1).query());
        sleep(3000);
        clickOn( query: "#OK");

        //wybor ilosci
        sleep(3000);
        doubleClickOn( query: "#quantityTextField");
        sleep(1000);
        write("7");
        sleep(3000);
        clickOn( query: "#OK");

        //akceptacja dialogu o błędnej
        sleep(2000);
        clickOn( query: "#OKWrongQuantity");
    }
}

```

*Automatyzacja testu z niepoprawną ilością*

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

```

@Test
public void testOrderCompletionDelivery() {
    testGeneratorClient.add004_5_6_7_8_10();
    //logowanie
    clickOn( query: "#username");
    sleep(1000);
    write("admin");
    sleep(1000);
    clickOn( query: "#password");
    sleep(1000);
    write("topsecret");
    sleep(1000);
    clickOn( query: "#login");

    //kompletowanie
    sleep(1000);
    clickOn( query: "#completeOrder");

    //wybor zamówienia
    sleep(1000);
    clickOn((Node) lookup( query: ".table-cell").nth(2).query());
    sleep(3000);
    clickOn( query: "#OK");

    //wydawanie zamówienia
    sleep(1000);
    clickOn( query: "#send");
    sleep(3000);
    clickOn((Node) lookup(NodeMatchers.hasText( string: "Potwierdź")).nth(0).query());
    moveTo( x: 12, y: 12);

    //akceptacja wysyłki
    sleep(3000);
    clickOn( query: "#OKShipment");
}

```

*Automatyzacja testu wydania z wysyłką do klienta*

Automatyzacja nastąpiła w narzędziu dedykowanym do testowania interfejsów w technologii JavaFX. Narzędzie o nazwie TestFX udostępnia podstawowe metody interakcji z aplikacją. M.in. możliwość przesuwania kursorem po ekranie, klikania elementów, dwuklik itp.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 15. Badanie jakości projektu (JDepend)

# Server

Package	TC	CC	AC	Ca	Ce	A	I	D	V
com.StoreX	1	1	0	0	2	0.0%	100.0%	0.0%	1
com.StoreX.api	5	0	5	1	4	100.0%	80.0%	80.0%	1
com.StoreX.api.impl	6	6	0	0	20	0.0%	100.0%	0.0%	1
com.StoreX.common.datatypes.bo	17	16	1	8	3	6.0%	27.000002%	67.0%	1
com.StoreX.common.datatypes.enumerations	3	3	0	5	1	0.0%	17.0%	83.0%	1
com.StoreX.common.datatypes.to	18	17	1	2	4	6.0%	67.0%	28.0%	1
com.StoreX.persistence.entity.AuthorizationEntities	1	1	0	2	1	0.0%	33.0%	67.0%	1
com.StoreX.persistence.entity.BilansEntities	2	2	0	5	3	0.0%	38.0%	62.0%	1
com.StoreX.persistence.entity.PrzyjecieWydanieEntities	4	4	0	4	4	0.0%	50.0%	50.0%	1
com.StoreX.persistence.entity.TowarEntities	3	3	0	7	1	0.0%	12.0%	88.0%	1
com.StoreX.persistence.entity.UmieszczenieEntities	2	2	0	5	2	0.0%	29.0%	71.0%	1
com.StoreX.persistence.entity.ZamowienieEntities	5	4	1	5	4	20.0%	44.0%	36.0%	1
com.StoreX.persistence.repository.AuthorizationRepository	1	0	1	3	100.0%	75.0%	75.0%	1	
com.StoreX.persistence.repository.BilansRepository	2	0	2	2	4	100.0%	67.0%	67.0%	1
com.StoreX.persistence.repository.PrzyjecieWydanieRepository	4	0	4	2	3	100.0%	60.000004%	60.000004%	1
com.StoreX.persistence.repository.TowarRepository	3	0	3	1	2	100.0%	67.0%	67.0%	1
com.StoreX.persistence.repository.UmieszczenieRepository	2	0	2	3	3	100.0%	50.0%	50.0%	1
com.StoreX.persistence.repository.ZamowienieRepository	5	0	5	2	3	100.0%	60.000004%	60.000004%	1
com.StoreX.persistence.repository.impl	1	1	0	0	2	0.0%	100.0%	0.0%	1
com.StoreX.service.AuthorizationServices	1	0	1	6	3	100.0%	33.0%	33.0%	1
com.StoreX.service.BilansServices	4	0	4	2	4	100.0%	67.0%	67.0%	1
com.StoreX.service.HelperServices	16	0	16	3	6	100.0%	67.0%	67.0%	1
com.StoreX.service.PozycjeWydaniePrzyjecieServices	2	0	2	2	3	100.0%	60.000004%	60.000004%	1
com.StoreX.service.UmieszczenieServices	1	0	1	2	3	100.0%	60.000004%	60.000004%	1
com.StoreX.service.ZamowienieServices	4	0	4	2	3	100.0%	60.000004%	60.000004%	1
com.StoreX.service.impl.AuthorizationServicesImpl	1	1	0	0	8	0.0%	100.0%	0.0%	1
com.StoreX.service.impl.BilansServicesImpl	4	4	0	0	11	0.0%	100.0%	0.0%	1
com.StoreX.service.impl.HelperServicesImpl	16	16	0	0	14	0.0%	100.0%	0.0%	1
com.StoreX.service.impl.PozycjeWydaniePrzyjecieServicesImpl	2	2	0	0	8	0.0%	100.0%	0.0%	1
com.StoreX.service.impl.UmieszczenieServicesImpl	1	1	0	0	9	0.0%	100.0%	0.0%	1
com.StoreX.service.impl.ZamowienieServicesImpl	4	4	0	0	12	0.0%	100.0%	0.0%	1

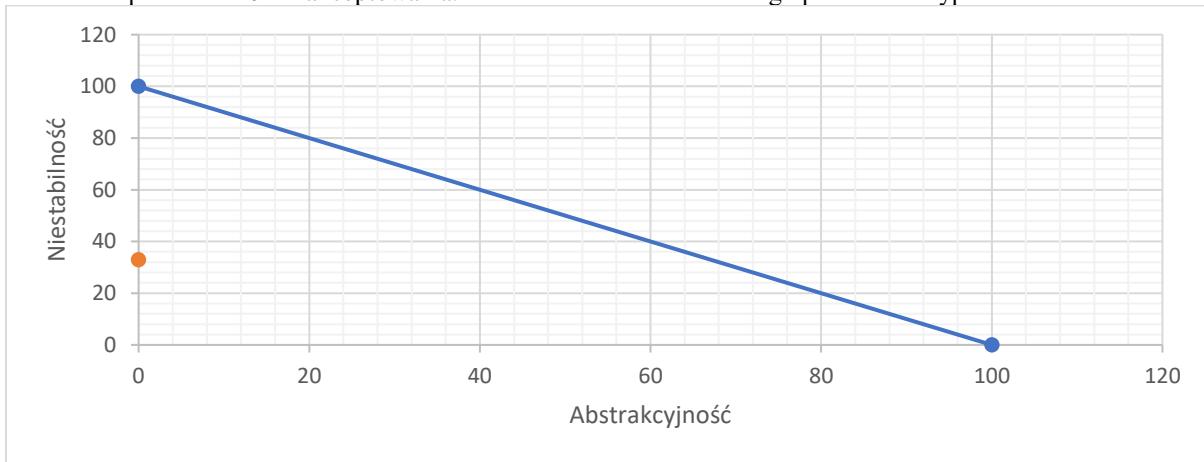
### Raport dla servera

com.StoreX					
Afferent Couplings		Efferent Couplings		Abstractness	
0		2		0.0%	
Abstract Classes		Concrete Classes		Used by Packages	
None		com.StoreX.StoreXApplication		None	Uses Packages java.lang org.springframework.boot

Pakiet poprawny – dystans zachowany, klasa zależna od stabilnych pakietów

com.StoreX.api					
Afferent Couplings		Efferent Couplings		Abstractness	
1		4		100.0%	
Abstract Classes		Concrete Classes		Used by Packages	
com.StoreX.api.AuthorizationApi		None		com.StoreX.api.impl	Uses Packages com.StoreX.common.datatypes.to java.lang java.util org.springframework.http
com.StoreX.api.BilansApi					
com.StoreX.api.PozycjaZamowieniaApi					
com.StoreX.api.UmieszczenieApi					
com.StoreX.api.ZamowienieZakupuApi					

Niestabilność na poziomie 80%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 20% - akceptowalna. Klasa zależna od niestabilnego pakietu datatypes.to



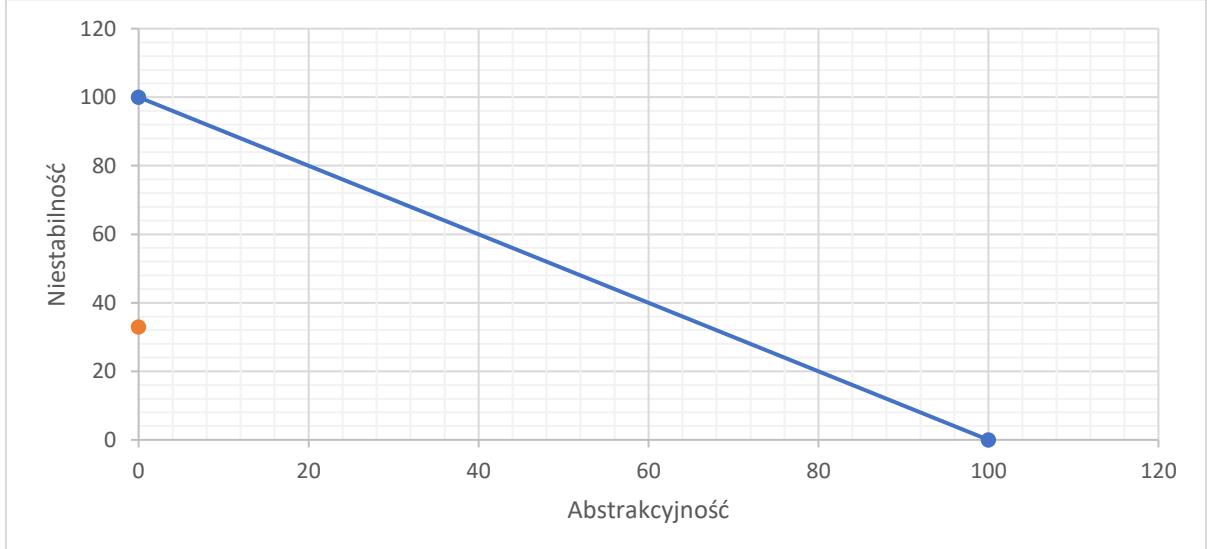
<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

com.StoreX.api.impl					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
0	20	0.0%	100.0%	0.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>		
None	com.StoreX.api.impl.AuthorizationApiImpl com.StoreX.api.impl.BilansApiImpl com.StoreX.api.impl.PozycjaZamowieniaApiImpl com.StoreX.api.impl.TestGeneratorImpl com.StoreX.api.impl.UmieszczenieApiImpl com.StoreX.api.impl.ZamowienieZakupuApiImpl	None	com.StoreX.api com.StoreX.common.datatypes.bo com.StoreX.common.datatypes.enumerations com.StoreX.common.datatypes.to com.StoreX.persistence.entity.BilansEntities com.StoreX.persistence.entity.PrzycieWydanieEntities com.StoreX.persistence.entity.TowarEntities com.StoreX.persistence.entity.UmieszczenieEntities com.StoreX.persistence.entity.ZamowienieEntities com.StoreX.service.AuthorizationServices com.StoreX.service.BilansServices com.StoreX.service.HelperServices com.StoreX.service.UmieszczenieServices com.StoreX.service.ZamowienieServices java.io java.lang java.util javax.naming org.modelmapper org.springframework.http		

Pakiet poprawny – dystans zachowany. Zależność od wielu niestabilnych pakietów – związanych z typami danych oraz serwisami.

com.StoreX.common.datatypes.bo					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
8	3	6.0%	27.000002%	67.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>		
com.StoreX.common.datatypes.bo.ZamowienieBO	com.StoreX.common.datatypes.bo.BilansBO com.StoreX.common.datatypes.bo.JednostkaBO com.StoreX.common.datatypes.bo.KategoriaBO com.StoreX.common.datatypes.bo.KlientBO com.StoreX.common.datatypes.bo.LokalizacjaBO com.StoreX.common.datatypes.bo.PozyczkaBO com.StoreX.common.datatypes.bo.PozyczkaWydanaBO com.StoreX.common.datatypes.bo.PozycjaWydanaBO com.StoreX.common.datatypes.bo.PozycjaZamowieniaBO com.StoreX.common.datatypes.bo.PrzycieZamowieniaBO com.StoreX.common.datatypes.bo.TowarBO com.StoreX.common.datatypes.bo.UmieszczenieBO com.StoreX.common.datatypes.bo.UserBO com.StoreX.common.datatypes.bo.WydanieZamowieniaBO com.StoreX.common.datatypes.bo.ZamowienieDostawyBO com.StoreX.common.datatypes.bo.ZamowienieZakupuBO	com.StoreX.api.impl com.StoreX.service.AuthorizationServices com.StoreX.service.BilansServices com.StoreX.service.impl.AuthorizationServicesImpl com.StoreX.service.impl.BilansServicesImpl com.StoreX.service.impl.PozyczkaWydanaServicesImpl com.StoreX.service.impl.UmieszczenieServicesImpl com.StoreX.service.impl.ZamowienieServicesImpl	com.StoreX.common.datatypes.enumerations java.lang java.util		

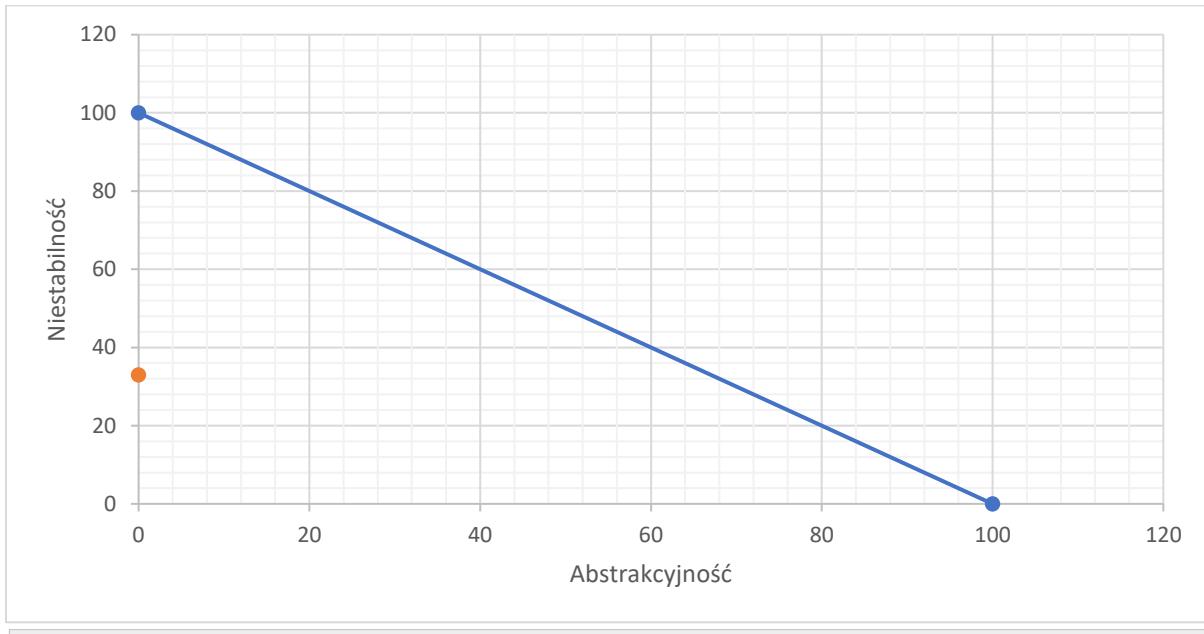
Pakiet zależny od stabilnych pakietów



com.StoreX.common.datatypes.enumerations					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
5	1	0.0%	17.0%	83.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>		
None	com.StoreX.common.datatypes.enumerations.StatusDostawy com.StoreX.common.datatypes.enumerations.StatusWydania com.StoreX.common.datatypes.enumerations.TypOdbioru	com.StoreX.api.impl com.StoreX.common.datatypes.bo com.StoreX.common.datatypes.to com.StoreX.persistence.entity.ZamowienieEntities com.StoreX.service.impl.ZamowienieServicesImpl	java.lang		

Pakiet zależny od stabilnych pakietów.

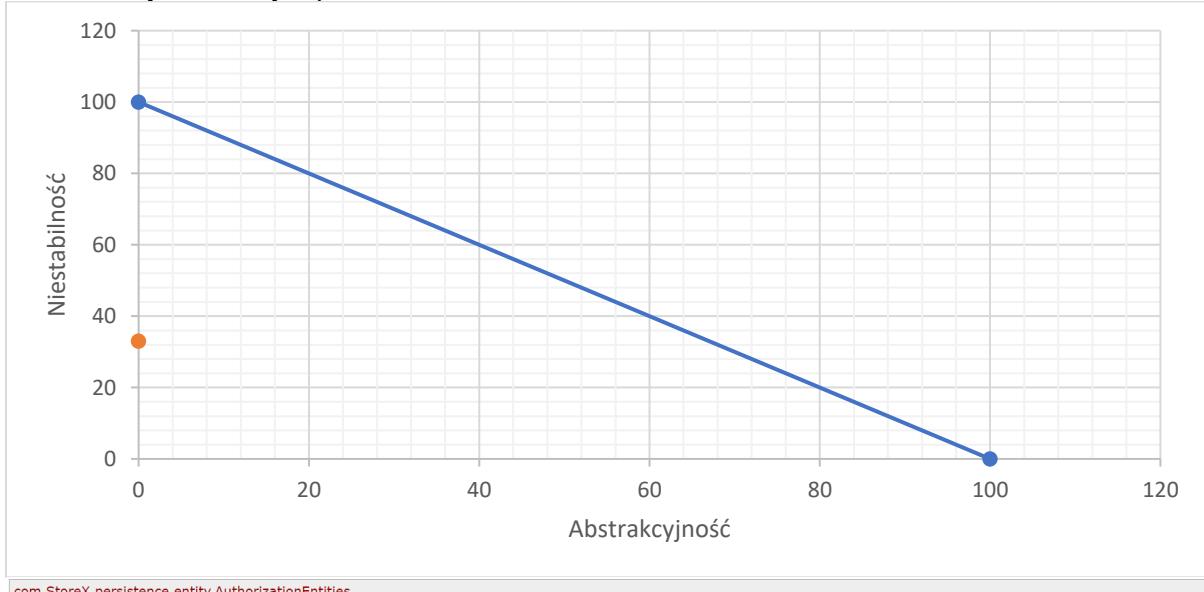
<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



#### com.StoreX.common.datatypes.to

Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
2	4	6.0%	67.0%	28.0%	
<b>Abstract Classes</b>		<b>Concrete Classes</b>			
com.StoreX.common.datatypes.to.ZamowienieTO		com.StoreX.common.datatypes.to.BilansTO com.StoreX.common.datatypes.to.KategoriaTO com.StoreX.common.datatypes.to.PozycjaTO com.StoreX.common.datatypes.to.KlientTO com.StoreX.common.datatypes.to.LokalizacjaTO com.StoreX.common.datatypes.to.PozycjaBalansuTO com.StoreX.common.datatypes.to.PozycjaPrzychecaTO com.StoreX.common.datatypes.to.PozycjaWydaniaTO com.StoreX.common.datatypes.to.PrzyjeteZamowieniaTO com.StoreX.common.datatypes.to.PrzyjeteZamowieniaTO com.StoreX.common.datatypes.to.SessionTO com.StoreX.common.datatypes.to.TowarTO com.StoreX.common.datatypes.to.UmorczenieTO com.StoreX.common.datatypes.to.UzytkownikTO com.StoreX.common.datatypes.to.WydanieZamowieniaTO com.StoreX.common.datatypes.to.ZamowienieDostawyTO com.StoreX.common.datatypes.to.ZamowienieZakupuTO			
		<b>Used by Packages</b>		<b>Uses Packages</b>	
		com.StoreX.api com.StoreX.api.impl		com.StoreX.common.datatypes.enumerations com.fasterxml.jackson.annotation java.lang java.util	

Pakiet zależny od stabilnych pakietów.

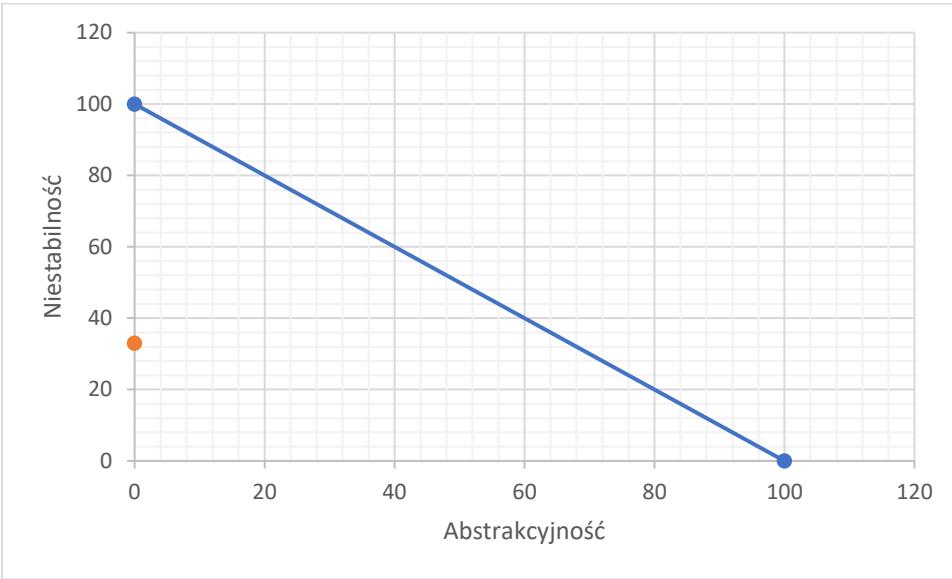


#### com.StoreX.persistence.entity.AuthorizationEntities

Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
2	1	0.0%	33.0%	67.0%	
<b>Abstract Classes</b>		<b>Concrete Classes</b>			
None		com.StoreX.persistence.entity.AuthorizationEntities.User			
		<b>Used by Packages</b>		<b>Uses Packages</b>	
		com.StoreX.persistence.repository.AuthorizationRepository com.StoreX.service.impl.AuthorizationServicesImpl		java.lang	

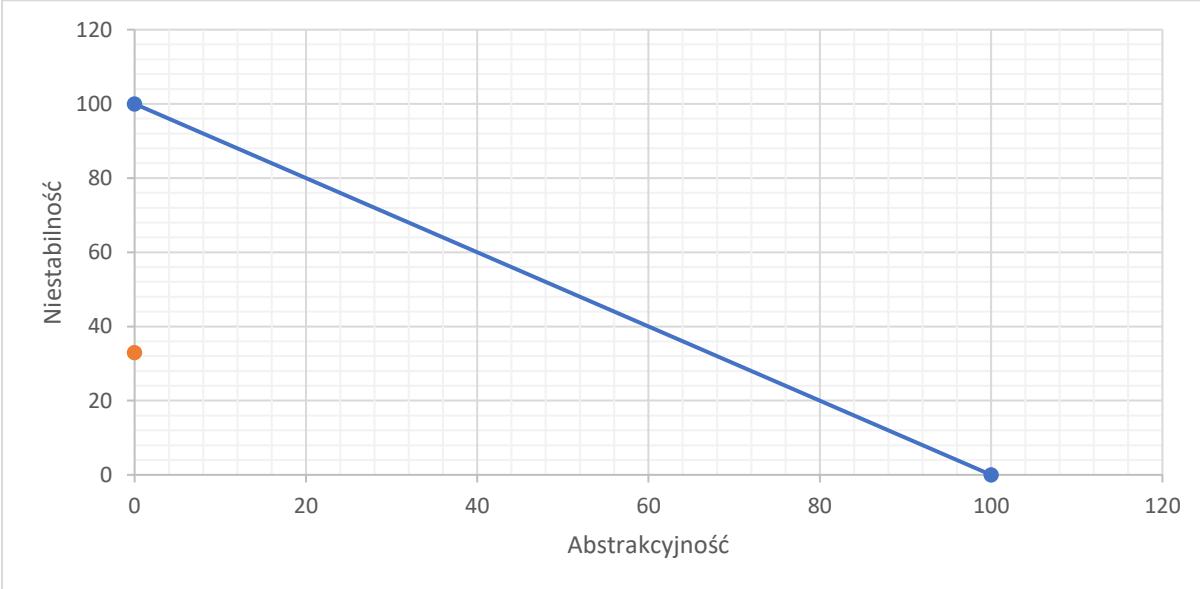
Pakiet zależny od stabilnych pakietów.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



com.StoreX.persistence.entity.BilansEntities					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
5	3	0.0%	38.0%	62.0%	
Abstract Classes	Concrete Classes				
None	com.StoreX.persistence.entity.BilansEntities.Bilans com.StoreX.persistence.entity.BilansEntities.PozycjaBilansu	com.StoreX.api.impl com.StoreX.persistence.repository.BilansRepository com.StoreX.service.HelperServices com.StoreX.service.impl.BilansServicesImpl com.StoreX.service.impl.HelperServicesImpl	com.StoreX.persistence.entity.TowarEntities java.lang java.util		

Pakiet zależny od stabilnych pakietów.



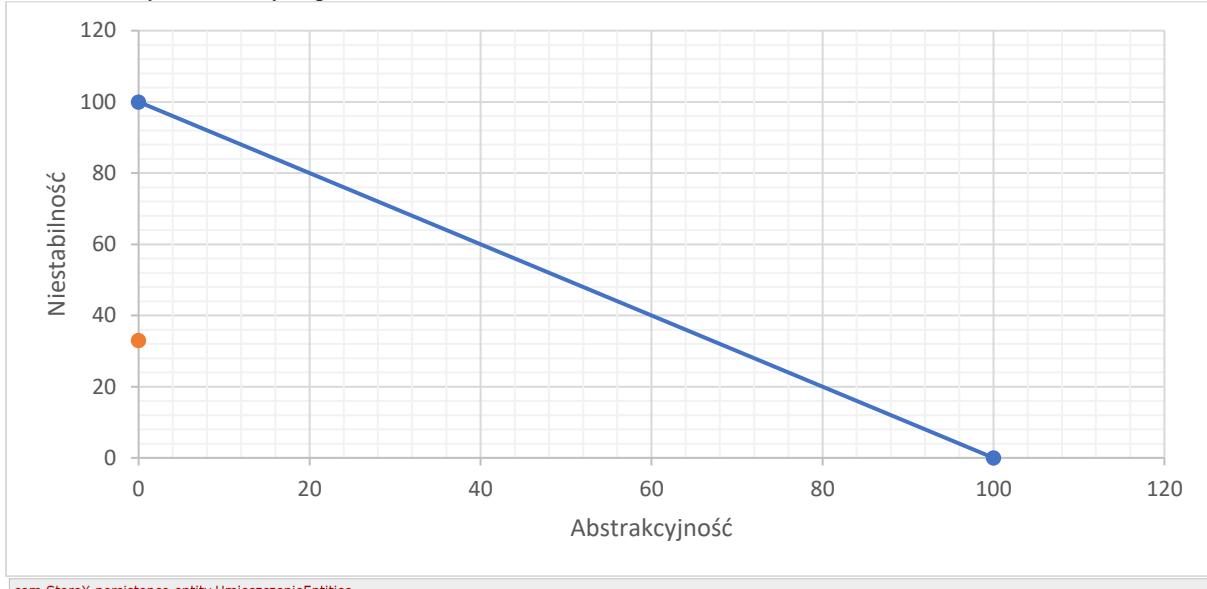
com.StoreX.persistence.entity.PrzyjecieWydanieEntities					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
4	4	0.0%	50.0%	50.0%	
Abstract Classes	Concrete Classes				
None	com.StoreX.persistence.entity.PrzyjecieWydanieEntities.PozycjaPrzyjecia com.StoreX.persistence.entity.PrzyjecieWydanieEntities.PozycjaWydania com.StoreX.persistence.entity.PrzyjecieWydanieEntities.PrzyjecieZamowienia com.StoreX.persistence.entity.PrzyjecieWydanieEntities.WydanieZamowienia	com.StoreX.api.impl com.StoreX.service.HelperServices com.StoreX.service.impl.HelperServicesImpl com.StoreX.service.impl.PozycjeWydanaPrzyjecieServicesImpl	com.StoreX.persistence.entity.TowarEntities com.StoreX.persistence.entity.ZamowienieEntities java.lang java.util		

Pakiet zależny od stabilnych pakietów, zależny też przechodnio od kolejnych pakietów, co pogłębia niestabilność.

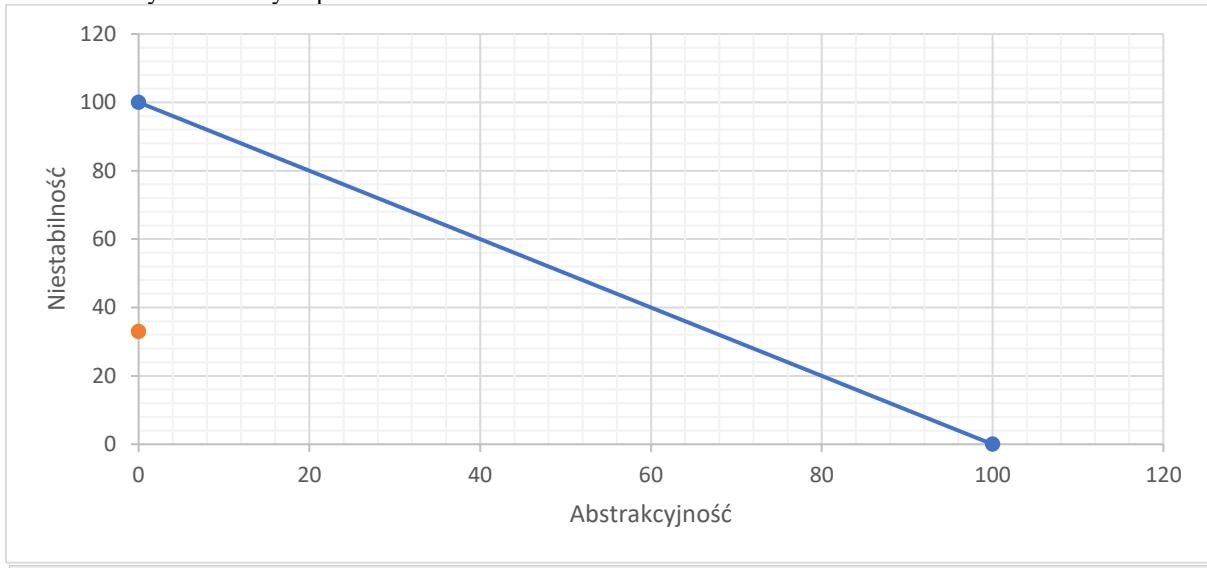
com.StoreX.persistence.entity.TowarEntities					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
7	1	0.0%	12.0%	88.0%	
Abstract Classes	Concrete Classes				
None	com.StoreX.persistence.entity.TowarEntities.Jednostka com.StoreX.persistence.entity.TowarEntities.Kategoria com.StoreX.persistence.entity.TowarEntities.Towar	com.StoreX.api.impl com.StoreX.persistence.entity.BilansEntities com.StoreX.persistence.entity.PrzyjecieWydanieEntities com.StoreX.persistence.entity.UmieszczenieEntities com.StoreX.persistence.entity.ZamowienieEntities com.StoreX.service.HelperServices com.StoreX.service.impl.HelperServicesImpl	java.lang		

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

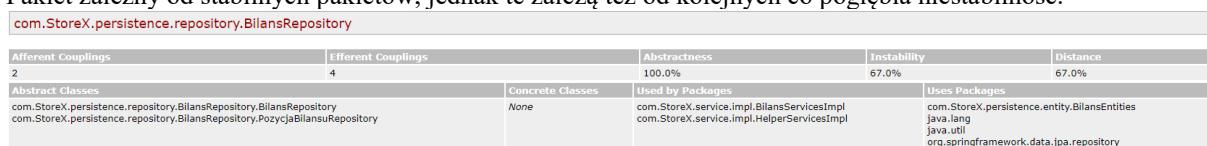
Pakiet zależny od stabilnych pakietów.



Pakiet zależny od stabilnych pakietów.



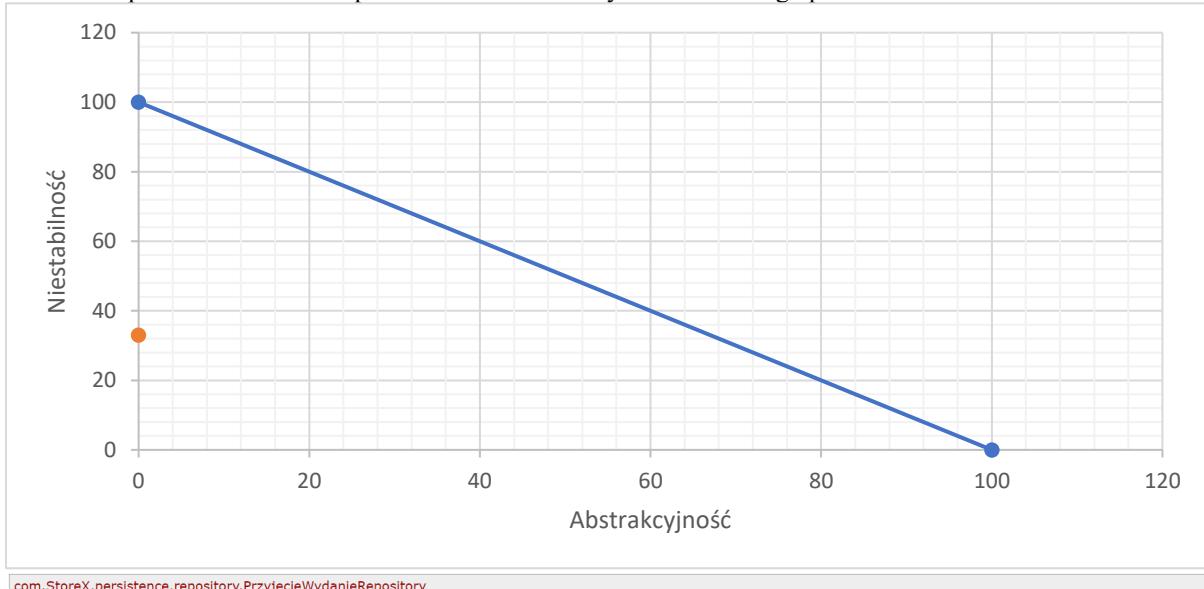
Pakiet zależny od stabilnych pakietów, jednak te zależą też od kolejnych co pogłębia niestabilność.



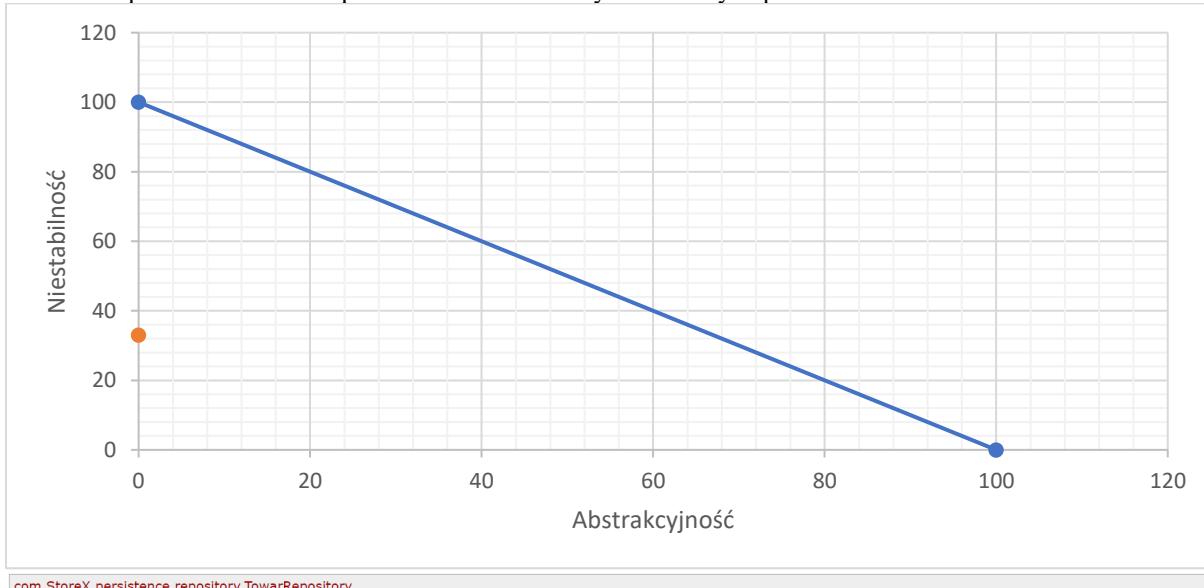
Niestabilność na poziomie 67%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 33% - akceptowalna. Pakiet zależny od niestabilnego pakietu BilansEntities.



Niestabilność na poziomie 60%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.



Niestabilność na poziomie 67%. Biblioteka java.lang jest standardową biblioteką języka, natomiast org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

com.StoreX.persistence.repository.UmieszczenieRepository					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used by Packages
3	3	100.0%	50.0%	50.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.StoreX.persistence.repository.UmieszczenieRepository.LokalizacjaRepository com.StoreX.persistence.repository.UmieszczenieRepository.UmieszczenieRepository	None	com.StoreX.service.impl.HelperServicesImpl com.StoreX.service.impl.UmieszczenieServicesImpl com.StoreX.service.impl.ZamowienieServicesImpl	java.lang java.util org.springframework.data.jpa.repository		

Niestabilność na poziomie 67%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

com.StoreX.persistence.repository.ZamowienieRepository					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used by Packages
2	3	100.0%	60.000004%	60.000004%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.StoreX.persistence.repository.ZamowienieRepository.KlientRepository com.StoreX.persistence.repository.ZamowienieRepository.PozycjaZamowieniaRepository com.StoreX.persistence.repository.ZamowienieRepository.BilansServicesRepository com.StoreX.persistence.repository.ZamowienieRepository.ZamowienieRepository com.StoreX.persistence.repository.ZamowienieRepository.ZamowienieZakupuRepository	None	com.StoreX.service.impl.HelperServicesImpl com.StoreX.service.impl.ZamowienieServicesImpl	java.lang java.util org.springframework.data.jpa.repository		

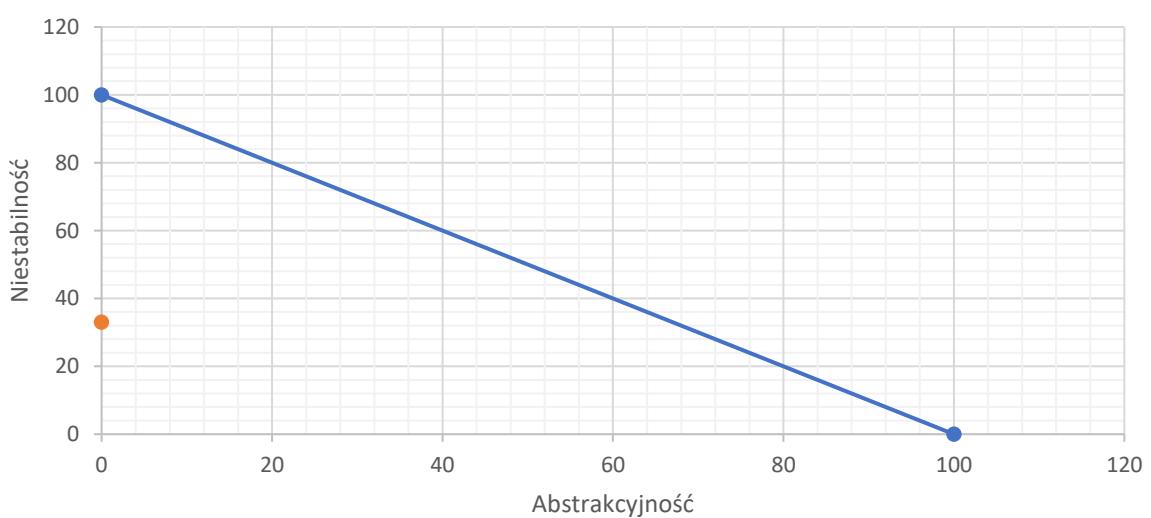
Niestabilność na poziomie 67%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast org.springframework jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

com.StoreX.persistence.repository.impl					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used by Packages
0	2	0.0%	100.0%	0.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
None	com.StoreX.persistence.repository.impl.UserRepositoryImpl\$1		None	java.lang java.util	

Pakiet poprawny – dystans zachowany. Pakiet zależny od stabilnych pakietów.

com.StoreX.service.AuthorizationServices					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used by Packages
6	3	100.0%	33.0%	33.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.StoreX.service.AuthorizationServices.AuthorizationService	None	com.StoreX.api.impl.AuthorizationServicesImpl com.StoreX.service.impl.BilansServicesImpl com.StoreX.service.impl.HelperServicesImpl com.StoreX.service.impl.UmieszczenieServicesImpl com.StoreX.service.impl.ZamowienieServicesImpl	com.StoreX.common.datatypes.bo java.lang java.util		

Niestabilność na poziomie 67%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 14% - akceptowalna. Pakiet zależny od stabilnych pakietów.



com.StoreX.service.BilansServices					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used by Packages
2	4	100.0%	67.0%	67.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.StoreX.service.BilansServices.BilansCreationService com.StoreX.service.BilansServices.BilansSearchService com.StoreX.service.BilansServices.PozyczkaBilansuCreationService com.StoreX.service.BilansServices.PozyczkaBilansuSearchService	None	com.StoreX.api.impl.BilansServicesImpl	com.StoreX.common.datatypes.bo java.lang java.util javax.naming		

Niestabilność na poziomie 67%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast javax.naming jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 33% - akceptowalna. Pakiet zależny od stabilnych pakietów.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

com.StoreX.service.PozycjeWydanPrzyjecServices				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
2	3	100.0%	60.000004%	60.000004%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>		<b>Uses Packages</b>
com.StoreX.service.PozycjeWydanPrzyjecServices.PozycjaPrzyjeciaSearchService com.StoreX.service.PozycjeWydanPrzyjecServices.PozycjaWydaniaSearchService	None	com.StoreX.service.impl.BilansServicesImpl com.StoreX.service.impl.PozycjeWydanPrzyjecServicesImpl		java.lang java.util javax.naming

Niestabilność na poziomie 60%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast javax.naming jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

com.StoreX.service.UmieszczenieServices				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
2	3	100.0%	60.000004%	60.000004%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>		<b>Uses Packages</b>
com.StoreX.service.UmieszczenieServices.UmieszczenieSearchService	None	com.StoreX.api.impl com.StoreX.service.impl.UmieszczenieServicesImpl		java.lang java.util javax.naming

Niestabilność na poziomie 60%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast javax.naming jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

com.StoreX.service.ZamowienieServices				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
2	3	100.0%	60.000004%	60.000004%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>		<b>Uses Packages</b>
com.StoreX.service.ZamowienieServices.PozycjaZamowieniaProceedService com.StoreX.service.ZamowienieServices.PozycjaZamowieniaSearchService com.StoreX.service.ZamowienieServices.ZamowienieZakupuSearchService com.StoreX.service.ZamowienieServices.ZamowienieZakupuUpdateService	None	com.StoreX.api.impl com.StoreX.service.impl.ZamowienieServicesImpl		java.lang java.util javax.naming

Niestabilność na poziomie 60%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast javax.naming jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 0% - akceptowalna. Pakiet zależny od stabilnych pakietów.

com.StoreX.service.impl.AuthorizationServicesImpl				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
0	8	0.0%	100.0%	0.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.AuthorizationServicesImpl.AuthorizationServiceImpl	None	com.StoreX.common.datatypes.bo com.StoreX.persistence.entity.AuthorizationEntities com.StoreX.persistence.repository.AuthorizationRepository com.StoreX.service.AuthorizationServices java.lang java.util org.modelmapper org.springframework.util	

Pakiet poprawny – dystans zachowany

com.StoreX.service.impl.BilansServicesImpl				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
0	11	0.0%	100.0%	0.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.BilansServicesImpl.BilansCreationServiceImpl com.StoreX.service.impl.BilansServicesImpl.BilansSearchServiceImpl com.StoreX.service.impl.BilansServicesImpl.PozycjaBilansuCreationServiceImpl com.StoreX.service.impl.BilansServicesImpl.PozycjaBilansuSearchServiceImpl	None	com.StoreX.common.datatypes.bo com.StoreX.persistence.entity.BilansEntities com.StoreX.persistence.repository.BilansRepository com.StoreX.service.BilansServices com.StoreX.service.HelperServices com.StoreX.service.PozycjeWydanPrzyjecServices java.lang java.util javax.naming org.modelmapper	

Pakiet poprawny – dystans zachowany

com.StoreX.service.impl.HelperServicesImpl				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
0	14	0.0%	100.0%	0.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.HelperServicesImpl.BilansServiceImpl com.StoreX.service.impl.HelperServicesImpl.KlientServiceImpl com.StoreX.service.impl.HelperServicesImpl.KategorieServiceImpl com.StoreX.service.impl.HelperServicesImpl.TowarServiceImpl com.StoreX.service.impl.HelperServicesImpl.LokalizacjaServiceImpl com.StoreX.service.impl.HelperServicesImpl.PozycjaBilansuServiceImpl com.StoreX.service.impl.HelperServicesImpl.PozycjaPrzyjeciaServiceImpl com.StoreX.service.impl.HelperServicesImpl.PozycjaZamowieniaServiceImpl com.StoreX.service.impl.HelperServicesImpl.WydanieZamowieniaServiceImpl com.StoreX.service.impl.HelperServicesImpl.ZamowienieDostawyServiceImpl com.StoreX.service.impl.HelperServicesImpl.ZamowienieServiceImpl com.StoreX.service.impl.HelperServicesImpl.ZamowienieZakupuServiceImpl	None	com.StoreX.persistence.entity.BilansEntities com.StoreX.persistence.entity.KategorieEntities com.StoreX.persistence.entity.TowarEntities com.StoreX.persistence.entity.UmieszczenieEntities com.StoreX.persistence.entity.ZamowienieEntities com.StoreX.persistence.repository.BilansRepository com.StoreX.persistence.repository.PrzyjecieWydanieRepository com.StoreX.persistence.repository.UmieszczenieRepository com.StoreX.persistence.repository.ZamowienieRepository com.StoreX.service.AuthorizationServices com.StoreX.service.HelperServices java.lang org.modelmapper	

Pakiet poprawny – dystans zachowany

com.StoreX.service.impl.PozycjeWydanPrzyjecServicesImpl				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
0	8	0.0%	100.0%	0.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>	<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.PozycjeWydanPrzyjecServicesImpl.PozycjaPrzyjeciaSearchServiceImpl com.StoreX.service.impl.PozycjeWydanPrzyjecServicesImpl.PozycjaWydaniaSearchServiceImpl	None	com.StoreX.common.datatypes.bo com.StoreX.persistence.entity.PrzyjecieWydanieEntities com.StoreX.persistence.repository.PrzyjecieWydanieRepository com.StoreX.service.PozycjeWydanPrzyjecServices java.lang java.util javax.naming org.modelmapper	

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

### Pakiet poprawny – dystans zachowany

com.StoreX.service.impl.UmieszczenieServicesImpl					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
0	9	0.0%	100.0%	0.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>		<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.UmieszczenieServicesImpl.UmieszczenieServiceImpl		None	com.StoreX.common.datatypes.bo com.StoreX.persistence.entity.UmieszczenieEntities com.StoreX.persistence.repository.UmieszczenieRepository com.StoreX.service.AuthorizationServices com.StoreX.service.UmieszczenieServices java.lang java.util javax.naming org.modelmapper	

### Pakiet poprawny – dystans zachowany

com.StoreX.service.impl.ZamowienieServicesImpl					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
0	12	0.0%	100.0%	0.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>		<b>Used by Packages</b>	<b>Uses Packages</b>	
None	com.StoreX.service.impl.ZamowienieServicesImpl.PozycjaZamowieniaProceedServiceImpl com.StoreX.service.impl.ZamowienieServicesImpl.PozycjaZamowieniaSearchServiceImpl com.StoreX.service.impl.ZamowienieServicesImpl.ZamowienieZakupuSearchServiceImpl com.StoreX.service.impl.ZamowienieServicesImpl.ZamowienieZakupuUpdateServiceImpl		None	com.StoreX.common.datatypes.bo com.StoreX.common.datatypes.enumerations com.StoreX.persistence.entity.UmieszczenieEntities com.StoreX.persistence.entity.ZamowienieEntities com.StoreX.persistence.repository.UmieszczenieRepository com.StoreX.persistence.repository.ZamowienieRepository com.StoreX.service.AuthorizationServices com.StoreX.service.ZamowienieServices java.lang java.util javax.naming org.modelmapper	

### Pakiet poprawny – dystans zachowany

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

# Client

Package	TC	CC	AC	Ca	Ce	A	I	D	V
com.crashcourse.restclient.api	5	0	5	3	3	100.0%	50.0%	50.0%	1
com.crashcourse.restclient.api.impl	9	9	0	1	8	0.0%	89.0%	11.0%	1
com.crashcourse.restclient.common	1	1	0	0	1	0.0%	100.0%	0.0%	1
com.crashcourse.restclient.controller	14	13	1	1	17	7.0%	94.0%	2.0%	1
com.crashcourse.restclient.controller.basic	3	2	1	3	8	33.0%	73.0%	6.0%	1
com.crashcourse.restclient.datatype	18	17	1	5	4	6.0%	44.0%	50.0%	1
com.crashcourse.restclient.datatype.enumeration	3	3	0	2	1	0.0%	33.0%	67.0%	1
com.crashcourse.restclient.main	1	1	0	0	15	0.0%	100.0%	0.0%	1
com.crashcourse.restclient.main.config	4	4	0	3	8	0.0%	73.0%	27.000002%	1
com.crashcourse.restclient.model	16	15	1	2	4	6.0%	67.0%	27.000002%	1
com.crashcourse.restclient.view	1	1	0	3	3	0.0%	50.0%	50.0%	1

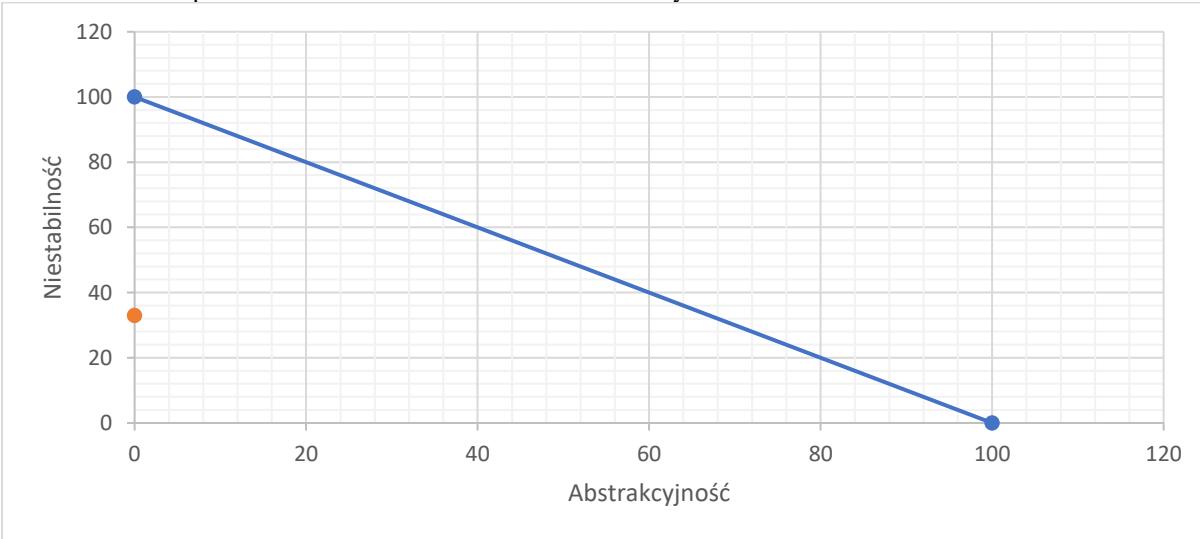
## Raport dla klienta

com.crashcourse.restclient.api					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
3	3	100.0%	50.0%	50.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.crashcourse.restclient.api.AuthorizationRestServiceClient	None				
com.crashcourse.restclient.api.BilansRestServiceClient					
com.crashcourse.restclient.api.PozycjaZamowieniaRestServiceClient					
com.crashcourse.restclient.api.UmieszczenieRestServiceClient					
com.crashcourse.restclient.api.ZamowienieZakupuRestServiceClient					

Niestabilność na poziomie 50%. Biblioteki java.lang oraz java.util są standardowymi bibliotekami języka, natomiast javax.naming jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można by określić na poziomie 25% - akceptowalna.

com.crashcourse.restclient.api.impl					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
1	8	0.0%	89.0%	11.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
None	com.crashcourse.restclient.api.impl.AuthorizationRestServiceClientImpl				
	com.crashcourse.restclient.api.impl.AuthorizationRestServiceClientImpl\$1				
	com.crashcourse.restclient.api.impl.BilansRestServiceClientImpl				
	com.crashcourse.restclient.api.impl.BilansRestServiceClientImpl\$1				
	com.crashcourse.restclient.api.impl.PozycjaZamowieniaRestServiceClientImpl				
	com.crashcourse.restclient.api.impl.PozycjaZamowieniaRestServiceClientImpl\$1				
	com.crashcourse.restclient.api.impl.PozycjaZamowieniaRestServiceClientImpl\$2				
	com.crashcourse.restclient.api.impl.UmieszczenieRestServiceClientImpl				
	com.crashcourse.restclient.api.impl.UmieszczenieRestServiceClientImpl\$1				
	com.crashcourse.restclient.api.impl.ZamowienieZakupuRestServiceClientImpl				
	com.crashcourse.restclient.api.impl.ZamowienieZakupuRestServiceClientImpl\$1				

Zależność wychodząca związana z konfiguracją serwisów autoryzacyjnych (Spring Beans) wprowadza zafalszowanie do pakietu. Oczekiwana niestabilność 100% - Dystans 0%



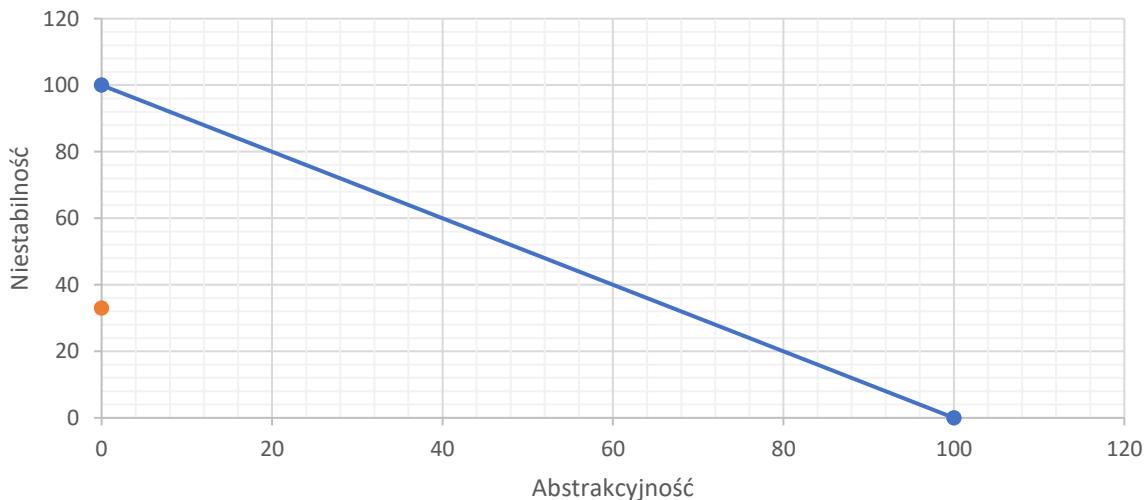
com.crashcourse.restclient.common					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
0	1	0.0%	100.0%	0.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
None	com.crashcourse.restclient.common.SerialVersionUIDProvider				

Pakiet poprawny – dystans zachowany

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

com.crashcourse.restclient.controller					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
1	17	7.0%	94.0%	2.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>				
com.crashcourse.restclient.controller.ArtifactsBaseController	com.crashcourse.restclient.controller.BalanceAcceptanceController com.crashcourse.restclient.controller.DefaultDialogController com.crashcourse.restclient.controller.InformationErrorController com.crashcourse.restclient.controller.LoginController com.crashcourse.restclient.controller.MainScreenController com.crashcourse.restclient.controller.OKEmailController com.crashcourse.restclient.controller.OKWaitingOrdersController com.crashcourse.restclient.controller.OKOutOfProductController com.crashcourse.restclient.controller.OKWrongDateController com.crashcourse.restclient.controller.OKWrongQuantityController com.crashcourse.restclient.controller.QuantityProductController com.crashcourse.restclient.controller.ShipmentDataController		Used by Packages	Used Packages	
			com.crashcourse.restclient.controller.basic	com.crashcourse.restclient.api com.crashcourse.restclient.controller.basic com.crashcourse.restclient.datatype com.crashcourse.restclient.main.config com.crashcourse.restclient.model com.crashcourse.restclient.view	
				java java.lang java.text java.util java.util.regex javafx.beans.property javafx.collections javafx.event javafx.scene.control javafx.stage javax.security.auth.login	

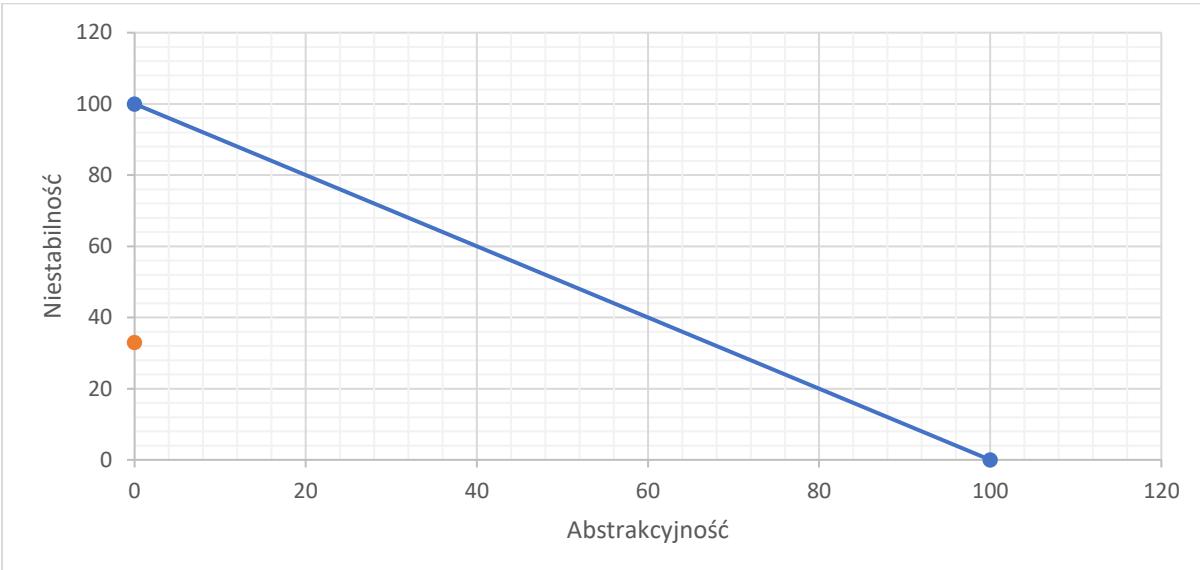
Pakiet poprawny – dystans zachowany, jednak refaktoryzacja może zostać rozważona, ze względu na nieprzewidywany dalszy rozwój pakietu basic i zmiany w nim – zagrożenie pomijalne



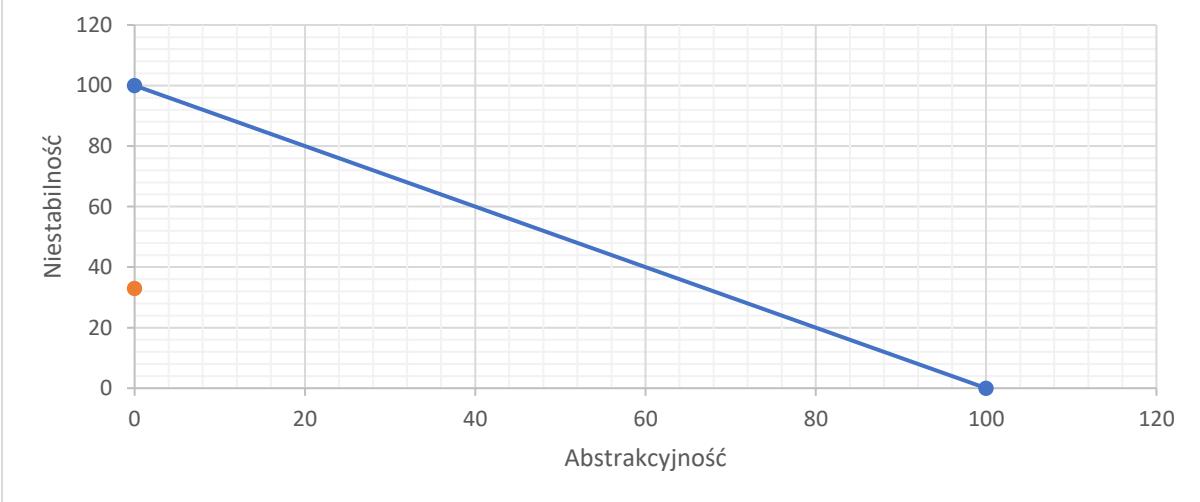
com.crashcourse.restclient.controller.basic					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	
3	8	33.0%	73.0%	6.0%	
<b>Abstract Classes</b>	<b>Concrete Classes</b>		Used by Packages	Used Packages	
com.crashcourse.restclient.controller.basic.DialogController	com.crashcourse.restclient.controller.basic.DialogManager com.crashcourse.restclient.controller.basic.ScreensConfiguration		com.crashcourse.restclient.controller com.crashcourse.restclient.main com.crashcourse.restclient.main.config	com.crashcourse.restclient.controller com.crashcourse.restclient.model com.crashcourse.restclient.view java.io java.lang java.util javafx.stage org.slf4j	

Niestabilność na poziomie 73%. Biblioteki java.lang, java.io, java.util są standardowymi bibliotekami języka, natomiast javafx.stage jest biblioteką frameworkową, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 50%.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

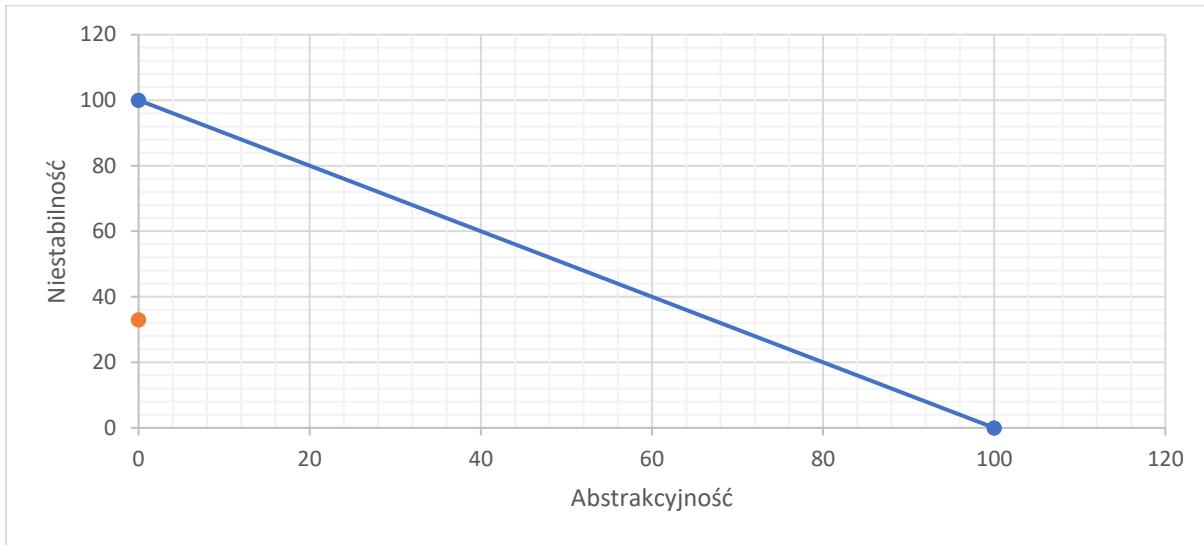


com.crashcourse.restclient.datatype				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
5	4	6.0%	44.0%	50.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>			
com.crashcourse.restclient.datatype.ZamowienieTO	com.crashcourse.restclient.datatype.BilansTO com.crashcourse.restclient.datatype.JednostkaTO com.crashcourse.restclient.datatype.KategoriaTO com.crashcourse.restclient.datatype.KlientTO com.crashcourse.restclient.datatype.LokalizacjaTO com.crashcourse.restclient.datatype.PozyczajBilansTO com.crashcourse.restclient.datatype.PozyczajPrzypadekTO com.crashcourse.restclient.datatype.PozyczajZadaniaTO com.crashcourse.restclient.datatype.PozyczjaZamowieniaTO com.crashcourse.restclient.datatype.PrzyjcieZamowieniaTO com.crashcourse.restclient.datatype.SessionTO com.crashcourse.restclient.datatype.TowarTO com.crashcourse.restclient.datatype.UzytkowniczeZamowieniaTO com.crashcourse.restclient.datatype.UserTO com.crashcourse.restclient.datatype.WydzialZamowieniaTO com.crashcourse.restclient.datatype.ZamowienieDostawyTO com.crashcourse.restclient.datatype.ZamowienieZakupuTO		com.crashcourse.restclient.api com.crashcourse.restclient.impl com.crashcourse.restclient.controller com.crashcourse.restclient.main.config com.crashcourse.restclient.model	com.crashcourse.restclient.datatype.enumeration com.fasterxml.jackson.annotation java.lang java.util



com.crashcourse.restclient.datatype.enumeration				
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
2	1	0.0%	33.0%	67.0%
<b>Abstract Classes</b>	<b>Concrete Classes</b>			
None	com.crashcourse.restclient.datatype.enumeration.StatusDostawy com.crashcourse.restclient.datatype.enumeration.StatusWydania com.crashcourse.restclient.datatype.enumeration.TypOdbioru		com.crashcourse.restclient.datatype com.crashcourse.restclient.model	java.lang

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>



com.crashcourse.restclient.main					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used Packages
0	15	0.0%	100.0%	0.0%	com.crashcourse.restclient.controller.basic com.crashcourse.restclient.main.config com.crashcourse.restclient.view java.lang java.util javafx.scene.control javafx.scene.input javafx.scene.layout javafx.stage org.glassfish.jersey.internal.util org.springframework.context org.springframework.context.annotation org.testfx.api org.testfx.framework.junit

### Pakiet poprawny – dystans zachowany

com.crashcourse.restclient.model					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used Packages
2	4	6.0%	67.0%	27.000002%	com.crashcourse.restclient.datatype com.crashcourse.restclient.datatype.enumeration java.lang javafx.beans.property

com.crashcourse.restclient.view					
Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance	Used Packages
3	3	0.0%	50.0%	50.0%	com.crashcourse.restclient.controller com.crashcourse.restclient.controller.basic com.crashcourse.restclient.main java.lang javafx.collections javafx.scene.control

Niestabilność na poziomie 50%. Biblioteka java.lang jest standardową biblioteką języka, natomiast javafx.collections oraz javafx.scene są bibliotekami frameworkowymi, wpływ na stabilność jest marginalny. Niestabilność można określić na poziomie 0% - akceptowalna.

Cycles					
[ summary ] [ packages ] [ cycles ] [ explanations ]					
Package	Package Dependencies				
com.crashcourse.restclient.api.impl	com.crashcourse.restclient.main.config com.crashcourse.restclient.api.impl				
com.crashcourse.restclient.controller	com.crashcourse.restclient.controller.basic				
com.crashcourse.restclient.controller.basic	com.crashcourse.restclient.controller				
com.crashcourse.restclient.main	com.crashcourse.restclient.controller.basic com.crashcourse.restclient.controller				
com.crashcourse.restclient.main.config	com.crashcourse.restclient.api.impl com.crashcourse.restclient.main.config				

Nie ustrzeżono się także przed cyklami, powinna zostać rozważona refaktoryzacja pakietów i usunięcie cykli. Pakiet basic jest podpakiem kontrolera, nie istnieje więc duże bezpieczeństwo. Jednak posiadanie odwołań do pakietu kontrolera w pakiecie głównym może prowadzić do problemów w dalszym rozwoju kodu.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

Klasy posiadające za dużą niestabilność ze względu na zależność od pakietów standardowych nie muszą zostać zmieniane.

<b>Storex</b>	
Jan Borowiak, Kamil Mlynarczyk	Data: <20/01/2018>

## 16. Podsumowanie

Projekt został zakończony sukcesem pod względem spełnienia wymagań. Dostarczony został produkt wraz z dokumentacją.

Zastosowana została architektura klient-serwer, wzorzec MVC (w kliencie).

Wykorzystane wzorce *projektowe*.

**Wartość edukacyjna:**

- Poprawa zrozumienia architektury Klient-Serwer, wzorca MVC
- Poprawa umiejętności implementacyjnych
- Poprawa umiejętności pracy w zespole:
  1. Projektowym
  2. Developerskim (Korzystanie z systemu kontroli wersji GIT)
- Poprawa zrozumienia dobrych praktyk projektowania bazy danych
- Poznanie wzorców projektowych
- Zwiększenia doświadczenia pracy nastawionej na spełnienie wymagań klienta
- Poznane frameworki: Spring, Hibernate
- Zrozumienie licznych problemów związanych z projektowaniem systemu oraz sposobów ich rozwiązania
- Poprawa umiejętności pisania testów \
- Nauka od podstaw JavaFX

**Popełnione błędy:**

- Implementacja nie dostarczyła prawdopodobnie wystarczająco wydajnych procesów biznesowych. Interfejs graficzny aplikacji został oparty o wrażenia z użytkowania prawdziwego systemu w magazynie jednego z projektantów aplikacji.
- System nie jest imponujący graficznie ze względu na ograniczone możliwości animacyjne i ograniczeniowe związane z technologią wykonania.
- W międzyczasie produkcji kodu wyszła nowa wersja Javy. Dziewiąta odsłona nie pozwala na korzystanie z zaimplementowanej aplikacji, wymagana byłaby natychmiastowa migracja na nową wersję JavaFX (3.0).
- Aplikacja prawdopodobnie byłaby prostsza w ewentualnym wprowadzaniu jej na platformy klienckie, gdyby została napisana jako aplikacja webowa.

Z pespektywy czasu należy zauważać wysokie zaangażowanie oraz zgranie zespołu implementacyjnego jak i projektowego (ten sam zespół). Projektowi przywiecały wartości edukacyjne, nie faktyczny zarobek, zastosowane wzorce i technologie zostały zastosowane drogą konsensusu między uczestnikami projektu. W różnych fazach projektowania były brane pod uwagę inne technologie.