

Automatyka i Robotyka  
Informatyka w Sterowaniu i Zarządzaniu  
Systemy Rozproszone  
2024

## **Rozproszony system zamówień**

Autorzy:

Łukasz Gakan

Artur Mazurkiewicz

Andrzej Janik

# Spis treści

1. Specyfikacja projektu .....	3
2. Scenariusze użycia .....	4
3. Diagram przypadków użycia (UC) .....	6
4. Model danych .....	7
5. Interfejsy oraz klasy .....	8
6. Aplikacja.....	9
7. Testy rozproszonego systemu zamówień.....	16
8. Podsumowanie .....	27

## 1. Specyfikacja projektu

### Cel systemu

Rozproszony system zamówień ma na celu umożliwienie klientom składania zamówień na różnorodne towary oraz ich dostarczanie poprzez integrację różnych dostawców w jedną platformę. System zapewnia klientom łatwy dostęp do ofert różnych dostawców oraz umożliwia skuteczne przeglądanie, porównywanie oraz składanie zamówień.

### Zakres systemu

Rozproszony system zamówień obejmuje szereg funkcji i modułów:

- Moduł przeglądania i wyboru ofert sprzedawców - klienci systemu mają dostęp do interfejsu umożliwiającego przeglądanie ofert różnych dostawców.
- Moduł składania zamówień - klienci przychodzą z listą zakupową, która zawiera określone towary oraz ich ilości do zakupu. Bazując na ofertach różnych sprzedawców system dodaje towary do koszyka zakupowego, a następnie system przeprowadza transakcje tak aby one były najbardziej optymalne.
- Zarządzanie ofertami sprzedawców - dostawcy mają możliwość dodawania, edytowania ofert towarów oraz zarządzania ich dostępnością i stanem magazynowym.

## 2. Scenariusze użycia

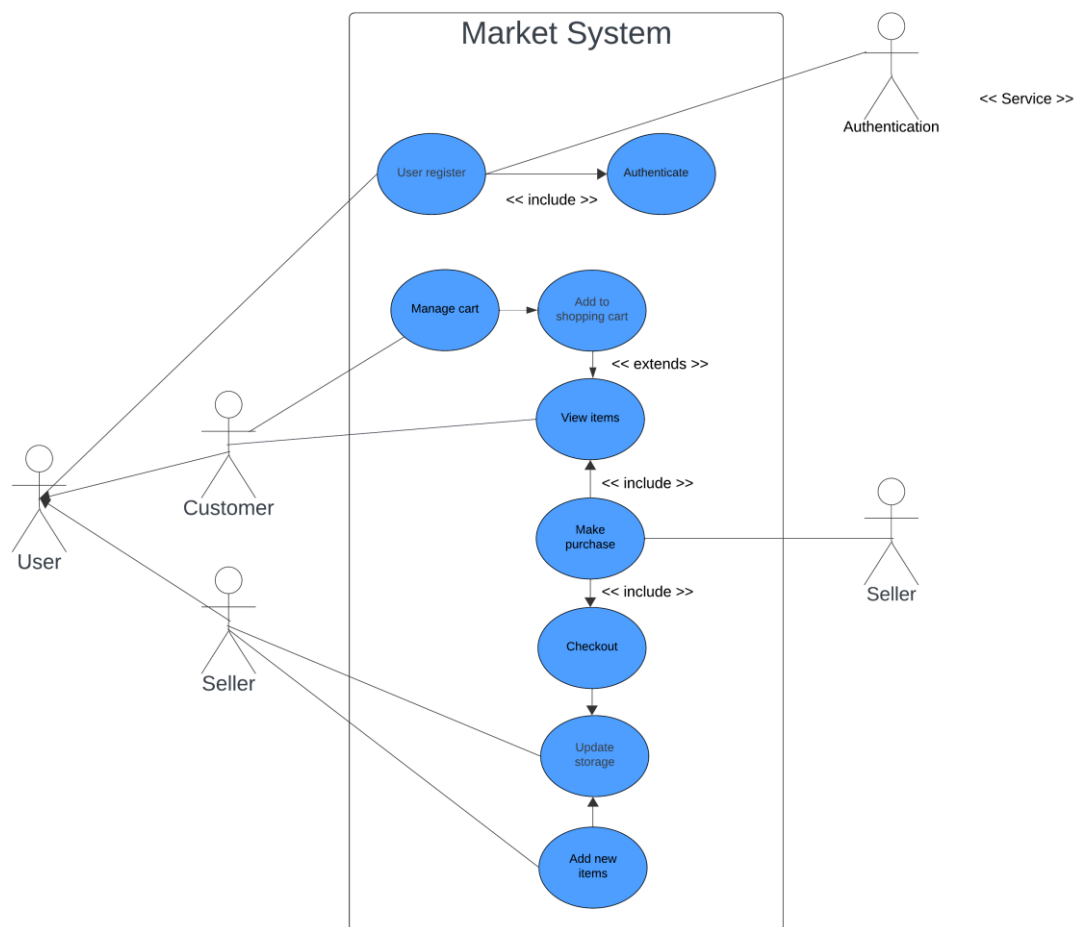
Pierwszy scenariusz użycia:

Identyfikator i nazwa przypadku użycia:	SR-WZM – Wykonanie zamówienia przez klienta		
Utworzony przez:	AJ	Data utworzenia:	11.03.2024
Aktor główny:	Klient	Aktorzy drugorzędni:	
Wyzwalacz	Klient zgłasza chęć zalogowania się do systemu zamówień.		
Opis:	Każdy klient <b>[customer_id]</b> , który będzie zamawiał w systemie musi zostać uwierzytelniony		
Warunki początkowe:	1. Klient posiada niepowtarzalny identyfikator <b>[customer_id]</b> , jednoznacznie identyfikujący tego klienta 2. System posiada zdefiniowane warunki uwierzytelnienia klienta w systemie		
Warunki końcowe:	Klient jest uwierzytelniony, pojawia się na liście klientów <b>[customers_list]</b> i może korzystać z systemu		
Przeptyw normalny:	Klient pokazuje listę zakupów składającą się z określonych towarów i ich ilości <b>[shopping_list]</b> . Następnie składane jest zamówienie i następuje jego kompletacja <b>[transaction]</b> .		
Przeptyw alternatywny:	Klient przegląda dostępne oferty sprzedawców i na ich podstawie wykonuje zamówienie.		
Wyjątki:	1.0.W1 Niemożliwe jest skompletowanie zamówienia u jednego sprzedawcy.		
Rozszerzenie scenariusza bazowego:	Klient pokazuje listę zakupów składającą się z określonych towarów i ich ilości. Następnie składane jest zamówienie i podczas składania zamówienia okazuje się, że sprzedawca nie posiada takiej ilości towarów to następuje jak najbardziej optymalne rozdzielenie zamówienia.		
Priorytet:	Wysoki		
Częstotliwość użycia:	Bardzo często, największe użycie w godzinach pracy 8-16.		
Reguły biznesowe:	Wyświetlenie aktualnej dostępności towarów u sprzedawców <b>[item_view]</b> .		
Inne informacje:	Usunięcie klienta odbywa się na poziomie systemu.		
Założenia wstępne:	1. Istnienie platformy autoryzacji klientów 2. Istnienie systemu zarządzania zamówieniami		

Drugi scenariusz użycia:

Identyfikator i nazwa przypadku użycia:	SR-ZAS – Zalogowanie sprzedawcy do systemu		
Utworzony przez:	AJ	Data utworzenia:	12.03.2024
Aktor główny:	Sprzedawca	Aktorzy drugorzędni:	
Wyzwalacz	Sprzedawca zgłasza chęć zalogowania się do systemu zamówień.		
Opis:	Każdy sprzedawca <b>[seller_id]</b> , który będzie wystawiał swoje towary w systemie musi zostać uwierzytelniony		
Warunki początkowe:	1. Sprzedawca posiada niepowtarzalny identyfikator <b>[customer_id]</b> , jednoznacznie identyfikujący tego sprzedawcę 2. System posiada zdefiniowane warunki uwierzytelnienia sprzedawcyw systemie		
Warunki końcowe:	Sprzedawca jest uwierzytelniony, pojawia się na liście klientów <b>[sellers_list]</b> i może korzystać z systemu		
Przeptyw normalny:	Sprzedawca wystawia swoje towary do systemu <b>[seller_items]</b> . Przegląda aktualne zamówienia i je kompletuje. <b>[complete_transaction]</b> .		
Wyjątki:	2.0.W1 Sprzedawca nie posiada wystarczającej ilości przedmiotów w magazynie.		
Rozszerzenie scenariusza bazowego:	Jeśli sprzedawca nie posiada wystarczającej ilości towarów do skompletowania zamówienia to przekazuje informacje o innych sprzedających w systemie posiadających dane towary oraz rozpoczyna proces uzupełnienia magazynu.		
Priorytet:	Wysoki		
Częstotliwość użycia:	Bardzo częsty, głównie w godzinach pracy sprzedawców (8-16).		
Reguły biznesowe:	Wyświetlenie aktualnej dostępności towarów u sprzedawcy <b>[item_view]</b> . Uzupełnienie towarów w magazynie u sprzedawcy. <b>[storage_replenishment]</b> .		
Inne informacje:	Usunięcie sprzedawcy odbywa się na poziomie systemu.		
Założenia wstępne:	1. Istnienie platformy autoryzacji sprzedawców 2. Istnienie systemu zarządzania zamówieniami		

### 3. Diagram przypadków użycia (UC)

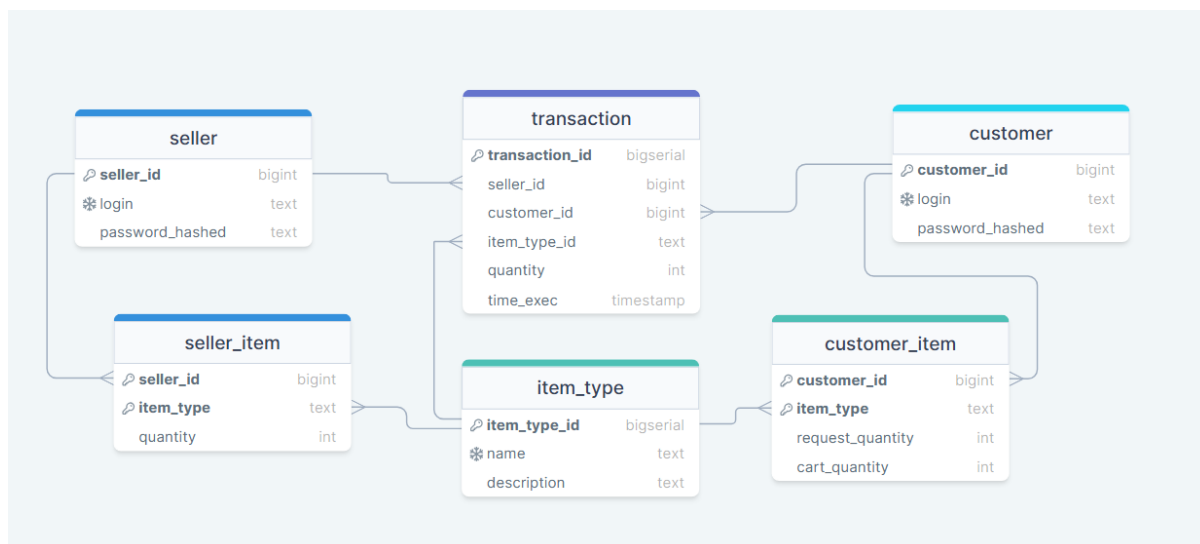


Rys. 1. Diagram przypadków użycia

Powyższy Rysunek 1 przedstawia diagram przypadków użycia dla rozproszonego systemu zamówień, ilustrujący interakcje między aktorami (użytkownikami systemu) – klientami i sprzedawcami – a systemem w ramach różnych przypadków użycia, czyli funkcjonalności, które mogą być wykorzystywane przez aktorów. Diagram ten służy do opisu funkcji systemu na etapie analizy wymagań, ułatwiając komunikację w zespole projektowym oraz definiowanie zakresu projektu.

#### 4. Model danych

W rozproszonym systemie zamówień model danych musi być odpowiednio zaprojektowany, aby umożliwić skuteczne zarządzanie towarami, zamówieniami, klientami i dostawcami.

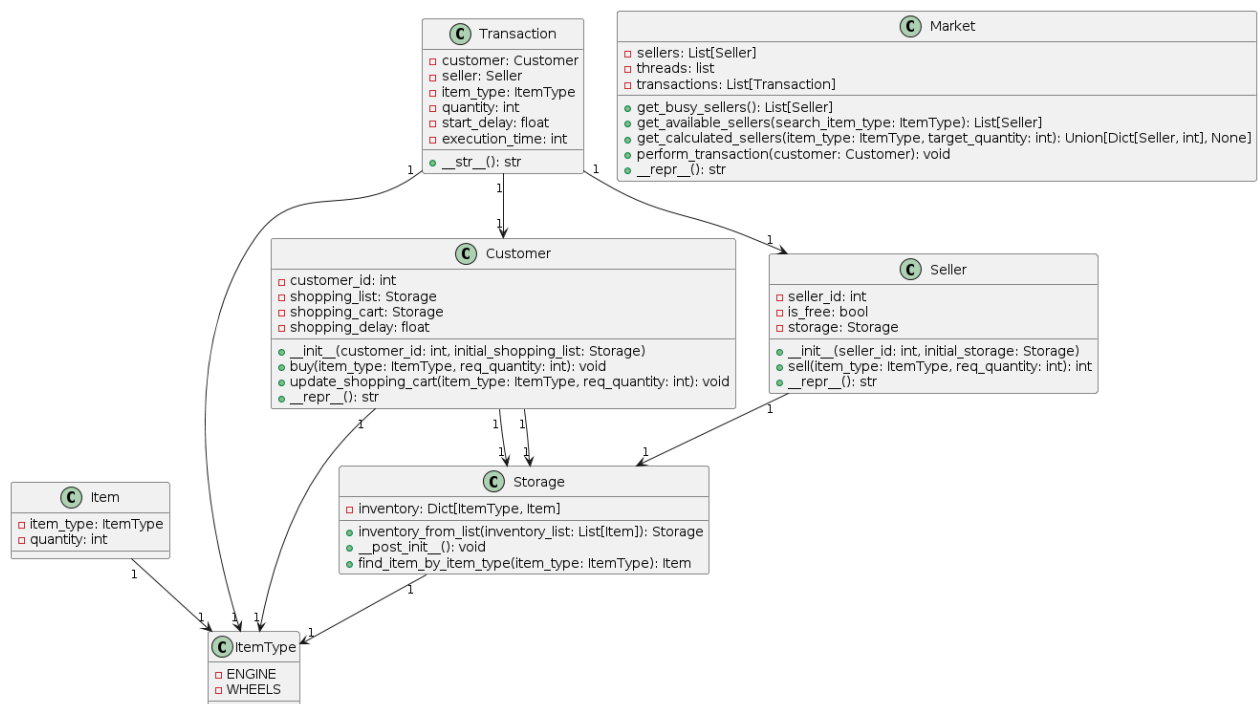


Rys. 2. Model danych

Na rysunku Rys.2. został przedstawiony model danych utworzony w projekcie. Składa się on z następujących tabel:

- **Tabela dostawców** – przechowuje dane dostawców, takie jak niepowtarzalny identyfikator jednoznacznie identyfikujący sprzedawcę.
- **Tabela klientów** – przechowuje dane klientów, takie jak niepowtarzalny identyfikator jednoznacznie identyfikujący klienta w systemie.
- **Tabela transakcji** – przechowuje informacje o danej transakcji, sprzedawcy, kliencie, towarach, ilości sprzedanych towarów oraz czasie wykonania transakcji.
- **Tabela towarów (item\_type)** przechowuje informację o przedmiotach jakie system jest w stanie przetwarzać.
- **Tabela seller\_item** składa się z informacji o dostępnych towarach u określonych sprzedawców.
- **Tabela customer\_item** reprezentuje listę przedmiotów, których klient planuje zakupić oraz listę przedmiotów zakupionych już przez klienta.

## 5. Interfejsy oraz klasy



Rys. 3. Diagram klas i interfejsów (UML)

W rozproszonym systemie zamówień zarówno interfejsy, jak i klasy, odgrywają kluczową rolę w organizacji i strukturyzacji kodu, umożliwiając separację logiki biznesowej od warstwy prezentacji oraz ułatwiając rozbudowę i utrzymanie systemu.

W prezentowanym systemie zaimplementowane zostały następujące klasy:

- **Transaction** – reprezentują transakcję. Zawiera wszystkie potrzebne elementy oraz obiekty składowe do wykonania transakcji.
- **Market** – reprezentuje rynek, na którym dostępni są sprzedawcy, klienci oraz przeprowadza transakcję w odpowiedni sposób, aby jak najbardziej były one optymalne. Zawiera odpowiednie metody do zarządzania zarówno klientami jak i sprzedawcami.
- **Customer** – zawiera informacje o klientach dostępnych w systemie, jak identyfikator, listę zakupów, koszyk zakupowy oraz metody zarządzające ich atrybutami.
- **Seller** – zawiera informację o dostępnych sprzedawcach w systemie oraz posiada odpowiednie metody do zarządzania ich magazynem.
- **Storage** – reprezentuje informację na temat magazynu każdego sprzedawcy, posiada zaimplementowane metody do przeszukiwania magazynu w celu szybkiego i prostego znajdowania określonego towaru.
- **Item** – przechowuje informację o dostępnych typach przedmiotów oraz ich ilości.
- **ItemType** – reprezentują typ dostępnych przedmiotów w systemie zamówień.



Zostały zaimplementowane również związki pomiędzy klasami jak dziedziczenie. Związki te określają, w jaki sposób klasy współpracują ze sobą oraz jakie występują między nimi zależności. Dlatego w prezentowanym systemie zachowana jest spójna i elastyczna struktura programu aplikacji.

## 6. Aplikacja

Do implementacji systemu zamówień wraz z graficznym interfejsem użytkownika wykorzystany został **Python (wersja 3.10)** - język wysokiego poziomu, oferujący rozbudowany pakiet bibliotek oraz składania cechującą się przejrzystością oraz prostotą.

Wykorzystane biblioteki:

- a) **SQLAlchemy (wersja - 2.0.27)** – biblioteka do obsługi relacyjnych baz danych. Zapewnia wygodne narzędzia do tworzenia oraz zarządzania bazami danych. Oferuje mechanizmy ORM (Object-Relational Mapping), które pozwalają mapować obiekty w języku Python na rekordy w bazie danych, co ułatwia pracę z danymi za pomocą obiektowości.
- b) **Psycopg2 (2.9.5)** – służy do wykonywania operacji bazodanowych jak wykonywanie zapytań SQL, wstawianie oraz aktualizację danych. Zapewnia interfejs do komunikacji z bazą danych PostgreSQL z poziomu języka Python.
- c) **Streamlit (1.33.0)** – biblioteka wykorzystana do utworzenia graficznego interfejsu użytkownika (GUI) dla rozproszonego systemu zamówień. Bardzo dużą zaletą była możliwość tworzenia aplikacji internetowej za pomocą zwykłego kodu napisanego w języku Python bez potrzeby wykorzystania języku HTML i JavaScript.
- d) **Streamlit-on-Hover-tabs (1.0.1)** – to dodatek do biblioteki Streamlit, który umożliwił stworzenie interaktywnych zakładek (tabs) w aplikacji.
- e) **Pandas (1.5.1)** – wydajne i łatwe narzędzie do analizy i przetwarzania danych. W systemie została wykorzystana do przetwarzania wejściowych plików o rozszerzeniu .csv oraz do zarządzania strukturami danych jak DataFrame, które pozwalają na efektywną pracę z danymi tabelarycznymi.

W systemie wykorzystywany jest obiektowo-relacyjny system baz danych PostgreSQL typu open-source. Połączenie z serwerem bazy PostgreSQL odbywa się z wykorzystaniem opisywanych wyżej bibliotek – dane uwierzytelniające zawarte są w kodzie systemu. Dodatkowo w projekcie używany jest pgAdmin – jest to bezpłatna aplikacja kliencka, która umożliwia łatwe zarządzanie bazami danych za pomocą graficznego interfejsu użytkownika.

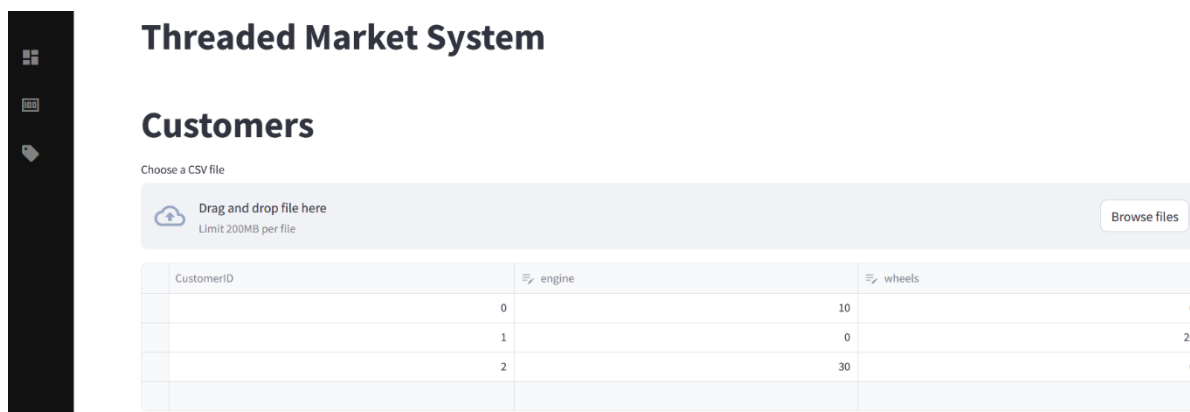
W celu uruchomienia rozproszonego systemu zamówień konieczne jest wykonanie następujących kroków:

1. Sklonowanie repozytorium systemu z platformy Github – „**git clone <link>**”
2. Przetączenie się na gałąź main – „**git checkout main**”.
3. Zainstalowanie potrzebnych bibliotek – „**pip install -r requirements.txt**”.
4. Wywołanie w terminalu komendy - „**streamlit run app.py**”.

Następnie w przeglądarce otworzona zostanie strona główna aplikacji. Kliknięcie przycisku „**Start**” - spowoduje uruchomienie systemu. Po zakończeniu rozproszonego algorytmu sprzedaży na ekranie wyświetlane są następujące tabele i wykresy zawierające informacje na temat:

- klientów (**przed uruchomieniem algorytmu**) wraz z listami zakupowymi.
- sprzedawców (**przed uruchomieniem algorytmu**) wraz z ich aktualnym stanem magazynu.
- przeprowadzonych transakcji.
- klientów (**po uruchomieniu algorytmu**) wraz z listami zakupowymi.
- sprzedawców (**po uruchomieniu algorytmu**) wraz z ich aktualnym stanem magazynu.

Użytkownik systemu może wprowadzić dowolną ilość klientów oraz sprzedawców za pomocą wgrania pliku o rozszerzeniu „**.csv**” o odpowiedniej strukturze. Może tego dokonać w zakładkach „*Sellers*” oraz „*Customers*” widocznych kolejno na rys. 4 i 5. Zakładki te umożliwiają również ręczne wprowadzanie rekordów przy pomocy operacji na tabeli z poziomu aplikacji.



CustomerID	engine	wheels	
0	10	0	
1	0	20	
2	30	0	

Rys. 4. Zakładka "Customers" do wprowadzenia żądań kupujących

**Threaded Market System**

## Sellers

Choose a CSV file

Drag and drop file here  
Limit 200MB per file

Browse files

SellerID	engine	wheels
0	10	0
1	0	20
2	30	0

Rys. 5. Zakładka "Sellers" do wprowadzenia dostępnych produktów sprzedających

Głównym procesem zachodzącym w systemie są transakcje. Zaimplementowano dwa podejścia, które określają sposób ich przetwarzania:

- Synchronicznie – przy użyciu pojedynczego wątku
- Asynchronicznie – przy użyciu wielu wątków

Na wykonanie transakcji składa się szereg czynności, które muszą zostać wykonane, aby poprawnie przetworzyć transakcję na podstawie żądania kupującego.

Przed przystąpieniem do realizacji transakcji, system umieszcza informacje o sprzedających w strukturach na podstawie zgromadzonych danych. Dla każdego rodzaju produktu tworzona jest pojedyncza struktura danych, tzn. liczba tworzonych struktur w całym systemie jest równa ilości unikalnych rodzajów produktów. System umieszcza w strukturach instancje sprzedających na podstawie ilości danego typu produktu, który jest związany z daną strukturą. Miejsce w strukturze jest powiązane z kolejnością, w jakiej dany sprzedający zostanie uwzględniony przy przetworzeniu kolejnej transakcji. Zaimplementowano dwie struktury danych, które mogą zostać użyte do tego celu:

- Lista - wszyscy sprzedawcy posiadający dany typ produktu ustawiani są w strukturę listy połączonej w kolejności malejącej względem ilości posiadanego typu produktu, tzn. pierwszym elementem listy jest sprzedający, który posiada największą ilość produktu danego typu z całego systemu, a ostatnim ten, który posiada jej najmniej.
- Kolejka priorytetowa – analogiczne podejście jak dla struktury listy, jednak zamiast listy połączonej użyto struktury kolejki o charakterze priorytetowym

Motywacją do zdefiniowania struktur, które są posortowane według ilości dostępnych produktów, jest minimalizacja liczby transakcji potrzebnych do spełnienia żądań kupujących.

Niezależnie od wyboru struktury, dane w nich zawarte będą sobie tożsame. Różnica uwidacznia się w sposobie przetwarzania transakcji.

Dla struktury listy kolejne transakcje są przetwarzane na podstawie wartości indeksu listy aktualnie wybranego sprzedawcy. Dla pierwszej transakcji indeks wskazuje na pierwsze miejsce listy, czyli na sprzedawcę, który posiada najwięcej produktów o danym typie. Kolejne transakcje powodują przesuwanie się indeksu elementów w stronę ostatniego elementu. Po osiągnięciu przez indeks wartości maksymalnej, czyli indeksu ostatniego elementu listy, wartość indeksu jest zerowana, co powoduje zapętlenie się procesu.

Po wyborze danego sprzedawcy następuje aktualizacja ilości dostępnych produktów dla danego sprzedawcy w wybranej strukturze. Pomimo zmiany ilości dostępnych produktów nie następuje zamiana poszczególnych elementów listy – sprzedawcy pozostają w niej w kolejności ustalonej na początku startu systemu. Przy dużej ilości transakcji w małych odstępach czasowych takie podejście pozwala na szybkie przydzielenie sprzedawcy do kupującego.

Celem sprostania problemu asynchronicznego przetwarzania transakcji każdy z kupujących posiada atrybut świadczący o tym, czy dany sprzedawca jest w trakcie przetwarzania innej transakcji. W sytuacji, w której znajduje się już w transakcji, to indeks listy zostaje przesunięty na następną pozycję.

Dla struktury kolejki priorytetowej transakcja pobiera zawsze pierwszy element z kolejki, czyli sprzedawcę, który posiada najwięcej produktów danego typu. Jeżeli wybrany sprzedawca nie sprzedał wszystkich swoich produktów w danej transakcji, to zostaje on umieszczony ponownie w kolejce ze zaktualizowaną wartością dostępnych produktów. Struktura zgodnie ze swoją charakterystyką i przeznaczeniem przesuwa odpowiednie elementy kolejki celem umieszczenia sprzedawcy o zaktualizowanej ilości dostępnych produktów w odpowiednim miejscu. Wówczas przy każdej transakcji żądanie jest obsługiwane przez sprzedawcę o najwyższej dostępności produktu.

W obrębie całego systemu struktura jest jednorodna dla wszystkich typów produktów oferowanych w sprzedaży.

Dla przetwarzania synchronicznego wybór sprzedających odbywa się na podstawie struktury listy połączonej. Kolejka priorytetowa została uwzględniona jedynie dla podejścia asynchronicznego z uwagi na swoją własność, zgodnie z którą wypychanie i umieszczanie elementów w tej strukturze jest bezpieczne wątkowo (ang. *thread-safe*). Upraszcza to wówczas użycie struktury i implementację sposobu wyboru sprzedających.

Ideą systemu jest obsłużenie żądań kupujących niezależnie od aktualnego obciążenia, tzn. powinien działać prawidłowo dla żądań pojawiających się niemalże jednocześnie

jak i żądań pojawiających się w większych odstępach czasowych. Celem zasymulowania tych sytuacji zaimplementowano dwa możliwe scenariusze:

- Żądania kupujących pojawiają się jednocześnie
- Każde z żądań kupujących pojawiają się z różnym wybranym losowo opóźnieniem czasowym

Z uwagi na mnogość możliwych kombinacji, które pojawiają się w zależności od podejścia i warunków pojawiania się żądań kupujących umożliwiono użytkownikowi wybór:

- Przetwarzania transakcji w sposób synchroniczny, bądź asynchroniczny
- Struktury listy, bądź kolejki priorytetowej dla przetwarzania asynchronicznego
- Uwzględnienia opóźnienia czasowego w pojawianiu się żądań kupujących, bądź pojawienia się ich jednocześnie

Wybór poszczególnych opcji może zostać zrealizowany na stronie głównej aplikacji (*Dashboard*) poprzez zaznaczenie odpowiednich pól przez wystartowaniem symulacji przetworzenia transakcji przyciskiem “Start!” (rys. 6).



Rys. 6. Strona główna aplikacji przed przystąpieniem do symulacji

Po wyborze opcji i kliknięciu przycisku “Start!” następuje proces przetworzenia transakcji dla danych umieszczonych w zakładkach “*Customers*” oraz “*Sellers*”.

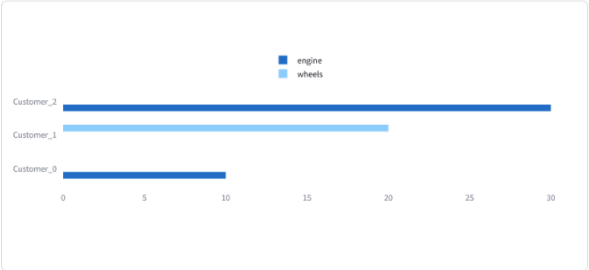
Po pomyślnym przetworzeniu transakcji poniżej przycisku “Start!” w sekcji “*Start Users*” zostają wyświetlone dane początkowe zadane przez użytkownika w postaci tabel i wykresów dla zarówno kupujących i sprzedających (rys. 7).

Start Users

Customers

CustomerID	engine	wheels
0	10	0
1	0	20
2	30	0

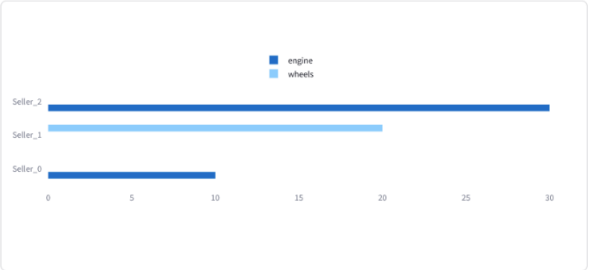
Customers' storage plot



Sellers

SellerID	engine	wheels
0	10	0
1	0	20
2	30	0

Sellers' storage plot



Rys. 7. Początkowe ilości dostępnych produktów sprzedających i żądań kupujących

W sekcji “Transactions” umieszczone zostały dane dotyczące dokonanych transakcji (rys. 8). Każdy rekord zawiera informację o typie produktu, ilości, która zostanie dostarczona do kupującego oraz ID zarówno kupującego oraz sprzedającego.

Transactions

customerID	sellerID	item_type	quantity
0	2	engine	10
1	1	wheels	20
2	2	engine	20
2	0	engine	10

Rys. 8. Dane dotyczące dokonanych transakcji zestawione w tabeli

W sekcji “End Users” umieszczone zostały zaktualizowane stany dostępnych produktów sprzedających i niespełnionych żądań kupujących po przetworzeniu transakcji (rys. 9).

End Users

Customers

CustomerID	engine	wheels
0	0	0
1	0	0
2	0	0

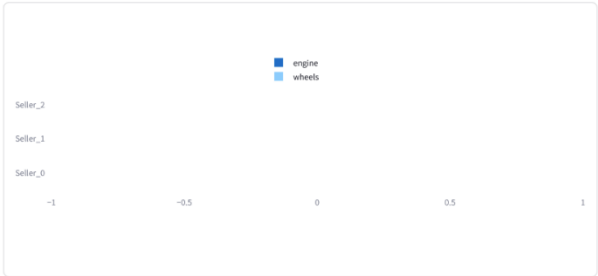
Customers' storage plot



Sellers

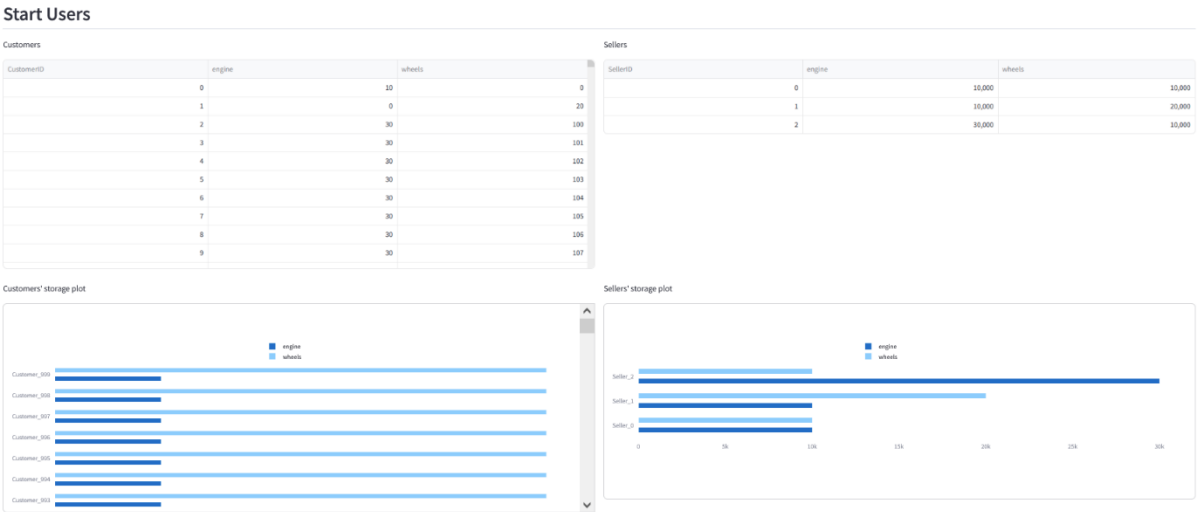
SellerID	engine	wheels
0	0	0
1	0	0
2	0	0

Sellers' storage plot



Rys. 9. Końcowe ilości pozostałych produktów sprzedających i niespełnionych żądań kupujących

Tabele i wykresy z rys. 7, 8 i 9 pozwalają na wizualizację i analizę danych dotyczących sprzedających, kupujących oraz transakcji w taki sposób, aby były czytelne oraz nie powodowały obciążenia aplikacji przy danych wejściowych o znacznym rozmiarze. Na rys. 10, 11, 12 widoczne są zakładki kolejno “Start Users”, “Transactions”, “End Users” dla 1000 kupujących.

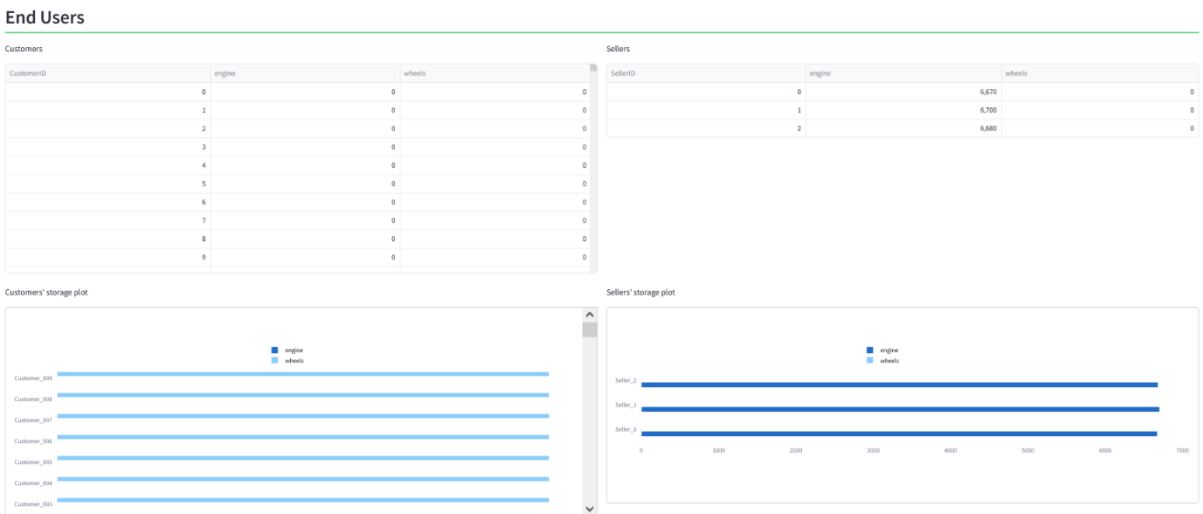


Rys. 10. Sekcja “Start Users” dla 1000 kupujących

Transactions

customerID	sellerID	item_type	quantity
0	2	engine	10
1	1	wheels	20
2	2	engine	30
3	1	wheels	100
3	2	engine	30
3	1	wheels	101
4	2	engine	30
4	1	wheels	102
5	2	engine	30
5	1	wheels	103

Rys. 11. Sekcja “Transactions” dla 1000 kupujących



Rys. 12. Sekcja “End Users” dla 1000 kupujących

## 7. Testy rozproszonego systemu zamówień

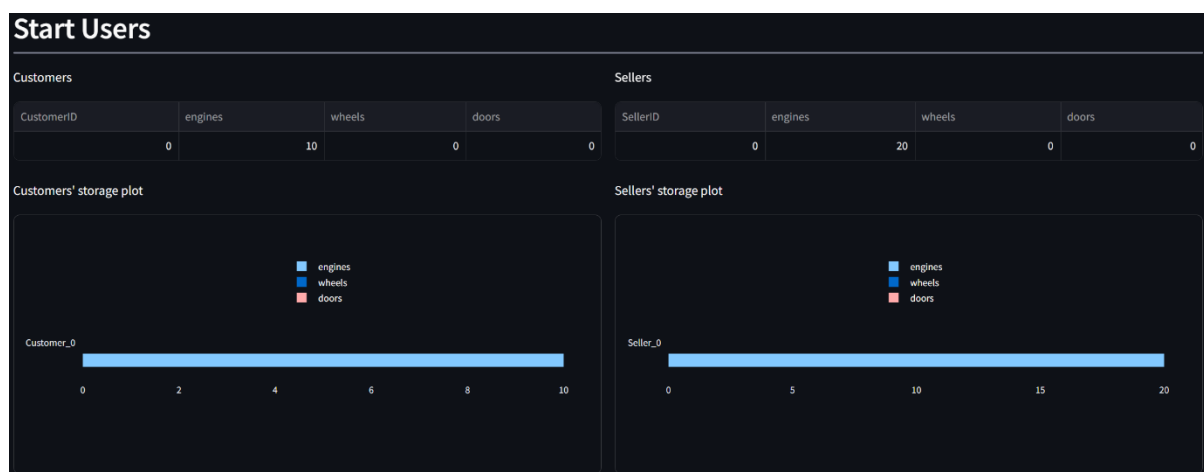
Stworzony system został przetestowany manualnie poprzez uruchamianie go dla wybranych scenariuszy, które miały za zadanie pokryć wszystkie możliwe typy sytuacji. Wszystkie testy zostały opisane w dalszej części rozdziału.

Każdy z testów polegał na dodaniu list kupujących i sprzedających, zawierających ich produkty do kupienia i sprzedaży, a następnie uruchomieniu systemu i potwierdzeniu poprawności wyników.

Dodatkowo, każde uruchomienie stworzonego systemu zostało porównane czasowo do sytuacji, gdyby system pracował w sposób nierozproszony i wszystkie operacje były wykonywane bez użycia wielowątkowości.

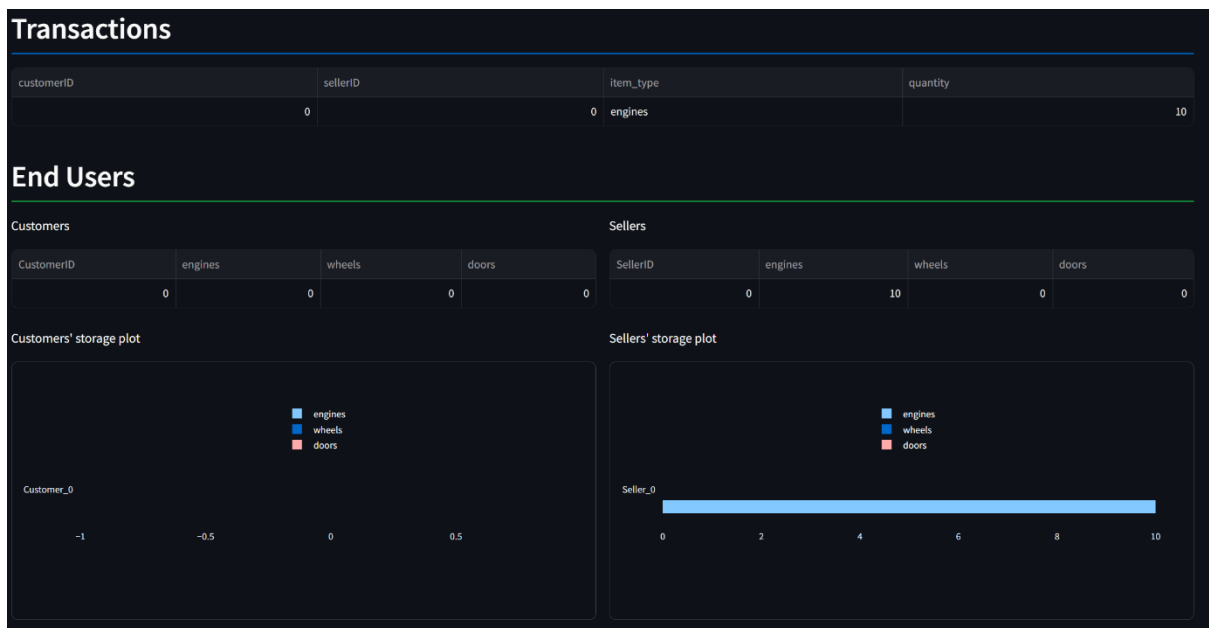
### Test nr. 1

**Pojedynczy** kupujący chce kupić **mniej** produktów niż **pojedynczy** sprzedawca posiada aktualnie na stanie magazynu.



Rys. 13. Dane wejściowe dla testu nr. 1.





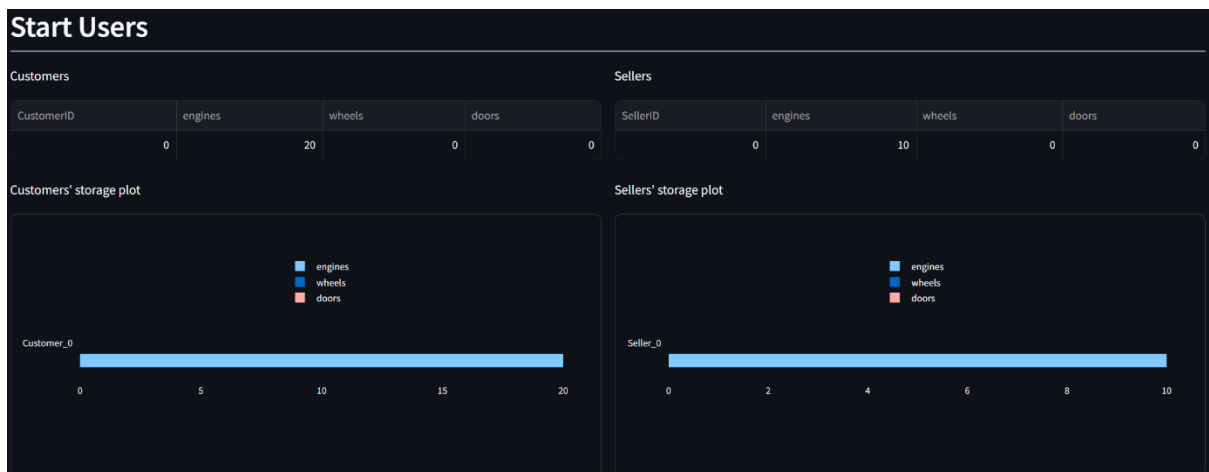
Rys. 14. Transakcje i dane wyjściowe dla testu nr. 1.

Czas wielowątkowości – kolejka : 0.001 sekundy  
 Czas wielowątkowości – lista : 0.001 sekundy  
 Czas jednowątkowości: 0.000 sekundy  
 Czas wielowątkowości – kolejka [opóźnienie]: 1.014 sekundy  
 Czas wielowątkowości – lista [opóźnienie]: 1.005 sekundy  
 Czas jednowątkowości [opóźnienie]: 1.007 sekundy

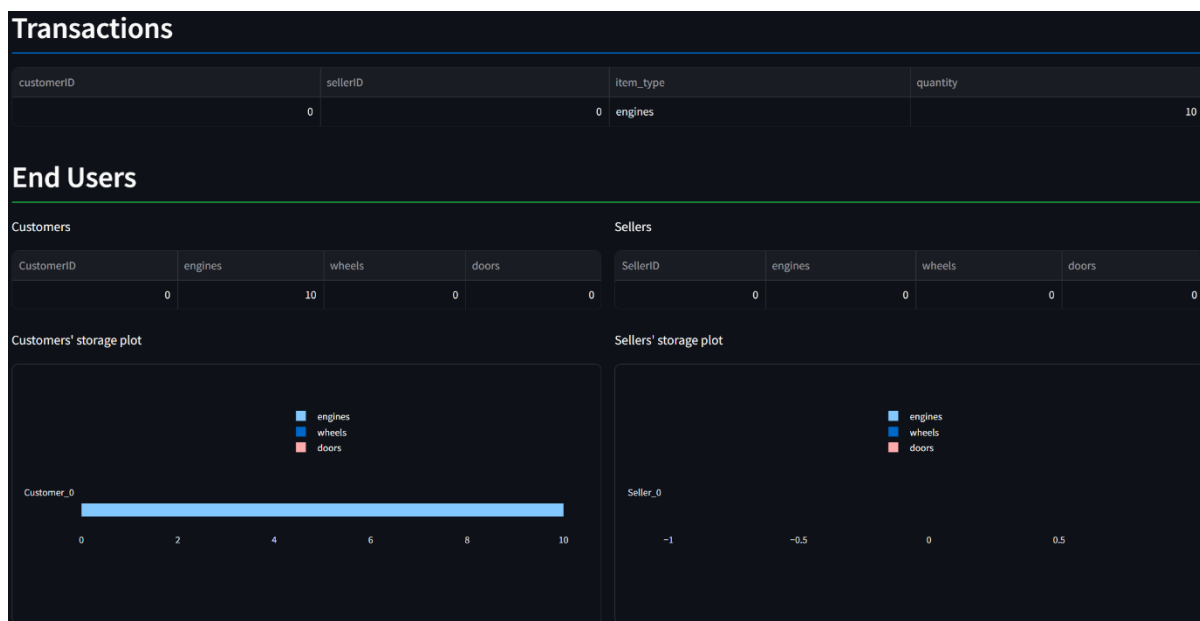
Zgodnie z wynikami, pojedynczy kupujący, chcący dokonać zakupu 10 przedmiotów typu "engines" od pojedynczego sprzedawcy, posiadającego 20 przedmiotów tego samego typu, dokonuje transakcji z sukcesem. Powoduje to wyzerowanie jego listy zakupów, jak i zmniejszenie listy dostępnych przedmiotów u sprzedawcy o 10 sztuk.

## Test nr. 2

**Pojedynczy** kupujący chce kupić **więcej** produktów niż **pojedynczy** sprzedawca posiada aktualnie na stanie magazynu.



Rys. 15. Dane wejściowe dla testu nr. 2.



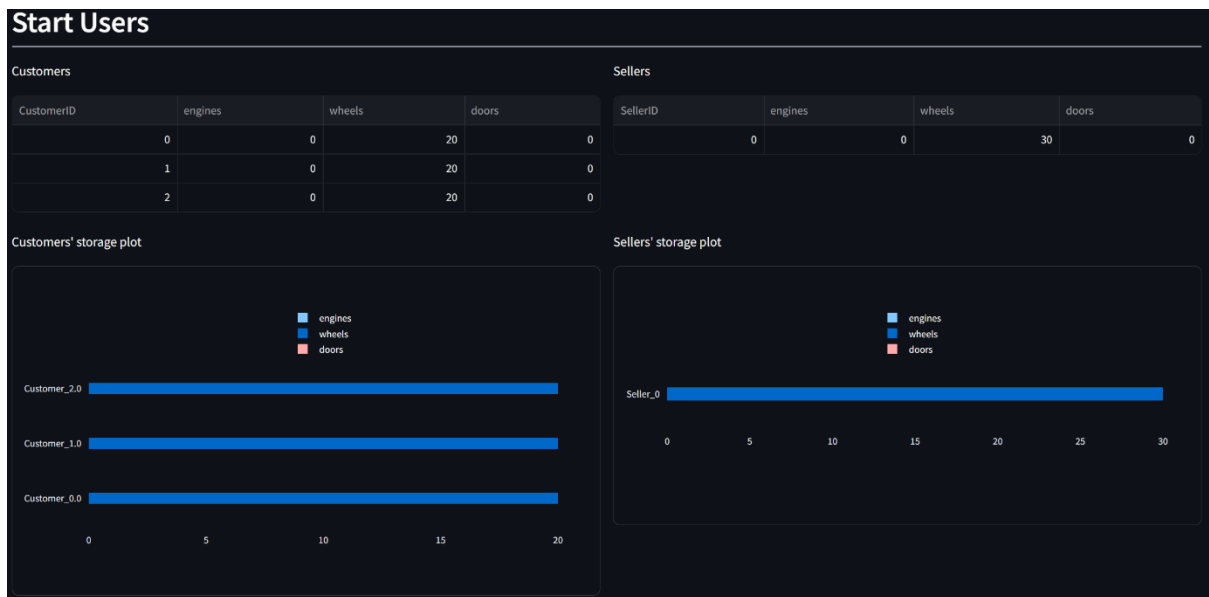
Rys. 16. Transakcje i dane wyjściowe dla testu nr. 2.

Czas wielowątkowości – kolejka : 0.001 sekundy  
 Czas wielowątkowości – lista : 0.002 sekundy  
 Czas jednowątkowości: 0.000 sekundy  
 Czas wielowątkowości – kolejka [opóźnienie]: 1.013 sekundy  
 Czas wielowątkowości – lista [opóźnienie]: 1.005 sekundy  
 Czas jednowątkowości [opóźnienie]: 1.005 sekundy

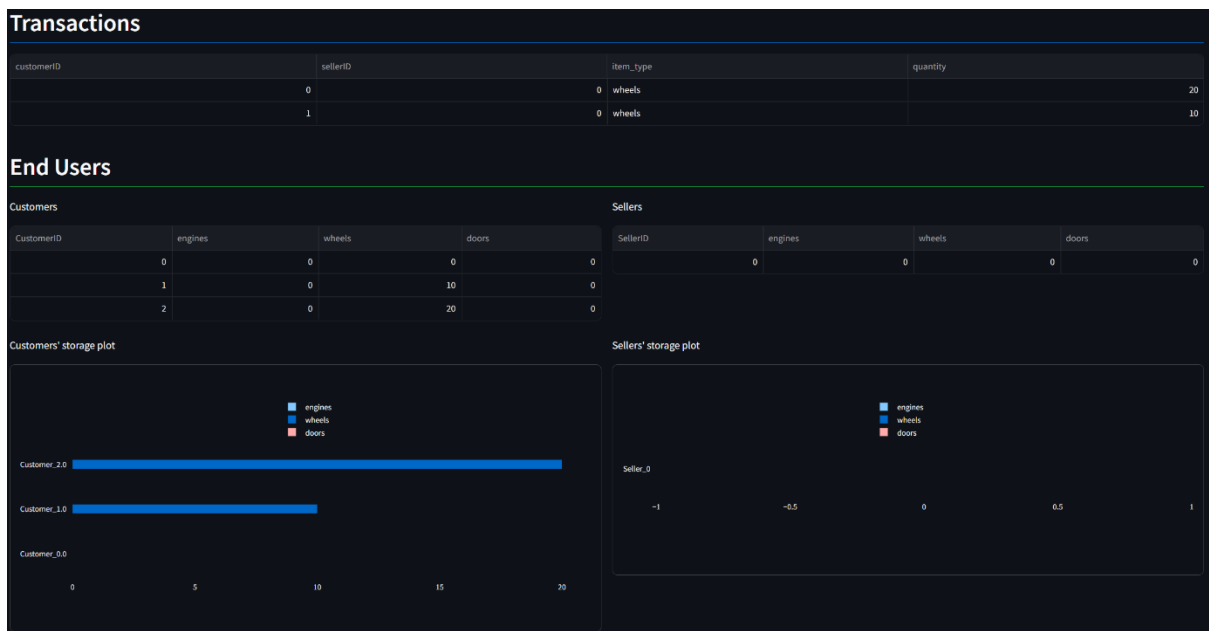
Jak widać, pojedynczy kupujący, chcący dokonać zakupu 20 przedmiotów typu "engines" od pojedynczego sprzedawcy, posiadającego 10 przedmiotów tego samego typu, dokonuje transakcji z sukcesem ale tylko 10 przedmiotów. Powoduje to zmniejszenie jego listy zakupów, oraz wyzerowanie listy dostępnych przedmiotów u sprzedawcy.

### Test nr. 3

**Wielu** (trzech) kupujących planuje kupić taką samą ilość produktów od **pojedynczego** sprzedającego, a ich sumaryczna ilość jest **większa** od ilość dostępnych produktów u sprzedawcy.



Rys. 17. Dane wejściowe dla testu nr. 3.



Rys. 18. Transakcje i dane wyjściowe dla testu nr. 3.

Czas wielowątkowości – kolejka :	0.004 sekundy
Czas wielowątkowości – lista :	0.002 sekundy
Czas jednowątkowości:	0.001 sekundy
Czas wielowątkowości – kolejka [opóźnienie]:	1.011 sekundy
Czas wielowątkowości – lista [opóźnienie]:	1.012 sekundy
Czas jednowątkowości [opóźnienie]:	3.018 sekundy

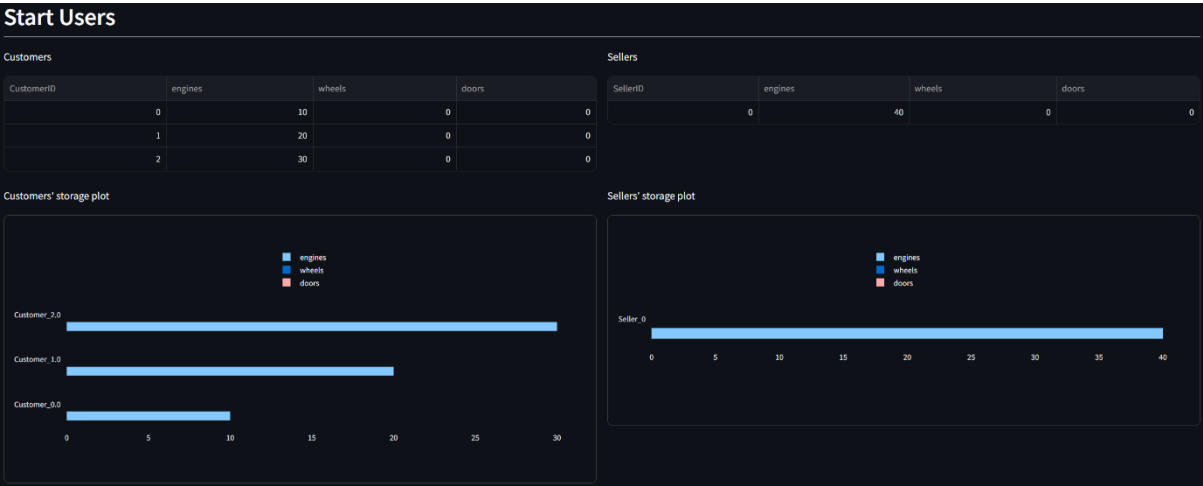
Jak widać, bez opóźnień, czas operacji w trybie wielowątkowym jest taki sam jak w trybie jednowątkowym. Wynika to z faktu, że operacje obejmują niewielką liczbę kupujących i jednego sprzedawcę. W rezultacie oba podejścia mają podobną wydajność i szybkość obliczeń.

W przypadku istniejącego opóźnienia można zaobserwować, że wielowątkowość zaczyna przewyższać jednowątkowość w rozważanym problemie.

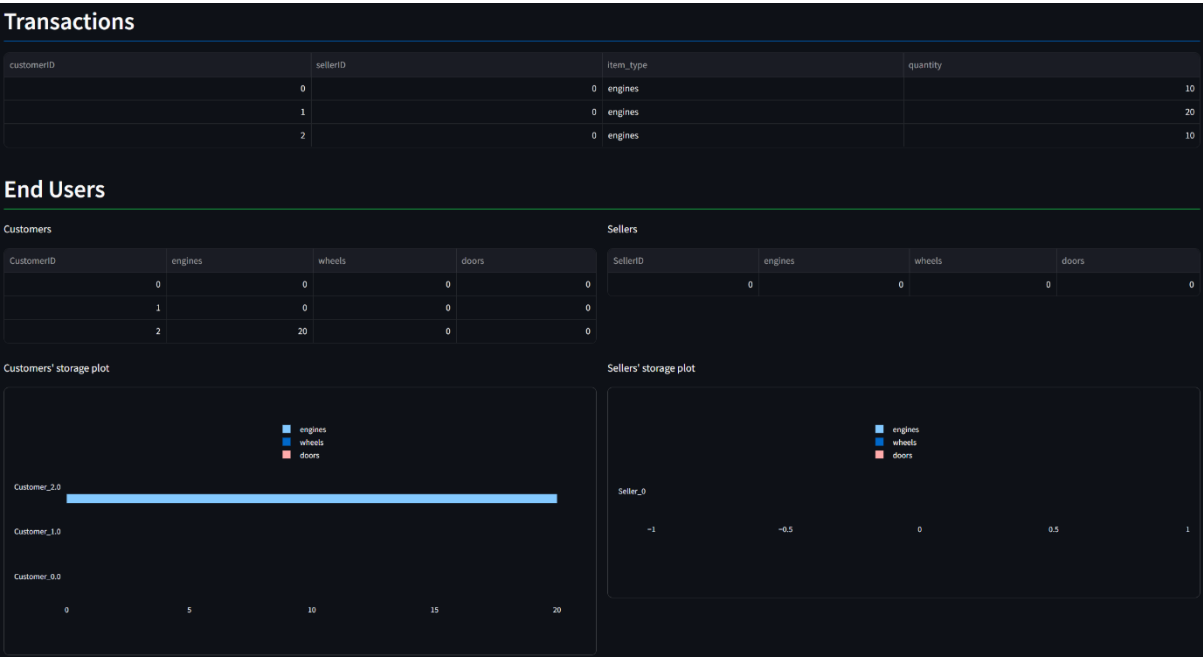
Transakcje i dane potwierdzają prawidłowe działanie systemu. Łącznie klienci chcą zakupić 60 przedmiotów typu „wheels”, podczas gdy pojedynczy sprzedawca dysponuje jedynie 30. Przeprowadzono dwie transakcje z dwoma kupującymi. W rezultacie lista zakupów jednego kupującego została wyzerowana, lista drugiego zmniejszona, a ilość dla trzeciego pozostała bez zmian.

Test nr. 4

**Wielu** (trzech) kupujących chce kupić różną ilość produktów od **pojedynczego** sprzedającego, sumaryczna ilość jest **większa** od ilości dostępnych produktów u sprzedawcy.



Rys. 19. Dane wejściowe dla testu nr. 4.



Rys. 20. Transakcje i dane wyjściowe dla testu nr. 4.

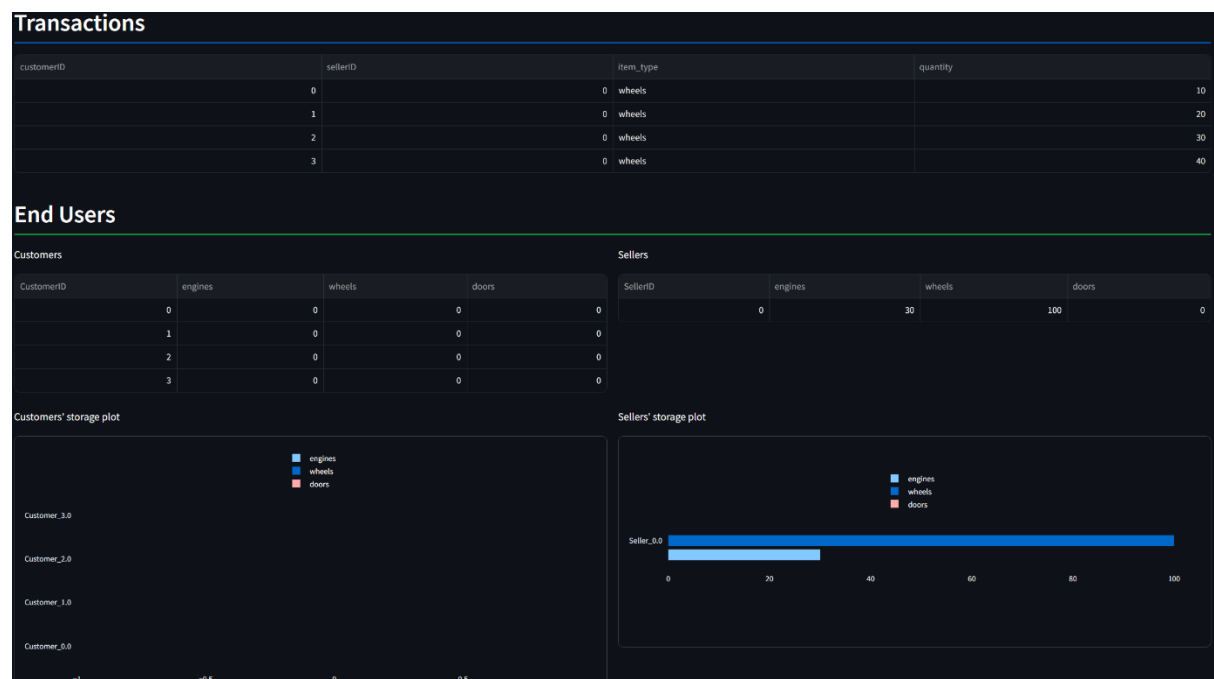
Czas wielowątkowości – kolejka :	0.002 sekundy
Czas wielowątkowości – lista :	0.003 sekundy
Czas jednowątkowości:	0.000 sekundy
Czas wielowątkowości – kolejka [opóźnienie]:	1.006 sekundy
Czas wielowątkowości – lista [opóźnienie]:	1.017 sekundy
Czas jednowątkowości [opóźnienie]:	3.025 sekundy

## Test nr. 5

**Wielu** (czterech) kupujących chce kupić dowolną ilość produktów od **pojedynczego** sprzedającego, której sumaryczna ilość jest **mniejsza** od ilości dostępnych produktów u danego sprzedawcy.



Rys. 21. Dane wejściowe dla testu nr. 5.

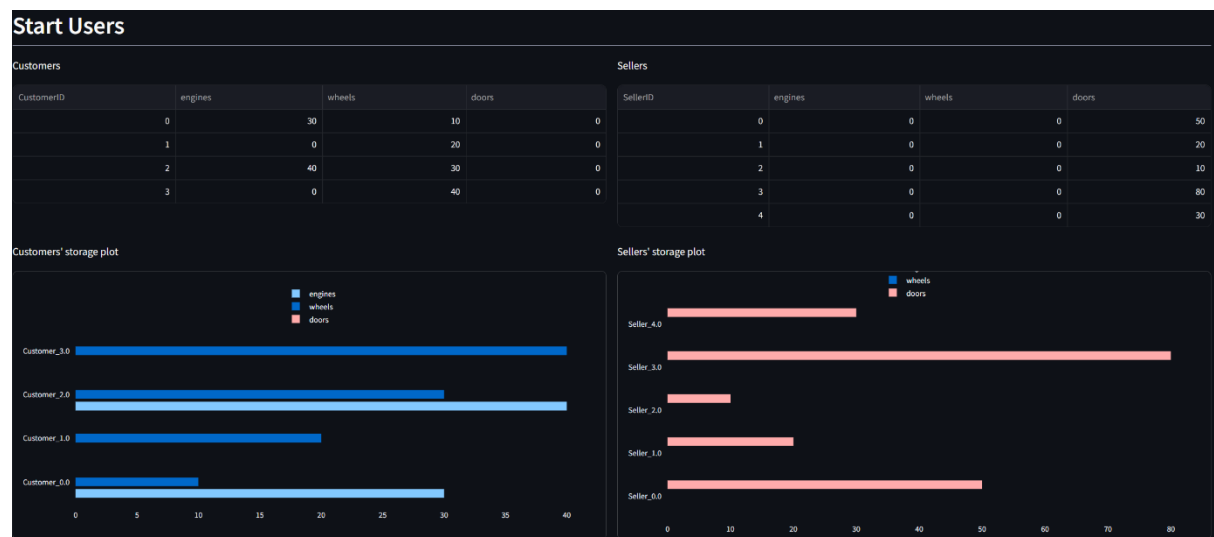


Rys. 22. Transakcje i dane wyjściowe dla testu nr. 5.

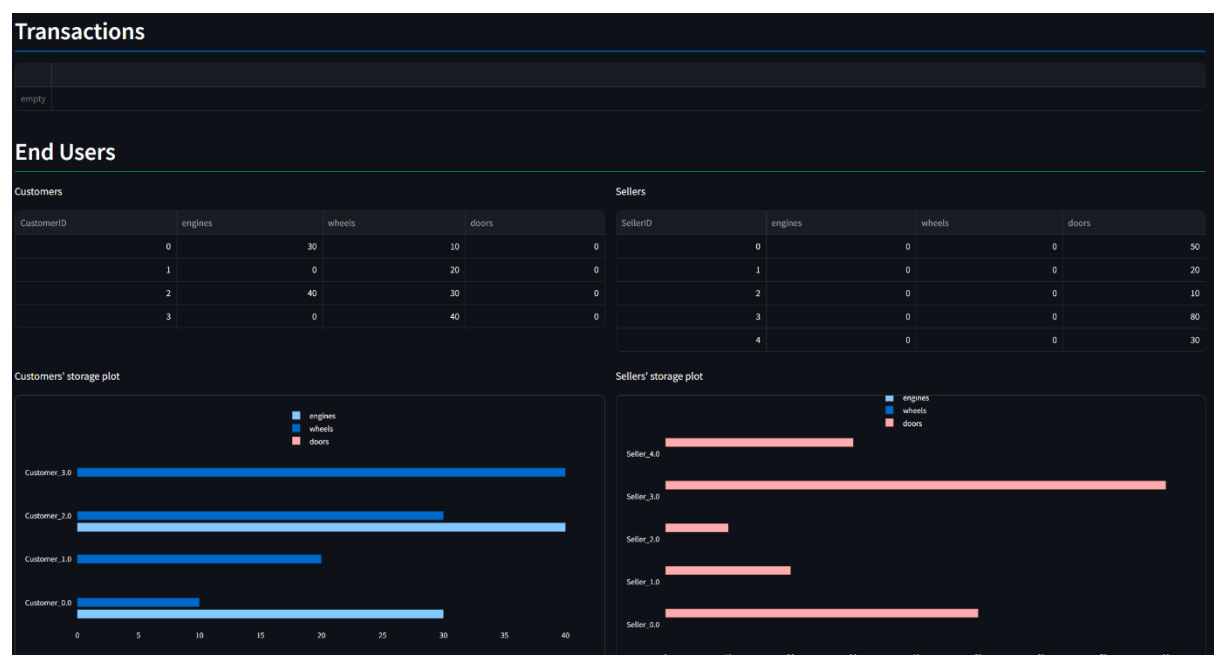
Czas wielowątkowości – kolejka :	0.003 sekundy
Czas wielowątkowości – lista :	0.002 sekundy
Czas jednowątkowości:	0.000 sekundy
Czas wielowątkowości – kolejka [opóźnienie]:	1.019 sekundy
Czas wielowątkowości – lista [opóźnienie]:	1.018 sekundy
Czas jednowątkowości [opóźnienie]:	4.044 sekundy

## Test nr. 6

**Wielu** (czterech) kupujący chcą kupić produkty tego samego typu. Niestety żaden z **wielu** (pięciu) sprzedających nie posiada tego typu produktów.



Rys. 23. Dane wejściowe dla testu nr. 6.



Rys. 24. Transakcje i dane wyjściowe dla testu nr. 6.

Czas wielowątkowości – kolejka : 0.002 sekundy  
 Czas wielowątkowości – lista : 0.003 sekundy  
 Czas jednowątkowości: 0.002 sekundy  
 Czas wielowątkowości – kolejka [opóźnienie]: 1.016 sekundy  
 Czas wielowątkowości – lista [opóźnienie]: 1.005 sekundy  
 Czas jednowątkowości [opóźnienie]: 4.047 sekundy

Według wyników, żadna operacja nie została wykonana, co jest oczywiście poprawnym wynikiem.

## Test nr. 7

**Wielu** (siedmiu) kupujących i **wielu** (dwunastu) sprzedawców. Dla każdego typu produktów istnieje jeden kupujący, który planuje złożyć dużo większe zamówienie niż reszta.

Start Users							
Customers				Sellers			
CustomerID	engines	wheels	doors	SellerID	engines	wheels	doors
0	20	10	0	0	50	40	50
1	30	20	10	1	50	10	20
2	40	30	0	2	50	0	10
3	200	30	0	3	10	20	80
4	0	200	0	4	0	0	30
5	0	0	200	5	0	100	30
6	10	10	10	6	0	0	50
				7	0	60	100
				8	100	0	80
				9	0	50	0
				10	40	50	0
				11	30	20	0

Rys. 25. Dane wejściowe kupujących i sprzedających dla testu nr. 7.

Transactions				
customerID	sellerID	item_type	quantity	
	6	8 engines		10
	6	5 wheels		10
	6	7 doors		10
	5	3 doors		80
	5	8 doors		80
	5	0 doors		40
	0	0 engines		20
	0	7 wheels		10
	2	1 engines		40
	2	9 wheels		30
	3	2 engines		50
	3	10 engines		40
	3	11 engines		30
	3	3 engines		10
	3	8 engines		70
	3	10 wheels		30
	4	0 wheels		40
	4	3 wheels		20
	4	11 wheels		20
	4	1 wheels		10
	4	5 wheels		90
	4	7 wheels		20
	1	0 engines		30
	1	9 wheels		20
	1	6 doors		10

Rys. 26. Transakcje przeprowadzone w teście nr. 7

End Users				
Customers				
CustomerID	engines	wheels	doors	
	0	0	0	0
	1	0	0	0
	2	0	0	0
	3	0	0	0
	4	0	0	0
	5	0	0	0
	6	0	0	0
Sellers				
SellerID	engines	wheels	doors	
	0	0	0	10
	1	10	0	20
	2	0	0	10
	3	0	0	0
	4	0	0	30
	5	0	0	30
	6	0	0	40
	7	0	30	90
	8	20	0	0
	9	0	0	0
	10	0	20	0
	11	0	0	0

Rys. 27. Dane wyjściowe kupujących i sprzedających dla testu nr. 7

Czas wielowątkowości – kolejka : 0.003 sekundy  
 Czas wielowątkowości – lista : 0.004 sekundy  
 Czas jednowątkowości: 0.000 sekundy  
 Czas wielowątkowości – kolejka [opóźnienie]: 1.018 sekundy  
 Czas wielowątkowości – lista [opóźnienie]: 1.009 sekundy  
 Czas jednowątkowości [opóźnienie]: 7.053 sekundy

System zadziałał poprawnie dla podanych danych wejściowych. Dowodem na to jest wyzerowanie list zakupów dla wszystkich kupujących, ponieważ łączna liczba produktów do kupienia była mniejsza niż łączna liczba produktów do sprzedaży w każdej kategorii. Najważniejszym wynikiem jest to, że czas wielowątkowości z opóźnieniem jest 7 razy krótszy niż czas jednowątkowości. Wnioskiem z tego testu jest to, że przy wystąpieniu opóźnienia, efektywność wielowątkowości wzrasta proporcjonalnie do liczby kupujących.



Test nr. 8

**Wielu** (dwunastu) kupujących i **wielu** (siedmiu) sprzedawców. Dla każdego typu produktów istnieje jeden kupujący, który planuje złożyć dużo większe zamówienie niż reszta.

Start Users							
Customers				Sellers			
CustomerID	engines	wheels	doors	SellerID	engines	wheels	doors
0	20	10	0	0	50	0	50
1	30	20	10	1	50	10	20
2	40	30	0	2	50	10	80
3	200	30	0	3	100	100	80
4	0	200	0	4	200	100	200
5	0	0	200	5	100	200	70
6	10	10	10				
7	60	60	60				
8	30	30	10				
9	0	50	10				
10	10	0	20				
11	60	30	10				

Rys. 28. Dane wejściowe kupujących i sprzedających dla testu nr. 8.

CustomerID	engines	wheels	doors	SellerID	engines	wheels	doors
0	20	10	0	4	engines		10
1	30	20	10	5	wheels		10
2	40	30	0	6	doors		10
3	200	30	0	7	engines		10
4	0	200	0	8	wheels		10
5	0	0	200	9	engines		10
6	10	10	10	10	wheels		10
7	60	60	60	11	doors		10
8	30	30	10	12	engines		10
9	0	50	10	13	wheels		10
10	10	0	20	14	doors		10
11	60	30	10	15	engines		10
				16	wheels		10
				17	doors		10
				18	engines		10
				19	wheels		10
				20	doors		10
				21	engines		10
				22	wheels		10
				23	doors		10
				24	engines		10
				25	wheels		10
				26	doors		10
				27	engines		10
				28	wheels		10
				29	doors		10
				30	engines		10
				31	wheels		10
				32	doors		10
				33	engines		10
				34	wheels		10
				35	doors		10
				36	engines		10
				37	wheels		10
				38	doors		10
				39	engines		10
				40	wheels		10
				41	doors		10
				42	engines		10
				43	wheels		10
				44	doors		10
				45	engines		10
				46	wheels		10
				47	doors		10
				48	engines		10
				49	wheels		10
				50	doors		10
				51	engines		10
				52	wheels		10
				53	doors		10
				54	engines		10
				55	wheels		10
				56	doors		10
				57	engines		10
				58	wheels		10
				59	doors		10
				60	engines		10
				61	wheels		10
				62	doors		10
				63	engines		10
				64	wheels		10
				65	doors		10
				66	engines		10
				67	wheels		10
				68	doors		10
				69	engines		10
				70	wheels		10
				71	doors		10
				72	engines		10
				73	wheels		10
				74	doors		10
				75	engines		10
				76	wheels		10
				77	doors		10
				78	engines		10
				79	wheels		10
				80	doors		10
				81	engines		10
				82	wheels		10
				83	doors		10
				84	engines		10
				85	wheels		10
				86	doors		10
				87	engines		10
				88	wheels		10
				89	doors		10
				90	engines		10
				91	wheels		10
				92	doors		10
				93	engines		10
				94	wheels		10
				95	doors		10
				96	engines		10
				97	wheels		10
				98	doors		10
				99	engines		10
				100	wheels		10
				101	doors		10
				102	engines		10
				103	wheels		10
				104	doors		10
				105	engines		10
				106	wheels		10
				107	doors		10
				108	engines		10
				109	wheels		10
				110	doors		10
				111	engines		10
				112	wheels		10
				113	doors		10
				114	engines		10
				115	wheels		10
				116	doors		10
				117	engines		10
				118	wheels		10
				119	doors		10
				120	engines		10
				121	wheels		10
				122	doors		10
				123	engines		10
				124	wheels		10
				125	doors		10
				126	engines		10
				127	wheels		10
				128	doors		10
				129	engines		10
				130	wheels		10
				131	doors		10
				132	engines		10
				133	wheels		10
				134	doors		10
				135	engines		10
				136	wheels		10
				137	doors		10
				138	engines		10
				139	wheels		10
				140	doors		10
				141	engines		10
				142	wheels		10
				143	doors		10
				144	engines		10
				145	wheels		10
				146	doors		10
				147	engines		10
				148	wheels		10
				149	doors		10
				150	engines		10
				151	wheels		10
				152	doors		10
				153	engines		10
				154	wheels		10
				155	doors		10
				156	engines		10
				157	wheels		10
				158	doors		10
				159	engines		10
				160	wheels		10
				161	doors		10
				162	engines		10
				163	wheels		10
				164	doors		10
				165	engines		10
				166	wheels		10
				167	doors		10
				168	engines		10
				169	wheels		10
				170	doors		10
				171	engines		10
				172	wheels		10
				173	doors		10
				174	engines		10
				175	wheels		10
				176	doors		10
				177	engines		10
				178	wheels		10
				179	doors		10
				180	engines		10
				181	wheels		10
				182	doors		10
				183	engines		10
				184	wheels		10
				185	doors		10
				186	engines		10
				187	wheels		10
				188	doors		10
				189	engines		10
				190	wheels		10
				191	doors		10
				192	engines		10
				193	wheels		10
				194	doors		10
				195	engines		10
				196	wheels		10
				197	doors		10
				198	engines		10
				199	wheels		10
				200	doors		10

Rys. 29. Transakcje przeprowadzone w teście nr. 8.

## End Users

Customers				Sellers			
CustomerID	engines	wheels	doors	SellerID	engines	wheels	doors
0	0	10	0	0	0	0	0
1	0	0	0	1	0	0	0
2	0	0	0	2	0	0	0
3	0	0	0	3	0	0	0
4	0	40	0	4	90	0	170
5	0	0	0	5	0	0	0
6	0	0	0				
7	0	0	0				
8	0	0	0				
9	0	0	0				
10	0	0	0				
11	0	0	0				

Rys. 30. Dane wyjściowe kupujących i sprzedających dla testu nr. 8.

Czas wielowątkowości – kolejka :	0.006 sekundy
Czas wielowątkowości – lista :	0.004 sekundy
Czas jednowątkowości:	0.008 sekundy
Czas wielowątkowości – kolejka [opóźnienie]:	1.013 sekundy
Czas wielowątkowości – lista [opóźnienie]:	1.018 sekundy
Czas jednowątkowości [opóźnienie]:	12.111 sekundy

Omawiany przypadek testowy jest bliźniaczym odpowiednikiem poprzedniego, siódmego testu. Różnica polega na zamianie ilości oraz list produktów sprzedających i kupujących. Widoczną na rysunkach (Rys. 26 i Rys. 29) zmianą było znaczne zwiększenie ilości wykonanych transakcji.

Poprawność transakcji potwierdzają dane wyjściowe przedstawione na rysunku (Rys. 30). Dla produktów z kategorii "engines" i "doors" sumaryczna ilość produktów do kupienia była niższa niż tych do sprzedaży. Natomiast w przypadku kategorii "wheels" liczba produktów do sprzedaży była mniejsza o 50 sztuk.

Istotną różnicą jest również różnica w czasach wielowątkowości i jednowątkowości dla przypadku z opóźnieniem, na korzyść tego pierwszego.

## 8. Podsumowanie

Projekt Rozproszonego Systemu Zamówień został stworzony w celu realizacji systemu, w którym składane są zamówienia na różne towary. Kluczowym elementem było to, że zamówienia te są dostępne u różnych sprzedawców, co sprawia, że klient chcący dokonać zakupu musi przeglądać wiele ofert i odwiedzać stoiska wielu sprzedawców.

Podstawowym rozwiązaniem tego problemu był prosty system jednowątkowy. Posiadając wszelkie informacje o sprzedających, system ten przyjmował każdego klienta pojedynczo i osobno przetwarzał jego zamówienie. Jest to jednak podejście dość nierealne, ponieważ żaden market z wieloma sprzedawcami nie przyjmowałby każdego klienta pojedynczo.

Celem tej pracy była implementacja wielowątkowości, będącej realizacją systemu rozproszonego. Zaimplementowane podejście różniło się od podstawowego tym, że zamiast wpuszczać każdego klienta osobno do marketu, wszyscy klienci byli wpuszczani jednocześnie.

Stworzony system przeszedł szereg testów, które miały na celu pokrycie różnych scenariuszy użycia i potwierdzenie poprawności jego działania. Testy obejmowały przypadki takie jak pojedynczy kupujący dokonujący zakupu mniejszej lub większej liczby produktów niż dostępne na stanie, wielu kupujących planujących zakupy od jednego sprzedawcy oraz testy efektywności wielowątkowości w porównaniu do jednowątkowości. Wyniki pokazały, że efektywność rozwiązania będącego przedmiotem tej pracy wzrastała proporcjonalnie do liczby kupujących.