**Prosty system giełdowy:**

Celem programu jest stworzenieaplikacji symulującej prosty system giełdowy, którego zadaniem jest przyjmowanie od dealerów zleceń kupna/sprzedaży i automatyczne zawieranie transakcji

Projekt będzie się składał z dwóch głównych komponentów działających jako dwie niezależne aplikacje

* Aplikacja pośrednicząca z webowym interfejsem użytkownika typu SPA - rejestracja/modyfikacja i przesyłanie ofert do właściwego rynku.
* Aplikacja backend - kojarzenie ofert i zawieranie transakcji dla danego rynku

Oraz z części UI która zapewni dostęp do funkcjonalności części pośredniczącej.

Komunikacja pomiędzy aplikacjami będzie zrealizowana za pomocą JMS (ActiveMQ).

**Funkcjonalność – aplikacja pośrednicząca:**

1. Pobierz listę produktów
   1. Argumenty: brak
   2. Rezultat: lista produktów obsługiwanych przez działające instancje backend.
2. Zgłoś zlecenie sprzedaży/kupna
   1. Argumenty:
      1. Nazwa produktu
      2. Wolumen zlecenia
      3. Typ zlecenia (kupno/sprzedaż)
      4. Cena
      5. Zlecający
   2. Rezultat zawiera identyfikator zlecenia jeśli zostało ono zaakceptowane przez odpowiedni backend, bądź informację o błędzie jeśli przesłanie zlecenia do backendu się nie powiodło.
3. Wyświetlaj informacje o zawartych transakcjach w danym okresie czasu
   1. Argumenty: zakres czasu
   2. Rezultat: lista zawartych transakcji.
4. Pobierz listę zleceń zalogowanego dealera
   1. Argumenty: zakres czasu
   2. Rezultat: lista zleceń dealera.
5. Przesyłanie zleceń sprzedaży i kupna do odpowiedniego backendu (przez JMS Queue)
6. Przesyłanie zleceń zmiany/anulowania zlecenia kupna/sprzedaży
7. Odbiór informacji o dostepnych produktach (przez JMS Queue)
8. Pobieranie informacji o zawartych transakcjach (przez JMS Queue)
9. Pobieranie informacji o liście zleceń (przez JMS Queue)
10. Odbiór informacji o realizacji transakcji (przez JMS Topic)
11. Wysyłanie informacji o zrealizowanych zleceniach poprzez WebSockets do UI

**Funkcjonalność – aplikacja backend:**

1. Po uruchomieniu wysyła do aplikacji pośredniczącej komunikat z informacją o obsługiwanych produktach (zaczytanych z konfiguracji)
2. Kojarzenie ofert i zawieranie transakcji wg algorytmu opisanego poniżej. Ponadto konieczne jest zagwarantowanie współbieżności tego procesu.
   1. Zlecenie sprzedaży z najniższą ceną ze zleceniem kupna tego samego produktu z najwyższą ceną.
   2. Jeśli dwa zlecenia kupna lub sprzedaży mają taką samą cenę to brane jest starsze zlecenie.
   3. Jeśli istnieją aktywne zlecenia kupna i sprzedaży dla tego samego produktu, gdzie cena kupna >= cena sprzedaży to transakcja jest uznana za zawartą (po cenie sprzedaży). Wolumen obu zleceń zostaje pomniejszony o wolumen transakcji. Jeśli pozostały wolumen zlecenia wynosi zero to zlecenie jest uznawane za spełnione.
3. Informacja o każdej zawartej transakcji jest rozpowszechniana (przez JMS Topic)

**Opis komunikatów przesyłanych przez JMS:**

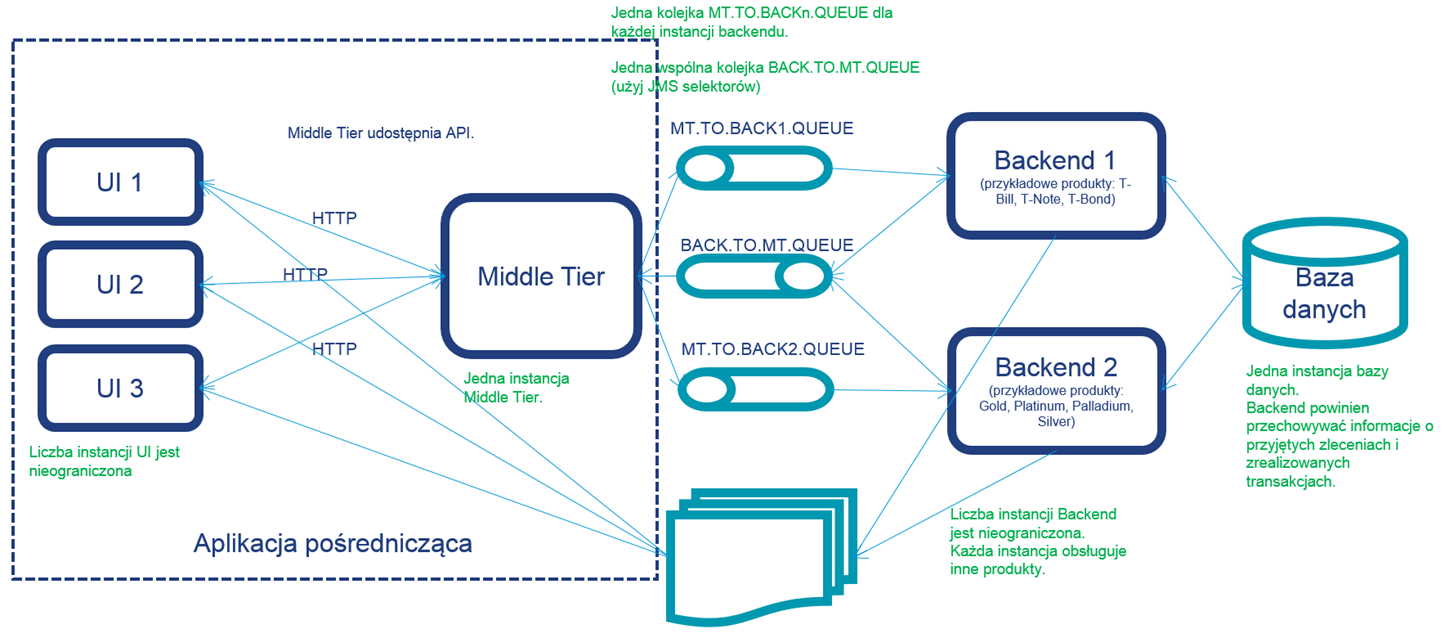
1. MT.TO.BACKn.QUEUE
   1. Komunikat: Nowe zlecenie  
      Atrybuty: produkt, wolumen zlecenia, typ zlecenia (kupno/sprzedaż), cena.
   2. Komunikat: Zmień zlecenie  
      Atrybuty: identyfikator zlecenia, produkt, wolumen zlecenia, typ zlecenia (kupno/sprzedaż), cena.
   3. Komunikat: Anuluj zlecenie  
      Atrybuty: identyfikator zlecenia
   4. Komunikat: Lista zawartych transakcji  
      Atrybuty: zakres czasu
   5. Komunikat: Lista zleceń  
      Atrybuty: zalogowany użytkownik, zakres czasu
2. BACK.TO.MT.QUEUE
   1. Komunikat: Nowa instancja backend uruchomiona  
      Atrybuty: ID instancji, lista produktów obsługiwanych przez instancję.
   2. Komunikat: Zlecenie przyjęto  
      Atrybuty: ID nowego zlecenia
3. EXECUTED.TRADES.TOPIC
   1. Komunikat: Transakcja zawarta  
      Atrybuty: produkt, wolumen transakcji, cena, ID zlecenia kupna, ID zlecenia sprzedaży

**Funkcjonalność – aplikacja front-end:**

Aplikacja front-end ma umożliwić wykonywanie operacji z aplikacji pośredniczącej. Sam jej kształt i sposób zaimplementownia nie podlega konkretnym wymaganiom.

1. Wyświetlenie listy transakcji odświeżanej na bieżąco poprzez WebSockets. Odświeżenie dotyczy listy trnsakcji jak i statusu każdej z transakcji.
2. Zgłoszenie nowej transakcji.
3. Anulowanie nie wykonanej transakcji.

**Schemat architektury:**

****