Programowanie III - projekt zaliczeniowy

Michał Gaj, Tomasz Januszek

Luty 2018

1 Założenie

Istotą projektu jest stworzenie mikroserwisu aplikacji, jako wariacji koncepcji SOA (*Service-Oriented Architecture*. Zastosowanie takiej architektury umożliwia tworzenie poszczególnych serwisów, które nie są zależne od innych, co pozwala na swobodne wprowadzanie zmian w jednym z nich, nie wpływając w żadnym stopniu na pozostałe. Takie rozwiązanie wiąże się ze stosunkowo większym nakładem pracy, co jest spowodowane dużym rozrzutem bloków kodu do wielu plików, jednakże "odwdzięcza" się to w momencie, kiedy przychodzi nam edytować poszczególne komponenty.

Nasz mikroserwis składa się z 3 mniejszych (niezwiązanych ze sobą bezpośrednio) mikroserwisów. Stanowi on część aplikacji dotyczącej symulowania gry na giełdzie, która cały czas jest rozwijana. Ta część projektu zakłada utworzenie modeli matematycznych w każdym z mikroserwisów, które przetwarzają dane oparte na wartościach walut na giełdzie i przesyłają wyniki do kontrolera, który przesyła dane do klienta. Komunikacja docelowo odbywa się poprzez sockety (gniazda) zawierający protokół przesyłu danych, adres oraz w naszym przypadku również numer portu.

Poszczególne mikroserwisy zawierające modele matematyczne traktowane są jako serwery oczekujące na połączenie i są przeznaczone jedynie do wykonania odpowiednich obliczeń i przesłania wyników. Zastosowanie koncepcji SOA pozwoli na stworzenie części aplikacji, która będzie mogła zostać użyta później, bez ryzyka przypadkowej ingerencji w jej kod.

2 Przygotowanie

W ramach przygotowań zrealizowano dwa założenia:

- Ustalono komunikację pomiędzy kontrolerem połączenia, a mikroserwisami:
  a) Ustalono adres i port komunikacyjny dla mikroserwisów i format przesyłu danych JSON.
  b) Ustalono format danych wejściowych:

  module: nazwa modułu,
  data:

  firstCurrency: [wartości waluty],

  secondCurrency: [wartości innej waluty]

- Utworzono modele matematyczne:

  a) Currency investment profitability:

  Moduł 1: Oblicza rentowność inwestycji w walutę na przestrzeni czasu podanej w danych wejściowych porównując iteracyjnie sąsiadujące ze sobą wartości.

  Moduł 2: Porównuje linię trendu dla dwóch podanych walut w tym samym przedziale czasowym. Porównanie następuje poprzez odjęcie przyrostu wartości waluty dodatkowej od przyrostu wartości waluty głównej w tym samym czasie na giełdzie. Wartości ujemne oznaczają, że waluta dodatkowa ma trend dodatni wyższy niż waluta główna.

  b) Percentage investment:

  Moduł 1: Oblicza dla zadanej tablicy wartości walut wzrost funduszy klienta przy inwestycji w daną walutę ze stałym oprocentowaniem z zakresu 1.0%-4.0%. Wyniki są zestawiane w postaci tablicy z krokami zgodnymi z czasem inwestycji.

  Moduł 2: Oblicza dla zadanej tablicy wartości walut wzrost funduszy klienta przy inwestycji w daną walutę ze zmiennym oprocentowaniem rozpoczynającym się na poziomie 2.5%. Procent rośnie w przypadku, gdy waluta ma w danym czasie trend rosnący, a spada w wypadku przeciwnym. Wartości graniczne dla oprocenowania to 1.0%-4.0%. Wyniki są zestawiane w postaci tablicy z krokami zgodnymi z czasem inwestycji.

  Moduł 3: Oblicza dla zadanej tablicy wartości walut wzrost funduszy klienta przy inwestycji w daną walutę ze zmiennym oprocentowaniem zmieniającym się z każdą zmianą wartości waluty. Zmiana ta nie jest sprzężona z wartościami waluty, jest ona losowa, mająca symulować zdarzenia losowa w trakcie statycznych inwestycji.

c) Stock value discrete change:

Moduł 1: Model ten ma obliczać wartości inwestycji giełdowej w walutę przy wpływie zewnętrznych czynników. Oznacza to, że na wartość akcji wpływa równocześnie wartość waluty oraz wcześniej wspomniane czynniki zewnętrzne generowane przez utworzony przez nas algorytm. Algorytm generuje zmienną stopę procentową zgodnie z normalnym rozkładem prawdopodobieństwa. Zakresy zmienności oprocentowania (z malejącym prawdopodobieństwem): 1.0%-3.0%, 3.1%-8.0%, 8.1%-15.0%, 15.1%-30.0%.

Kolejno wygenerowany procent jest zestawiany z procentem wynikającym ze wzrostu wartości waluty (procent rośnie) lub ze spadku wartości waluty (procent spada). Zestawiony procent jest przemnażany z wartością akcji w danym czasie, zgodnym z czasem zmiany wartości waluty.

3 Realizacja

Serwis został strukturalnie podzielony na główny mikroserwis odpowiadający za komunikację i przesył danych, 3 mikroserwisy zawierające jedynie modele matematyczne oraz ścieżkę zawierającą potrzebne biblioteki oraz programy pomocnicze (m.in. konwertery JSON). Do kontroli stanu poszczególnych mikroserwisów oraz całej aplikacji użyto systemu kontroli wersji Git.

Pierwszym krokiem było utworzenie plików odpowiadających za model mikroserwisów. Utworzono modele, a następnie kontrolery zapewniające nasłuchiwanie komunikacji, wywoływanie modułów oraz jeszcze wtedy nie utworzonych konwerterów. Postawiony serwer działał jako plik wykonywalny.

W następnym kroku prowadzono równocześnie pracę nad konwerterami oraz głównym serwisem aplikacji. Taka praca możliwa była dzięki rozdzieleniu zadań pomiędzy poszczególne serwisy aplikacji. Utworzono konwertery w oparciu o bibliotekę rapidjson. Utworzono SocketManager, który odpowiadał za komunikację klienta z serwerami. Po wykonaniu testów napotkano duże problemy z konwersją danych, co zostało opisane w sekcji *Obsługa błędów i napotkane problemy.*

Po usunięciu napotkanych przeszkód zaistniał problem związany z komunikacją z mikroserwisami, co również zostało opisane w w.w. sekcji. Po zredagowaniu kodu mikroserwisy odpowiadały na wysyłane żądania i przestały generować błędy. Dzięki zastosowanemu podejściu działają one cały czas, a po zamknięciu połączenia automatycznie oczekują na kolejne uwalniając uprzednio pamięć komputera.

4 Obsługa błędów i napotkane problemy

Problem dotyczący konwersji danych w postaci JSON związany był ze stopniem trudności konwersji nawet najprostszych danych i nakładu w postaci linii kodu. Dodatkowym problemem była niejasna dokumentacja i nastawienie biblioteki na czytanie plików JSON, ale nie ich tworzenie. Zmiana biblioteki na bibliotekę Nielsa Lohmanna "JSON for Modern C++" rozwiązała problem. Było to możliwe dzięki uniwersalnemu podejściu biblioteki i bardzo rozbudowanemu systemowi rzutowania danych.

Ostatnim napotkanym problemem była komunikacja z serwerem. Początkowo działający mikroserwis generował błędy i zatrzymywał swoje działanie po ponownym jego wywołaniu. Problemem okazała się próba odwołania do tych samych miejsc pamięci w mikroserwisie zamiast ponownej alokacji. Po usunięciu błędu i wprowadzeniu zarządzania alokacją pamięci pojawił się następny problem. Mikroserwisy po wielokrotnym wywołaniu spowalniały, aż do momentu zatrzymania się. Z tego powodu uściślono zarządzanie pamięcią, upewniono się, że pamięć jest całkowicie uwalniana po każdym uruchomieniu mikroserwisu i na nowo alokowana przy kolejnym wywołaniu.