PIISW, W08, IO, 2020/2021, semestr letni
Lista zadań nr 3: Tworzenie i testowanie backendu: serwisy RESTowe

Maciej Małecki
maciej.malecki@pwr.edu.pl

13 marca 2021

Zasady Pracy

- Rozwiązania zadań muszą być umieszczone w prywatnym repozytorium na portalu `github.com`.
- Prowadzący musi mieć uprawnienia do odczytu i zapisu dla tego repozytorium.
- Zadanie 1 jest obowiązkowe – w razie jego braku dalsza część listy nie będzie sprawdzana.

Wprowadzenie

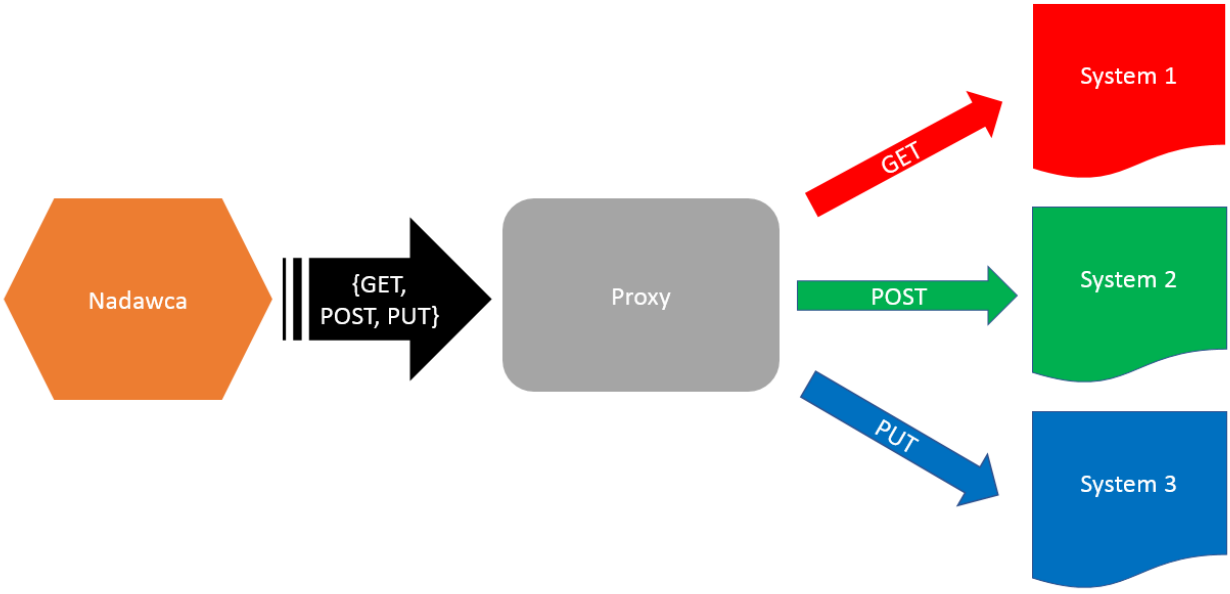
W zadaniu 1 należy zaimplementować uproszczoną wersję reverse-proxy. Reverse-proxy nie ma żadnej logiki biznesowej i służy jedynie jako pośrednik odbierający żądania i przekazujący je dalej. Reverse-proxy ukrywa architekturę systemów po stronie odbiorcy przed nadawcą, on (nadawca) niezależnie od rodzaju żądania cały czas rozmawia z jednym systemem.

Oceny

Punkty:	< 7	$7 - 8$	$9 - 10$	$11 - 12$	$13 - 14$	15
Ocena:	$2,0$	$3,0$	$3,5$	$4,0$	$4,5$	$5,0$

Zadania

1. (4 pkt) Reverse-Proxy - zadanie podstawowe.
   Zaimplementuj szary element z rysunku 1, posiadający interfejsy RESTowe: wejściowy i wyjściowy.
   - Interfejs wejściowy przyjmuje żądanie od nadawcy i w zależnosci od jego typu przekazuje do odpowiedniego adresata (system 1-3). Gdy adresat przetworzy żądanie, jego odpowiedź jest zwracana do nadawcy.
   - Ponieważ systemy zewnętrzne nie istnieją, należy je "zamokować" z użyciem dowolnego narzędzia (wiremock, json-server, ...).
   - Konfiguracja adresów systemów zewnętrznych powinna znaleźć się w pliku `application.properties` w formie np. `destination.get=localhost:8899`.



Rysunek 1: Diagram obrazujący sposób działania reverse proxy implementowanego w ramach zadania 1.

Wskazówka: Wygeneruj aplikację SpringBoot z wykorzystaniem Maven oraz z modułem Web (wykorzystaj w tym celu `https://start.spring.io`) oraz zaznajom się z `RestController` i `RestTemplate`.

2. (3 pkt) Testowanie automatyczne.
   Napisz testy automatyczne dla komponentu reverse-proxy stworzonego w zadaniu 1.

   - Dla każdego rodzaju żądania należy napisać co najmniej jeden test automatyczny.
   - Jeśli rodzaj testu tego wymaga, zależność (czyli obecność systemu będącego na końcu testowanego interfejsu) powinna być uruchamiana przez sam test, a nie przez użytkownika – ręcznie.

   Wskazówka: Zaznajom się z `TestRestTemplate` oraz np. `Wiremock` i jego wsparciem dla testów JUnit.

3. (2 pkt) Obsługa wyjątków.
   Reverse-proxy nie interpretuje błędów, jakie mogą wystąpić podczas komunikacji systemami zewnętrznymi. Błędy takie należy przekazywać dalej.

   - Do obsługi wyjątków należy użyć `ControllerAdvice`.
   - W zależnosci od implementacji testy automatyczne powinny być rozszerzone o przypadki sprawdzające sytuacje, w których system obsługuje (przekazuje) wyjątki.

4. (3 pkt) Testowanie integracyjne.

   - Do testowania integracyjnego należy użyć aplikację `Postman`. Przykłady testów API z użyciem tego narzędzia można znaleźć na stronie:
     `http://blog.getpostman.com/2014/03/07/writing-automated-tests-for-apis-using-p`
   - Dla każdego rodzaju żądania należy napisać co najmniej dwa test integracyjne (jeden pozytywny – tzw. *happy path* – i jeden negatywny, testujący sytuację błędną).

5. (3 pkt) CircleCI.
   Aktywuj system CI `circleci.com` dla repozytorium z rozwiązaniem listy 3. Upewnij się, że aplikacja jest budowana oraz że uruchamiane są testy jednostkowe (nie ma konieczności uruchamiania testów integracyjnych w ramach CircleCI). Status aplikacji powinien być "zielony".