

# PIISW, W08, IO, 2018/2019, semestr letni

## Lista zadań nr 3: Tworzenie i testowanie backendu: serwisy RESTowe

Adam Puchalski  
adam.puchalski@pwr.edu.pl

26 lutego 2019

### Zasady Pracy

- Rozwiązania zadań z tej listy muszą znaleźć się w prywatnym repozytorium na `github.com` (może to być to samo repozytorium, które założono w ramach prac nad listą nr 2).
- Prowadzący zajęcia musi mieć uprawnienia do odczytu i zapisu do tego repozytorium.
- Zadanie 1 jest obowiązkowe. W razie jego braku dalsza część listy nie będzie sprawdzana.

### Wprowadzenie

- W zadaniu 1 trzeba zaimplementować uproszczoną wersję reverse-proxy. Jako takie reverse-proxy nie ma żadnej logiki biznesowej i służy jedynie jako pośrednik odbierając requesty i przekazując je dalej. Reverse-proxy ukrywa architekturę systemów po stronie odbiorcy przed nadawcą, on (nadawca) niezależnie od rodzaju requestu cały czas rozmawia z jednym systemem (proxy).
- Kolejne zadania uzupełniają implementację serwera o testy Junitowe (jednostkowe bądź integracyjne), obsługę wyjątków oraz typowe testy integracyjne.
- Ostatnie zadanie polega na podłączeniu CI do repozytorium z zadaniem i doprowadzenie stanu builda do zieleni.

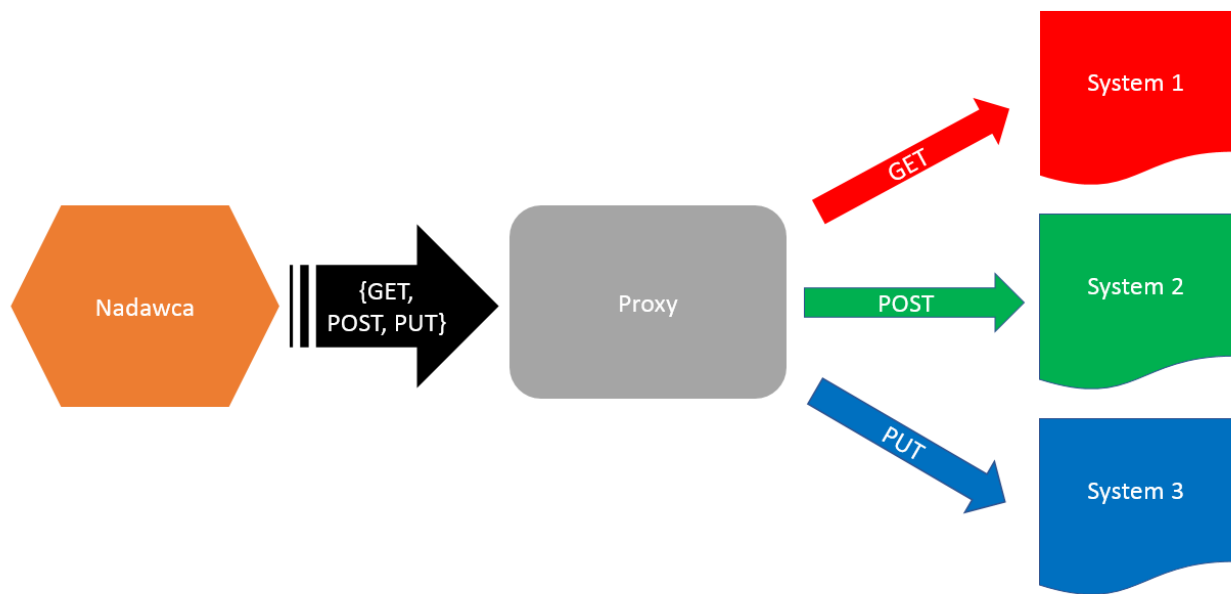
### Oceny

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.  
W ramach ćwiczeń do zaimplementowania jest szary element z rysunku 1. Posiada on interfejs RESTowy wejściowy i wyjściowy.

- Interfejs wejściowy przyjmuje request od nadawcy i w zależności od jego typu przekazuje do odpowiedniego adresata (System 1-3). Gdy adresat przetworzy request, jego odpowiedź jest zwracana do nadawcy.
- Ponieważ systemy zewnętrzne nie istnieją, należy je zamockować 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:** Zalecane jest rozpoczęcie od nowego springbootowego projektu z modulem Web oraz zaznajomienie się z `RestController` i `RestTemplate`.

## 2. (3 pkt) Testowanie automatyczne.

W ramach ćwiczenia trzeba stworzone w zadaniu 1 reverse-proxy pokryć testami automatycznymi.

- Dla każdego rodzaju requestu powinien zostać napisany co najmniej jeden test automatyczny.
- Jeśli rodzaj testu tego wymaga, zależność (czyli obecność systemu będącego na końcu testowanego interfejsu) musi być uruchamiana przez sam test, a nie przez użytkownika zanim testy zostaną uruchomione

**Wskazówka:** Zalecane jest zaznajomienie się z `TestRestTemplate` oraz np. Wiremock i jego wsparciem dla JUnitów.

## 3. (2 pkt) Obsługa wyjątków.

Reverse-proxy nie jest w stanie interpretować błędów jakie mogą się przytrafić podczas komunikacji systemami zewnętrznymi, musi więc komunikaty o błędach odpowiednio przekazywać dalej.

- Do obsługi wyjątków należy użyć `ControllerAdvice`.

- W zależności od implementacji testy automatyczne powinny być rozszerzone o przypadki sprawdzające sytuacje w których system musi obsługiwać wyjątki.

4. (3 pkt) Testowanie integracyjne.

- Do testowania integracyjnego zalecane jest użycie aplikacji **Postman**. Przykłady odnosnie pisania testów api z użyciem tego narzędzia są tutaj:  
<http://blog.getpostman.com/2014/03/07/writing-automated-tests-for-apis-using-p>
- Dla każdego rodzaju requestu powinny zostać napisane conajmniej dwa test integracyjne (1 pozytywny i 1 negatywny).

5. (3 pkt) Travis CI.

W ramach tego ćwiczenia trzeba do serwisu **travis-ci.org** podłączyć swoje repozytorium z zaimplementowanym reverse-proxy i doprowadzić build do zieleni - aby się kompilował oraz żeby wszystkie testy były "zielone".