Ćwiczenia nr 2 – Mock-owanie backend-u

Wstęp

Celem ćwiczeń jest zapoznanie się z pojęciem mock-owania serwisów http. Do ćwiczenia wykorzystamy już istniejący backend Narodowego Banku Polskiego. Dokumentację API można znaleźć pod adresem http://api.nbp.pl/

Aby zacząć wykonaj następujące kroki:
1. Ściągnij projekt z poprzednich zajęć za pomocą instrukcji:
'git clone https://github.com/anicos/quickstart.git'
2. Przełącz się na branch 'cw2'
3. Ściągnij zależności używając komendy 'npm install'
4. Uruchom test używając komendy 'npm test'
5. W pliku 'src/app/services/gold-price-service.service.integration.spec.ts' znajdziesz przykład testu integracyjnego mock-ujacego backend.
6. Rozwiąż poniższe zadania bazując na powyższym przykładzie

Ćwiczenie nr1 – obliczanie ceny złota na zadany dzień (ocena 3 lub 4)

Aktualnie w kodzie znajduje się serwis zwracający dzisiejszą cenę złota 'src/app/services/gold-price-service.service.ts'. Serwis ten należy rozszerzyć o metodę zwracającą cenę złota na zadany dzień w tym celu powinniśmy zaimplementować metodę

```
getGoldPriceByDate(date: String): Observable<Number>
```

Kryteria akceptacji:

- dopisanie testów w następujących miejscach:

a) src/app/services/gold-price-http-client.service.spec.ts

b) src/app/services/gold-price-service.service.integration.spec.ts

Wskazówki:

Aby pobrać cenę złota na zadany dzień należy wykonać następujące zapytanie na backend

http://api.nbp.pl/api/cenyzlota/{data}?format=json gdzie *data* w nawiasach klamrowych jest dniem w formacie 'yyyy-mm-dd'.

Poniżej przykład zapytania na dzień 13. marca roku 2014:

http://api.nbp.pl/api/cenyzlota/2014-03-13?format=json

Ćwiczenie nr 2 – Serwis pobierający walutę z najwyższym kursem średnim.

Do projektu został dodany nowy service ‘AverageExchangeRatesService;. Serwis ten zawiera metodę ‘getCurrencyWithTheHighestCurrentRate’ która, powinna zwracać walutę o najwyższym kursie średnim. W tym celu używamy zapytania GET http://api.nbp.pl/api/exchangerates/tables/a?format=json .

Serwis ten(AverageExchangeRatesService) powinien być zbudowany analogicznie do GoldPriceService czyli powinien być kompozycją AverageExchangeHttpClient i AverageExchangeCalculator .

W myśl zasady Single Responsability AverageExchangeHttpClient jest odpowiedzialny za pobranie danych z serwera a AverageExchangeCalculator za wyliczenie waluty z najwyższym kursem średnim.

Każda klasa (AverageExchangeCalculator, AverageExchangeHttpClient, AverageExchangeRatesService) powinna posiadać testy analogicznie do serwisów związanych z ceną złota.

W myśl zasady TDD należy zacząć pracę od testu integracyjnego dla serwisu AverageExchangeRatesService.