09.12.2016

Integracja Aplikacji i Systemów

Dokumentacja projektu

Angelika Osińska (164179), Marek Osiński (164165) IwB 08

Integracja Aplikacji i Systemów

Dokumentacja projektu

Temat:
Integracja RESTful services – hub i klient.

Polecenie:

1. *Napisać aplikację (HUB), pobierającą dane od dwóch dowolnych dostawców RESTful web services i zwracającą te dane w ujednoliconej postaci.*
2. *Napisać aplikację będącą klientem aplikacji HUB.*

Opis pomysłu

Aplikacja, która służy do porównywania aktualnych kursów walut:

- Narodowego Banku Polskiego (Tabela A)
- kursów Europejskiego Banku Centralnego (EBC) względem PLN.

Implementacja

Aplikacje zostały napisane w języku Java z użyciem następujących technologii:

- Java EE 1.7
- JAXB
- JSF
- JAX-RS

Ponadto użyto poniższych biblioteki i API:

- Jersey API
- Jackson API
- PrimeFaces UI Framework

Obydwie aplikacje mogą być uruchamiane w kontenerach serwerów GlassFish lub Tomcat.

Aplikacja HUB (*hub2*)

Aplikacja jest jednocześnie serwerem i klientem RESTful web services. Na żądanie klienta, HUB pobiera w sekwencji dane od dwóch dostawców za pomocą RESTful web services. HUB buforuje dane służące do udzielenia odpowiedzi klientom. Dane od dostawców pobierane są tylko w dwóch przypadkach: encja wynikowa jest pusta (pierwsze uruchomienie) lub data systemowa różni się od daty notowań, znajdującej się w encji wynikowej. Dzięki temu zapytania do dostawców mogą się odbywać tylko raz dziennie. Po otrzymaniu odpowiedzi od dostawców, po zmapowaniu danych JSON i XML na obiekty, odbywa się proces integracji, mający na celu przygotowanie jednolitej odpowiedzi do klienta. Integracja polega na konwersji i skojarzeniu danych o kursach walut pochodzących od dwóch dostawców. W związku z tym, że koszyk walut Banku ECB jest podzbiorem koszyka walut NBP, podczas kojarzenia kursów walut, waluty nieobecne w koszyku ECB są pomijane. W tym celu zastosowano mechanizm refleksji (reflection) do odkrywania nazwy metody zwracającej kurs konkretnej waluty z encji *ECBRates*. Gdy *getter()* dla określonej waluty nie zostanie znaleziony, mechanizm obsługi błędów nie dopuszcza do dodania tej waluty do wynikowej listy.

Poniżej przedstawiono fragment kodu odpowiedzialny za kojarzenie kursów walut (klasa *XchangeRatesService*):

```
private double getECBrateByCode(ECBRates ecbRates, String code) throws
NoSuchMethodException, IllegalAccessException, InvocationTargetException {

        String methodName;

        methodName = "get" + code;

        Method getNameMethod = ecbRates.getClass().getMethod(methodName);
        double rate = (double) getNameMethod.invoke(ecbRates);
        return rate;

    }
```

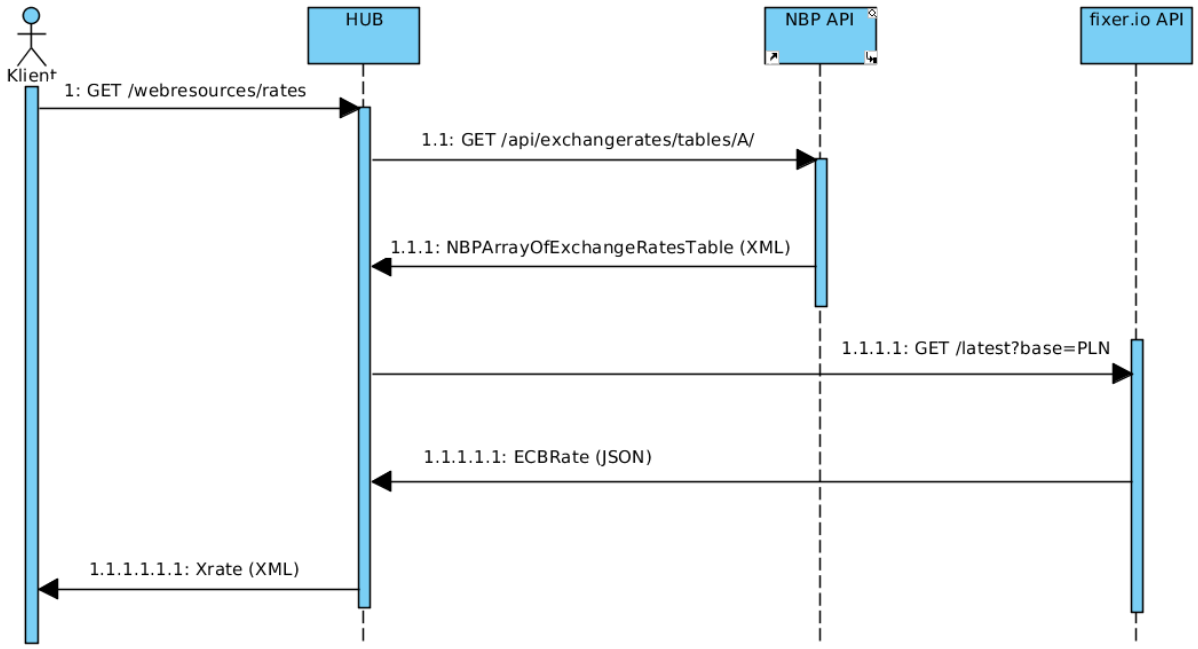
Fragment kodu tworzący wynikową encję (wynikowy koszyk kursów walut):

```
ecbRate = this.round(1 / ecbRate, 8);
this.rates.getXrates()
                    .getXrate()
                    .add(new Xrate(rate.getCurrency(),
                                   rate.getCode(),
                                   String.format("%.8f",rate.getMid(),
                                   String.format("%.8f", ecbRate)));
```

Ponadto kursy ECB są podawane względem PLN, zatem kursy wynikowe należy przedstawić jako odwrotność kursów ECB (1/x). Zaokrąglone wartości *Double* (z precyzją 8 miejsc po przecinku) zamieniane są na *String* i formatowane w taki

sposób, żeby wynik nie pojawił się w notacji naukowej. Po zakończeniu procedury integracji, hub udziela odpowiedzi REST klientowi w formacie XML.

Uproszczony diagram sekwencji dla aplikacji HUB:



Dostawcy i struktura danych

Dostawca 1 (Narodowy Bank Polski)

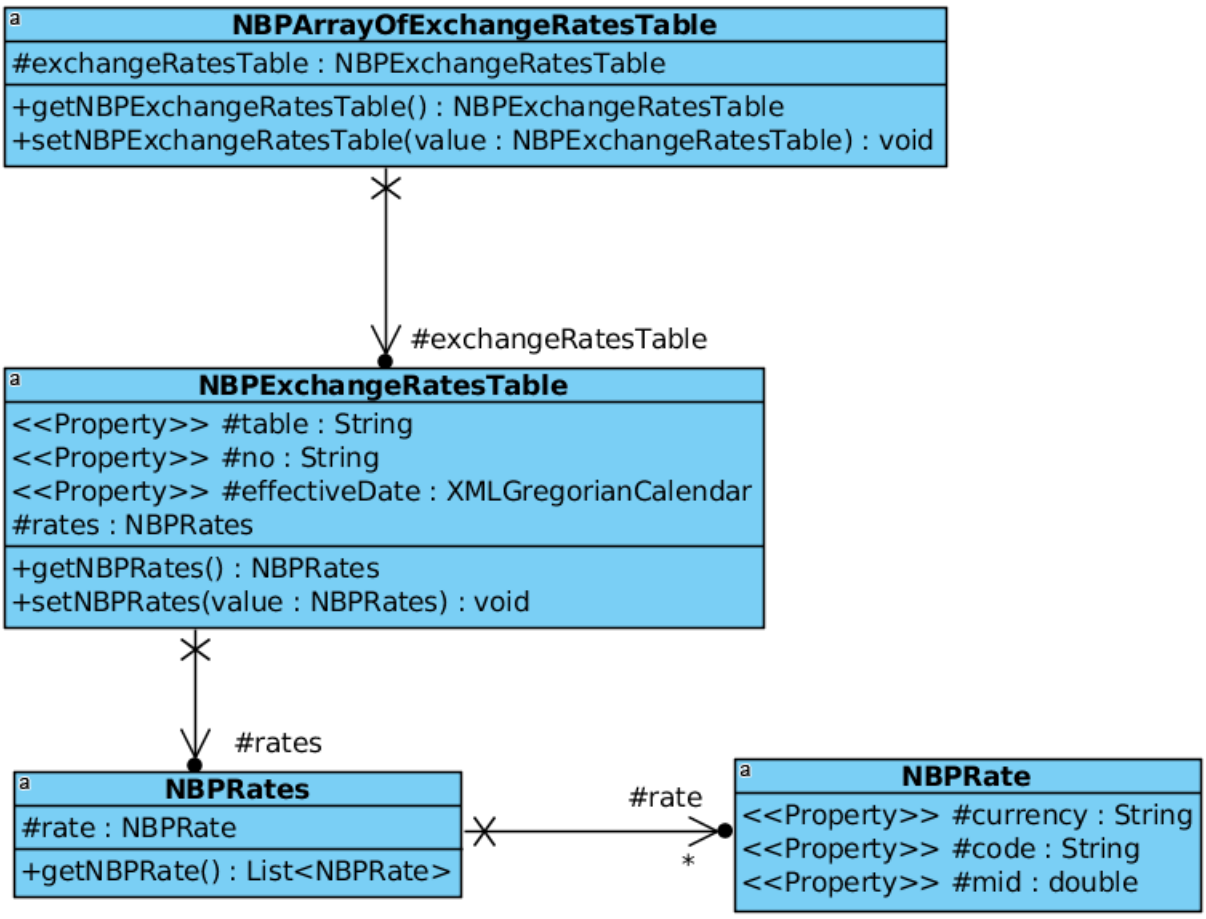
Wywołanie: GET http://api.nbp.pl/api/exchangerates/tables/A

W odpowiedzi otrzymujemy ciąg XML zawierający dane o bieżących kursach walut NBP wg tabeli A, który opisuje poniższy schemat:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Currency" type="xs:string"/>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="Mid" type="xs:float"/>
  <xs:element name="Rate">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Currency"/>
        <xs:element ref="Code"/>
        <xs:element ref="Mid"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Table" type="xs:string"/>
  <xs:element name="No" type="xs:string"/>
  <xs:element name="EffectiveDate" type="xs:date"/>
  <xs:element name="Rates">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Rate" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExchangeRatesTable">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Table"/>
        <xs:element ref="No"/>
        <xs:element ref="EffectiveDate"/>
        <xs:element ref="Rates"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ArrayOfExchangeRatesTable">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ExchangeRatesTable"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Dla reprezentacji danych zawartych w odpowiedzi XML, w aplikacji utworzono klasy:



Pola:

NBPArrayOfExchangeRatesTable:
#exchangeRatesTable – tabela A kursów walut NBP

NBPExchangeRatesTable:
#table – symbol tabeli
#no – numer tabeli
#effectiveDate – data notowania
#rates – lista kursów

NBPRates:
#rate – lista kursów

NBPRate:
#currency – nazwa waluty
#code – kod waluty
#mid – aktualny kurs waluty

Dostwca 2 (Europejski Bank Centralny za pośrednictwem fixer.io API):

Witryna fixer.io udostępnia API dostarczające dane o bieżących notowaniach kursów Europejskiego Banku Centralnego (ECB)

Wywołanie: GET http://api.fixer.io/latest?base=PLN

W odpowiedzi otrzymujemy ciąg JSON zawierający dane o bieżących kursach walut Europejskiego Banku Centralnego względem PLN, który opisuje poniższy schemat:

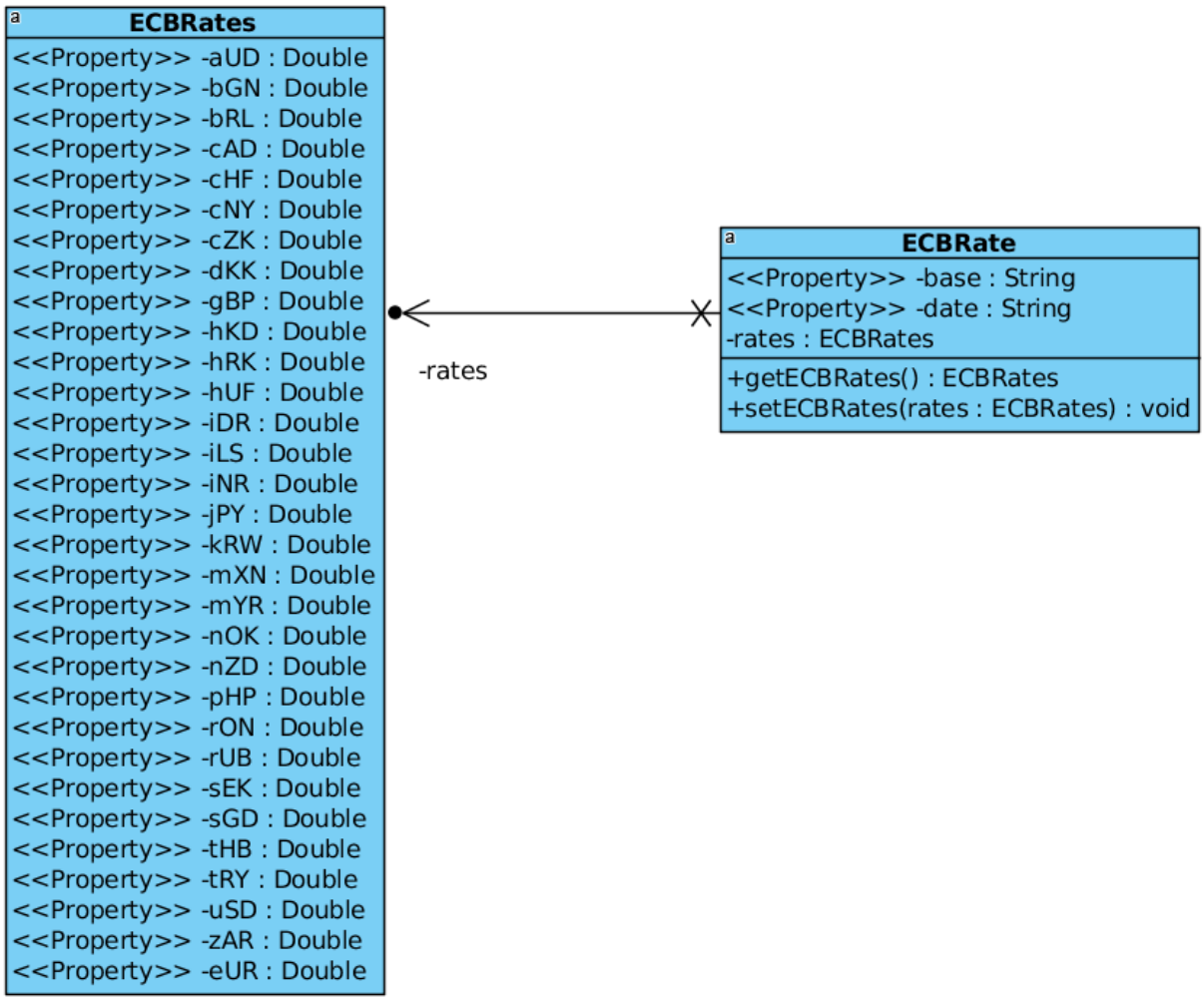
```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "base": {
      "type": "string"
    },
    "date": {
      "type": "string"
    },
    "rates": {
      "type": "object",
      "properties": {
        "AUD": {
          "type": "number"
        },
        "BGN": {
          "type": "number"
        },
        "BRL": {
```

```
      "type": "number"
    },
    "CAD": {
      "type": "number"
    },
    "CHF": {
      "type": "number"
    },
    "CNY": {
      "type": "number"
    },
    "CZK": {
      "type": "number"
    },
    "DKK": {
      "type": "number"
    },
    "GBP": {
      "type": "number"
    },
    "HKD": {
      "type": "number"
    },
    "HRK": {
      "type": "number"
    },
    "HUF": {
      "type": "number"
    },
    "IDR": {
      "type": "number"
    },
    "ILS": {
      "type": "number"
    },
    "INR": {
      "type": "number"
    },
    "JPY": {
      "type": "number"
    },
    "KRW": {
      "type": "number"
    },
    "MXN": {
      "type": "number"
    },
    "MYR": {
      "type": "number"
    },
    "NOK": {
      "type": "number"
    },
    "NZD": {
      "type": "number"
    },
    "PHP": {
```

```
      "type": "number"
    },
    "RON": {
      "type": "number"
    },
    "RUB": {
      "type": "number"
    },
    "SEK": {
      "type": "number"
    },
    "SGD": {
      "type": "number"
    },
    "THB": {
      "type": "number"
    },
    "TRY": {
      "type": "number"
    },
    "USD": {
      "type": "number"
    },
    "ZAR": {
      "type": "number"
    },
    "EUR": {
      "type": "number"
    }
  },
  "required": [
    "AUD",
    "BGN",
    "BRL",
    "CAD",
    "CHF",
    "CNY",
    "CZK",
    "DKK",
    "GBP",
    "HKD",
    "HRK",
    "HUF",
    "IDR",
    "ILS",
    "INR",
    "JPY",
    "KRW",
    "MXN",
    "MYR",
    "NOK",
    "NZD",
    "PHP",
    "RON",
    "RUB",
    "SEK",
    "SGD",
```

```
          "THB",
          "TRY",
          "USD",
          "ZAR",
          "EUR"
        ]
      }
    },
    "required": [
      "base",
      "date",
      "rates"
    ]
}
```

Dla reprezentacji danych zawartych w odpowiedzi JSON, w aplikacji utworzono klasy:



Pola:

ECBRate
-base – waluta bazowa
-date – data notowania
-rates – zbiór kursów walut

ECBRates
aUD – kurs AUD
bGN – kurs BGN
…
…
…
eUR – kurs EUR

Odpowiedź aplikacji HUB

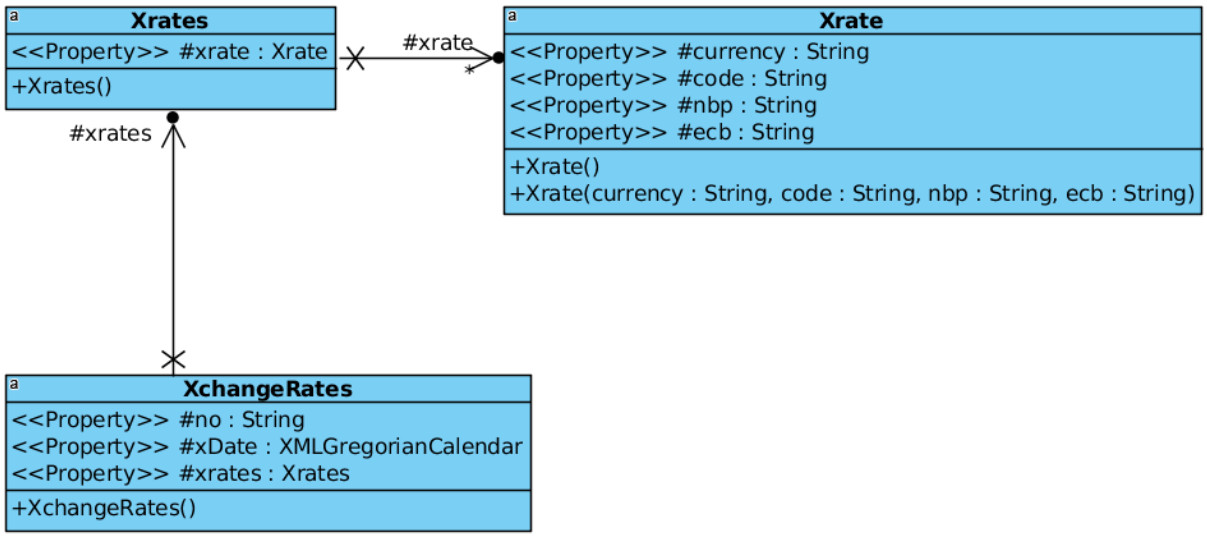
Wywołanie: GET http:// <hub address>:8080/hub2/webresources/rates

HUB udziela odpowiedzi na zapytania w formacie XML o następującym schemacie:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Currency" type="xs:string"/>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="nbp" type="xs:string"/>
  <xs:element name="ecb" type="xs:string"/>
  <xs:element name="Xrate">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Currency"/>
        <xs:element ref="Code"/>
        <xs:element ref="nbp"/>
        <xs:element ref="ecb"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="No" type="xs:string"/>
  <xs:element name="XDate" type="xs:date"/>
  <xs:element name="Xrates">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Xrate" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="XchangeRates">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="No"/>
        <xs:element ref="XDate"/>
        <xs:element ref="Xrates"/>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Dla reprezentacji danych zawartych w odpowiedzi XML, w aplikacji utworzono klasy:



Pola:

XchangeRates
#no – numer tabeli NBP
#xDate – data notowania
#xRates – lista kursów walut

Xrates
#xrate – lista kursów walut

Xrate
#currency – nazwa waluty
#code – kod waluty
#nbp – kurs NBP
#ecb – kurs ECB

Wszystkie klasy reprezentujące encje zostały skonstruowane w sposób umożliwiający mapowanie ich na obiekty XML lub JSON (za pomocą odpowiednich adnotacji).

Podczas implementacji napotkano następujące problemy:

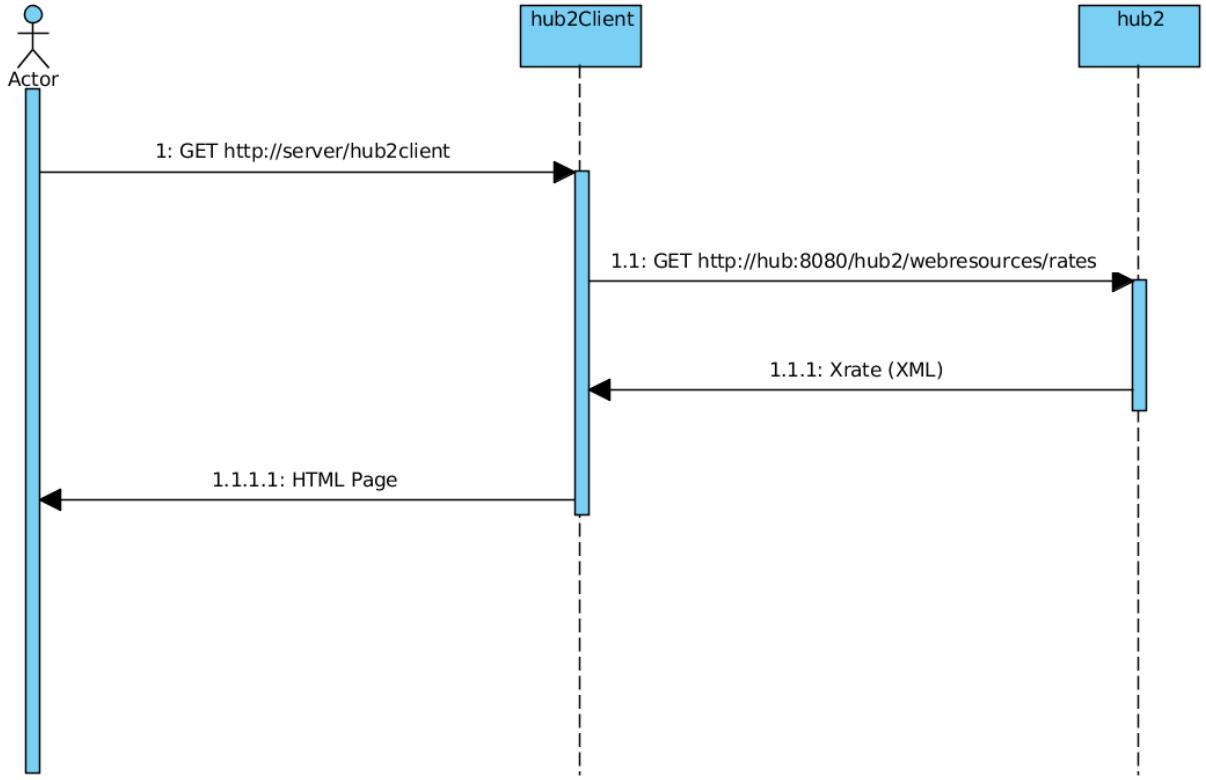
— Niewygodny sposób reprezentacji danych z fixer.io
(Koszyk kursów walut nie jest przedstawiony w formie listy, tylko osobnych pól typu property dla każdej waluty).

W kolejnej iteracji można byłoby zaimplementować szczegółową obsługę błędów dla web services i skorzystać z obiektów transferowych (DTO) żeby zwiększyć elastyczność w zakresie struktur danych przetwarzanych przez hub.

Aplikacja Klient (*hub2Client*)

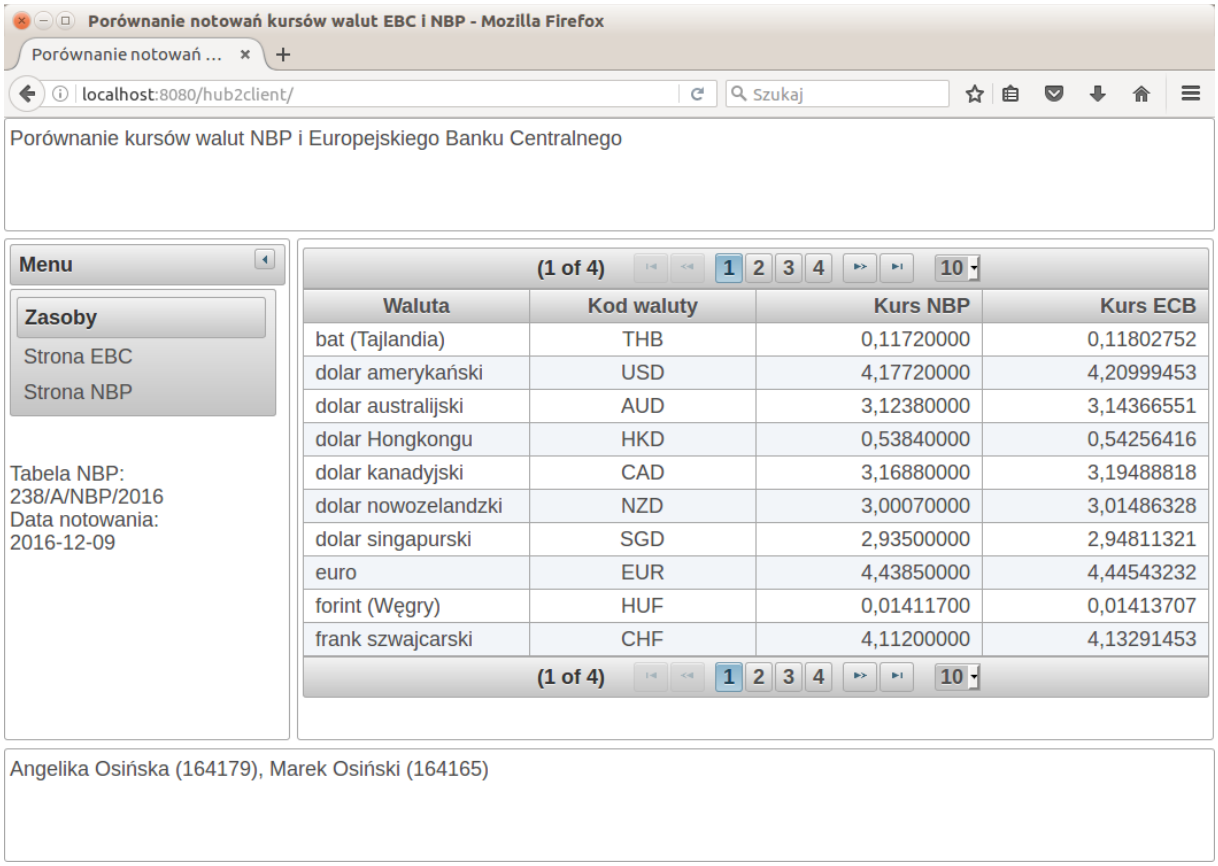
Program jest aplikacją webową będącą jednocześnie klientem RESTful web services. Aplikacja pobiera dane zwracane przez aplikację hub i przedstawia je w formie tabeli w przeglądarce. Wywołanie strony WWW aplikacji powoduje zainicjowanie procesu klienta REST i pobierania danych z serwera *hub2*. Po otrzymaniu odpowiedzi z serwera hub2, dane XML są mapowane na encje, zostaje wyodrębniona lista z koszykiem kursów walut i jest ona przekazywana do klienta (przeglądarki) w formie tabeli osadzonej na stronie WWW. Klient nie buforuje danych i przy każdym odświeżeniu strony aplikacji, nowe dane są ponownie pobierane z serwera aplikacji *hub2*. Oprócz danych o aktualnych kursach, na stronie wyświetla się również informacja o numerze tabeli NBP oraz dacie notowań. Do realizacji interfejsu użytkownika użyto komponentów z biblioteki PrimeFaces UI.

Uproszczony diagram sekwencji:



url: http://server:8080/hub2client

Wygląd strony WWW:

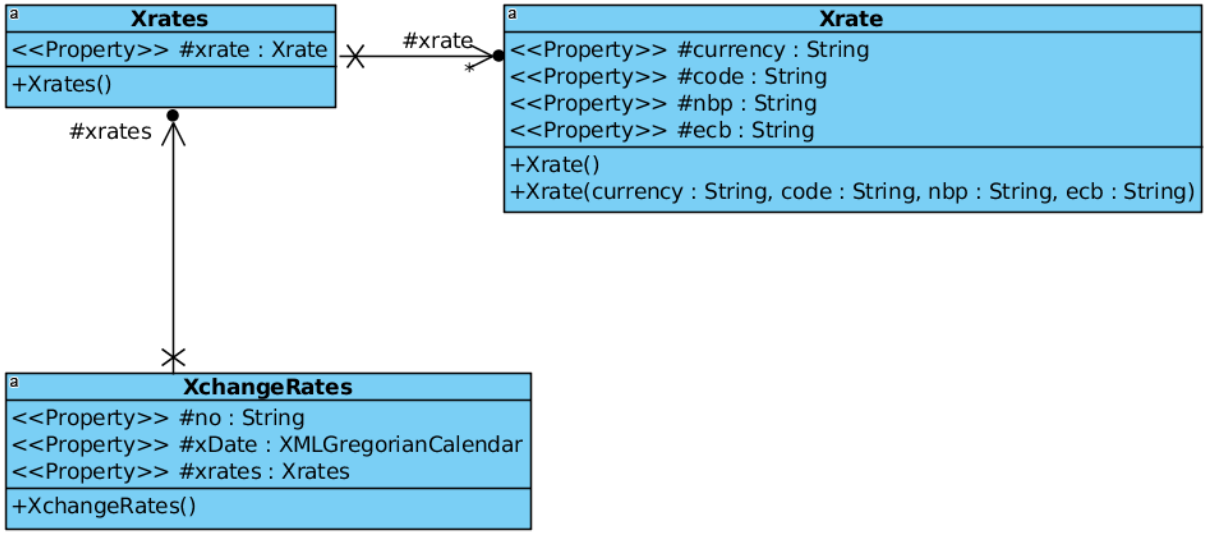


Aplikacja konsumuje odpowiedź REST w formacie XML o następującym schemacie:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Currency" type="xs:string"/>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="nbp" type="xs:string"/>
  <xs:element name="ecb" type="xs:string"/>
  <xs:element name="Xrate">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Currency"/>
        <xs:element ref="Code"/>
        <xs:element ref="nbp"/>
        <xs:element ref="ecb"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="No" type="xs:string"/>
  <xs:element name="XDate" type="xs:date"/>
  <xs:element name="Xrates">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="Xrate" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="XchangeRates">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="No"/>
        <xs:element ref="XDate"/>
        <xs:element ref="Xrates"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Dla reprezentacji danych zawartych w odpowiedzi XML, w aplikacji utworzono klasy:



Pola:

XchangeRates
#no – numer tabeli NBP
#xDate – data notowania
#xRates – lista kursów walut

Xrates
#xrate – lista kursów walut

Xrate
#currency – nazwa waluty
#code – kod waluty
#nbp – kurs NBP
#ecb – kurs ECB

Wszystkie klasy reprezentujące encje zostały skonstruowane w sposób umożliwiający mapowanie ich na obiekty XML lub JSON (za pomocą odpowiednich adnotacji).

Podczas pracy nad aplikacją nie napotkano żadnych problemów.

W kolejnej iteracji można byłoby upiększyć interfejs graficzny i zaimplementować DTO oraz możliwość eksportu danych np. do Excel'a.

Podział pracy:

Angelika Osińska: Pomysł, opracowanie struktur danych, i dokumentacja.

Marek Osiński: Programowanie, testowanie i dokumentacja.

Kod źródłowy:

hub2:

https://github.com/164165/hub2.git

hub2Client:

https://github.com/164165/hub2client.git