Projekt Rest Web Service

Sklep internetowy z kartami do gry Magic: The Gathering
Specyfikacja Wejście-Wyjście
wersja 1.0

Paweł Marczak i Łukasz Kosmaty

7 czerwca 2021

Spis treści

1 Wstęp

1.1 Przeznaczenie

Celem zaprojektowanego web service-u jest automatyzacja procesu składania zamówień na karty w sklepie sprzedającym single (karty na sztuki) do gry Magic: The Gathering.

W obecnej wersji, web service pozwala na przegląd magazynu sklepu (narazie zdefiniowanego statycznie), uzupełnienie koszyka zamówienia przez użytkownika, złożenie zamówienia (z kontrolą poprawności zamówienia i odpowiednią aktualizacją stanu magazynu) oraz wydrukowanie potwierdzenia zamówienia w formacie pdf. By obsługiwać serwis można korzystać z przygotowanego przez nas klienta obsługiwanego za pośrednictwem html.

Serwis został stworzony z myślą o dalszym rozwoju. Przykładowymi ścieżkami rozwoju są: integracja z zewnętrznymi bazami danych, możliwość otwartej rejetracji, szyfrowanie przesyłanych komunikatów (np. przy użyciu standardu SSL), integracja z zewnętrznymi systemami płatniczymi.

1.2 Zakres

Dokument opisuje standardy sieciowe, według których zbudowana jest usługa oraz prezentuje przykładowy sposób jej używania.

2 Specyfikacja usługi Web Service

Usługa „Sklep Mtg" zaimplementowana została jako usługa sieciowa (Web Service) z użyciem architektury Rest.

Usługa dostępna jest przez protokół HTTP (np. za pośrednictwem zaprojektowanego w HTML klienta).

2.1 Adres usługi

Usługa sklepu dostępna jest pod adresem http://25.76.141.122:8080/RestProjectServer/webresources/sklep/ , a odpowiendnie metody do jego obsługi i ścieżki do nich zostaną opisane w dalszej części.

2.2 Specyfikacja wadl

Listing 1: Plik wadl

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <application xmlns="http://wadl.dev.java.net/2009/02">
3     <doc xmlns:jersey="http://jersey.java.net/"
          jersey:generatedBy="Jersey: 2.28 2019-01-25 15:18:13"
          />
4     <doc xmlns:jersey="http://jersey.java.net/" jersey:hint="
          This is simplified WADL with user and core resources
          only. To get full WADL with extended resources use
          the query parameter detail. Link: http:
          //25.76.141.122:8080/RestProjectServer/webresources/
          application.wadl?detail=true"/>
5     <grammars>
6         <include href="application.wadl/xsd0.xsd">
7             <doc title="Generated" xml:lang="en"/>
8         </include>
```

```
9     </grammars>
10    <resources base="http://25.76.141.122:8080/
          RestProjectServer/webresources/">
11        <resource path="/sklep">
12            <method id="getMagazyn" name="GET">
13                <response>
14                    <representation mediaType="application/
                          json"/>
15                </response>
16            </method>
17            <resource path="/stan/{cardName}">
18                <param xmlns:xs="http://www.w3.org/2001/
                      XMLSchema" name="cardName" style="
                      template" type="xs:string"/>
19                <method id="getStan" name="GET">
20                    <response>
21                        <ns2:representation xmlns:ns2="http:
                              //wadl.dev.java.net/2009/02"
                              xmlns="" element="stan" mediaType
                              ="application/json"/>
22                    </response>
23                </method>
24                <method id="updateKoszyk" name="PUT">
25                    <request>
26                        <param xmlns:xs="http://www.w3.org
                              /2001/XMLSchema" name="ilosc"
                              style="query" type="xs:int"/>
27                    </request>
28                    <response>
29                        <ns2:representation xmlns:ns2="http:
                              //wadl.dev.java.net/2009/02"
                              xmlns="" element="coreLinki"
                              mediaType="application/json"/>
30                    </response>
31                </method>
32            </resource>
33            <resource path="/user/koszyk">
34                <method id="getKoszyk" name="GET">
35                    <response>
36                        <representation mediaType="
                              application/json"/>
37                    </response>
38                </method>
39                <method id="zlozZamowienie" name="POST">
40                    <response>
41                        <ns2:representation xmlns:ns2="http:
                              //wadl.dev.java.net/2009/02"
                              xmlns="" element="potwZamowienia"
                               mediaType="application/json"/>
```

```
42                      </response>
43                  </method>
44              </resource>
45              <resource path="/user/koszyk/{cardName}">
46                  <param xmlns:xs="http://www.w3.org/2001/
                        XMLSchema" name="cardName" style="
                        template" type="xs:string"/>
47                  <method id="deleteFromKoszyk" name="DELETE">
48                      <response>
49                          <ns2:representation xmlns:ns2="http:
                                //wadl.dev.java.net/2009/02"
                                xmlns="" element="coreLinki"
                                mediaType="application/json"/>
50                      </response>
51                  </method>
52                  <method id="zwrocpozycjeZKoszyka" name="GET">
53                      <response>
54                          <ns2:representation xmlns:ns2="http:
                                //wadl.dev.java.net/2009/02"
                                xmlns="" element="stan" mediaType
                                ="application/json"/>
55                      </response>
56                  </method>
57              </resource>
58              <resource path="konto">
59                  <method id="getKonto" name="GET">
60                      <response>
61                          <ns2:representation xmlns:ns2="http:
                                //wadl.dev.java.net/2009/02"
                                xmlns="" element="daneKlienta"
                                mediaType="application/json"/>
62                      </response>
63                  </method>
64              </resource>
65          </resource>
66      </resources>
67  </application>
```

2.3 Operacje usługi

Usługa obsługuje operacje :

- getMagazyn - zwraca informacje o aktualnym stanie magazynu sklepu
- getStan- zwraca informacje o stanie pojedynczej pozycji z magazynu
- updateKoszyk - ustawia w koszyku klienta podaną liczbę kopii danej karty

- getKoszyk - zwraca informacje o aktualnym stanie koszyka klienta
- zwrocpozycjeZKoszyka-zwraca informacje o aktualnym stanie wybranej pozycji z koszyka klienta
- deleteFromKoszyk - usuwa z koszyka daną pozycję (wszystkie karty o danej nazwie)
- zlozZamowienie - składa zamówienie (weryfikuje poprawność koszyka, aktualizuje magazyn i stan konta użytkownika, wysyła informacje o potwierdzeniu zamówienia)

2.4 Zwracane obiekty

Szczegółowiej opisane zostaną tylko elementy budzące wątpliwości.

2.4.1 Stan

Obiekty zawierające informacje o danym stanie związanym z daną kartą(zarówno magazyn jak i koszyk klienta składają się z listy stanów ArrayList<Stan>), tzn.:

- karta (Karta)- obiekt karta zawierający informacje o karcie
- na_stanie (int) -liczba kopii karty karta w danym stanie
- cena (float)- cena pojedynczej kopii karty karta brutto
- wartosc_razem- cena wszystkich kart w danym stanie brutto (cena*na_stanie)

2.4.2 Karta

Obiekty zawierające informacje o danej karcie, tzn.:

- nazwa (String)
- opis (String) - krótki opis karty: kolor, Set, rzadkość
- ilustracja (String)- grafika karty zakodowana w formacie Base64

2.4.3 Magazyn

Obiekty zawierające dane o stanie magazynu, tzn.:

- ArrayList<Stan> lista_st - lista stanów, z których złożony jest magazyn

2.4.4 Potw_zamowienia

Obiekty zawierające informacje o statusie złożonego zamówienia, tzn.:

- kwota (Float)- kwota brutto w pln.
- kwota_netto(Float)- kwota netto w pln.
- dane_klienta(Dane_Klienta)- informacje o kliencie
- dane_sklepu (Dane_sklepu)- informacje o sprzedawcy.
- czy_zatwierdzono(Boolean) -True, jeśli zamówienie zostało pomyślnie złożone, False- jeśli nie zostało pomyślnie złożone
- message (String)- komentarz dotyczący statusu zamówienia

2.5 Implementacja HATEOAS (Hypermedia as the Engine of Application State)

Do zwracanych obiektów dodane zostały linki ułatwiające nawigowanie po usłudze. Są to dodatkowe elementy w formacie

- rel- String opisujący do czego odnosi się link
- uri- adres url do zasobu

Rysunek 1: Przykładowy link zawarty w zwracanych obiektach

2.6 Specyfikacja wejście-wyjście opercji Web Service

By operacje działały poprawnie użytkownicy muszą trzymać się określonych niżej standardów wyznaczonych przez nasz Restful Webservice.

2.6.1 Nagłówki odpowiedzi

Nagłówki każdej odpowiedzi serwisu składają się z 4 elementów, z których można odczytać m.in. format zawartości odpowiedzi. Postać nagłówka prezentuje się jak poniżej:

Rysunek 2: Przykładowy nagłówek odpowiedzi serwisu

2.6.2 Operacja getMagazyn

Dane wejściowe:

- operacja GET
- ścieżka do operacji:http://25.76.141.122:8080/RestProjectServer/webresources/sklep
- pole login przekazane w nagłówku z loginem do obsługiwanego konta
- pole haslo przekazane w nagłówku z hasłem do obsługiwanego konta

Dane wyjściowe:

- obiekt typu Magazyn zgodny z aktualnym stanem magazynu sklepu
- zwraca wyjątek, jeżeli login i/lub hasło podane w nagłówku są nieprawidłowe

```
1 GET /RestProjectServer/webresources/sklep HTTP/1.1
2 Host: 25.76.141.122:8080
3 login: lukasz
4 haslo: kosmaty
5 Cache-Control: no-cache
6 Postman-Token: 584eca22-2c38-7b04-b509-df965dd1f068
7
```

Rysunek 3: Przykładowy HTTP request operacji zwrocMagazyn

Rysunek 4: Przykładowy response body (JSON) operacji zwrocMagazyn

2.6.3 Operacja getStan

Zwraca wybraną pozycję z magazynu. Dane wejściowe:

- operacja GET
- ścieżka do operacji: http://25.76.141.122:8080/RestProjectServer/webresources/sklep/{nazwa karty}
- nazwa karty (przekazana w ścieżce zapytania)

Dane wyjściowe:

- obiekt typu Stan zgodny z aktualnym stanem magazynu sklepu

```
GET /RestProjectServer/webresources/sklep/stan/Drannith_Stinger HTTP/1.1
Host: 25.76.141.122:8080
Cache-Control: no-cache
Postman-Token: bd34af9a-84f5-9a7c-1cf6-a608d2bbc4ff
```

Rysunek 5: Przykładowy HTTP request operacji zwrocStan

Rysunek 6: Przykładowy response body (JSON) operacji zwrocStan

2.6.4 Operacja updateKoszyk

Operacja nie dodaje pozycji do koszyka, a ustawia w koszyku podaną przez klienta pozycję w podanej liczbie sztuk.
Dane wejściowe:

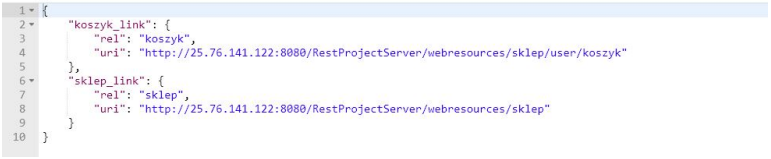
- operacja PUT
- ścieżka do operacji: http://25.76.141.122:8080/RestProjectServer/webresources/sklep/{nazwa karty}?ilosc={liczba}
- nazwa karty (przekazana w ścieżce zapytania)
- liczba (int)- liczba kopii karty, którą chcemy ustawić w koszyku przekazana jako query param w ścieżce do odpowiedniej karty z magazynu

Dane wyjściowe:

- linki do sklepu i koszyka
- zwraca wyjątek, jeżeli nie ma karty o podanej nazwie w magazynie.

```
PUT /RestProjectServer/webresources/sklep/stan/Drannith_Stinger?ilosc=5 HTTP/1.1
Host: 25.76.141.122:8080
Cache-Control: no-cache
Postman-Token: aeb28f31-390f-f9f3-166e-839d58882c28
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
```

Rysunek 7: Przykładowy HTTP request operacji updateKoszyk



Rysunek 8: Przykładowy response body (JSON) operacji updateKoszyk

2.6.5 Operacja getKoszyk

Dane wejściowe:

- operacja GET
- ścieżka do operacji:http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk

Dane wyjściowe:

- obiekt typu Koszyk zgodny z aktualnym stanem koszyka klienta

Rysunek 9: Przykładowy HTTP request operacji getKoszyk

Rysunek 10: Przykładowy response body (JSON) operacji getKoszyk

2.6.6 Operacja zwrocpozycjeZKoszyka

Zwraca wybraną pozycję z koszyka. Dane wejściowe:

- operacja GET
- ścieżka do operacji:
  http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk{nazwa karty}
- nazwa karty (przekazana w ścieżce zapytania)

Dane wyjściowe:

- obiekt typu stan odpowiadający pozycji z koszyka

```
GET /RestProjectServer/webresources/sklep/user/koszyk/Drannith_Stinger HTTP/1.1
Host: 25.76.141.122:8080
Cache-Control: no-cache
Postman-Token: fe285b5d-2036-8d3b-775a-2cbe6f707508
```

Rysunek 11: Przykładowy HTTP request operacji zwrocpozycjeZKoszyka


```
{
    "cena": 0.89,
    "karta": {
        "nazwa": "Drannith_Stinger",
        "opis": "Kolor: czerwony; set: Ikoria, Lair of Behemoths; rzadkosc: common"
    },
    "koszyk_link": {
        "rel": "koszyk",
        "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk"
    },
    "na_stanie": 5,
    "self": {
        "rel": "pozycja w koszyku",
        "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk/Drannith_Stinger"
    },
    "sklep_link": {
        "rel": "sklep",
        "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep"
    },
    "wartosc_razem": 4.45
}
```


Rysunek 12: Przykładowy response body (JSON) operacji zwrocpozycjeZ-Koszyka

2.6.7 Operacja deleteFromKoszyk

Operacja usuwa podaną pozycję z koszyka podanego konta.
Dane wejściowe:

- operacja DELETE
- ścieżka do operacji:
  http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk/{nazwa karty}

- nazwa karty (przekazana w ścieżce zapytania)

Dane wyjściowe:

- zwraca linki do sklepu i koszyka
- zwraca wyjątek, jeżeli w bazie serwera nie ma konta o loginie i haśle podanych w zapytaniu, lub nie ma karty o podanej nazwie w koszyku.

```
DELETE /RestProjectServer/webresources/sklep/user/koszyk/Drannith_Stinger HTTP/1.1
Host: 25.76.141.122:8080
Cache-Control: no-cache
Postman-Token: cbbd8a57-8789-3f7f-ba6f-b98f487bbb8f
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

```

Rysunek 13: Przykładowy HTTP request operacji deleteFromKoszyk


```
{
    "koszyk_link": {
        "rel": "koszyk",
        "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk"
    },
    "sklep_link": {
        "rel": "sklep",
        "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep"
    }
}
```


Rysunek 14: Przykładowy response body (JSON) operacji deleteFromKoszyk

2.6.8 Operacja zlozZamowienie

Operacja sprawdza poprawność koszyka (spójność z magazynem), stan konta użytkownika i zwraca potwierdzenie transakcji oraz aktualizuje stan magazynu i opróżnia koszyk.
Dane wejściowe:

- operacja POST
- ścieżka do operacji:
  http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk

Dane wyjściowe:

- obiekt typu Potw_zamowienia z wszelkimi informacjami dotyczącymi transakcji
- jeżeli zamówienie nie zostało potwierdzone, wtedy w potwierdzeniu zwracane jest czy_zatwierdzono = False i komentarz w polu message.

```
POST /RestProjectServer/webresources/sklep/user/koszyk HTTP/1.1
Host: 25.76.141.122:8080
Cache-Control: no-cache
Postman-Token: 883a9702-43ee-f59a-90d4-f80721263764
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
```

Rysunek 15: Przykładowy HTTP request operacji zlozZamowienie


```
{
    "czy_zatwierdzono": true,
    "dane_klienta": {
        "adres": "ul. szkolna 17",
        "email": "ookosmaty@gmail.com",
        "haslo": "Kosmaty",
        "imie": "Lukasz",
        "kod_pocztowy": "15-888",
        "login": "Lukasz",
        "miasto": "Bialystok",
        "nazwisko": "Kosmaty",
        "numer_telefonu": "666662222",
        "stan_konta": 999191.8
    },
    "dane_sklepu": {
        "NIP": "68830299831",
        "adres": "Boboli 10",
        "email": "Magiaaa@gmail.com",
        "kod_pocztowy": "15-431",
        "miasto": "Bialystok",
        "nazwa": "Znowu trzeba w magica grac.Magiaaa!",
        "wlasciciel": "Pawel Roszkowski"
    },
    "koszyk": [
        {
            "cena": 0.89,
            "karta": {
                "nazwa": "Drannith_Stinger",
                "opis": "Kolor: czerwony; set: Ikoria, Lair of Behemoths; rzadkosc: common"
            },
            "koszyk_link": {
                "rel": "koszyk",
                "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk"
            },
            "na_stanie": 5,
            "self": {
                "rel": "pozycja w koszyku",
                "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep/user/koszyk/Drannith_Stinger"
            },
            "sklep_link": {
                "rel": "sklep",
                "uri": "http://25.76.141.122:8080/RestProjectServer/webresources/sklep"
            },
            "wartosc_razem": 4.45
        }
    ],
    "kwota": 4.45,
    "kwota_netto": 3.62,
    "message": "zamowienie potwierdzone"
}
```


Rysunek 16: Przykładowy response body (JSON) operacji zlozZamowienie