PatientData-dokumentacja.md

Dokumentacja mikrousługi danych PatientData

Autorzy:

Maciej Włodarczyk

Marcin Dadura

1. Cel

Mikrousługa PatientData ma za zadanie dostarczyć interfejs REST, który może służyć do obsługi prostych zapytań dotyczących danych zgromadzonych w tej usłudze. Interfejs powinien być przejrzysty oraz nie ujawniać szczegółów implementacji.

2. Opis danych

Mikrousługa służy do obsługi danych dotyczących pacjentów. Dane przetrzymywane są w najlepszym do tego formacie XML.

Każda instancja pacjenta zawiera pola:

- `Id` - unikalny identyfikator pacjenta,
- `Name` - jest to imię i nazwisko pacjenta,
- `Sex` - płeć pacjenta, możliwa `male` lub `female`
- `Pesel` - PESEL pacjenta,
- `Conditions` - lista obiektów typu `Condition` przechowujących informacje o pojedyńczym schorzeniu.

Każda instancja schorzenia - `Condition` przechowuje:

- `Type` - typ schorzenia,
- `DiagnosisDate` - data zdiagnozowania choroby u pacjenta.

2.1 Plik .xsd - formalny model danych

Plik definiujący bazę danych to `schema.xsd`. Zawierają się w nim wsyztskie informacje opisujące bazy danych, jej struktura oraz możliwe wartości przechowywane w pliku `database.xml`. Plik można odnaleźć w katalogu `PatientsData/PatientsData.Web/Utility/Resources/schema.xsd`.

2.2 Plik bazy danych - przykładowe dane

Cały plik można odnaleźć w katalogu `PatientsData/PatientsData.Web/Utility/Resources/database.xml`. Stworzyliśmy również generator pliku `.xml`, który pozwala nam wygenerować dowolną ilość pacjentów w bazie danych.

Przykładowy zestaw danych jednego pacjenta w bazie danych w formacie pliku XML:

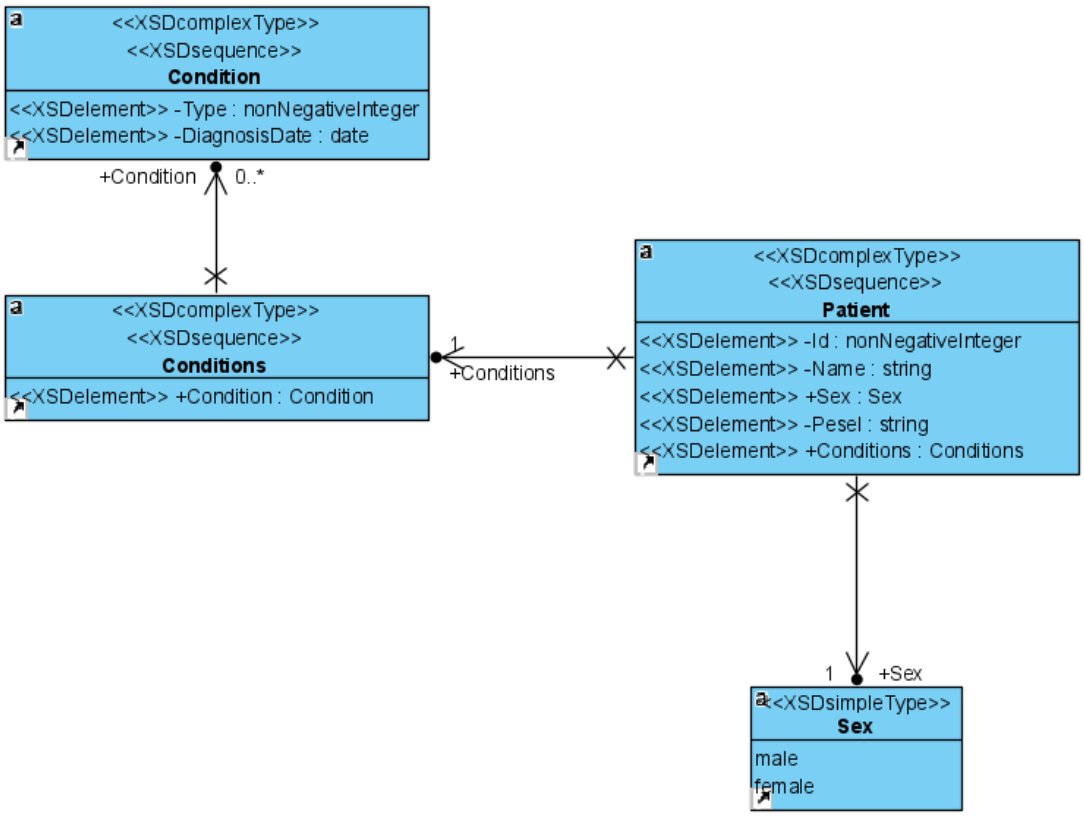
```
<?xml version="1.0" encoding="utf-8"?>
<PatientsList  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <Patients>
        <Patient>
                <Id>1</Id>
                <Name>Honorata Bosek</Name>
                <Sex>female</Sex>
                <Pesel>86392074438</Pesel>
                <Conditions>
                        <Condition>
                                <Type>5</Type>
                                <DiagnosisDate>2005-02-09T00:00:00</DiagnosisDate>
                        </Condition>
                </Conditions>
        </Patient>
```

```
        </Patients>
</PatientsList >
```

2.3 UML

Poniżej znajduje się diagram UML bazy danych. Można na nim zaobserwować następujące elementy:

- `Patient` - obiekt pacjent zawierający w sobie odpowiednie atrybuty, dokładniej opisane w punkcie 1.
  - `Sex` - enum, element obiektu Patient, przyjmuje wartość `male` lub `female`
  - `Conditions` - lista schorzeń, element obiektu Patient
    - `Condition` - schorzenie, element obiektu Conditions



3. Struktura projektu

```
PatientsData
├── PatientsData.Domain
│   ├── Models
│   │   ├── Condition.cs - model do przetrzymywania schorzeń
│   │   ├── Patient.cs - model do przetrzymywania danych pacjenta
│   │   ├── PatientsList.cs - model pomocniczy do czytania bazy danych w xml
│   │   └── Sex.cs - model do przetrzymywania danych o płci
│   └── PatientsData.Domain.csproj - plik opisujący projekt
├── PatientsData.Infractructure
│   ├── PatientsData.Infractructure.csproj - plik opisujący projekt
│   └── Repositories
│       ├── IPatientRepository.cs - interfejs do repozytorium pacjentów
│       └── PatientRepository.cs - repozytorium pacjentów odpowiedzialne za wszystkie operacje na bazie danych w XML.
├── PatientsData.Web
│   ├── Applictaion
│   │   ├── Dtos
│   │   │   ├── ConditionDto.cs - model do wysyłania danych o schorzeniach
│   │   │   └── PatientDto.cs - model do wysyłania danych o pacjentach
│   │   ├── Mapper
│   │   │   └── Mapper.cs - klasa zawierająca statyczne metody do mappowania dto na model
│   │   └── Queries
│   │       ├── IPatientQueriesHandler.cs - obsługa zapytań dotyczących pacjentów - przekazuje zapytanie do repozytorium
│   │       └── PatientQueriesHandler.cs - interfejs, który implementuje obsługę zapytań dla pacjentów
```

```
│   ├── Controllers
│   │   └── PatientController.cs - kontroler odpowiedzialny za obsługę end-pointów - wywołuje metody z PatientQueriesHandler
│   ├── PatientsData.Web.csproj - plik opisujący projekt
│   ├── Program.cs - znajduje się tutaj funkcja wejściowa do programu
│   ├── Properties
│   │   └── launchSettings.json - plik JSON z ustawieniami ładowany przy uruchomieniu aplikacji
│   ├── Startup.cs - klasa zajmująca cię wstepną konfiguracją mikrousługi przy uruchomieniu
│   ├── Resources - katalog z plikiem bazy danych oraz schema bazy
│   │   ├── database.xml
│   │   └── schema.xsd
│   ├── Utility
│   │   └── GenEntries - w katalogu znajduje się generator przykładowych danych w pliku database.xml
│   │       ├── generateEntires.py
│   │       ├── imiona.txt
│   │       └── nazwiska.txt
│   ├── appsettings.Development.json - ustawienia aplikacji dla stanu Development
│   └── appsettings.json - ustawienia aplikacji dla stanu Production
└── PatientsData.sln - plik opisujący rozwiązanie
```

Zapytania obsługiwane są w następującym schemacie. Zapytanie przechwytuje `controller` i przekazuje je do `QueryHandlera`. Następnie zapytanie realizuje `PatientRepository` i zwraca wyniki w odwrotnej kolejności. `PatientRepository` działa bezpośrednio na danych. `QueryHandler` mapuje je na obiekty typu `Data Transfer Object` i zwraca je do `Controllera`, który wystawia dane na wyjście.

4. API

Usługa wystawia interfejs z którego można korzystać za pomocą HTTP REST.

Usługa wystawia następujące metody na punktach końcowych:

- metody `GET`:
  - `/patients` - zwraca listę wszyskich pacjentów dostępnych w repozytorium danych, nie przyjmuje argumentów
  - `/select-condition` - metoda zwracająca listę pacjentów z danymi schorzeniami, przyjmuje liczbę całkowitą, jako typ schorzenia
  - `/patient-id` - metoda zwracająca pacjenta o danym id, przyjmuje liczbę całkowitą jako ID pacjenta
  - `/patient-pesel` - metoda zwracająca pacjenta o danym peselu, przyjmuje stringa jako Pesel
- metoda `POST`:
  - `/add-patient` - wysyłając obiekt json metoda ta pozwala na dodanie pacjenta

Przykładowy obiekt `Patient` w pliku Json. W metodach `GET` dostajemy listę tych obiektów lub jeden obiekt, tak jak poniżej. Aby dodać pacjenta metodą `POST` musimy wysłać taki obiekt jak poniżej:

```
{
    "id": 1,
    "name": "Modest Tomaszek",
    "sex": "male",
    "pesel": "25206423085",
    "conditions": [
        {
            "type": 9,
            "diagnosisDate": "28.04.2011 00:00:00"
        }
    ]
}
```

5. Testy

Do mikrousługi zostały dostarczone również testy w postaci skryptu `curl`. Aplikacja przechodzi testy i spełnia swoją funkcjonalność. Testy można odnaleźć pod ścieżką `Projekt/Testy/PatientsData`.