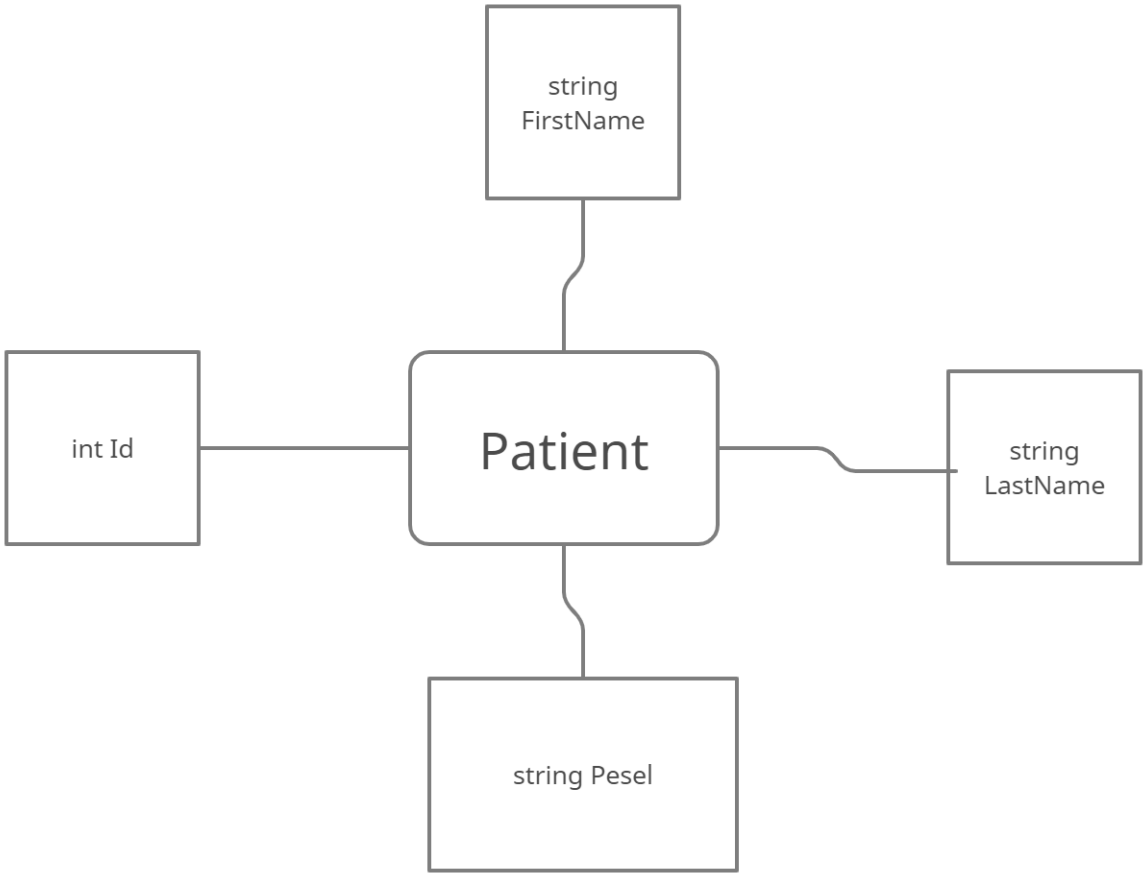
1. Diagram encji



2. Formalny model danych

Pacjent ma następujące dane:

- Id – identyfikator, typ int
- Pesel – numer pesel, typ string
- LastName – nazwisko, typ string
- FirstName – imię, typ string

3. Przykładowe dane

```
<net:Patients>
    <net:Patient id="0" pesel="00000000000" lastName="Kowalski" firstName="Jan" />
    <net:Patient id="1" pesel="00000000001" lastName="Nowak" firstName="Andrzej" />
    <net:Patient id="2" pesel="00000000002" lastName="Kot" firstName="Tytus" />
    <net:Patient id="3" pesel="00000000003" lastName="Stoch" firstName="Piotr" />
    <net:Patient id="4" pesel="00000000004" lastName="Steinbach" firstName="Paweł" />
    <net:Patient id="5" pesel="00000000005" lastName="Makowski" firstName="Artur" />
</net:Patients>
```

4. Struktura rozwiązania

1. Projekt - Patient.Logic:

- *PatientsFromXml* – klasa odpowiadająca za logikę mikrousługi, korzysta z czytacza (Klasa *Reader*) do odczytania danych
- *Reader* – klasa odpowiadająca za czytanie danych z pliku xml

2. Projekt - Patients.Model
   - *IPatients* – interfejs zawierający wzór metod dla klasy Patients
   - *Patient* – klasa definiująca model lekarza
3. Projekt - Patients.Rest
   - *Controller* – klasa tworząca kontroler serwisu Rest
4. Projekt - Patients.Rest.Model
   - *DataConverter* – klasa konwertująca dane odczytane z pliku xml do danych, które będą przesyłane w sieci
   - *PatientData* – klasa definiująca model danych (Patient) przesyłanych w sieci
   - *IPatientService* - interfejs zawierający wzór metod dla kontrolera Rest

5. Interfejs

1. Operacje usługowe:

1. Pobranie wszystkich pacjentów z bazy danych - GetAllPatients():
   - Dane wejściowe: brak
   - Dane wyjściowe: tablica z danymi wszystkich pacjentów

2. Pobranie wybranego pacjenta z bazy danych - GetPatient(int id):
   - Dane wejściowe: id pacjenta
   - Dane wyjściowe: tablica z danymi szukanego pacjenta o podanym id