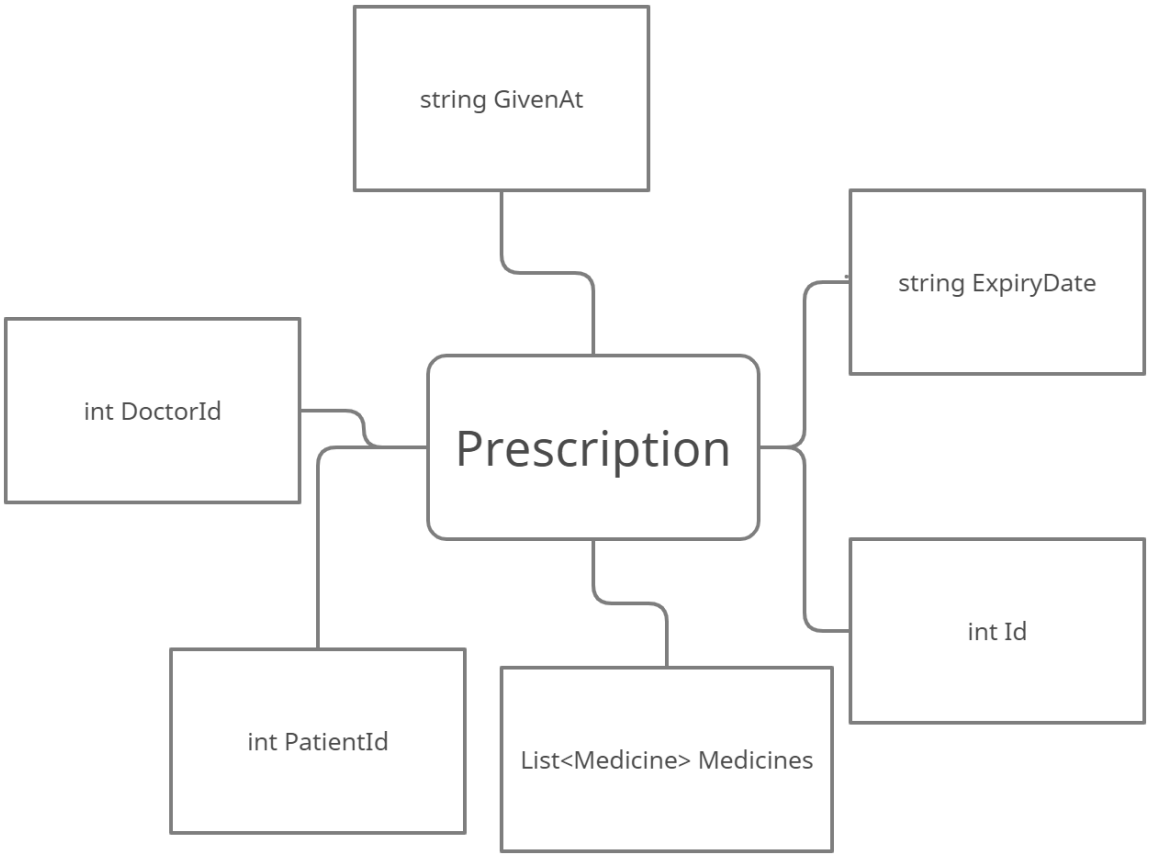
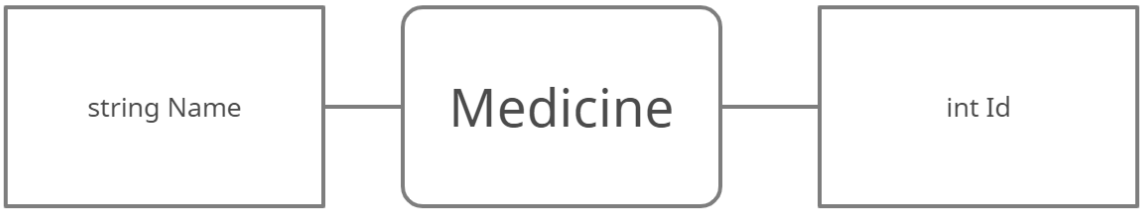
Mikrousługa danych – Recepty

1. Diagram encji





2. Formalny model danych

Wizyta ma następujące dane:

- Id – identyfikator, typ int
- DoctorId – identyfikator doktora, typ int
- PatientId – identyfikator pacjenta, typ int
- GivenAt – data nadania recepty, typ string
- ExpiryDate– data ważności recepty, typ string

3. Przykładowe dane

```
<net:Prescription id="0" patientId="1" doctorId="1"  givenAt="04.01.2021" expiryDate="04.03.2021" >
    <net:Medicines>
        <net:Medicine id="0" name="Aspirin"/>
        <net:Medicine id="1" name="Strepsils"/>
        <net:Medicine id="2" name="Ketonal"/>
    </net:Medicines>
</net:Prescription>
<net:Prescription id="1" patientId="0" doctorId="1"  givenAt="04.01.2021" expiryDate="04.03.2021" >
    <net:Medicines>
        <net:Medicine id="0" name="Aspirin"/>
        <net:Medicine id="2" name="Ketonal"/>
        <net:Medicine id="3" name="Apap"/>
    </net:Medicines>
</net:Prescription>
<net:Prescription id="2" patientId="0" doctorId="2"  givenAt="04.01.2021" expiryDate="04.03.2021" >
    <net:Medicines>
        <net:Medicine id="0" name="Aspirin"/>
        <net:Medicine id="2" name="Ketonal"/>
        <net:Medicine id="3" name="Apap"/>
    </net:Medicines>
</net:Prescription>
<net:Prescription id="3" patientId="2" doctorId="0"  givenAt="04.01.2021" expiryDate="04.03.2021" >
    <net:Medicines>
        <net:Medicine id="0" name="Aspirin"/>
        <net:Medicine id="2" name="Ketonal"/>
        <net:Medicine id="3" name="Apap"/>
    </net:Medicines>
</net:Prescription>
```

4. Struktura rozwiązania

1. Projekt - Prescriptions.Logic:
    - *PrecscriptionsFromXml* – klasa odpowiadająca za logikę mikrousługi, korzysta z czytacza i klasy zapisującej (Klasy *Reader i Writer*) do odczytania danych
    - *Reader* – klasa odpowiadająca za czytanie danych z pliku xml
    - *Writer* – klasa odpowiadająca za zapisywanie danych do pliku xml
2. Projekt - Prescriptions.Model
    - *IPrescriptions* – interfejs zawierający wzór metod dla klasy Prescriptions
    - *Prescription* – klasa definiująca model recept

- *Medicine* – klasa definiująca model leków

3. Projekt - Prescriptions.Rest
   - *Controller* – klasa tworząca kontroler serwisu Rest
4. Projekt - Prescriptions.Rest.Model
   - *DataConverter* – klasa konwertująca dane odczytane z pliku xml do danych, które będą przesyłane w sieci
   - *PrescriptionData* – klasa definiująca model danych (Prescription) przesyłanych w sieci
   - *IPrescriptionsService* - interfejs zawierający wzór metod dla kontrolera Rest
   - *MedicineData* - klasa definiująca model danych (Prescription) przesyłanych w sieci

5. Interfejs

1. Operacje usługowe:

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

2. Dodanie wizyty do bazy danych – AddPrescription(PrescriptionData PrescriptionData):
   - Dane wejściowe: Id, DoctorId, PatientId, Problem, Date, Room
   - Dane wyjściowe: brak
3. Pobranie recepty o zadanym id – GetPrescription(int id):
   - Dane wejściowe: Id
   - Dane wyjściowe: żądana recepta