Opracowanie i implementacja podstaw optymalnego przechowywania i transferowania danych medycznych (ontologie)

Patryk Bąk, Marek Kendziorek, Wojciech Inglot

AGH Kraków, 2013

1 Abstrakt

Systemy medyczne należą do jednych z najbardziej rozbudowanych systemów w branży IT. Cechują się one nie tylko ogromnymi ilosciami danych, które przetwarzają, ale również bardzo dużą złożonoscią budowych tych danych oraz relacji, które między nimi zachodzą. Własnie branża e-health jako jedna z pierwszych zainteresowała się zastosowaniem ontologii oraz ich reprezentacji w zagadnieniu reprezentowania danych. [7] Niniejszy artykuł porusza problem zastosowania optymalnych technologii i metodologii do implementacji systemu opartego o ontologię z naciskiem na przechowywanie oraz transfer ontologii.

Zaproponowane przez nas rozwiązanie problemu oparte zostało na języku programowania java oraz opartych na nim technologiach i narzędziach. Do transferu danych wykorzystano FUSEKI, które oferuje transfer danych protokołem REST z wykorzystaniem języka zapytań SPARQL. Dodatkowo przy definiowanu ontologii zaproponowano wykorzystanie opartego i interfejsy i stałe słownika danych.

Zaproponowana przez nas implementacja reprezentacji oraz transferu danych medycznych posiada wiele cech pożądanych przy projektowaniu tego typu systemów. Rozwiązanie to cechuje się bardzo dużą skalowalnoscią zarówno od strony wielkosci samego systemu jak i możliwosci rozbudowy przechowywanych w nim danych. [6] Zastosowanie architektury typu SOA przy budowie systemu medycznego pozwala na łatwą, modułową rozbudowę takiego systemu oraz ułatwioną integrację z innymi systemami medycznymi, zupełnie niezależnymi od siebie.

2 Wstęp

Dane medyczne są podstawowym zbiorem informacji odności pacjenta, historii jego choroby oraz planowanego leczenia. W 40 milinonowym kraju, takim jakim jest Polska, liczba tych danych rośnie w zatrważającym tempie. Coraz to nowe firmy prześcigają się w podawaniu gotowych rozwiązań dotyczących przetwarzania i przechowywania tychże danych. Problem ten dotyka nie tylko naszego kraju ale niemal wszystkich wysokorozwiniętych nacji świata.

Jest to problem złożony. Możemy podzielić go na kilka części:

- gromadzenie danych
- aktualizowanie zgromadzonych danych
- przekazywanie zgromadzonych i aktualizowanych danych
- ochrona danych
- wprowadzanie nowych podmiotów mających dostęp do zgromadzonych danych
- utrata i odzyskiwanie danych
- a wreszcie utylizacja danych

Poszególne problemy pokrótce przeanalizujemy na podstawie polskiego systemu służby zdrowia co pozwoli nam łatwiej zrozumieć działalność systemu HL7, który obecnie jest jednym z popularniejszych na świecie. [20] Przypomnijmy iż w Polsce używa się obecnie systemu eWUŚ (Elektroniczna Weryfikacja Uprawnień Świadczeniobiorców) oraz zintegrowanego z nim systemu ZIP (Zintegrowany Informator Pacjenta).

Problem z gromadzeniem danych można uznać za problem najważniejszy gdyż istotną w nim rolę odgrywa czynnik ludzki. Jedna pomyłka podczas wprowadzania informacji rzutuje na działalności całego systemu. [25] Przykład: Błędnie wprowadzone nazwisko podczas rejestracji pacjenta pociągnie za sobą konsekwencje w postaci nie odnalezienia go wśród pacjentów ubezpieczonych oraz automatyczne generowanie faktury za udzielone świadczenia.

Aktualizowanie danych sugeruje nam, a wręcz niejako wymusza aby nasza baza danych była skalowalna. [21] A co za tym idzie musi posiadać określoną pojemność, która będzie rozszerzalna. Dlatego przechowywanie takich danych musi zostać rozproszone w celach zarówno wydajności działania jaki i bezpieczeństwa. Skoro jesteśmy już przy bezpieczeństwie, warto wspomnieć o ochronie danych. Wspomniane przez nas systemy są chronione w sposób prosty. Dostęp do nich mają określone podmioty zarejestrowane w konkretnej bazie, posługujące się przypisanymi tylko im loginami i hasłami. [25] Przykład: Weryfikacja uprawnień świadczeniobiorcy odbywa się poprzez wpisanie jego numeru ewidencyjnego PESEL do wyszukiwarki bazy danych. Osoba, która ten numer wpisuje musi być uprzednio zalogowna do systemu a przed każdą próbą skorzystania z niego, potwierdzać swoją tożsamość.

W ten oto sposób pokrótce opisaliśmy wybrane problemy dotyczące przechowywania i przesyłania informacji. Ninejszym wskazaliśmy, w którą stronę podążają wspomniane problemy. W kolejnych rozdziałach wspróbujemy zastanowić się wspólnie nad uszeregowaniem zarówno kłopotów jak i przypisywanych im rozwiązań, które pojawiają się w temacie systemów przechowywania i przesyłania danych medycznych.

3 Wykorzystane technologie

3.1 Ontologia

Ontologia jest formalną reprezentacją pewnej dziedziny wiedzy. Składają się na nią zapis zbiorów pojęć i relacji między nimi. Zapis ten tworzy schemat pojęciowy, który jest opisem danej dziedziny wiedzy. Może służyć jednocześnie jako podstawa do wnioskowania o właściwości opisywanych ontologią pojęć. [7]

Pod pojęciem ontologii mogą się kryć różne struktury wiedzy. Ich przeznaczenie czy zakres stosowania może być wieloraki.

Ontologie możemy podzielić ze względu na stopień ich formalizacji:

- nieformalne
  - predefiniowane słownictow
  - słowniki
  - tezarusy
  - taksonomie
- formalne
  - ontologie oparte na danych
  - ontologie oparte na logice

Podział następuje także ze względu na zakres stosowania:

- ontologie wysokiego poziomu (*upper ontologies*)
- ontologie dziedzinowe (*domain ontologies*)
- ontologie aplikacyjne

Istnieje szereg języków pozwalających na zapis lub wspierających ontologie:

- OWL (*Ontology Web Language*)
- RDF (*Resource Description Framework*)
- RDF Vocabulary Description Language (RDF Schema, RDFS)
- OCML (Operational Conceptual Modeling Language)
- XML (Extensible Markup Language)

3.2 OWL

OWL (Ontology Web Language) jest opartym na składni XML językiem służącym do opisu ontologii. Stanowi on rozszerzenie języka RDF (Resource Definition Language). Istnieją 3 odmiany OWL: OWL Lite, OWL DL oraz OWL Full. W 2004 roku język ten został uznany za standard przez organizację W3C. [8]

OWL został stworzony do przetwarzania informacji o obiektach oraz relacji między nimi, nie do przechowywania tych informacji w formie czytelnej dla człowieka. OWL jest zbudowany na bazie RDF, tak więc języki te są ze sobą w pełni kompatybilne. OWL posiada jednak znacznie mocniejszą składnię.

Do elementów języka OWL należą takie atrybuty jak:

- Class - definiuje grupę indywidualnosci, które posiadają pewne wspólne cechy. Klasy można organizować w hierarchie za pomocą atrybutu subClassOf.
- rdf:Property - okresla pewne relacje między indywidualnosciami np. samochod może posiadać drzwi, osoba moze posiadać dziecko albo rodzica.
- rdfs:domain - ogranicza indywidualności, do których można zastosować relację (property). Jeżeli relacja łączy jedną indywidualność z drugą, i jednocześnie posiada ona domenę na określoną klasę, to obie indywidualności muszą należeć do tej klasy.

– rdfs:range - ogranicza nie tylko indywidualności, ale także wartość relacji (property)
– Individual - indywidualności są instancjami klas, które są połączone pewnymi relacjami (property).

Wszystkie powyższe atrybuty są częścią specyfikacji OWL Lite.

3.3 Protege

Protege jest otwarto-źródłowym narzędziem stworzonym przez Stanford Medical Informatics [26]. Jak przy większości narzędzi modelujących, architektura Protege jest przejrzyście podzielona na dwie części - model i widok. Modele są wewnętrzną reprezentacją mechanizmy ontologii i baz wiedzy. Widoki zapewniają interfejs użytkownika, dzięki któremu można wyświetlać i manipulować modelami.

Protege posiada możliwość wczytania, edycji i zapisywania w wielu popularnych dla ontologii formatach, np:

– RDF
– OWL
– XML
– UML
– Bazy relacyjne

3.4 SPARQL

SPARQL jest językiem zapytań RDF, czyli pozwala na wyciaganie lub manipulowanie danymi w bazie danych składującej dane w formacie RDF (eng. Resource Description Framework). SPARQL pozwala również na pracę na danych w formacie pochodnym od formatu RDF, np. OWL, jednak z pewnymi ograniczeniami. SPARQL operując na danych w formacie OWL musi pominąć elementy języka które nie mają przełożenia na format RDF.

Konstrukcja zapytań opiera się o zestaw trzech parametrów:

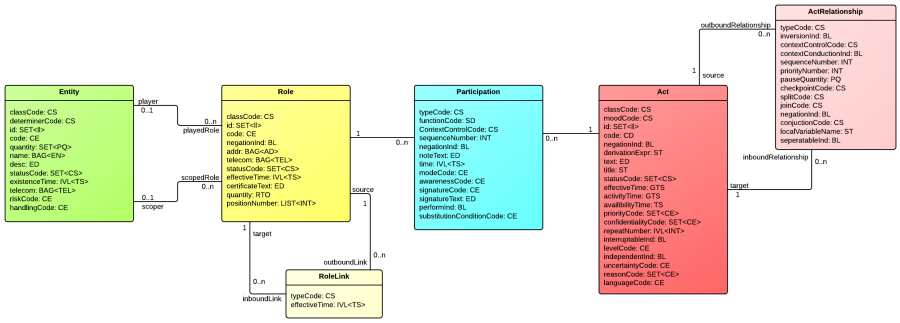
– spójnika (eng. conjuction)
– alternatywy (eng. disjunction)
– wzorzec (opcjonalnie) (eng. pattern)

Idea jaka przyświecała twórcom SPARQL określała go jako przełom w tworzeniu internetu czytelnego dla maszyn. Jeżeli format ten przyjmie się jako standard otrzymamy uniwersalny sposób otrzymywania informacji z internetowych „skupisk"danych, co da nowe możliwości łączenia danych z różnych źródeł.

3.5 REST

REST jest protokołem komunikacji między aplikacjami, który tak naprawdę jest tylko pewną dodatkową warstwą nałożoną na protokół HTTP. Budowanie zapytań restowych z poziomu języków programowania jest tak samo proste, jak zrozumienie działania protokołu HTTP. Istnieje wiele bibliotek i prostych frameworków do budowania serwisów, odbierających i wysyłających zapytania HTTP. Należą do nich np Jersey-RS, czy zbudowana na jego bazie trochę większa biblioteka Apache CXF. Biblioteka Fuserki, będąca częścią specyfikacji Apache Jena oferuje możliwość transferu danych właśnie protokołem REST, przy użyciu języka zapytań SPAQRL. [1]

Ze względu na swoją prostotę i oparcie o standard HTTP, wybór tego protokołu jako sposobu komunikacji między aplikacjami jest bardzo dobrą decyzją. Umożliwia to znakomitą skalowalność naszego systemu, poprzez rozdzielenie go na szereg indywidualnych



Rysunek 1. HL7 RIM

aplikacji, które nie muszą być napisane przy użyciu tej samej technologii, muszą jedynie współdzielić tą metodę komunikacji oraz ewentualnie sposób serializacji danych.

3.6 Fuseki

Fuseki jest to serwer HTTP w standardzie REST przyjmujący zapytania HTTP zawierające w parametrach zapytania SPARQL. Dzięki niemu możemu w ustandaryzowany sposób pobierać lub manipulowac danymi w modelu OWL lub RDF.

Fuseki jest jednym z subsystemów systemu Jena. Twórcy zadbali o to żeby serwer SPARQL udało się włączyć nie tylko poprzez konsole, ale również bezpośrednio z aplikacji JAVA z wykorzystaniem biblioteki Jena.

3.7 HL7

HL7 (Health Level 7) jest organizacją ustanawiającą standardy, akredytowaną przez American National Standards Insitute (ANSI). Opracowali protokół komunikacyjny szeroko używany w USA, obecnie coraz częściej rozpoznawany i wykorzystywany w pozostałych częściach świata. [14]

RIM (Reference Informational Model) został wykorzystany w trzeciej wersji HL7 (HL7 Version 3) w celu standaryzacji powiązań pomiędzy danymi przenoszonymi przez wiadomości protokołu HL7. [16]

Diagram UML na rysunku 1 przedstawia podstawowe klasy istniejące w RIM. [15]

RIM opiera się na pięciu podstawowych założeniach:

- Każde zdarzenie jest opisane przez klasę Act
  - zdarzeniem może być: procedura, obserwacja, dostawa zapasów, rejestracja, itp.
- Relacje pomiędzy zdarzeniami opisuje klasa ActRelationship
- Klasa Participation definiuje kontekst wystąpienia zdarzenia
  - autor, wykonujący, podmiot, lokalizacja itp.
- uczestnicy zdarzenia są opisanie przez klasę Role
  - pacjent, dostawca, praktykant, pracownik itp.
- role są reprezentowane przez klasę Entity
  - osoby, organizacja, materiały, miejsca, urządzenia, itp.

4 Implementacja platformy

Rozwiązanie zaprezentowane w niniejszym artykule zostało napisane w języku Java, z wykorzystaniem Ontology API biblioteki Apache Jena. Biblioteka ta pozwala na mapowanie różnego rodzaju reprezentacji ontologii na klasy Java. Do transferowania naszych danych wykorzystamy bibliotekę Apache CXF, a dokładniej jej moduł REST, który pozwoli nam na przesyłanie zserializowanych danych przez protokół HTTP. Schema klienta został przedstawiony na Rys. 2

4.1 Reprezentacja danych

Ontology API biblioteki Apache Jena pozwala między innymi na budowanie oraz importowanie modeli ontologii. Stworzenie nowego modelu ontologii oraz zaimportowanie do niego już istniejącej ontologii zapisanej w formacie OWL przedstawiono na rysunku. Przykładowy graf wygenerowany z części ontologii widoczny jest na rysunkach 3 oraz 4

4.2 Transfer danych

W projekcie uruchomiony został serwer HTTP Fuseki. Pozwala on na przechowywanie i udostepnienie danych wczytanych z plikow OWL poprzez zapytania HTTP. W jego wewnętrznej bazie danych budowane są grafu do których aplikacje klienckie odwołują się w zapytania SPARQL.

Instalacja Instalacja Jeny i Fuseki ogranicza się do dodania do zmiennych środowiskowych ścieżek do paczek.

```
export JENA=
    /Users/XXXX/Fuseki/apache
    -jena-2.11.0/
export FUSEKI=
    /Users/XXXX/Fuseki/jena
    -fuseki-1.0.0/
```

Uruchomienie serwera Standardowo serwer FUSEKI uruchamiany jest pod wirtualną domeną „localhost"i na porcie 3030.

Uruchomienie serwera FUSEKI:

```
./fuseki-server --update
    --loc=/Users/~/MyTDB/ /ds &
```

Załadowanie pliku z ontologią Załadowania pliku OWL (poprzez konsolową komendę):

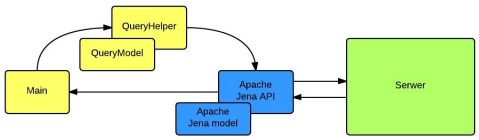
```
./s-put
    http://localhost:3030/ds/data
    default
    /Users/XXXX/RIMV3OWL.owl
```

Tworzenie zapytania SPARQL Wysłanie zapytania SPARQL wypisującego „koncepty"z załadowanego grafu:

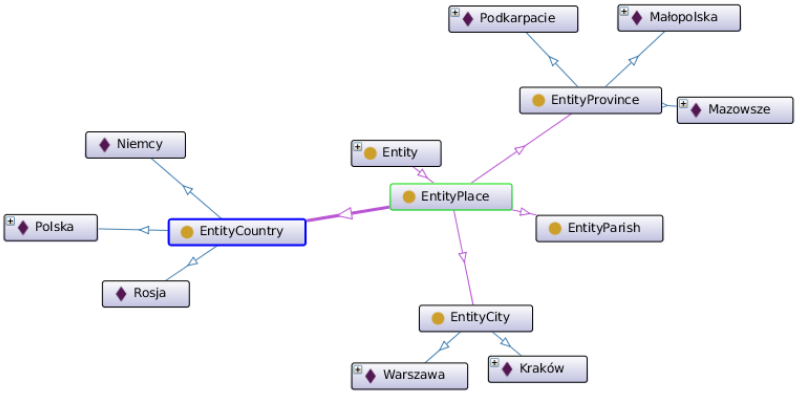
1. Tworzenie zapytania.

   ```
   select distinct ?Concept
       where {[] a ?Concept}
   ```

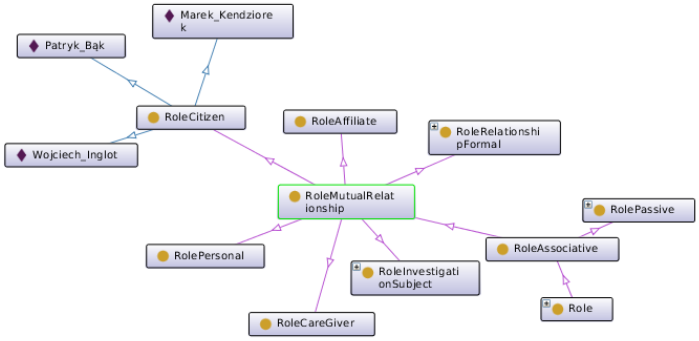
2. Tworzenie zapytania HTTP POST, gdzie:
   - jako URI podajemy sufix prowadzący do punktu końcowego (eng. endpoint) i doklajemy końcówkę /query
   - w parametrze POST/GET „query"podajemy stworzone zapytanie
   - opcjonalne parametry: output (np. „json"), default-graph-uri
3. W efekcie otrzymujemy zapytanie (dla punktu końcowego „ds":



Rysunek 2. Schemat klienta JAVA



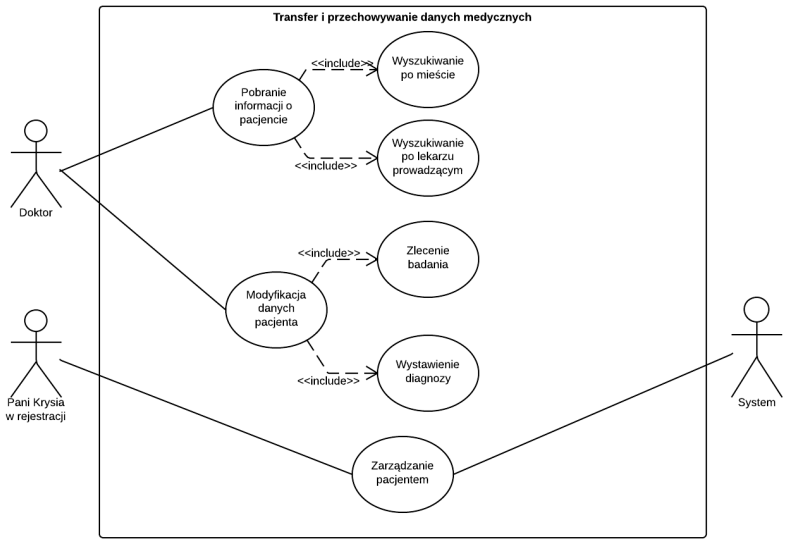
Rysunek 3. Przykład Ontologii Opartej o HL7 (EntityPlace)



Rysunek 4. Przykład Ontologii Opartej o HL7 (Role)

5 Przypadek użycia

Zaprojektowany system będzie wykorzystywany w sposób pokazany na Rys. 5. Celem implementacji jest stworzenie działającego przypadku użycia „Pobranie informacji o pacjencie", który będzie wykorzystywany przez więcej niż jednego „aktora (Doktor)", a dane będą składowane w centralnej bazie.



Rysunek 5. Diagram przypadków użycia

5.1 Pobranie informacji o pacjencie

Opis Przypadek użycia „Pobranie informacji o pacjencie"(Rys. 5) przedstawia jedną z możliwych interakcji klienta (w tym wypadku lekarza) z systemem. Klient ma możliwość przefiltrowania pacjentów wg. miast i lekarzy prowadzących (lekarz może mieć pacjentów w róznych miastach).

Zapytania SPARQL Przykładowe zapytanie użyte do pobrania pacjentów z danego miasta:

```
SELECT ?patient WHERE {
    ?patient hl7:livesIn
    hl7:Nazwa_miasta }
```

Implementacja po strone klienta - JAVA Aplikacja kliencka umożliwia pobranie listy pacjentów, parametryzowane wybranym miastem (poprzez pole select). W analogiczny sposób można dodawać nowe funkcjonalności - dodając nowe elementy interfejsu i przypisując odpowiednie zapytania SPARQL.

6 SPARQL - zastosowanie

Poniżej zamieszczone są przykładowe zapytania sparql do pobierania informacji z grafów przedstawionych na rysunku 1 i rysunku 2.

Jako, że platforma opiera się o scentralizowanej bazie danych udostępnionej przez protokół HTTP - aplikacje klienckie aby komunikować się z bazą muszę wykorzystywać zapytania SPARQL. Dlatego poniżej zamieszczamy przykłady możliwych zapytań. Przez to chcemy pokazać jak łatwa jest rozbudowa istniejącej aplikacji klienckiej, lub dorobienie nowych klientów (wykorzystujących inne przypadki użycia).

Wyszukiwanie węzłów-rodzeństw (poprzez użycie „relacji sameAs”
Zapytanie:

```
PREFIX rdf:
    <http://www.w3.org/
    1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:
    <http://www.w3.org/
    2000/01/rdf-schema#>
PREFIX owl:
    <http://www.w3.org/
    2002/07/owl#>
PREFIX HL7:
    <http://sim.ontology/HL7#>
SELECT ?subject ?object
        WHERE { ?subject
            owl:sameAs ?object
            }
```

Wynik:

```
{
  "head": {
    "vars": [ "subject" ,
        "object" ]
  } ,
  "results": {
    "bindings": [
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Krakow" } ,
        "object": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Warszawa" }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Marek_Kendziorek"
            } ,
        "object": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Wojciech_Inglot"
            }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Patryk_Bak" } ,
        "object": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Wojciech_Inglot"
            }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Mazowsze" } ,
        "object": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Malopolska" }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Mazowsze" } ,
        "object": { "type":
            "uri" , "value":
```

```
            "http://sim.ontology/
            HL7#Podkarpacie" }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Malopolska" } ,
        "object": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Podkarpacie" }
      }
    ]
  }
}
```

Wyszukiwanie miast (węzły typu hl7:EntityCity) podrzędnych do węzła „Polska", relacja: „isPartOf" Zapytanie:

```
PREFIX rdf:
    <http://www.w3.org/
    1999/02/22-rdf-syntax-ns#>
PREFIX owl:
    <http://www.w3.org/
    2002/07/owl#>
PREFIX xsd:
    <http://www.w3.org/
    2001/XMLSchema#>
PREFIX rdfs:
    <http://www.w3.org/
    2000/01/rdf-schema#>
PREFIX hl7:
    <http://sim.ontology/HL7#>
SELECT ?subject
        WHERE {
                ?subject
                    rdf:type
                    hl7:EntityCity
                    .
                ?subject
                    hl7:isPartOf
                    hl7:Polska
        }
```

Wynik:

```
{
  "head": {
    "vars": [ "subject" ]
  } ,
  "results": {
    "bindings": [
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Krakow" }
      } ,
      {
        "subject": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Warszawa" }
      }
    ]
  }
}
```

Wyszukiwanie miasta w którym mieszka dana osoba Zapytanie:

```
PREFIX rdf:
    <http://www.w3.org/
    1999/02/22-rdf-syntax-ns#>
PREFIX owl:
    <http://www.w3.org/
    2002/07/owl#>
PREFIX xsd:
    <http://www.w3.org/
    2001/XMLSchema#>
PREFIX rdfs:
    <http://www.w3.org/
    2000/01/rdf-schema#>
PREFIX hl7:
    <http://sim.ontology/HL7#>
SELECT ?city ?country
        WHERE {
                hl7:Wojciech_Inglot
                    hl7:livesIn
                    ?city .
                ?city
                    hl7:isPartOf
                    ?region .
                ?region
                    hl7:isPartOf
                    ?country
```

```
                    }
```

Wynik:


```
{
  "head": {
    "vars": [ "city" ,
        "country" ]
  } ,
  "results": {
    "bindings": [
      {
        "city": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Krakow" } ,
        "country": { "type":
            "uri" , "value":
            "http://sim.ontology/
            HL7#Polska" }
      }
    ]
  }
}
```

7 Podsumowanie

Zaproponowana przez nas implementacja reprezentacji oraz transferu danych medycznych posiada wiele cech pożądanych przy projektowaniu tego typu systemów. Rozwiązanie to cechuje się bardzo dużą skalowalnoscią zarówno od strony wielkosci samego systemu jak i możliwosci rozbudowy przechowywanych w nim danych. Zastosowanie architektury typu SOA przy budowie systemu medyczneo pozwala na łatwą, modułową rozbudowę takiego systemu oraz ułatwioną integrację z innymi systemami medycznymi, zupełnie niezależnymi od siebie. Idąc dalej można skorzystać również z możliwosci oferowanych przez narzędzia jeszcze wyższych rzędów, takie jak szyny biznesowe, dzięki którym integracja mogłaby być jeszcze skuteczniejsza, nie naruszając jednoczesnie architektur oraz implementacji poszczególnych systemów.

Poprzez prezentacje przypadku użycia pokazaliśmy jak łatwo można tworzyć aplikacje klienckie do przygotowanej przez nas platformy. Dodawanie obsługi nowych przypadków użycia lub dodawanie nowych aktorów (np. pacjentów, recepcjonistek) nie powinno stanowić problemów. Proces ten proponujemy przeprowadzać analogicznie do przygotowanego przez nas przykładu: zaczynając od ustalenia przypadku użycia, następnie poprzez przygotowanie zapytań sparql (modyfikujących OWL lub pobierających dane), kończąc na implementacji aplikacji klienckiej.

W kwestii reprezentacji danych, wykorzystanie takiego frameworku jak Apache Jena pozwala na intuicyjne operowanie na dowolnej reprezentacji ontologii, ich rozbudowę w srodowisku programowania java, oraz ich graficzną reprezentację. Wykorzystanie tutaj w pracy srodowiska java jednoczesnie czyni znacznie prostszą integrację tego rozwiązania z opisanym wczesniej mechanizmem transferowania informacji, który również napisany jest w tym języku.

Literatura

1. Erik Sundvall, Mikael Nystrom, Daniel Karlsson, Martin Eneling, Rong Chen, Hakan Orman: Applying REST architecture to archetype-based electronic health record systems
2. Wiesław Wajs, Krzysztof Rączka, Paweł Stoch, Piotr Kruczek: Integration Platform As Central Service Of Data Replication In Distributed Medical System
3. Bartosz Jędrzejec: Pozyskiwanie wiedzy z dużych zbiorów danych z zastosowaniem adaptacyjnych procedur generowania zapytań, AGH Kraków 2008
4. Dortje Loper, Meike Klettke, Livio Bruder, Andreas Heuer: Enabling flexible integration of healthcare information using entity-attribute-value storage model
5. IHE cross-enterprise document sharing for imaging: interoperability testing software - Rita Noumeir, Berube Renaud - Springer Link database
6. Integrating Clinical Trial Imaging Data Resources Using Service-Oriented Architecture and Grid Computing - Stefan Baumann El-Ghatta, Thierry Cladé, Joshua C. Snyder - Springer Link database
7. Welcome to Health Information Science and Systems - Yanchun Zhang LabKey Server: An open source platform for scientific data integration, analysis and collaboration - Elizabeth K Nelson, Britt Piehler - Springer Link database
8. A database application for pre-processing, storage and comparison of mass spectra derived from patients and controls - Mark K Titulaer, Ivar Siccama - Springer Link database
9. PASSIM – an open source software system for managing information in biomedical studies - Juris Viksna, Edgars Celms - Springer Link database
10. The HL7 Clinical Document Architecture - Robert H Dolin, Liora Alschuler, Calvin Beebe; Journal of the American Medical Informatics Association Volume 8 Number 6
11. The HL7 Clinical Document Architecture, Release 2 - Robert H Dolin, Liora Alschuler, Calvin Beebe; Journal of the American Medical Informatics Association Volume 13 Number 1
12. HL7 Document Patient Record Architecture: An XML Document Architecture Based on a Shared Information Model - Robert H. Dolin, MD, Liora Alschuler, Fred Behlen, PhD, Paul V. Biron, MLIS, Sandy Boyer, RPh; Dan Essin, MD, Lloyd Harding, Tom Lincoln, MD, John E. Mattison, MD, Wes Rishel, Rachael Sokolowski, John Spinosa, MD, PhD, Jason P. Williams, MS
13. HL7 Version 3—An object-oriented methodology for collaborative standards development - George W Beeler A health-care data model based on the HL7 Reference Information Model - Eggebraaten, T.J. Development of a Clinical Data Warehouse for Hospital Infection Control - Mary F Wisniewski, Piotr Kieszkowski, Brandon M Zagorski,
14. The HL7 Reference Information Model Under Scrutiny - Gunther Schadow , Charles N. Mead, D. Mead Walker Biomedical Engineering (Chapter 19 - Toward Multi-Service Electronic Medical Records Structure) - Bilal I. Alqudah, Suku Nair
15. Introduction to: HL7 Reference Information Model (RIM) - Health Level Seven International
16. HL7 RIM: An Incoherent Standard - Barry Smith, Werner Ceusters
17. Redundancy in electronic health records corpora: analysis, impact on text mining performance and mitigation strategies - Raphael Cohen, Michael Elhadad, Noemie Elhadad

18. Initial experience with asynchronous transfer mode for use in medical imaging network - Minh Dovan, Louis M. Humphrey, Geri Cox, Carl E. Ravin
19. A database application for pre-processing, storage and comparison of mass spectra derived from patients and controls - Mark T Titulaer, Ivar Siccama, Lennard J Dekker
20. Adding HL7 version 3 data types to PostgreSQL - Yeb Havinga, Willem Dijkstra, Ander de Keijzer
21. Clinical data integration of distributed data sources using Health Level Seven (HL7) v3-RIM mapping - Teeradache Viangteeravat, Matthew N Anyanwu, Venkateswara Ra Nagisetty, Emin Kuscu, Mark Eijiro Sakauye, Duojiao Wu
22. Electronic Medical Records vs. Electronic Health Records: Yes, There Is a Difference - Dave Garets and Mike Davis
23. The "New" America Electronic Medical Record (EMR)—Design Criteria and Challenge - Ralph Grams
24. Security of Medical Data Transfer and Storage in Internet. Cryptography, Antiviral Security and Electronic Signature Problems, which Must Be Solved in Nearest Future in Practical Context - Piotr Kasztelowicz, Marek Czubenko, Iwona Zięba
25. Barriers to implement Electronic Health Records (EHRs) - Sima Ajami, Razieh Arab-Chadegani
26. Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen - The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications