

Elektroniczna karta pacjenta

Słowa kluczowe: elektroniczna dokumentacja medyczna, standard HL7 FHIR

Opis

Celem projektu jest stworzenie aplikacji pełniącej rolę prostej elektronicznej karty pacjenta. W szczególności aplikacja powinna wykorzystywać standard FHIR w celu pobierania i aktualizacji danych z wybranego serwera testowego (ograniczenie do kilku wybranych zasobów -- szczegóły w opisie wymagań obowiązkowych) oraz pozwalać na prezentację danych w czytelnej formie uwzględniającej ich aspekt czasowy. Poniżej przedstawiono przykłady interfejsu użytkownika z systemu Eskulap, które można potraktować jako inspirację.

26 - 12 - 1987
Jan Ryszard Nowak Kowalski

Jednostka organizacyjna
1AB

Nr księgi oddziałowej 2313/312313
Nr księgi głównej 432/432/42

HISTORIA LECZENIA

Szukaj zdarzenia...

ROZWIŃ FILTRUJ TYP ZDARZENIA FILTRUJ PRACOWNIKA FILTRUJ CZAS WIĘCEJ FILTRÓW

środa, 2017-10-11
5 dzień pobytu

14:58:05
środa, 2017-10-11

Karta Karta oceny stanu pacjenta przyjętego do Oddziału.
Karty szpitalne

14:58:05
środa, 2017-10-11

Procedura medyczna - Wykonanie zdjęcia RTG.
Procedury zakładowe

14:58:05
środa, 2017-10-11

Skierowanie diagnostyki obrazowej do pracowni ZDO
Skierowanie i zlecenie badań

14:47:51
środa, 2017-10-11

Wywiad
Wywiady

14:47:51
środa, 2017-10-11

Boldaloin 30 tabl. tabl. mg doustnie mg
Zlecenia leków

14:47:51
środa, 2017-10-11

Obserwacja
Obserwacja

13:12:35
środa, 2017-10-11

Konsultacja lekarska
Konsultacja lekarska

12:11:47
środa, 2017-10-11

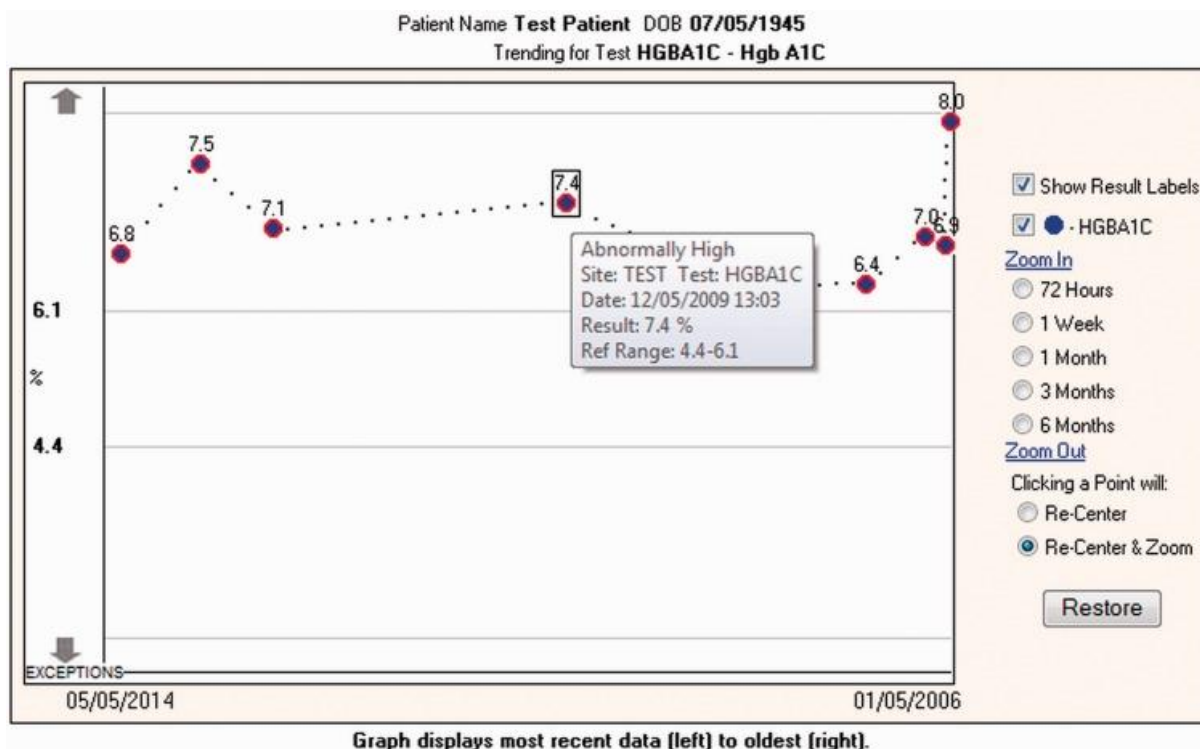
Technologia wykorzystana do budowy aplikacji oraz platforma, na której ma ona działać (np. webowa, desktopowa, mobilna) są dowolne. Proszę tylko zwrócić uwagę, że niektóre technologie (zwłaszcza Java) są lepiej wspierane niż inne, jeśli chodzi o dostępne biblioteki i narzędzia.

Wymagania obowiązkowe

- Aplikacja powinna wyświetlać listę pacjentów (z możliwością filtrowania po nazwisku), po wyborze pacjenta powinna pobierać komplet jego danych -- w tym celu można skorzystać z operacji [everything](#). Po odczytaniu dane powinny być wizualizowane w trybie do odczytu (**bez możliwości edycji**), czy czym powinna istnieć możliwość określenia okresu, który ma być wizualizowany (np. wybrany miesiąc).
- Aplikacja powinna wizualizować następujące zasoby: [Patient](#), [Observation](#), [Medication](#) oraz [MedicationStatement](#)¹ (pozostałe zasoby znajdujące się w zbiorze pobranych informacji można pominąć).
- W przypadku bardziej złożonych zasobów należy zastosować dwa poziomy prezentacji -- podstawowe informacje na diagramie reprezentującym zasoby na osi czasu oraz szczegóły po wyborze (np. dwukrotnym kliknięciu) na wybranym zasobie. **Podczas wyświetlania zasobu można wykorzystać jego podsumowanie tekstowe (text²).**

Wymagania na 4.0

- Aplikacja powinna realizować wymagania podstawowe. Poza tym dla wybranych obserwacji liczbowych (np. waga, temperatura) powinna pozwalać na tworzenie wykresów pokazujących zmianę tych parametrów w czasie. Przykładowe rozwiązanie prezentowane jest poniżej. Na wykresie wystarczy pokazać przebieg jednego parametru, przy czym powinna być możliwość dodatkowego ograniczenia okresu, dla którego prezentowany jest wykres (np. zmiany we wskazanym tygodniu).



- Do stworzenia wykresów należy wykorzystać gotowe biblioteki lub komponenty.

¹ Na osi czasu powinny być wyświetlane zasoby MedicationStatement -- Medication zawiera opis danego leku.

² Pole w każdym zasobie, generowane zazwyczaj automatycznie, które powinno stanowić zwięzłą i czytelną jego reprezentację.

Wymagania na 5.0

- Aplikacja powinna realizować wymagania na 4.0. Poza tym powinna ona umożliwiać edycję wybranych zasobów (ograniczenie 2-4 pól istotnych dla danego zasobu) oraz zapisywać zmiany na serwerze.
- Aplikacja powinna także zaznaczać zasoby dostępne w kilku wersjach i pozwalać na przechodzenie między wersjami oraz prezentację każdej z nich. Oprócz wartości zasobu dla wersji powinna być również wyświetlana informacja o dacie modyfikacji. Warto również rozważyć możliwość zaznaczania kolorem pól zmienionych w stosunku do poprzedniej wersji.

Linki

- Strona z opisem standardu HL7 FHIR: <https://www.hl7.org/fhir/>
 - zasoby: <https://www.hl7.org/fhir/resourceindex.html>
 - usługi: <https://www.hl7.org/fhir/http.html>
 - narzędzia: <https://www.hl7.org/fhir/downloads.html>
- Biblioteka HAPI-FHIR: <http://hapifhir.io/>
- Testowy serwer FHIR: <http://fhirtest.uhn.ca/>
- Klient FHIR dla Python-a: <http://docs.smarthealthit.org/clients/python/>

Inne materiały

- Zalecenia CUI odnośnie prezentowania danych na osi czasu (*timeline*): <http://webarchive.nationalarchives.gov.uk/20160921150545/http://systems.digital.nhs.uk/data/cui/ug/timelineview.pdf> (zostały one opracowane dla leków, ale mogą zawierać praktyczne sugestie i wskazówki)

Przygotowanie własnego serwera

Z uwagi na zgłaszane problemy związane z dostępnością serwera testowego FHIR oraz z jakością danych dostępnych na tym serwerze poniżej zamieszczam opis procedury uruchamiania własnego serwera oraz wypełniania go danymi przykładowymi.

Serwer FHIR

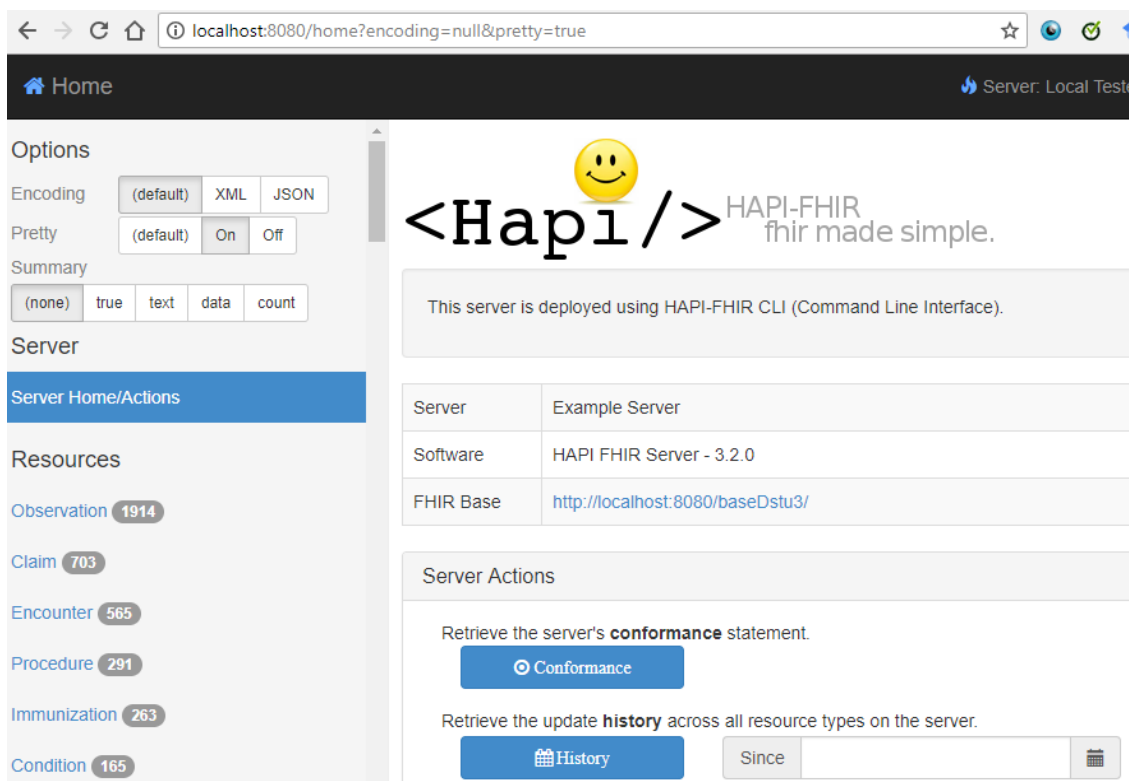
W ramach projektu HAPI FHIR dostępna jest “paczka” z samodzielnym serwerem, który jest uruchamiany z linii poleceń i nie wymaga żadnego dodatkowego oprogramowania (poza JRE 1.8). Sposób wykorzystania serwera został przedstawiony [tutaj](#), natomiast skompiowane oprogramowanie można pobrać z [GitHub-a](#).

Uwaga: polecam skorzystać z wersji 3.2 -- wersja 3.3 sprawia problemy (“sypie się” podczas uruchamiania).

Uruchomienie serwera sprowadza się do wydania polecenia:

```
hapi-fhir-cli run-server
```

Domyślnie serwer jest dostępny pod adresem <http://localhost:8080> (proste GUI przedstawione poniżej³) oraz <http://localhost:8080/baseDstu3> (zasoby i usługi).



Przykładowe dane

Projekt [Synthea](#) udostępnia generator danych medycznych obejmujący cały cykl życia pacjenta. Na stronie projektu udostępniono również przykładowe zbiory (1000+ pacjentów wraz z informacjami o stwierdzonych diagnozach, zebranych obserwacjach, przepisanych lekach) w kilku formatach, w tym FHIR (JSON). Pobierane dane powinny być zgodne z wersją FHIR obsługiwaną przez serwer -- w przypadku ustawień domyślnych należy pobrać [plik w wersji DSTU3](#).

Uwaga: w przypadku wykorzystania tych danych należy wizualizować zasoby [MedicationRequest](#) zamiast MedicationStatement.

Załadowanie danych na serwer

Do załadowania danych na serwer można wykorzystać narzędzie tag-uploader, które wprowadza odpowiednie poprawki do definicji zasobów (np. zmienia typ zasobu Bundle z collection na transaction, dzięki temu komplet danych dla jednego pacjenta może być w całości przesłany na serwer). Do zainstalowania i uruchomienia tego narzędzia niezbędny jest Node.js.

Instalacja tag-uploader-a wygląda następująco:

```
git clone https://github.com/smart-on-fhir/tag-uploader.git
cd tag-uploader
```

³ Ekran przedstawia stan serwera po załadowaniu części danych testowych. Nowa instalacja serwera nie zawiera żadnych danych (zasobów).

```
npm install
```

Proces ładowania danych uruchamiany jest następującym poleceniem:

```
node . -d <synthea-data> -S <base-address>
```

gdzie <synthea-data> to katalog zawierający rozpakowane przykładowe dane (każdy pacjent w osobnym pliku .json), a <base-address> to adres bazowy serwera. Jeśli dane znajdują się w katalogu sample-data, a serwer działa z ustawieniami domyślnymi, wówczas polecenie to będzie następujące:

```
node . -d ./sample-data -S http://localhost:8080/baseDstu3
```

Dane można też wysłać na serwer testowy:

```
node . -d ./sample-data -S http://fhirtest.uhn.ca/baseDstu3
```

Uwaga: ładowanie pacjentów jest powolne (5-10 sekund dla pacjenta w przypadku wykorzystania lokalnego serwera), dlatego proponuję załadować podzbiór przykładowych pacjentów.

Mapowanie wybranych kodów

Część pól zasobów zawiera kody (np. LOINC dla obserwacji, RxNorm dla leków). Odpowiadające im czytelne określenia można znaleźć w źródłach generatora danych (pakiet [modules](#)) unikając w ten sposób przeszukiwania pełnej terminologii.