AMIA 2018 Annual Symposium

View Abstract

Title: FHIR Developer's Toolkit: A UI-Component-Driven Reactive App with Analytics

Abstract Body:

Project abstract (1050 Characters): LibreHealth Toolkit serves as the foundational API and data model for many Health IT applications with backend being forked from the OpenMRS data model. There is a need to move towards the FHIR specification to overcome nteroprability issues. LibreHealth is working on adopting the latest v3 STU FHIR across the entire platform.
This project was divided into three sub-projects. The first sub-project builds reusable FHIR user-interface components. These components implement the FHIR RESTful services for developer to build EHR system with FHIR's REST interfaces. The second sub-project migrates the Toolkit's OpenMRS data model to a more modular Spring Data mechanism. The third sub-project implements a UI for the EHR user to analyze data and execute SQL-like queries on FHIR Resources.
The three sub-projects combined create a powerful, novel and cutting-edge functionality based on the FHIR standards, this project is a Work In Progress and in a very early stage to prove its clinical value with evaluation studies among clinicians.

Project rationale, impact and innovation (3500 Characters): Proposed Problem:
1. Implementing a FHIR-based EHR system or apps require a lot of boilerplate code to be repeated throughout an app. Nearly all apps written on FHIR do GET, POST, PUT, DELETE to FHIR RESTful web services either communicating using JSON or XML representations. It would speed-up building apps, and let developers focus on clinical workflow and functionality if some pre-built components that implemented FHIR were available.
2. To move away from the proprietary OpenMRS data model, promote efficient reading and writing of clinical data and improve interoperability, we need to move to the FHIR (Fast Healthcare Interoperability Resources) Specification. Also, providers would like to get immediate notifications rather that continuously pull data, when patient information in EHR system changes.
3. Analyzing data requires transformation to a format that is compatible with analytical engines. Many organizations store FHIR representation of patient information as documents. These documents might be in a file system or cloud storage and data extraction can be difficult. Some organizations use NoSQL and RDBMS databases to store FHIR resources which provide flexibility to query data with less difficulty. But modeling schemas to store FHIR resources is often complex and requires combining data from columns to recreate the FHIR resource.

Proposed solution to the above problems being implemented in this project
1. To reduce repeated coding, enable web service communication using JSON or XML and enable developers to concentrate more on workflow, web components were built. We used a JavaScript framework that implements the latest W3C web components standard called Google Polymer. This enabled EHR developers to reuse UI components that were coded to communicate with FHIR v3 Resources. The Resources from modules in level 3 and level 4 of FHIR with Maturity Level above 3 were created. These resources were specifically selected as they contribute to the patient-centric UI of the EHR. The UI component uses Material Design. Thus, EHR developers who use our web components can get a usability-focused EHR by default, that they can further customize to meet their EHR workflow needs.
2. To replace or migrate the Toolkit's existing persistence schema, the backend development provides an entirely reactive and modular solution to the above problems. As it uses Spring Data Repositories, there is no need to revamp the whole codebase whenever the FHIR specification is updated. In addition to this, it does not hardcode the FHIR resources, rather it retrieves them as a dependency of the HAPI library and maps them to the database supported types using a conversion mechanism.
3. To overcome difficulties analyzing gigabytes of data in FHIR specification, a module was developed where the user can analyze data via simple search UI and UI to execute SQL-like queries on FHIR resources. This module may be useful for researchers who wish to write complex queries using Spark SQL. It uses Cerner's Bunsen along and Apache Spark to build the platform for FHIR analytics.

Project design and implementation (7000 characters): The FHIR Web Components:

The web components that we developed are reusable components, which can be used as the smallest unit of EHR systems user-interface and assembled together on web pages to communicate with any FHIR v3 compatible EHR server. These web components follow Material Design principles as its shaping the way people see and interact with interfaces because of clear design and usability guidelines. It has the following characteristics:

1. Each component can be used for POST (user input and then send data to the FHIR-based EHR server) and GET of patient data.
2. Each field from a component (that matches with the FHIR Resource's field) can be removed by a developer just by using 'true' or 'false' in properties. eg. <fhir-person-identifier systemIdentifier = "false"> removes the systemIdentifier field from the component. Thus, each component is flexible to the needs of the EHR system.
3. On GET, if the patient data has multiple points of data for the same component, the extra fields pertaining to that data get created so as to accommodate that data and hence avoid loss of patient data.
4. If any component does not GET values for all the fields they contain, the fields not getting values remain unfilled due to partial filling of component.

These components can be used to develop an EHR system or apps by simply importing the component and using the custom HTML with tags as shown below.

The Reactive NoSQL DataStore for FHIR:

The new data store for the LibreHealth Toolkit was built as a Reactive Spring Data app which uses Cassandra as its NoSQL datastore to provide support for create, read, update and delete (CRUD) operations using its REST API. We made use of the HAPI FHIR DSTU3 structures, so that it is easy to update the FHIR Resource version changes. The app supports CRUD operations for Patient, Observation and Encounter and partial *search* operation for Patient with support for more resources to come. To avoid local modifications, we created sub-classes of the original HAPI structures and annotated properties, which implement certain functions such as defining the primary key, ignoring certain fields etc. To correctly save the data Cassandra needed to understand the complex data types of HAPI structures. The solution was to implement various converters which act as a layer between the models and the persistence layers and work to convert the complex data types to primitive data types which can be easily understood by Cassandra when writing the data and can just as easily be converted back to the complex data types at the time of reading the data. All of this is available in a Reactive way, so that the REST API can be exposed as a Stream, which will provide updates, as soon as the data store receives any updates.

The FHIR Analytics Engine using Spark SQL:

The FHIR Analytics Module provides the ability for making functional queries of FHIR data through Spark SQL which is a SQL-like language query using Apache Spark. The FHIR analytics module also provides a user interface to perform queries on each resource for five important FHIR resources - Patient, Encounter, Observation, DiagnosticReport and MedicationRequest.

FHIR Analytic Using Spark SQL

The FHIR Analytics Module Provide UI to execute Spark SQL against the data available in the system.

FHIR Resource Base Analytics

If there is difficulty writing Spark SQL queries, the analytics module provides a simple UI to perform analytics. Here is the patient demographics-based search UI. A user can search for an exact match or a contains search by using the contains checkbox. Also, a user can do a search with combining attributes.

Data Upload User Interface

The FHIR Analytics Module provides functionality to upload data from a file. A user can upload a FHIR Resource or a Bundle and perform analysis of this resource.

Project evaluation and sustainability (3500 characters): Since this project is a Work In Progress, and very recently developed, it has not been evaluated yet in a clinical setting. But it showcases many novel ideas based on modern web technologies that could influence FHIR developers from across the globe.

Twitter project summary (140 characters): https://twitter.com/librehealthio?lang=en

Any other information about the project we should know about (1500 characters)?: LINK TO VARIOUS REPOS USED FOR THISPROJECT:

First Sub-project:

Link for the web components repo:

https://gitlab.com/librehealth/lh-toolkit-webcomponents

Link for a rudimentary app that is created as an example to use our FHIR web components:
https://gitlab.com/parumenon.pm/lh-toolkit-app
Second Sub-project:
https://gitlab.com/yashdsaraf/reactive-lh-toolkit

Third Sub-project:
https://gitlab.com/kavindya89/librehealth-fhir-analytics

VARIOUS DISCUSSION ON PLANNING OF THIS PROJECT IN SUMMER :
https://forums.librehealth.io/c/projects/lh-toolkit

Customers: No

Solution Date: 04/01/2018

Implementation Date: 08/14/2018

How many users does your solution have or how many patients have been impacted by it (please indicate time frame)?: Not implemented yet (WIP)

Authors: Parvati Ravindranathan Menon Naliyatthaliyazchayil[1], Prashadi Bandara[3], Yash Saraf[2], Judy W. Gichoya[4], Saptarshi Purkayastha[1]

Institutions: 1. Department of Biohealth Informatics, Indiana University Purdue University Indianapolis (IUPUI), Indianapolis, IN, United States.
2. Jai Hind College, Mumbai, India.
3. University of Canberra, Canberra, ACT, Australia.
4. Oregon Health and Science University, Portland, OR, United States.