Propuesta para el Desarrollo de un Índice Maestro de Pacientes con el Estándar FHIR de HL7

Carlos Alejandro Martinez[1], Ricardo Cimarelli[2], Tamara Fazio[1], Gerardo Fuentes[3]

[1] Hospital Regional Dr. Antonio J. Scaravelli. Comité de Docencia e Investigación
{carlos_martinez, tgfazio}@mendoza.gov.ar
[2] Hospital Dr. Victorino Tagarelli. Informática
ricardo.cimarelli@gmail.com
[3] Dirección de Informática del Ministerio de Salud, D. Social y Deportes de la Provincia de Mendoza. Departamento de Proyectos
gfuentes@mendoza.gov.ar

Resumen. Un problema importante que se presenta en sistemas de salud compuesto por muchos efectores es poder interoperar entre ellos manteniendo una persona en forma única en todo el sistema de salud evitando la duplicación de los datos. La utilización de estándares como FHIR con la definición de una arquitectura orientada a servicios es fundamental para lograr un diseño que solucione en forma efectiva el problema. En este trabajo se presenta una propuesta de arquitectura utilizando software open source para permitir la integración de los datos de las personas y la interoperabilidad entre distintos sistemas. El objetivo es la utilización del estándar FHIR para la definición de los componentes que permitan obtener una aplicación que cumple con los requerimientos de un Índice Maestro de Pacientes.

Palabras Clave: Índice Maestro de Pacientes, FHIR, Interoperabilidad.

1 Introducción

El problema a resolver consiste en la falta de interoperabilidad entre los efectores del sistema de salud. Cada efector gestiona sus propios datos sin existir la comunicación necesaria para mantener la integración entre los mismos. Esta falta de comunicación provoca inconsistencias entre los sistemas de información de salud. Una misma persona puede estar dada de alta en diferentes efectores, y sus datos de filiación pueden variar de uno a otro. Como se explica en [1], la identificación errónea de una persona por otra, como la omisión de registros previos, pueden ocasionar confusiones graves en procedimientos médicos, quirúrgicos, prescripción y entrega de medicamentos, realización de estudios y demás intervenciones asistenciales o administrativas. Esto puede provocar consecuencias potencialmente graves para el paciente como para la institución. Es decir, la identificación errónea de personas puede producir la pérdida de información o mala praxis.

Teniendo en cuenta la problemática expuesta es necesario definir una arquitectura utilizando los estándares existentes. Esta arquitectura debe contener los componentes necesarios que sirvan de base para lograr un sistema:

- Interoperable, utilizando estándares específicos de salud y servicios web, que asegure la independencia de la tecnología.
- Basado en estándares de salud existentes como FHIR (Fast Healthcare Interoperability Resources) [2].
- Escalable, permitiendo agregar paulatinamente módulos.

El presente trabajo tiene como objetivo presentar una arquitectura que sirva de base para el desarrollo de un sistema de salud utilizando el estándar FHIR. Se plantea como primer componente de la arquitectura el Índice Maestro de Pacientes (MPI - por sus siglas en inglés Master Patient Index).

Los autores en [3], presentan una propuesta implementada en el Hospital Regional Antonio J Scaravelli [4], destinada a funcionar localmente focalizando la integración de los sistemas internos, la identificación unívoca de los pacientes en la institución y soportando un proceso de auditoría. Lo expuesto en este artículo es una evolución de esta, incluyendo la interoperabilidad a través de estándares internacionales así como una arquitectura orientada a servicios. Se utiliza como base para el actual desarrollo del Índice maestro de pacientes del Ministerio de Salud, Desarrollo Social y Deportes de la Provincia de Mendoza de la República Argentina.

El artículo está organizado de la siguiente forma: En la sección 2, se presenta la metodología y relación con otras propuestas. En la sección 3, se explica la arquitectura. En la sección 4, se describe el recurso Patient. En la sección 5, se presenta una discusión sobre temas claves de la arquitectura y finalmente en la sección 6 se exponen las conclusiones y el trabajo futuro.

2 Metodología y Relación con Otras Propuestas

Se utiliza una metodología de desarrollo de software ágil siguiendo un proceso continuo, iterativo e incremental que atraviesa por varias etapas de refinamiento y validación. Como resultado de este proceso se obtiene la versión del Índice Maestro de Pacientes que actualmente está en fase de prueba. El proceso iterativo utilizado para gestionar y concretar el proyecto se basa en Scrum [5] y se utiliza la plataforma para la gestión de proyectos Open Source, Taiga [6].

De acuerdo al objetivo planteado para la definición de la arquitectura básica de un sistema de salud y el Desarrollo de un Índice Maestro de Pacientes, el siguiente trabajo ha tenido en cuenta los temas desarrollados en: [7], [8], [9], [10] y [11].

En [7] se define cómo se tiene que construir un identificador único de un paciente. En [8] se describen los pasos para una solución óptima de implementación de un MPI. De [9] se obtiene una guía fundamental para el desarrollo de este tipo de sistemas, como referencias a métodos de bloqueos. En [10] el estándar de la OMG PIDS nos brinda las pautas para cumplir con el objetivo de soportar la identificación de pacientes a través

de múltiples organismos de salud. En [11] se define el perfil de integración PIX (Patient Identifier Cross Referencing) de la IHE (Integrating the Healthcare Enterprise) [12], este perfil soporta la correlación de identificadores de pacientes procedentes de múltiples dominios.

Existen varios productos con la funcionalidad de un Índice Maestro de Pacientes como los que se comparan en [13]. Entre estos se evalúo OpenEmpi (por sus siglas en inglés: Open Enterprise Master Patient Index) [14]. Este es un producto de código abierto que funciona con servicios REST. El presente trabajo se diferencia de [14] en: la implementación de las búsqueda de candidatos, en los componentes de la arquitectura y en la forma de adaptación del estándar FHIR a la funcionalidad requerida, especialmente la del recurso Patient y el módulo de Auditoría.

3 Arquitectura Propuesta

En base a los objetivos planteados anteriormente se presenta una propuesta de arquitectura utilizando el estándar FHIR y compuesta por software Open Source para la implementación de un Índice Maestro de Pacientes. En la Figura 1, se muestra el esquema con la arquitectura propuesta y luego se describen los componentes.

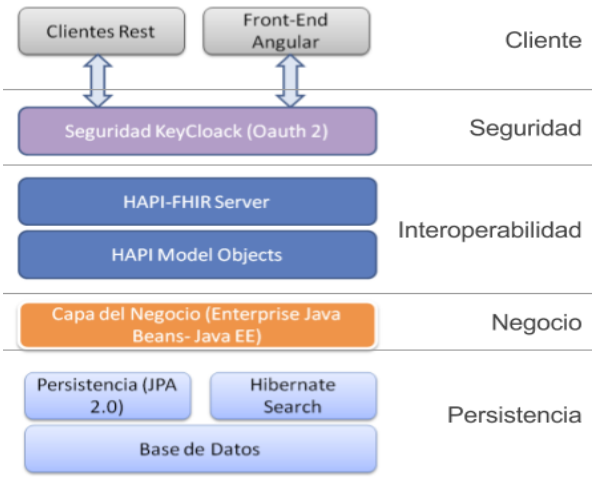

Fig. 1. Arquitectura Propuesta

3.1 Clientes REST

Los clientes interactúan vía REST [15] con el servidor HAPI-FHIR [16] previamente autenticandose, solicitando autorización y el acceso a los recursos por medio del componente de seguridad. Se desarrolla una aplicación cliente con Angular [17], que contiene la funcionalidad necesaria para la gestión de un índice maestro de pacientes

como: empadronamiento, búsqueda MPI y procesos de auditoría como la fusión de personas. El resto de los efectores tienen sus sistemas hospitalarios en distintas tecnologías y accederán al índice maestro de pacientes directamente con REST o indirectamente por medio de un motor de gestión de mensajes como Mirth [18].

3.2 Capa de Seguridad

Para la autenticación y/o autorización de usuarios, el estándar FHIR recomienda el uso del perfil Smart-On-FHIR [19], OpenID Connect [20], y Oauth 2 [21]. En este trabajo implementamos el componente de seguridad con el gestor de acceso e identidad: Keycloack [22]. Keycloak soporta los protocolos estándares recomendados por FHIR: OpenID Connect y OAuth 2.0. Se implementa como un servidor de seguridad para los servicios REST (Representational State Transfer).

3.3 Capa de Interoperabilidad

HAPI-FHIR es una librería open source desarrollada en Java que utiliza la especificación definida en el estándar FHIR. HAPI FHIR propone varias formas de implementación, como se explica en [16]. La elegida en este proyecto utiliza el servidor basado en servlets que provee capacidades RESTFull. Se definen varios proveedores que son clases que permiten el acceso a los recursos. Se utilizan varios de estos, como: Patient, Organization y AuditEvent.

3.4 Capa del Negocio

En esta capa se trabaja con Java EE [23] y la aplicación se despliega sobre el servidor de aplicaciones JBOSS [24]. En la capa de negocio se definen las entidades y el módulo que interactúa con los recursos provistos por el servidor HAPI-FHIR. Las entidades implementadas contemplan los atributos definidos en el estándar y otros necesarios para los procesos del MPI. Los enterprise services bean implementan la lógica del negocio compuesta por las operaciones CRUD (por sus siglas en inglés: Create, Read, Update and Delete) sobre los recursos, validaciones y reglas del negocio.

3.5 Capa de Persistencia y Búsqueda

La capa de persistencia se implementa con el API de persistencia JPA [25]. JPA es un framework basado en POJO que permite el mapeo objeto/relacional en aplicaciones Java EE. JPA se utiliza en conjunto con EJB a través de hibernate. El componente de búsqueda utiliza el software HibernateSearch [26] con Apache Lucene [27]. El mismo se utiliza como motor de búsqueda en texto completo por las siguientes razones: Alta integración con el Api de Persistencia de Java, rápida indexación en disco, soporte nativo en servidor de aplicaciones JBOSS y gran variedad de algoritmos de bloqueo y similitud. Este componente permite indexar el contenido del modelo de datos para ofre-

cer búsquedas avanzadas con texto libre. Se utiliza como analizador fonético el algoritmo DoubleMetaphone [28] configurado para su uso con el idioma español. Este método permite por cada cadena, obtener un código que permite su normalización para su posterior comparación con otras cadenas.

3.6 Base de Datos

La tecnología empleada permite la utilización de distintas tecnologías de Base de datos por la utilización de Hibernate en la capa de persistencia de la aplicación. En este caso se utiliza la base de datos PostgreSQL [29].

4 Recurso Patient

Se utiliza el recurso Patient que se encuentra en un nivel de madurez 5 de acuerdo al estándar FHIR. Se utilizan los atributos de este recurso y se siguen las recomendaciones para la definición de los procesos.

Para identificar en forma única a una persona hay que usar atributos que perduren o cambien poco en el tiempo. En este trabajo se utilizan algunos de los propuestos en [1] y utilizados en [3], los cuales están formados por: apellidos (family), nombres (given), sexo (gender), fecha de Nacimiento (birthDate), tipo y número de documento de identidad (identifier). A los atributos que se utilizan para la identificación única de una persona se denomina Conjunto Mínimo de Datos. El uso de un conjunto básico de atributos es conveniente complementarlo con el uso de algún tipo de registro biométrico como la foto (photo) de la cara o más avanzados como el de retina o reconocimiento de rostro. El uso de registros biométricos permite la identificación unívoca de pacientes en forma rápida y precisa. En la Figura 2, se muestra el conjunto mínimo de datos del recurso Patient.


```
"resourceType":"Patient",
   "id":"10",
   "meta":{
      "versionId":"0"
   },
   "active":true,
   "name":[
      {
         "family":[
            "LOPEZ"
         ],
         "given":[
            "JUAN"
         ]
      }
   ],
   "gender":"male",
   "birthDate":"1999-09-09"
}
```


Fig. 2. Conjunto Mínimo de Datos del Recurso Patient

El identificador oficial forma parte opcional del conjunto mínimo de datos. Es decir, que la búsquedas por set mínimo de datos pueden opcionalmente usar como parámetro algún identificador oficial como documento único (DU) de identidad de Argentina.

Para cumplir con el objetivo de soportar la identificación de pacientes a través de múltiples organismos de salud se considera lo propuesto por FHIR y el esquema propuesto por PIDS [10]. Cada organismo de salud asigna identificadores (ID) que identifica a pacientes dentro de su dominio local de valores de ID. Fuera de ese sistema u organización dichos IDs carecen de significado. Utilizando el concepto de identificador del recurso Patient de FHIR se puede soportar de manera simultánea la asignación de IDs dentro de un dominio particular y la correlación de IDs entre múltiples dominios. En la Figura 3, se puede ver un ejemplo de identificador del recurso Patient.

```
"identifier":[
{
  "use":"official",
  "system":"http://www.renaper.gov.ar/dni",
  "value":"66789098",
  "period":{
    "start":"2017-04-06",
    "end":"0000-00-00"
  },
{
  "use":"usual",
  "system":"http://www.hospital-scaravelli.mendoza.gov.ar/hc",
    "value":"1001",
    "period":{
      "start":"2017-04-06",
      "end":"0000-00-00"
}}  ]
```

Fig. 3. Identificadores del Recurso Patient

Se crean extensiones en el caso que lo contenido en el estándar no permite cumplir con los requerimientos. En la extensión estado (state), se define los estados por los que pasa el recurso Patient. Estos estados son: permanente (PERMANENT) y validado (VALIDATED). Está en estado permanente cuando se ha dado de alta y todavía no se han validado sus datos contra otro servicio seguro. El rol de auditor de los datos del MPI también tiene el permiso para validar los datos. El recurso pasa al estado validado cuando el auditor o un proceso automático valida el conjunto mínimo de datos del paciente. Este proceso automático se ejecuta en forma paralela y filtra las instancias de Patient que están en estado permanente. Luego en base a la lista producto del filtro se va contrastando los datos contra un servicio web provisto por SISA [30]. En el caso de encontrar coincidencia por el conjunto mínimo de datos de más del 95% el recurso queda en estado validado. En otro caso pasa al auditor para su revisión. En la Figura 4, se muestran algunas de las extensiones definidas para el recurso patient.

```
"extension":[
    {
      "url":"http://wwww.mendoza.gov.ar/extensions#state",
```

```
        "valueString":"PERMANENT"
      },
      {
        "url":"  http://hl7.org/fhir/StructureDefinition/patient-national-
ity",
        "valueString":"{\"id\":200,\"nombre\":\"ARGENTINA  \"}"
      },
      {
        "url":"http://wwww.mendoza.gov.ar/extensions# denounced",
        "valueString":"true"
      }
  ],
```


Fig. 4. Extensiones del Recurso Patient

La fusión de uno o más pacientes es una de las tareas del proceso de auditoría. Consiste en la unificación de instancias del recurso Patient en una sola instancia validada. Esto sucede cuando se encuentra un mismo paciente repetido una o varias veces en la base de datos. FHIR contempla las fusión mediante el elemento link. Como se muestra en la Figura 5, el recurso a pasado al estado inactivo (false) y en link Figura la referencia al recurso Patient que ha quedado en la base de datos. El tipo que se utiliza en este caso es replace, indicando que el paciente se reemplazó por otro. En la Figura 5, se puede ver un ejemplo de fusión del recurso Patient.


```
"active":false,
"link":[
      {
        "other":{
          "reference":"2"
        },
        "type":"replace"
      }
  ]
```


Fig. 5. Fusión del Recurso Patient

Otro elemento para la implementación de un MPI con FHIR es la búsqueda MPI. Implementar el proceso de búsqueda de candidatos es esencial para la detección de datos duplicados. Aquellas instancias que se detectan como duplicados son denunciadas para su posterior verificación por un auditor. Para las denuncias se utiliza la extensión "denounced" que se muestra en la Figura 4. Para la búsqueda MPI en FHIR se utiliza el formato (POST [base]/Patient/$match), arriba de la Figura 6, se muestra un ejemplo de consulta de candidatos. Como se puede ver en la misma Figura 6, el resultado de la consulta de candidatos utiliza una extensión "algorithmic-match" por cada recurso devuelto en la lista. El código del valor de esta extensión puede ser: cierto (certain), probable (probable), posible (posible) y de ninguna manera (certainly-not). Esta clasificación se define en base a reglas del negocio según el peso de las instancias devueltas por la búsqueda. El peso también se especifica en el elemento score.

Consulta de Candidatos

http://mpisaluddev.mendoza.gov.ar:8080/mpi-fhir/fhir/Patient?_query=**match&family=perez&given=susana&gender=female&birthdate=1937-08-11&identifier=http://www.renaper.gov.ar/dni|3057478**

Resultado devuelto

"resourceType":"Patient",
"id":"10",
....
"search": {
"extension": [
{"http://hl7.org/fhir/StructureDefinition/algorithmic-match",
"valueCode": "certain"}],
"score": 1}

Fig. 6. Búsqueda MPI

5 Discusión

La definición de la arquitectura orientada a servicios siguiendo el estándar FHIR puede funcionar sin problemas para la implementación de un índice maestro de pacientes y recursos como Organization, que se implementaron para poder cumplir con los requerimientos. Algunos de los servicios web REST definidos son los siguientes:

Consulta de Candidatos

WS001 - Consulta de Candidatos

Recurso Patient

WS002 - Consulta de Recurso Patient por Id Provincial
WS003 - Alta - Agregar Recurso Patient
WS004 - Modificación - Actualización Recurso Patient
WS005 - Fusionar un Paciente
WS006 - Desfusionar un Paciente
WS007 - Consulta de Pacientes Fusionados
WS008 - Consulta Pacientes Denunciados
WS009 - Denunciar a un Paciente
WS010 - Quitar Denuncia a un Paciente
WS011 - Verificar Paciente Manualmente
WS012 - Quitar Verificación de Paciente Manualmente
WS013 - Paciente Fallecido
WS014 - Paciente No Fallecido
WS015 - Historia del Paciente
WS016 - Consulta de Paciente por Tipo (System) y Número de Identificador
WS017 - Consulta de Pacientes Sin Validar

Recurso Organization
WS018 - Consulta de Recurso Organization por Id
WS019 - Alta - Agregar un Recurso Organization
WS020 - Modificación - Actualización Recurso Organization

Solo para el MPI se crearon 20 servicios REST. Para contemplar el crecimiento del sistema es necesario agregar a la arquitectura un Bus de Servicio Empresarial ESB (Enterprise Service Bus), que actúe como intermediario entre las aplicaciones. Teniendo en cuenta que los servicios provistos pueden estar en distintos puntos de accesos, el ESB centraliza los llamados y enruta los mensajes a donde corresponda. Otras necesidades de la arquitectura es la integración de un motor BPMN 2 (Business Process Model and Notation) para la definición de procesos y la orquestación de servicios. Además de un motor de reglas de negocio para la gestión centralizada de las mismas.

A nivel del estándar FHIR en este artículo se presenta la implementación realizada sobre el recurso Patient. Si bien la capa HAPI FHIR valida la sintaxis de los elementos que componen al recurso Patient; las reglas del negocio y validaciones necesarias para trabajar con el recurso Patient que llega con la operación REST (POST, PUT, GET) se complica especialmente con los cambios de estado del recurso. Logramos refactorizar y simplificar el código fuente Java EE de la aplicación, mediante el uso de operaciones expuestas como servicio REST que solo modifican el estado de un atributo del recurso. Por ejemplo, verificar un paciente manualmente. Esta operación cambia el estado del recurso Patient de Permanente a Validado sin necesidad de procesar la totalidad del recurso.

6 Conclusiones y Trabajo Futuro

En este trabajo se plantea una propuesta para solucionar el problema de la falta de integración e interoperabilidad de los datos de los pacientes de los efectores de salud de la provincia de Mendoza. Para esto se logra diseñar e implementar una arquitectura que contempla la interoperabilidad, la escalabilidad y la utilización del estándar FHIR de HL7.

El aporte principal presentado en este artículo, son los componentes que sirven de base para el desarrollo e implementación de un índice maestro de pacientes. El uso de estándares y el uso de la tecnología de servicios web permite que el sistema pueda crecer incorporando progresivamente nuevos componentes y exponer nuevos servicios con el formato de FHIR. En base al estándar se utiliza el recurso Patient para poder implementar un índice maestro de pacientes. Alrededor de este recurso se definieron los procesos, reglas del negocio y validaciones para cumplir con los requerimientos de un MPI.

Como trabajo futuro se van a ir agregando nuevos componentes a la arquitectura base propuesta para lograr un sistema de salud basado en el estándar FHIR.

Referencias

1. Garfi L., Navajas P., Gómez A., Luna D., González Bernardo de Quirós F: Implementación de un sistema centralizado para la identificación de pacientes en un hospital de alta complejidad.5to Simposio de Informática en Salud - 31 JAIIO. – 2002.
2. Estándar FHIR-HL7. https://www.hl7.org/fhir/. Último acceso 25/04/2017.
3. Martinez, C. A., Cimarelli, R., Fazio, T., Fuentes, G. - Desarrollo de un Índice Maestro de Pacientes Utilizando Estándares y Software Open Source. 6º Congreso Argentino de Informática y Salud (CAIS). 2015. ISSN: 2451-7607 (67-77)
4. Hospital Regional Antonio J. Scaravelli.www.hospital-scaravelli.mendoza.gov.ar. Último acceso 25/04/2017.
5. Scrum. www.scrum.org. Último acceso 15/02/2017.
6. Taiga. https://taiga.io/. Último acceso 20/04/2017.
7. Appavu, S., Analysis of Unique Patient Identifier Options. 1997, The National Committee on Vital and Health Statistics (NCVHS) - Department of Health and Human Services (U.S.).
8. Weber, G.I. Achieving a patient unit record within electronic record systems. in Toward an ElectronicPatient Record. 1995: Newton, Mass.
9. Erik A Sauleau, Jean-Philippe Paumier, AntoineBuemi. Medical record linkage in health information systems by approximate string matching and clustering. 2005 Sauleau et al licensee BioMed Central Ltd. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1274322/
10. Person Identification Service (PIDS) Specification. www.omg.org/spec/PIDS/1.1. Último acceso 20/04/2017.
11. Patient Identifier Cross Referencing (PIX). http://wiki.ihe.net/index.php/Patient_Identifier_Cross-Referencing. Último acceso 20/04/2017.
12. Integrating the Healthcare Enterprise (IHE). http://wiki.ihe.net/index.php/Main_Page. Último acceso 20/04/2017.
13. MPI/EMPI Feature Comparison Matrix. http://www.caredatasystems.com/index.php/empi-feature-comparison. Último acceso 20/04/2017.
14. OpenEmpi. www.openempi.org. Último acceso 20/04/2017.
15. REST. https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional. Último acceso 20/04/2017.
16. HAPI-FHIR. http://hapifhir.io/index.html. Último acceso 20/04/2017.
17. Angular.https://angular.io/. Último acceso 20/04/2017.
18. Mirth https://www.mirth.com/. Último acceso 20/04/2017.
19. Smart-On-FHIR. http://docs.smarthealthit.org/. Último acceso 20/04/2017.
20. OpenID Connect. http://openid.net/connect/. Último acceso 20/04/2017.
21. Oauth 2. https://oauth.net/2/. Último acceso 20/04/2017.
22. Keycloak. http://www.keycloak.org/.Último acceso 20/04/2017.
23. Java EE. www.oracle.com/technetwork/java/javaee/overview/. Último acceso 20/04/2017.
24. JBOSS. www.jboss.org/. Último acceso 20/04/2017.
25. JPA. hibernate.org/orm/. Último acceso 20/04/2017.
26. HibernateSearch. hibernate.org/search/. Último acceso 20/04/2017.
27. Apache Lucene.https://lucene.apache.org/. Último acceso 20/04/2017.
28. DoubleMetaphone. https://xlinux.nist.gov/dads/HTML/doubleMetaphone.html. Último acceso 20/04/2017.
29. PostgreSQL. https://www.postgresql.org/. Último acceso 20/04/2017.
30. SISA WebServices: https://sisa.msal.gov.ar/sisa/sisadoc/docs/0203/ws_sisa.jsp. Último Acceso 24/4/2017