Curso de FHIR, Unidad 3:

FHIR Avanzado

Material de Lectura

Language: Español

V4.0ES

Course Overview

Tabla de Contenidos

Contenido de la unidad y objetivos didácticos

Esta unidad discute algunos temas avanzados: transacciones, mensajes y documentos usando FHIR, la relación entre FHIR y CDA R2, y los enfoques arquitectónicos en FHIR: que usar y cuándo.
El poder de FHIR, un poder que Ud. precisa liberar, es que, para su caso de uso o escenario específico, en el que puede intercambiar un conjunto de recursos con uno o más servidores, FHIR no restringe ni los tipos de recursos ni su cantidad ni en qué momento se realiza el intercambio. Ni siquiera restringe el transporte utilizado para el intercambio de los recursos.
Ud. es libre de seleccionar cualquier recurso combinado con cualquier otro, y utilizar cualquiera de los paradigmas que exploraremos en esta Unidad.
Esta libertad viene con una carga: debe Ud. definir qué recursos intercambiar, como combinarlos, cómo se relacionan, y cómo y cuándo deben ser intercambiados. El único límite acerca de qué tipo de servidores se van a implementar y que características van a soportar está dada por un acuerdo entre las aplicaciones que se van a conectar.
En esta unidad exploraremos las opciones, y algunas técnicas avanzadas y buenas prácticas, y revisaremos al menos un ejemplo de cada paradigma – REST (incluyendo transacciones), mensajes, documentos y operaciones.

Paradigmas FHIR

FHIR soporta cuatro paradigmas de interoperabilidad: REST, Documentos, Mensajes y Servicios. Se habla también de dos paradigmas adicionales emergentes: Base de Datos o Storage y Transferencia Masiva (Bulk Transfer).

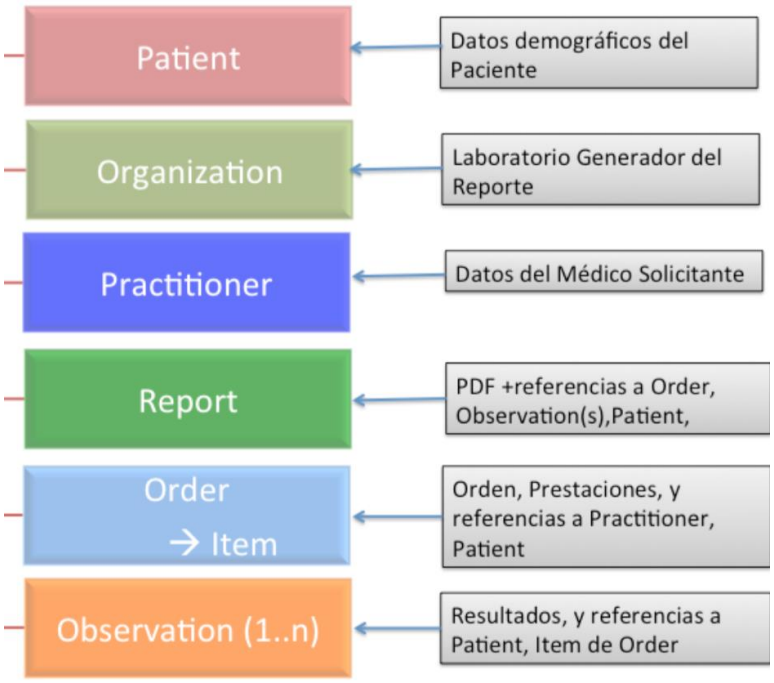
Revisaremos los seis enfoques, brindando herramientas para decidir cuál es más apropiado para cada proyecto que enfrente. No existe un ‘mejor’ enfoque. Hay enfoques ‘más apropiados’ para cada escenario, capacidad tecnológica o inclusive temas no técnicos como políticas o presupuesto.
Los cuatro paradigmas originales son: REST (+ transacciones), Documentos, Mensajes y Servicios (u operaciones).
Los dos paradigmas adicionales son: Almacenamiento y Transferencia Masiva

FHIR REST + Transacciones

El problema

Cuando intercambiamos recursos FHIR con un servidor en tiempo real, nos encontramos muy frecuentemente con el problema de enviar un grupo de recursos (relacionado o no), en oposición a lo habitual que es enviar un solo recurso.
Podemos elegir (como clientes FHIR) utilizar las interacciones REST habituales para cada recurso, o usar Transacciones FHIR.
Un ejemplo de esto es un reporte de laboratorio. Un reporte de laboratorio puede ser representado como una serie de recursos relacionados (paciente, médico solicitante, resultados, reporte en formato PDF, orden asociada). Si necesitamos enviar estos recursos a un servidor (asumiendo que se trata del mismo servidor en todos los casos) , podemos enviar los recursos de a uno, como puede verse en la Figura 1.

Figura 1. Intercambiar múltiples recursos implican múltiples viajes al servidor FHIR



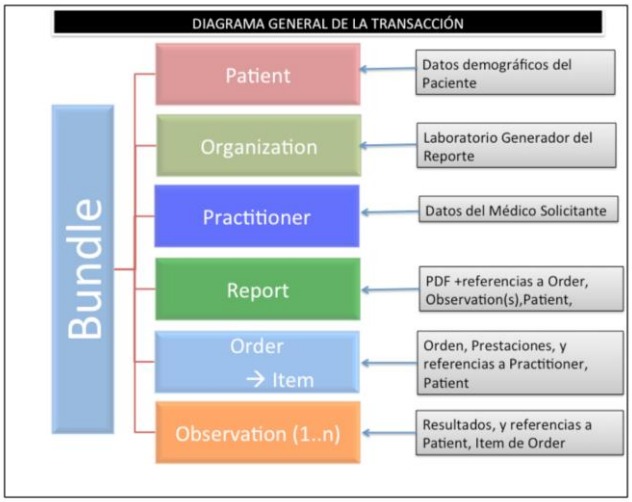
Las flechas bidireccionales en la Figura 1 significan, en el paradigma RESTful puro, asumiendo que no conocemos el identificador lógico del servidor para cada recurso (recurso.id), que debemos realizar una búsqueda, y luego realizar un POST (para crear el recurso) o PUT (para modificar el recurso) según sea necesario.
Nota: el nombre de la clase de recurso FHIR R4 para nuestra Orden es 'RequestGroup' y cada ítem es un 'ServiceRequest'

Ocurriría un diálogo parecido a éste: (las preguntas son 'búsquedas' / GET al recurso apropiado) y luego PUT/POST según sea necesario.

Cliente: ¿Tienes algún paciente con DNI 40928383?
Server: ¡Sí! Es mi Patient.id=338
Cliente: ¿Y conoces a la Organización 'Laboratorio Clínico', Identificador Nacional de Organizaciones INO # 333321?
Server: No, no tengo ninguna organización con ese número de INO
Cliente: Entonces agregar por favor, esta es la información: "Laboratorio Clínico, INO # 333321, Calle 3 entre 18 y 20, San Bernardo, Calafayún (Código postal: CA9201DA)'
…

Y así para cada uno de los recursos…
Imagine un reporte de laboratorio con 1 paciente, 1 organización, 1 médico solicitante, 1 orden con 10 ítems, 1 reporte con 40 analitos y un PDF adjunto….esto significa alrededor de 100 interacciones con el servidor (buscar…actualizar o crear), solamente para mandar UN reporte.
Las múltiples interacciones requieren múltiples viajes de ida y vuelta al servidor y además introducen el riesgo de la pérdida de integridad referencial si una interacción posterior falla (ejemplo: una entrada en un índice de documentos y su documento relacionado).
Otra opción es enviar todos los recursos agrupados (**bundled** : recordar la palabra mágica 'bundle') en una sola transacción: una transacción envía un conjunto de acciones a realizar en un servidor a través de una única solicitud/respuesta http, como se muestra en la figura 2.

Figura 2. Múltiples recursos agrupados en una sola transacción en un reporte de laboratorio



La responsabilidad por todo el procesamiento queda ahora del lado del servidor, incluyendo resolver los identificadores (como el DNI mencionado anteriormente) a identificadores lógicos locales, lo que se indica usando las cláusulas de 'Creación Condicional' o 'Actualización Condicional'.
El cliente indicará si desea que el procesamiento sea en el modo Lote ('batch') o Transacción ('transaction').
La única diferencia entre los modos es que el modo 'transaction' solamente será exitoso si TODAS las operaciones involucradas son exitosas, y una operación en modo 'batch' puede ser exitosa en forma parcial: algunas operaciones se podrán realizar y otras no.
Puede mezclar interacciones una transacción o lote, incluyendo múltiples interacciones en distintos tipos de recursos. Los servidores pueden inclusive definir sus propias operaciones (y lo veremos más adelante en esta misma unidad).
Las transacciones son útiles cuando necesitamos evitar múltiples interacciones.

Para realizar una transacción en un servidor FHIR, solo necesitamos efectuar un POST a la dirección base FHIR del servidor, en cualquiera de los formatos soportados (JSON/XML)
POST [base] {?_format=[mime-type]}
¿Recuerda la palabra mágica? La palabra mágica es '**Bundle**', porque una transacción, para FHIR es solamente un tipo específico de **Bundle**, con sus propias reglas.
El contenido o cuerpo del POST que se realiza al servidor es un recurso de tipo **Bundle.**
Para las transacciones, el elemento **Bundle.type** debe estar valuado como **"transaction"**, y para los lotes **"batch"**. Los elementos Bundle se componen de una o más entradas de tipo **entry**. Cada entrada en una transacción debe contener un recurso (**resource**) y una solicitud (**request**).
Los recursos son los **datos (**la información con el recurso que estamos enviando al servidor), la solicitud es la acción (que hacemos con los datos recibidos). Cada acción será procesada individualmente, y se generará una respuesta para cada acción (¿Se creó el recurso? ¿Cuál fue el resource.id asignado?)

Receta para Transacciones usando Bundle

Recuerde:

1. La transacción debe ser un **Bundle** de tipo "**transaction**", con una o más entradas (**entry**)
2. Cada **entry** debe contener un elemento **resource** y un elemento **request**.
3. El elemento **request** tiene dos elementos mandatorios:

method = "POST" / "PUT" / "DELETE" / "GET"
url= La dirección para el método y recurso, incluyendo también un parámetro de búsqueda de ser necesario (operaciones GET, DELETE, inserciones o actualizaciones condicionales)
En XML:

```
<Bundle>
  <type value="transaction"/>
  <entry>
        <resource>
            <Patient> <-- o cualquier otro ->
            ... contenido del recurso
            </Patient>
        </resource>
        <request>
            <method value="POST"></method>
            <url value="patient"></url>
        </request>
    </entry>
   ... (n Entradas)
</Bundle>
```

Reglas para procesamiento de transacciones

1. El servidor **aceptará todas las acciones** y devolverá una respuesta http 200 OK o rechazará todas las acciones y devolverá una respuesta http 400 o 500.
2. **No hay un orden requerido** para los recursos dentro de un Bundle para que sean procesados adecuadamente. El orden de procesamiento es el siguiente: **DELETE, POST, PUT, GET**. Así que el servidor tratará de eliminar todos los recursos que le solicitemos eliminar, luego creará todos los recursos nuevos, luego actualizará los recursos que deba actualizar, y finalmente procesará las búsquedas o GET directos (..recurso/id).
3. Una transacción puede incluir referencias de un recurso a otro en el mismo bundle. Algunas de estas referencias tienen un formato especial, que discutiremos en el apartado 'Interacciones Condicionales'

Respuestas a Transacciones

El servidor siempre retornará un Bundle (¡la palabra mágica nuevamente!) para que el cliente conozca los resultados de la transacción.
El tipo de Bundle será **"transaction-response"** con tantas entradas como acciones hayan sido requeridas por el cliente, en el mismo orden con el código de estado http, ubicación (nuevos identificadores lógicos de recurso asignados para las entradas), y ETag (información de versionado para cada recurso), el elemento response.
Tratemos de realizar un POST con una pequeña transacción y ver como es la respuesta del servidor y qué significa. Con este POST entenderemos 1) El formato y contenido de la respuesta del servidor y 2) Lo que el servidor realmente hace con los recursos agrupados en el bundle.
La transacción a enviar está en el archivo **SmallTransaction.xml**. Corte y pegue el contenido en su cliente REST y realice un POST al servidor FHIR de nuestro curso.
Recuerde que la dirección base de nuestro servidor FHIR es: http://fhir.hl7fundamentals.org/r4

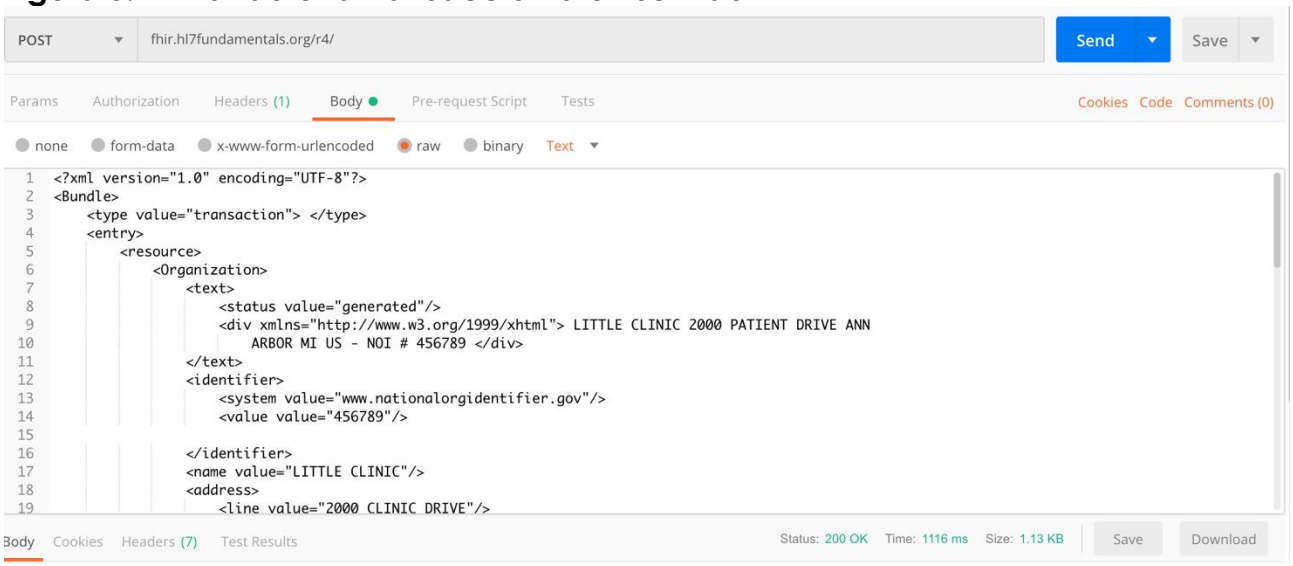
Las transacciones se envían a la dirección base FHIR (sin especificar recurso), asi que necesitará enviar el contenido del archivo a la dirección base FHIR de nuestr servidor.
Recuerde:

a- Fijar el encabezado para asegurar que el servidor entiende el formato del contenido o cuerpo: **Content-Type** : **application/fhir+xml**

b- Los identificadores de recurso que obtendrá serán diferentes de los mostrados en este documento.

La pantalla será parecida a la que tenemos en la Figura 3.

Figura 3. Enviando una transacción a un servidor FHIR



Revisemos la respuesta del servidor (en caso que ud. no pueda probarlo, un ejemplo de respuesta se adjunta en el archivo SmallTransactionAnswer.xml)

Y lo puede ver aquí mismo en las figuras Figures 4 and 5.

Figure 4. Encabezado de la Respuesta http del servidor para una transacción

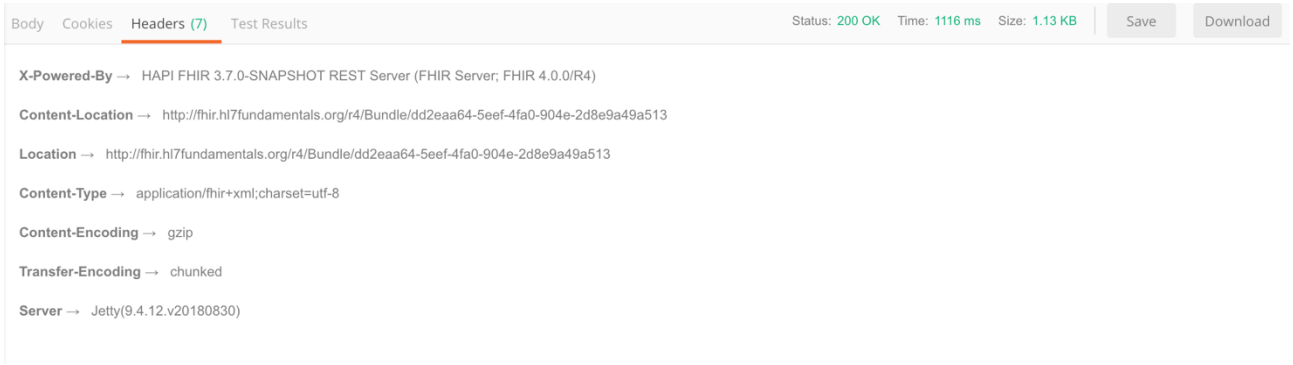


Figure 5. Cuerpo de la respuesta del servidor a la transacción

```
<Bundle xmlns="http://hl7.org/fhir">
    <id value="dd2eaa64-5eef-4fa0-904e-2d8e9a49a513"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Organization/17449/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:11:04.385+00:00"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Patient/17450/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:11:04.385+00:00"/>
        </response>
    </entry>
</Bundle>
```

Encabezado: hemos recibido un código de estado HTTP status code **200 OK**, así que todos nuestros recursos/acciones fueron procesados correctamente (recuerde que las transacciones fallan o tienen éxito en forma completa)
Cuerpo: hemos recibido un Bundle de tipo "transaction-response", con un elemento **entry** por cada elemento **entry** en nuestra transacción. La primera entrada nos indica que el servidor creó un nuevo recurso de tipo Organization ("201 Created"), con el identificador asignado por el servidor 556, y un nuevo recurso de tipo Patient, con el identificador asignado 557.
Ahora…en realidad queremos que los recursos estén RELACIONADOS. Necesitamos declarar que este paciente específico está siendo tratado por esta organización específica.
Esto puede conseguirse referenciando en el recurso Patient a través del elemento ManagingOrganization
Así que esto debería ser sencillo, simplemente agregar ManagingOrganization al recurso Patient
El problema es que ManagingOrganization requiere una referencia, y esta referencia debe ser a un recurso **Organization/id** en el mismo (u otro) servidor. Y no conoceremos la identificación interna del recurso hasta que no realicemos el POST de la transacción y obtengamos los resultados.
¿Así que estamos trabados? Afortunadamente no, la Identificación Temporaria de Recursos está allí para rescatarnos.

Identificación Temporaria de Recursos (ITR)

Este esquema especial de identificación se utiliza dentro de las transacciones para incluir una referencia a un identificador (asignado por un servidor) que todavía no conocemos.
El alcance de la identificación temporaria de recursos se limita exclusivamente al interior del Bundle de la transacción. Esto significa que se pueden reusar ITRs de una transacción a otra, pero no en la misma transacción.
Suponga que debemos enviar en el mismo bundle, el recurso Patient y el recurso DiagnosticReport, y que en DiagnosticReport hay una referencia al recurso Patient. ¿Cómo podemos hacer una referencia a un recurso cuya identidad aún no conocemos?
Por ejemplo, DiagnosticReport contiene un elemento 'Subject' que lo relaciona el reporte al paciente. Este elemento 'subject' requiere una referencia al recurso específico de paciente. Ejemplo: 'http://www.fhirserver.org/Patient/128'
Pero el originador de la transacción a) no conoce el ID específico que el servidor asignará a este paciente b) no quiere realizar más de una transacción con el servidor.
NOTA: Este no es un problema cuando el servidor permite a los clientes asignar su propio id a los recursos. Pero esto no siempre ocurre. Y no todos los servidores implementan la identificación temporaria de recursos. Pero una de las dos opciones estará disponible.

Hay una manera de incluir la solicitud de reemplazar el ID con el asignado por el servidor tanto en el recurso que contiene el ID como en sus referencias: identificación temporaria de recursos. El mecanismo es sencillo: simplemente incluya un GUID en la entry correspondiente al recurso referenciado, e incluya el mismo GUID en lugar de la referencia, como se puede apreciar aquí:

```
<<Bundle>
  <type value="transaction"/>
  <entry>
        <fullUrl value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>

        <resource>
           <Patient>

     ... Patient Resource contents
           </Patient>
        </resource>
        <request>
           <method value="POST"></method>
           <url value="patient"></url>
        </request>
    </entry>
    <entry>
        <DiagnosticReport>
          … other diagnostic report information
           <subject>
              <reference value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
           </subject>
         … more diagnostic report information
        </DiagnosticReport>
    </entry>
   ... (n Entries)
</Bundle>
```

El servidor creará los identificadores lógicos internos que sean apropiados para él, pero honrará las referencias entre los recursos, preservando de esta manera las relaciones entre los mismos. Así, volviendo a nuestro pequeño ejemplo, simplemente agregaremos un ITR a Organization, y lo referenciaremos desde el elemento ManagingOrganization del recurso Patient, como se aprecia en la Figura 6.

Figure 6. Identificación Temporaria de Recursos y Referencia

```
<?xml version="1.0" encoding="UTF-8"?>
<Bundle xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://hl7.org/fhir">
    <type value="transaction"> </type>
    <entry>
        <fullUrl value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
        <resource>
            <Organization>
                <text>
                    <status value="generated"/>
                    <div xmlns="http://www.w3.org/1999/xhtml"> LITTLE CLINIC 2000 PATIENT DRIVE ANN
                        ARBOR MI US - NOI # 456789 </div>
                </text>

                    <country value="US"/>
                </address>
                <managingOrganization>
                    <reference value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
                </managingOrganization>
                </Patient>

        </resource>
```

Puede ver esto en el archivo SmallTransactionLinked.XML
Puede también tratar de realizar un POST con este ejemplo a nuestro servidor también. Y examinar cuidadosamente la respuesta del servidor…ahora que sabe cómo leerla.

Examinemos la respuesta del servidor que presentamos en la Figura 7: el servidor no utilizó el GUID como identificador de recurso, lo reemplazó con el identificador interno que le asignó.

Figura 7. Respuesta de Servidor / Identificación Temporaria de Recursos

```
<Bundle xmlns="http://hl7.org/fhir">
    <id value="7bc3f27d-e86d-4738-a0db-300822b8db21"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Organization/17453/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:42:40.391+00:00"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Patient/17454/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:42:40.391+00:00"/>
        </response>
    </entry>
</Bundle>
```

El servidor creó dos recursos, y si recuperamos el recurso Patient (por favor hágalo ud. mismo), obtendremos el recurso Patient apropiadamente vinculado al recurso Organization que corresponde a su organización a cargo, como se ve en la Figura 8.

Figura 8. Referencia Correcta de una Identificación Temporaria de Recurso

```xml
<Patient xmlns="http://hl7.org/fhir">
    <id value="17454"/>
    <meta>
        <versionId value="1"/>
        <lastUpdated value="2019-04-14T15:42:40.391+00:00"/>
    </meta>
    <text>
        <status value="generated"/>
        <div xmlns="http://www.w3.org/1999/xhtml"> EVERYWOMAN EVE 2000 PATIENT DRIVE ANN
                ARBOR MI MPI #123456 </div>
    </text>
    <identifier>
        <system value="www.mypatientidentifier.com"/>
        <value value="123456"/>
    </identifier>
    <name>
        <family value="EVERYWOMAN"/>
        <given value="EVE"/>
    </name>
    <address>
        <line value="2000 PATIENT DRIVE"/>
        <city value="ANN ARBOR"/>
        <state value="MI"/>
        <country value="US"/>
    </address>
    <managingOrganization>
        <reference value="Organization/17453"/>
    </managingOrganization>
</Patient>
```

Pero…todavía hay un problema. Ud. puede haberlo notado. O no.
Cada vez que enviamos el mismo recurso (mismo paciente y organización) al servidor, lo crea nuevamente. Si utilizamos transacciones para nuestros reportes de laboratorio, tendremos cada reporte relacionado a un paciente, pero se generará un nuevo recurso paciente cada vez que grabemos un nuevo reporte (¡aun cuando ya exista un paciente con los mismos datos!)
El problema todavía continúa, aun utilizando transacciones e identificaciones temporarias. ¿Cómo podemos asegurarnos que no terminamos agregando el mismo recurso dos veces? ¿Estamos trabados nuevamente? ¿O necesitamos un nuevo superhéroe? ¡Interacciones condicionales al rescate!

Interacciones Condicionales

Las interacciones condicionales permiten a un cliente FHIR evitar la creación de un nuevo recurso basado en algún criterio de identificación distinto del id interno lógico del recurso. Para lograr esto, el cliente realiza un PUT como se muestra aquí :: PUT [base]/[type]/?[search parameters]
Para los nuevos registros, cuando el servidor procesa la interacción, realiza una búsqueda FHIR estándar para el tipo de recurso, con el objetivo de resolver a un solo identificador lógico para esta solicitud. La acción resultante dependerá de cuantos registros devuelve la búsqueda:
Ningún registro: El servidor realzará una interacción de creación de recurso (CREATE)
Un Registro: El servidor realizará la actualización del registro pre-existente.
Múltiples registros: El servidor devuelve un error http 412 Precondition Failed indicando que el criterio de búsqueda del cliente no fue lo suficientemente selectivo.
Esta variante funciona aún mejor con PUT para que pueda buscar un recurso pre-existente, pero por identificador de negocios (resource.identifier) en lugar de por el identificador asignado (resource.id), con el valor agregado de que además puede actualizar el recurso.
Por ejemplo un cliente puede actualizar el estado de un reporte de laboratorio de 'preliminar' a 'final' usando PUT /Observation?identifier=http://my-lab-system|123

Nota importante: el soporte para transacciones y creación/actualización/borrado condicional es complejo, y no se espera que todos los servidores los implementen. Los servidores que no soportan actualizaciones condicionales deben devolver un error http 400 y un recurso OperationOutcome explicando que la operación no está soportada.

Entonces, ¿cómo aplicar esto a nuestra pequeña transacción? ¿cómo asegurar que el servidor no creará el paciente y la organización cada vez que realizamos un POST con el paciente/y la organización dentro de una transacción?
Cambiemos un poco nuestra transacción. Los cambios están disponibles en el archivo SmallTransactionLinkedConditional.xml.
Observe que no estamos utilizando los mismos identificadores para la organización y el paciente ya que obtendríamos múltiples registros en la búsqueda y el servidor devolvería un error http 412 como se ha descripto previamente (puede intentarlo Ud. mismo).
El resultado de hacer el POST de este Bundle es que los recursos serán actualizados si ya existían, y serán creados si no existían. Y los identificadores temporarios serán reemplazados con el identificador asignado por el servidor para cada recurso. Obtendremos un estado http 200 OK para cada recurso.

```
<Bundle xmlns="http://hl7.org/fhir">
    <id value="f423a183-c128-48a5-a670-1046f44a95db"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="200 OK"/>
            <location value="Organization/17455/_history/1"/>
            <etag value="1"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="200 OK"/>
            <location value="Patient/17456/_history/1"/>
            <etag value="1"/>
        </response>
    </entry>
</Bundle>
```

Esto concluye nuestra discusión de transacciones FHIR, pero revisaremos más tipos de Bundle, relacionados con los paradigmas de interoperabilidad con mensajería, documentos y servicios.

Mensajería con FHIR

Conceptos básicos de Mensajería con FHIR

En el paradigma de mensajería, las interacciones entre clientes y servidores se asemejan a las definidas por la familia de estándares HL7 V2.x: hay un **emisor** y un **receptor** , hay un **evento disparador** (algo que ocurre en el mundo real que dispara la creación y comunicación de un mensaje específico) , un mensaje de solicitud , y un mensaje de respuesta.
El mensaje de solicitud FHIR es una transacción FHIR con algunas reglas especiales:

1. Un **mensaje FHIR es un Bundle** con Bundle.type=**message**, y el primer recurso es siempre un recurso especial denominado **MessageHeader**
2. El **MessageHeader** contiene los **metadatos** sobre el mensaje:

    a. **EventCoding** (elemento codificado): el evento disparador

    b. **Sender** (la aplicación que envía el mensaje)

    c. **Receiver** (la aplicación que debería recibir el mensaje)

    d. **Message Identifier** (identificador único para el mensaje, utilizado para auditoría y para asignarle la respuesta que podría ser asincrónica)

    e. **Date/Time (**Fecha y Hora de Creación del mensaje)

No hay necesidad que el transporte sea RESTful sobre http, se pueden enviar y recibir mensajes usando transferencia de archivos, carpetas compartidas, HTTP, MLLP, Websphere, RabbitMQ, o cualquier aplicación de encolamiento. El único requisito para el transporte es asegurar la entrega de los mensajes y sus respuestas.
Los mensajes FHIR se clasifican según su impacto:
Consecuencia: el mensaje representa un cambio que puede ser ejecutado sólo una vez. Este tipo de mensaje tendría solo UN mensaje de respuesta.
Notificación: El contenido de este mensaje es ‘actual’, y puede ser reprocesado. Este tipo de mensaje solo debería recibir UN mensaje de respuesta.
Respuesta a Consulta: Este mensaje es una respuesta a una consulta, y se supone que la información está actualizada. Siendo una consulta, este mensaje puede recibir múltiples mensajes de respuesta.

Lista de Eventos de FHIR

La lista de eventos de FHIR (lista de tipos de mensajes y eventos con una lista de los recursos agrupados para cada uno) está ubicada en: http://www.hl7.org/fhir/valueset-message-events.html
La lista actualmente es muy pequeña comparada con la variedad de mensajes y eventos definidos en HL7 V2.x and V3, principalmente debido al hecho que la mayoría de las implementaciones han seleccionado otros paradigmas (RESTful, transacciones, documentos, operaciones), pero está abierta a sugerencias.
Estos son algunos ejemplos de la lista y sus recursos asociados:
diagnosticreport-provide (notificación): proveer o actualizar in reporte diagnóstico: DiagnosticReport, Patient, Performer, Specimen, Image
MedicationAdministration-Update (consecuencia): actualizar un registro de administración de medicamentos
admin-notify (notificación): cambio en un recurso administrativo (Device, Location, Patient, Practiioner).

Identificadores en la Cabecera de un Mensaje

Cada mensaje contiene DOS identificadores, el elemento **Bundle.id** (obligatorio) y el elemento **MessageHeader.id** (también obligatorio). Estos identificadores deben ser únicos para el emisor, y es mejor aún que sean globalmente únicos, utilizando un OID o UUID. Cada vez que el mensaje se envíe, el elemento Bundle.id debe modificarse, pero el MessageHeader.id debe mantenerse.
Cuando un receptor procesa el mensaje, responderá con un nuevo Bundle, con un nuevo Bundle.id, conteniendo un nuevo mensaje, con un nuevo MessageHeader.id, y el MessageHeader.id en el elemento Messageheader.response.identifier

Receta para Mensajes FHIR

Recuerde,

1. La transacción debe ser un **Bundle** de tipo **value="message**", con al menos una entrada: MessageHeader y una o más entradas por cada recurso requerido para el tipo de mensaje.
2. El elemento **Bundle.id** es obligatorio y tiene que ser (globalmente) único.
3. El elemento **MessageHeader.id** es obligatorio y tiene que ser (globalmente) único. Será utilizado para relacionar la respuesta al mensaje con el mensaje.
4. No hay elemento **MessageHeader.timestamp asi que el Bundle.timestamp** es obligatorio y representa la fecha y hora de generación del mensaje.
5. El elemento **MessageHeader.source** es obligatorio y representa el **endpoint** (obligatorio) del emisor y su elemento **name** (nombre del emisor, opcional)
6. El elemento **MessageHeader.destination** es opcional y representa el **endpoint** (obligatorio) del receptor y su elemento **name** (nombre del receptor, opcional)
7. El elemento **MessageHeader.focus es obligatorio,** y apunta a la primera entrada o entrada focal, el primer recurso 'con contenido útil' del mensaje – recuerde que el recurso MessageHeader contiene implemente metadatos acerca del mensaje, no acerca del hecho real que aconteció - .

In XML:

```
<Bundle xmlns="http://hl7.org/fhir">
  <id value="a4b0eb3c-a3f3-4739-aef3-db9f718a0b15"></id>
  <type value="message"> </type>
  <timestamp value="2019-01-04T09:10:14Z"></timestamp>

    <entry>
        <resource>
            <MessageHeader>
                <id value="21448097-009c-4eec-b8d3-6aba818b3a74"></id>
                <eventCoding>
                    <code value="admin-notify"></code>
                </eventCoding>
                <destination>
                    <name>RECEIVING SYSTEM</name>
                    <endpoint value="www.receivingsystem.com"></endpoint>
                </destination>
                <source>
                    <name>SENDING SYSTEM</name>
                    <endpoint value="www.sendingsystem.com"></endpoint>
                </source>
                <focus>
                    <reference value="urn:uuid:3a78d9c4-5d87-4264-b3b4-
f4d0506a373c"></reference>
                </focus>
            </MessageHeader>
        </resource>
    </entry>
    <entry>
        <fullUrl value="urn:uuid:3a78d9c4-5d87-4264-b3b4-f4d0506a373c"></fullUrl>
        <resource>
        …
        </resource>
     </entry>

…any other entries
</Bundle>
```

Procesamiento de Mensajes

Los mensajes pueden ser procesados utilizando la operación $process-message, solamente mediante POST, o directamente a la dirección base FHIR. La operación acepta un mensaje, lo procesa de acuerdo a la definición del evento en la cabecera del mensaje y devuelve uno o más mensajes de respuesta. El servidor puede almacenar los recursos incluidos en la transacción, y posibilitar el acceso a los mismos, o realizar cualquier decisión de negocios basada en la información contenida en los recursos. El servidor puede devolver un código de estado http 200 OK si el mensaje fue procesado exitosamente.
Otros códigos de estado:
Errores 4xx: el error está en el contenido del mensaje. No reenviar.
Errores 5xx: el error está del lado del servidor. Por favor re-enviar.
Usualmente debería haber una respuesta por cada mensaje, conteniendo el estado detallado del procesamiento: el recurso usado para mostrar el estado del procesamiento es OperationOutcome

Receta para Respuestas a Mensajes FHIR

```
<Bundle xmlns="http://hl7.org/fhir">
    <id value="0522aa28-8cf1-4b83-a452-8f7164c76d72"></id>
    <type value="message"> </type>
    <timestamp value="2016-10-04T09:10:14Z"></timestamp>

    <entry>
        <resource>
            <MessageHeader>
                <id value="0522aa28-8cf1-4b83-a452-8f7164c76d72"></id>

                 <response>
                  <identifier value="efdd254b-0e09-4164-883e-35cf3871715f"/>
                   <code value="ok"/>
                   <details>
                    <reference value="OperationOutcome/94cee761-74c0-495c-82dd-
5e5686b0218e"/>
                   </details>
                </response>
             <source>
                    <name>SISTEMA RECEPTOR </name>
                    <endpoint value="www.receivingsystem.com"></endpoint>
                </source>
                <destination>
                    <name>SISTEMA EMISOR</name>
                    <endpoint value="www.sendingsystem.com"></endpoint>
                </destination>
                <data>
                    <reference value="urn:uuid: e7b49eb3-64d8-40d3-a013-
c60a8df5e897"></reference>
                </data>
            </MessageHeader>
        </resource>
    </entry>
    <entry>
        <fullUrl value="urn:uuid:e7b49eb3-64d8-40d3-a013-c60a8df5e897"></fullUrl>
        <resource>
          <OperationOutcome xmlns="http://hl7.org/fhir">
          <id value="94cee761-74c0-495c-82dd-5e5686b0218e"/>
         <text>
                <status value="generated"/>
                <div xmlns="http://www.w3.org/1999/xhtml">
                <p>…detalle de todos los errores …</p>
                 </div>
          </text>
           <issue>
                 <severity value="…"/>
                 <code value="…"/>
                 <diagnostics value="…"/>
          </issue>
          </OperationOutcome>
        </resource>
    </entry>

</Bundle>
```

Mapeo de Mensajes HL7 V2.x a Mensajes FHIR

Por cada recurso en la especificación FHIR hay una sección de mapeo donde se compara y mapea el recurso los campos, segmentos, mensajes y eventos disparadores relevantes en HL7 V2.x
Esto NO SIGNIFICA que seamos capaces de mapear automáticamente mensajes, segmentos, campos, y eventos HL7 V2.x a su equivalente en recursos o mensajes FHIR. No hay proyectos en HL7 que hayan intentado hacer esto, pero algunos proyectos intentaron cubrir la mensajería básica (ADT, ORU, ORM, etc.).
Estas son algunas interesantes lecturas y opiniones sobre la experiencia (en inglés):
Dr. David Hay (Nueva Zelanda) – blog post
http://fhirblog.com/2014/10/05/mapping-hl7-version-2-to-fhir-messages/
Rene Spronk (Holanda) – whitepaper
http://www.ringholm.com/docs/04350_mapping_HL7v2_FHIR.htm
Simonne Heckman (Alemania) - video
https://vimeo.com/128126357

Ejemplo de Mensajería FHIR

Este es un pequeño ejemplo de mensaje FHIR, en el archivo SmallMessage.xml, conteniendo información acerca de un paciente a través del evento **admin-notify**.
Solo unos pocos servidores de referencia soportan el paradigma de mensajería, pero puede postear el mensaje al Endpoint/Bundle de cualquiera de ellos y lo almacenarán. Puede ver este video de René Spronk en DevDays 2018 “Mensajería, El paradigma olvidado” si está interesado en mensajería con FHIR: https://www.youtube.com/watch?v=_ov5bYIRTpg

Documentos Clínicos con FHIR

Características de los Documentos Clínicos

Hay seis características principales definidas para los documentos clínicos en HL7 CDA R2: **Persistencia, Cuidado, Potencial para la autenticación, Contexto, Completitud y Legibilidad Humana.** Discutiremos primero estos conceptos aplicados al estándar FHIR.

Persistencia: El documento debe ser almacenado y permanecer asequible en su forma original (como fue firmado) por el período de retención legal. FHIR no requiere que los recursos sean almacenados tal como fueron enviados, pero esto es generalmente una expectativa para los documentos.

Potential para la autenticación: El documento puede ser firmado. En FHIR, la autenticación es del contenido completo, y de una vista específica del contenido incluido. FHIR brinda mejores herramientas que CDA para especificar firmas digitales (a través del elemento **signature**) y proveniencia de los distintos recursos (a través del recurso asociado **Provenance**)

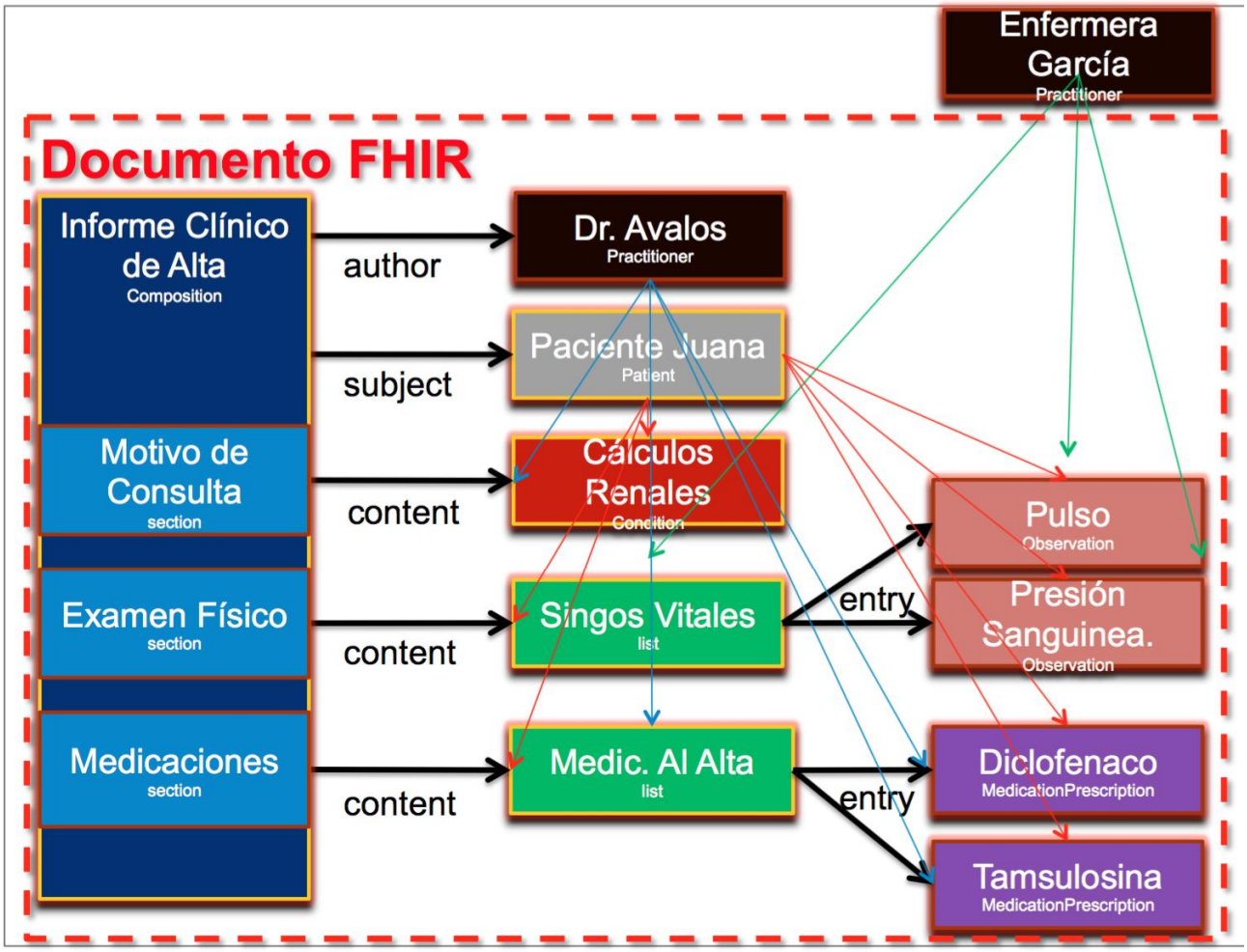
Contexto: En CDA R2, está dado por la cabecera de documento. EN FHIR, el contexto existe independientemente en cada recurso. Sin embargo, los documentos indican un grupo de datos que se prevé consumir en conjunto (contexto humano vs contexto técnico). Veremos esto cuando discutamos la estructura de los documentos: se consigue a través del uso del recurso **Composition**.

Completitud: la autenticación y el contexto aplican al contenido completo del documento, y no a partes específicas.

Legibilidad humana: La legibilidad humana es opcional para los recursos FHIR, pero obligatoria para los documentos, incluyendo reglas específicas de visualización.

Los documentos FHIR son más flexibles en su estructura que los documentos CDA R2, son una estructura genérica, pueden ser acerca de cualquier sujeto (todos los documentos CDA R2 son acerca de un paciente): diseño de una prueba clínica, reporte de un dispositivo, casos de salud pública, etc. Finalmente, los documentos pueden ser procesados de formas que no cumplen con las 6 características de los documentos clínicos definidos por CDA R2 (por ejemplo, podrías simplemente analizar los recursos incluidos y NO almacenar o mostrar el documento completo tal como fue enviado por el creador original del mismo). Esto no excluye a un proyecto específico de poder basarse en documentos FHIR y preservar las características de los documentos clínicos.

FIGURA 9 – Vista conceptual de un Documento FHIR



Estructura de documentos FHIR

El contenido que enviaremos (POST) a un servidor para un documento FHIR es también un recurso **Bundle**. El elemento **Bundle.type** debe ser valuado como **"document"**. El documento incluirá una lista de entradas (elementos de tipo **Entry**). Para los documentos, la primera entrada será un recurso Composition, y debe haber algunos otros recursos también obligatoriamente (no puede haber solamente un recurso Composition como único recurso del Bundle). Estos otros recursos deben estar incluidos en el Bundle (recomendado para mantener la completitud) o referenciados externamente desde el recurso Composition. Un documento, sin importar que tan complejo y anidado sea, se achata a una lista de entradas, con el encabezado o Composition como primera entrada. El encabezado del documento (y cualquier otro recurso) tiene referencias al resto de las entradas para reflejar la vinculación y anidamiento. Puede haber una firma digital (elemento **signature**) para todo el recurso Bundle, para autenticar el contenido del documento.

Receta para Documentos FHIR

Recuerde,

1. La transacción debe ser un **Bundle** con el elemento type con **value="document"**, con al menos un elemento entry que contiene un recurso Composition y una o más entradas para cada referencia interna (al menos una es requerida). Esta es una receta MINIMA, así que solo listaremos los elementos obligatorios, y sus referencias, pero en su escenario específico Ud. podría necesitar otros elementos y referencias adicionales a los descriptos aquí abajo.

Elementos del recurso Composition (atributos o metadatos del documento en sí):

2. El elemento **Bundle.id** : identificación del bundle, tiene que ser (globalmente) única.
3. El elemento **Composition.id :** tiene que ser (globalmente) única. Es el identificador del documento.
4. El elemento **Composition.date :** la fecha/hora de creación del documento.
5. El elemento **Composition.title :** título legible para los humanos para el documento
6. El elemento **Composition.code :** tipo de documento, del vocabulario controlado http://www.hl7.org/fhir/valueset-doc-typecodes.html (que refiere a todas las clases de documento LOINC)
7. El elemento **Composition.status :** representa el estado actual del documento. Opciones: "preliminary" (preliminar), "final" (final), "amended" (corregido), "entered-in-error" (ingresado por error)

Referencias de la composición (y entradas asociadas si se encuentran en el mismo Bundle)

8. Composition.subject: Sobre qué o quién versa la composición (usualmente es sobre el paciente)
9. Composition.author: Profesional, dispositivo, paciente o persona relacionada que crea el documento (pueden ser varios).
10. Composition.attester.party: Paciente o profesional que atestigua la exactitud de la composicion
11. Composition.custodian: Recurso Organization, que representa a la entidad que mantiene la composición
12. Composition.event.detail: El evento que se documenta
13. Composition.encounter: El encuentro clínico que brinda contexto a la composición.
14. Composition.section.entry: el contenido clínico del documento

Debido al tamaño del documento, no podemos incluirlo en este material de lectura, pero puede revisar el ejemplo para examinar su estructura.

Ejemplo de Documento FHIR

El archivo SmallDocument.xml es un ejemplo mínimo y semánticamene válido de documento FHIR.
Nuevamente, si desea probar este documento utilizando un servidor FHIR, necesitará utilizar este servidor (no todos soportan el paradigma de documentos):

http://fhir.hl7fundamentals.org/r4/Bundle

Presentación de Documentos FHIR

El documento debe ser presentado en este orden: Narrativa del recurso Composition, Narrativa del recurso Subject, Narrativas de los elementos section.text, utilizando el estilo básico para narrativas explicado aquí: http://www.hl7.org/fhir/narrative.html#css
Además, un documento puede referenciar o contener una o más hojas de estilo adicionales que contienen estilos que aplican a una narrativa específica.
Esto se logra incluyendo una refencia a la hoja de estilos en el recurso Bundle. La referencia puede ser un archivo existente en un servidor web o un recurso incluido en el mismo Bundle.

```
<Bundle xmlns="http://hl7.org/fhir">
<link>
 <relation value="stylesheet"/>
<url value="[uri]"/>
…
</Bundle>
```

Transporte de Documentos FHIR

Hay varias opciones para enviar documentos a un servidor FHIR, a través de distintos endpoints. Ninguno de ellas es obligatorio. Los documentos pueden intercambiarse en cualquiera de las maneras que prefiera (Sistema de archivos, FTP, USB, etc.)
***baseurl*/Bundle:** Esto funciona como un endpoint normal para manejar un tipo de recurso, pero trabaja con documentos completos. Específicamente, una operación de lectura devuelve un bundle, una actualización cambia un bundle y una búsqueda devuelve un bundle de bundles.
***baseurl*/Binary:** Almacena un documento entero como secuencia de bytes, y devuelve exactamente la secuencia cuando se la solicita. Usualmente esto se asocia con un recurso DocumentReference, asi las aplicaciones pueden encontrar y procesar el documento.
***baseurl* (Transacción):** Ignora el hecho que el bundle es un documento, y procesa todos los recursos como recursos individuales. Los clientes NO deben esperar que un servidor sea capaz de rearmar el documento exactamente como se lo envió. Aún si el servidor pudiera rearmar el documento, el resultado puede no estar en el mismo orden, y es muy probable que la firma digital fuera invalidada.

Consideraciones de Arquitectura de Documentos FHIR

Persistencia de Documentos: No hay un orden definido para el orden de las entradas en un documento más allá de la primera entrada, así que la transformación entre formatos XML / JSON y RDF puede producir diferentes órdenes. Aún con el proceso de canonicalización, las firmas pueden no ser válidas si los documentos se almacenan como sus partes constituyentes. Así, si le interesa la firma digital, almacene los documentos en recursos de tipo **binary.**

Firma Digital: Cualquier bundle (incluyendo documentos) puede tener una firma digital. Las firmas no son requeridas para usar documentos FHIR. Otras maneras de verificar la integridad son aceptadas también. Uno o más recursos en un documento pueden ser firmados también incluyendo el recurso Provenance.

Etiquetas y Seguridad: Las etiquetas (elementos **tag)**, y los metadatos de seguridad (elementos **security**) pueden aparecer en las entradas y a nivel de Bundle. Si se usan etiquetas de seguridad, debe establecerse una precedencia. Lo más seguro es 'aplicar siempre lo más restrictivo', pero puede haber cambios según las reglas de negocio locales. Las etiquetas dentro de las entradas de documento no pueden modificarse sin invalidar la firma digital del documento.

Documentos FHIR y CDA R2

Proyectos relacionados con FHIR en HL7 International

Hay dos proyectos principales relacionados con FHIR:

1) **CDA ON FHIR:** Proyecto que define como deben ser los “documentos clínicos” en FHIR. Incluye: Perfil para el recurso Composition (veremos que es un perfil la semana próxima), Mapeo del encabezado de CDA a Composition y un mapeo de las entradas de alto nivel a un recurso correspondiente en FHIR. Ver **https://www.hl7.org/fhir/cda-intro.html**

2) **C-CDA ON FHIR:** Proyecto para mapear entre C-CDA y su equivalente en FHIR, incluyendo perfiles. La visión a largo plazo es una nueva versión de C-CDA con equivalentes exactos votados en paralelo. Ver:

   http://wiki.hl7.org/index.php?title=C-CDA_on_FHIR

3) Para una comparación completa entre CDA R2 y FHIR, ver
   https://www.hl7.org/fhir/comparison-cda.html

Operaciones en FHIR

Operaciones extendidas en la API RESTful

Cuando no alcanza con el conjunto de interacciones comunes definidas por la API RESTful de FHIR (Crear, Leer, Actualizar, Borrar, Buscar), FHIR permite la definición de operaciones utilizando un paradigma de operaciones o servicios parecido al de RPC.

1. Se ejecutan operaciones nominadas, con sus parámetros de ingreso y de salida.
2. Las operaciones se utilizan:

    a. Cuando el servidor necesita desempeñar un rol activo en formular el contenido de la respuesta

    b. Cuando el propósito es causar efectos colaterales como la modificación de recursos existentes o la creación de nuevos recursos.

3. La especificación FHIR describe un entorno de operaciones liviano que extiende la API RESTful de la siguiente manera:

    a) Cada operación tiene un nombre

    b) Cada operación tiene una lista de parámetros de entrada y de salida.

    c) Los parámetros son recursos, tipos de datos, o parámetros de búsqueda.

    d) Las operaciones están sujetas a las mismas restricciones de seguridad y restricciones que la API RESTful.

    e) Las URIs para los puntos de entrada (endpoints) se basan en el esquema existente de direcciones de la API RESTful.

    f) Las operaciones pueden hacer uso del repositorio existente de recursos en sus definiciones.

    g) Las operaciones pueden ser realizadas sobre un recurso específico, un tipo de recurso, o un sistema completo.

Ejecutando Operaciones

Las operaciones se ejecutan mediante un POST a un punto de entrada FHIR, donde el nombre de la operación se prefija con el signo $ (“$”). Por ejemplo:
POST http://fhir.someserver.org/fhir/Patient/1/$everything
Las operaciones pueden ser invocadas en cuatro tipos de puntos de entrada:
1. **El punto de entrada “base”** (como http://fhir.someserver.org/fhir): Opera en toda la escala del servidor. Ejemplo “devolver todas las extensiones conocidas por este servidor”.
2. **Un tipo de recurso** (como http://fhir.someserver.org/fhir/Patient): Opera sobre todas las instancias de un tipo de recurso
3**. Una instancia de recurso** (como http://fhir.someserver.org/fhir/Patient/1): Involucra solo una instancia de un recurso. Ejemplo: la operación $everything.
4. **Una versión de una instancia específica de un recurso:**
(http://fhir.someserver.org/fhir/Patient/1/_history/4): Solo para permitir la manipulación de los metadatos profile y tag de versiones anteriores.

Parametros de una Operación

El cuerpo de la invocación contiene un recurso especial de infraestructura denominado **Parameters,** que representa una colección de parámetros nominados como pares <clave, valor>, donde el valor puede ser cualquier tipo de datos primitivo o complejo, o inclusive un recurso. TAmbien puede incluir cadenas alfanuméricas formateadas como tipos de parámetro de búsqueda.

Respuesta a Operaciones

Luego de completarse, la operación devuelve otro recurso Parameters, conteniendo uno o más parámetros de salida. Esto significa que una operación FHIR puede tomar cualquier conjunto de parámetros de entrada y devolver un conjunto de parámetros de salida. Tanto el cuerpo del POST como el resultado que se devuelve son siempre un recurso. Si la operación tiene éxito, se devuelve un código de estado http 200 OK. Un código de estado http 4xx o 5xx indica un error, y puede devolver además un recurso OperationOutcome.

Operaciones definidas por FHIR

La especificación de FHIR define varias operaciones, puede encontrar los detalles aquí:
http://www.hl7.org/fhir/operations.html#defined
Dos importantes operaciones que usaremos son $everything (aplica a un paciente específico) y $validate (valida un recurso contra su definición y/o un perfil específico). Veremos más acerca de perfiles en la próxima unidad.
Puede probar la operación $everything contra cualquiera de los recursos que ha subido a nuestro servidor FHIR, realizando un GET como este, reemplazando [id] por el identificador asignado al paciente:
http://fhir.hl7fundamentals.org/baseDstu2/Patient/[id]/$everything
Devolverá un Bundle con todos los recursos actuales almacenados en el servidor que pertenezcan al paciente.

Operaciones definidas por las Implementaciones

Cada implementación puede definir sus propias operaciones además de aquellas definidas por la especificación de FHIR. Esto es parte de la definición de perfiles en FHIR, y lo exploraremos la próxima semana.

¿Cuándo usar? ¿Cuándo evitar?

Cuando usar interacciones RESTful y transacciones FHIR

Interoperabilidad sencilla **– plug and play**
Aprovechar el protocolo http **que hace funcionar la web**
Usar solamente operaciones pre-definidas: crear, leer, actualizar, borrar **(CRUD)**, historia, leer versión, buscar, validar, conformidad y lotes.
Arquitecturas cliente-servidor con orquestación por el cliente **(móviles, PHRs, registros de pacientes/prestadores/profesionales)**

Cuando evitar las interacciones RESTful

Orquestación compleja o manejada por el servidor
La unidad de trabajo no es el recurso
No hay un servidor 'natural' para los recursos

Cuando utilizar mensajería FHIR

Flujo de trabajo de solicitud/respuesta
Necesidad de un modo asincrónico
Conducta más compleja **que CRUD**
Operaciones CRUD con transportes distintos a http

Cuando evitar la mensajería FHIR

Cuando hay otra manera estandarizada de lograr lo mismo

Cuando usar documentos FHIR

Cuando el foco está en la persistencia
No hay flujos de trabajo involucrados, solo guardar y recuperar documentos
Necesidad de reglas acerca del contenido autenticado
Necesidad de comunicar múltiples recursos con control sobre cómo se presentan los datos
Los datos se reparten en múltiples recursos

Cuando evitar los documentos FHIR

Necesidad de manejar el **flujo de trabajo**
Solicitud y Respuesta con **soporte a las decisiones**
Los datos son dinámicos: queremos ver la información actual, no cuando se generó un documento.
Múltiples autores a lo largo del tiempo
Se requiere **acceso independiente a los recursos**

Cuando usar operaciones o servicios FHIR

Para implementar **operaciones no-CRUD** sin la sobrecarga de la infraestructura de mensajería
Flujo de trabajo **más complejo que solicitud/respuesta**
Mezclar documentos y conducta

Cuando evitar los servicios FHIR

Si necesita control sobre como se muestra la información
Si desea evitar la pre-negociación de la conducta

Guía General

No hay absolutos. Esta sección es solo un resumen. Su propia opción puede estar influenciada por la preferencia y su legado tecnológico, y otras consideraciones no técnicas. No se limite a una sol opción: mezcle y ajuste según sus escenarios, casos de uso, y socios en la comunicación.

Paradigma/Aspecto	RESTful	TRANSACCION	MENSAJES	DOCUMENTOS	OPERACIONES
Uso de HTTP/CRUD	Solamente	Solamente	No limitado	No limitado	No limitado
Otras interacciones NO CRUD	No	No	w/Sobrecarga	No limitado	Si
Recurso como unidad	Si	Si	No limitado	No	No
Conducta/Flujo trabajo complejo	Manejado por el cliente	Manejado por el cliente	Solicitud/Respuesta	No	Complejo
Persistencia	No	No	No	Si	No
Contenido Autenticado	No	No	No	Si	No
Control sobre la presentación	No	No	No	Si	No
Multiples autores en el tiempo	Si	Si	SI	No	Si

Bonus Track: Dos paradigmas adicionales

Estos dos paradigmas son emergentes, de reciente desarrollo, pero es interesante comenzar a discutirlos y evaluar su utilidad: transferencia masiva de datos y almacenamiento.

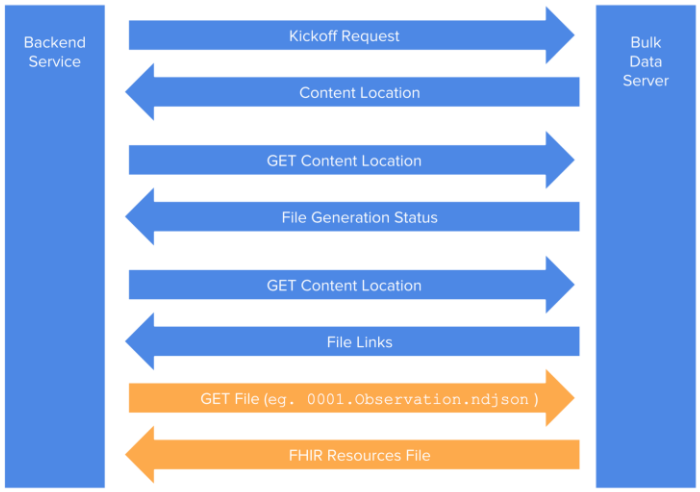
FHIR Bulk Transfer Data API

Resumen: Transferencia masiva de datos utilizando las APIs de FHIR. Es una especificación en estudio, que será probada durante lo que resta de 2018 y liberada como STU durante 2019.
Casos de Uso: Integración de sistemas de salud poblacional con historias clínicas electrónicas, machine-learning basado en datos de HCE, alimentación de repositorios internos para datawarehousing, envío de datos regulatorios (ejemplo: enfermedades transmisibles), operaciones realizadas con pacientes de un financiador específico.
Problema: usualmente el envío masivo de datos se hace a través de formatos propietarios o CSV, generando trabajo repetitivo y manual para mapeo. Se requiere una extracción de datos para cada caso de uso o escenario. Las APIs de FHIR son buenas para pequeñas consultas basadas en la información de un grupo limitado de pacientes (en general, uno) ya que son sincrónicas y paginadas.
Propuesta de Mejora a FHIR para soportar acceso masivo de datos: modelo estándar para simplificar parseo o mapeo, API estandarizada para iniciar la extracción de datos
Objetivos de Diseño: Usar tecnologías maduras y estables, limitar la cantidad de opciones de consulta y formatos, reusar la semántica de FHIR, estructurar para una generación y carga eficiente de grandes conjuntos de datos (operación asíncrona, un tipo de datos por archivo, streaming)
Seguridad: Utiliza autorización Smart-On-FHIR para back-end. Requiere registración previa, tokens firmados con clave privada de corto plazo un scope de lectura para los recursos
Formato de Representación:
El formato de representación es **ndjson** que permite transmitir una cantidad de recursos FHIR separados por \n (CHAR 10) . Cada línea es un recurso JSON válido.

```
{"id": "06eb35fc-09e6-48 ... "given": ["Lucille"],"family": "Bluth"}]}
{"id": "cf53f382-6eb6-4f ... "given": ["George", "Oscar"],"family": "Bluth","suff
{"id": "406a9c3e-50f9-4c ... "given": ["Michael"],"family": "Bluth"}]}
```

Más detalles sobre FHIR Bulk Data Transfer API: Puede obtener más detalles en
http://wiki.hl7.org/index.php?title=201801_Bulk_Data

Arquitectura
La arquitectura es un poco compleja, pero permite la operación asincrónica pero sin perder el control del proceso y su gestión por el servidor.



Kick-Off Request:
Un servicio de back-end solicita a un servidor FHIR bulk data la información que necesita
Invocación: **[fhir_base]/[recurso]/$export**
Parámetros de Consulta:
_outputFormat: formato de salida. El único formato soportado por ahora es ndjson
_since: fecha de última modificación de recurso
_type: tipos de recurso que se desean obtener, delimitados por coma
[group_id]: conjunto pre-negociado de recursos (empleador, cohorte de prueba clínica, miembros de un financiador específico)
Ejemplos
a. Todos los pacientes: [FHIR Server Base]/Patient/$export
b. Un grupo de pacientes: [FHIR Server Base]/Patient/[group_id]/$export
c. Se utiliza el encabezado de http: Prefer: respond-async

Kick-Off Response:
El servidor FHIR bulk data contesta aceptando el query y definiendo el lugar donde obtener el estado de la solicitud
El servidor deberá responder en caso de aceptar la consulta un código de estado http 202 Accepted, y el header Content-Location: [URL del end point donde consultar el estado del query]

Ejemplo:
HTTP Status: **202 Accepted**
Content-Location: www.fhir-server/fhir/bulk/status

Status Request:

El servicio de back-end solicita el estado de la operación a través de un GET al endpoint.

Ejemplo: GET www.fhir-server/fhir/bulk/status

Status Response:

El servidor FHIR Bulk da respuestas temporarias de porcentaje del proceso se ha completado en el encabezado de la respuesta http a través de un estado http 202 Accepted y los parámetros X-Progress con el estado y la cantidad de segundos para volver a solicitar el estado. Finalmente entrega un estado http 200 OK con un parámetro Expires que indica hasta que momento estarán disponibles los archivos, y el detalle de la ubicación de los mismos.

Ejemplo (mientras el proceso no concluye)
Status: **202 Accepted**
X-Progress: **"50% complete"**
Retry-After: **120**

Ejemplo (proceso concluido)
Status: **200 OK**
Expires: **Mon, 12 Mar 2018 23:59:59 GMT**

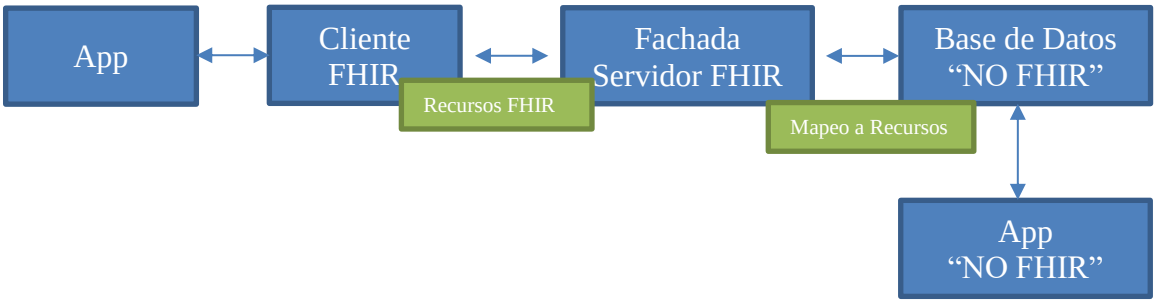
Status Response Body:

En el cuerpo de la respuesta http el servidor FHIR Bulk incluye el detalle del request, y la lista de archivos a recuperar a través de un GET directo.

```
{
   "transactionTime": "[instant]",
   "request" : "[base]/Patient/$export?_type=Patient,Observation",
   "secure"  : true,
   "output"  : [{
       "type" : "Patient",
       "url"  : "http://serverpath2/patient_file_1.ndjson"
   },{
       "type" : "Patient",
       "url" : "http://serverpath2/patient_file_2.ndjson"
   },{
       "type" : "Observation",
       "url" : "http://serverpath2/observation_file_1.ndjson"
   }]
}
```

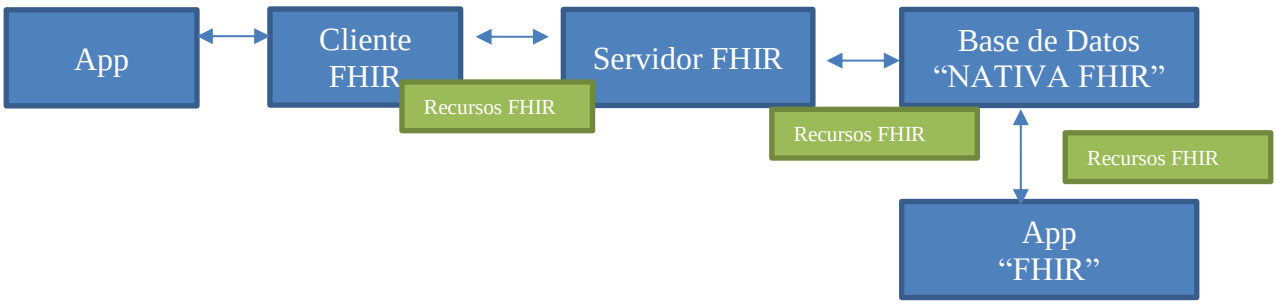
Paradigma FHIR de Base de Datos o Almacenamiento

FHIR es un estándar para INTERCAMBIAR INFORMACIÓN electrónicamente.
Por ende, la idea de utilizar los recursos intercambiados DIRECTAMENTE para almacenamiento y, finalmente como la base para una historia clínica electrónica o cualquier otro uso clínico, administrativo o estadístico, no es uno de los objetivos de su diseño.
Lo habitual, entonces, es que lo que esté basado en FHIR sea lo que está 'entre las aplicaciones'.



Sin embargo, además de las estrategias que comentamos en la primera unidad (FHIR como fachada de un servidor de datos legado o mapeando mensajes V2.x o en formatos propietarios, y, en general, como puente para la interoperabilidad moderna), en algunas implementaciones se han utilizado los recursos FHIR como tales para el almacenamiento de la información clínica y/o administrativa.
Normalmente esto se asocia con análisis de datos, pero no es necesario que sea con ese objetivo.
Los patrones más comunes son:

- Crear algún tipo de almacenamiento persistente (base de datos SQL o No-SQL con soporte para JSON/XML, gran repositorio de datos o RDF)
- Las aplicaciones contribuyen recursos a este almacenamiento
- Otras aplicaciones usan los datos en el almacenamiento más tarde con cualquier objetivo de negocios (análisis de datos, historia clínica compartida, facturación, etc.)
- Algunas de las tareas que deben realizarse para asegurar la calidad de los datos que se contribuyen son: Detección y prevención de duplicados y la resolución de referencias (tanto en la base de datos local como externa).



Cómo este paradigma es relativamente nuevo, todavía está en discusión si puede estandarizarse, y cuál es el efecto de los cambios de versión en el estándar sobre los datos que ya están almacenados (después de todo, están almacenados en un JSON con un esquema basado en una versión específica del estándar).

Para decidir si este paradigma es adecuado a nuestras necesidades específicas debemos evaluar si los requerimientos de nuestro sistema de información están claramente expresados y entendidos, y no pueden cambiar de manera incontrolada.
En este caso, diseñar un sistema de almacenamiento completamente adecuado es mucho más eficiente que almacenar recursos FHIR.

Por otra parte, si los requerimientos aún no están claros, y tenemos que almacenar y devolver cualquier recurso que nos envíen, los recursos FHIR son una manera muy adecuada y robusta, incluyendo la capacidad estructurada y documentada de extender los recursos.
En general, los sistemas de información se mueven entre estos dos extremos.

Lo que estamos viendo en general, es que los diseñadores **adoptan métodos mixtos o híbridos:** para alguna porción de la información utilizan un enfoque tradicional estructurado e indexado, y para lo que no pueden o no desean controlar, almacenan los recursos FHIR nativamente.

Uno de los problemas más complicados en este paradigma es como resolver las referencias: para poder ser de utilidad (específicamente para hacer agregación de datos, estadísticas, análisis, etc.) en el estado actual del arte, se necesita que las referencias estén completas en el servidor donde se realiza el análisis, esto implica 'desreferenciar' los elementos reference con urls o identifier y almacenar el dato real o al menos la información requerida en el servidor donde se analizará la información.

Para más detalles en la discusión de este paradigma, ver las dos entradas en el blog de Graham Grieve al respecto:

http://www.healthintersections.com.au/?p=2776
http://www.healthintersections.com.au/?p=2753

La actividad de esta semana

Trataremos de crear una transacción para un dispositivo wearable (falso) y enviar/recibir información sobre signos vitales a nuestro repositorio FHIR. ¡Vamos, ya! ¡Esta es la parte divertida!

Resumen de la unidad y conclusiones

FHIR no hará nada por Ud., pero permite la construcción de sistemas sólidos e interoperables. Esta unidad le brindó algunas pistas sobre los paradigmas de FHIR, y cuándo y cómo usarlos.
Trataremos de resumir en una página lo que leyó, por si alguien le pregunta en el ascensor lo que aprendió esta semana.

REST-ful: Operaciones CRUD solamente. Intercambiar información de a un recurso a la vez GET para saber si existe en el servidor, PUT/POST para actualizar o crear de ser necesario)

Transacciones: Bundle.type=**transaction**. Mezclar y ajustar cualquier tipo de recurso – relacionados entre sí-. Fallar o tener éxito por completo. Necesita algunos trucos cuando los identificadores lógicos de los recursos se desconocen (identificación temporaria de recursos, interacciones condicionales).

Lotes: Bundle.type=**batch**. Igual que las transacciones, pero pueden tener éxito parcial, y los recursos pueden no estar relacionados.

Documentos: Bundle.type=**document**. Primera entrada: Composition, con metadatos acerca del contexto de las entradas clínicas incluidas. Algunos atributos y referencias a entradas son obligatorios.

Mensaje: Bundle.type=**message**. Primera entrada: MessageHeader, con metadatos acerca del mensaje, emisor y receptor. Basado en el paradigma de eventos disparadores. Los eventos son definidos por HL7.

Operaciones: extienden la conducta de los servidores para operaciones complejas específicas. Comienzan con el signo $. Pueden aplicar al servidor completo, tipos de recurso o instancias de recursos específicas.

Guía de Cuándo Usar

Paradigma/Aspecto	RESTful	TRANSACCION	MENSAJES	DOCUMENTOS	OPERACIONES
Uso de HTTP/CRUD	Solamente	Solamente	No limitado	No limitado	No limitado
Otras interacciones NO CRUD	No	No	w/Sobrecarga	No limitado	Si
Recurso como unidad	Si	Si	No limitado	No	No
Conducta/Flujo trabajo complejo	Manejado por el cliente	Manejado por el cliente	Solicitud/Respuesta	No	Complejo
Persistencia	No	No	No	Si	No
Contenido Autenticado	No	No	No	Si	No
Control sobre la presentación	No	No	No	Si	No
Multiples autores en el tiempo	Si	Si	SI	No	Si