

```
"resourceType" : "Patient"
"text" : {
  "status" : "generated"
  "_status" : {
    "id" : "12344"
  },
  "div" : "<div
},
"identifier"
{
  "use"
  "ty"
```

FHIR FUNDAMENTALS COURSE

**FAST
HEALTHCARE
INTEROPERABILITY
RESOURCES**



Curso de FHIR, Unidad 2:

Operaciones y consultas a un servidor FHIR

Material de Lectura

Tabla de contenido

1.	Contenido de la unidad y Objetivos.....	3
2.	Rest.	4
3.	Seguridad y Auditoria	8
	Paradigmas de Interoperabilidad	8
4.	Trabajar con recurso paciente en FHIR.	10
	Crear un nuevo recurso.....	10
	Leyendo un recurso existente.....	12
	Actualizando un recurso	14
	Borrar un recurso.	16
5.	Búsquedas en FHIR.....	17
	Recuperar un recurso	20
	Parámetros de búsqueda Standar	21
	Tipos de Parámetros de búsqueda.	22
	Modificadores.....	23
	Prefijos	24
	Búsquedas a través de recursos.....	26
	Paginamiento	26
6.	Resumen de la unidad y conclusiones.....	28

1. Contenido de la unidad y objetivos.

En esta unidad nos centramos en la primera interacción con un server FHIR, los conceptos básicos de REST y vamos a ejecutar distintas operaciones con recursos.

La idea es crear, recuperar y actualizar recursos y aprender como consultar al servidor de diferentes formas utilizando diferentes parámetros.

2. Rest.

En esta primera parte vamos a echar un vistazo a lo que es REST en el contexto de FHIR.

La especificación de FHIR detalla cómo funciona REST y FHIR juntos y es lo que abarcaremos con más detalle.

FHIR trabaja con diferentes paradigmas de interoperabilidad, pero REST, está establecido a esta altura, por lejos el de más fácil uso y realmente lo van a ver. Es actualmente usado por compañías como Google, Amazon y Twitter para proveer acceso a sus servicios.

A pesar de lo complicado de su nombre, en realidad es verdaderamente simple, de hecho es cómo funciona la web hoy.

El término REST fue usado por un tipo poco conocido llamado Roy Fielding es uno de esos nerds inteligentes que definen muchas de las cosas y de los protocolos web que disfrutamos actualmente. En una disertación de doctorado, él tomó algunos conceptos básicos de la web y los describió como un “estilo de arquitectura” que luego fue tomado en una forma fácil y simple de implementar.

Lo básico es realmente simple.

REST está construido en la parte más alta del protocolo http. Es el mismo protocolo que se usa cuando solicitan una página web en un servidor http (la versión corta para hipertext transfer protocol)

Es simplemente un conjunto de reglas que tanto el servidor como el cliente, en este caso el browser, usa para intercambiar datos en la web. Http es un esquema de pedido / respuesta. Uno hace una consulta que puede contener información y el server responde, eso es todo.

REST está desarrollado y pensado en términos de recursos, cada uno de los cuales está identificado unívocamente en cada servidor y de esta forma puede ser trabajado o manipulado.

Esto es el URI (Uniform Resource Identifier), y que unívocamente especifica la ubicación del recurso. Esto es particular y específico para cada servidor.

Cuando trabajamos con recursos, hay un número fijo de métodos que se pueden ejecutar.

Los que realmente nos interesan en FHIR incluyen:

GET. Es para recuperar un recurso en particular basado en su identificador, (en la ubicación en el servidor y el ID). FHIR usa GET tanto para traer un único recurso como para ejecutar consultas al servidor que posiblemente retorne muchos recursos.

POST. Este método se usa para crear nuevos recursos en un servidor específico. En REST el servidor le va a asignar un id, y esa combinación se transforma en el identificador único para ese recurso.

PUT. Este método es para actualizar un recurso ya existente. El cliente que ejecuta esta operación necesita saber el identificador del recurso. En FHIR creará una nueva versión del recurso.

DELETE. Este método es para borrar un recurso. Otra vez, necesitamos saber cuál es el identificador único de ese recurso. Es recomendación de FHIR guardar una copia de este recurso en el servidor y que sea ubicable de alguna forma para accederlo. De todas formas, si uno ejecuta directamente un GET de ese recurso, no debería retornar a aquellos que fueron borrados.

OPTIONS. Este es un método “especial” en FHIR que está dirigido a la raíz del servidor. Devuelve el conformance de cada recurso. Esto significa cuales son las capacidades que tiene el servidor. Veremos esto en unidades siguientes.

Cuando un servidor responde a un pedido REST, procesa la llamada de acuerdo al método y a su inteligencia / regla de negocio interna, y le responde al cliente. La respuesta tiene diferentes partes que detallamos:

En el cuerpo de la respuesta está, en general, lo que el cliente está pidiendo del servidor. Mucho de esto siempre es en respuesta a un GET. La mayoría de los otros métodos no retornan una respuesta.

El server siempre revuelve el código de estado, que le informa al cliente cual fue el resultado de esa operación. En la respuesta se indica si la operación fue correcta o si falló, con algunos detalles adicionales acerca del problema de la operación.

El código de estado es un número, el cual está muy bien definido cada uno de los significados en el protocolo http. FHIR define un recurso para esto que se llama `operationOutcome`, que el servidor usa para darle al cliente más información acerca de la falla.

Tanto el pedido como la repuesta contienen cabeceras. Esto le da información extra acerca del pedido o de la repuesta para ayudar al server o al cliente a procesar apropiadamente la transacción.

Ejemplo de esto pueden ser:

- Tanto la pregunta como la respuesta pueden usar `content-type` en la cabecera para decirle a la otra parte si el recurso está en XML o JSON.
- El cliente va a usar el `acceptHeader` para decirle al servidor en que formato (XML o JSON) preferirá que vuelva la respuesta.

- El server usará el locationHeader para indicarle al cliente cual es la ubicación de ese nuevo recurso creado.

Hay un buen tutorial acerca de REST en esta dirección: <http://rest.elkstein.org>

Para introducción es suficiente, hagamos algo con FHIR.

Recuperemos un recurso paciente de uno de nuestros servidores tester en línea.

Copie el siguiente comando (que es la ubicación del recurso) en su navegador:

`http://fhir.hl7fundamentals.org/r4/Patient/836`

Lo que acaba de hacer con esta instrucción es decirle a su navegador que haga una operación GET para recuperar un identificador específico y mostrarlo. En este caso usted debiera ver la versión JSON del recurso paciente cuyo id es 836.

Desarmando la instrucción veremos

- `http://` es el protocolo
- `fhir.hl7fundamentals.org/r4` es el punto de entrada al servidor FHIR y donde responde las operaciones.
- `/Patient` Determina que la operación se realiza sobre el recurso paciente.
- `836` es el id del recurso

Un par de observaciones:

Para recuperarlo, se necesita conocer cuál es el ID.

Esto es bastante seguro en este caso porque es nuestro servidor y con recursos establecidos, y al menos que alguien borre estos recursos, debería estar disponible. (Pruebe otro número, cambie Id y vea que pasa).

Se podría usar esta técnica para recuperar cualquier tipo de recursos de cualquier servidor FHIR, obviamente sabiendo el ID, y conociendo los mecanismos de seguridad que tenga ese servidor.

La única operación que puede hacer a través de su navegador es la operación GET, recuperar recursos. En realidad, el servidor envuelve el recurso en una interfaz

de usuario. Veremos cómo hacer consultas desde un cliente como PostMan en los siguientes pasos.

3. Seguridad y Auditoria

El tema de seguridad en FHIR todavía está en activo desarrollo, y en este aspecto IHE es que está contribuyendo significativamente en esto. FHIR en si mismo no define una funcionalidad en cuanto a seguridad, pero depende de otros servicios para proveer lo que se necesita. Actualmente OAuth es uno de los más importantes.

Usar https es opcional, pero todos los intercambios de información en salud deberían usar SSL y todos los métodos de seguridad apropiados. Muchas de las operaciones requerirán usar autenticación de usuario y otras pueden depender de algún consentimiento apropiado y permisos.

La especificación de cómo implementar el tema de seguridad puede variar significativamente de acuerdo a cual sea el paradigma que vamos a usar. Rest, mensajes, documentos o servicios.

Hay unos cuantos recursos que se pueden usar para registrar las operaciones que requieren auditoria.

Esto incluye :

Provenance. Este recurso se usa para indicar desde donde viene un recurso en particular.

SecurityEvent. Son recursos equivalentes a los que usa IHE en el perfil de ATNA para registros de auditoria.

Paradigmas de Interoperabilidad

Es esperable que los mismos recursos de FHIR se usen en todos los distintos paradigmas de interoperabilidad en salud. Por ejemplo, el recurso paciente es el mismo no importa cómo se han ido intercambiando.

Estos paradigmas incluyen otros además de REST.

Mensajes

Un mensaje se usa cuando se quiere enviar información de un sistema a otro y espera que quien lo reciba actualice la información que recibe y después simplemente borre el mensaje. No es necesario guardarlo (solamente se guardan por un tema de auditoria). Ejemplo de esto HL7 versión 2 que es todo acerca de mensajería

Documentos

Un documento es registrar todo el conjunto de información de atención que aplica en un momento del tiempo a un paciente. Por ejemplo una epicrisis, un re-

sumen de una internación o una interconsulta. CDA trata con documentos. FHIR también tomará el tema de documentos y los describiremos en las unidades siguientes.

Servicios

El servicio está destinado para usarse en información en tiempo real, pero la diferencia con respecto a REST es que el servicio puede incorporar un flujo de trabajo más complejo que la simple interfaz REST. Por ejemplo podría usarse para solicitar una orden y hay que darle a ese servicio conceptos básicos de decisiones. En base a eso, la orden se puede ejecutar, modificar, autorizar o ser rechazada.

También el uso de servicios lo veremos más adelante

4. Trabajar con recurso paciente en FHIR.

Iremos un poco más en profundidad y empezaremos a manipular un simple recurso en FHIR. Vamos a crear un nuevo recurso, recuperarlo y luego actualizarlo.

La primera tarea será establecer un par de herramientas que necesitaremos para trabajar. Hay dos herramientas que necesitaremos para hacer esto de manera apropiada.

Algo para enviar y recibir consultas http. Esto nos permitirá especificar el contenido, el método y establecer las cabeceras. También la necesitaremos para ver la respuesta del servidor el StatusCode, y cualquier respuesta en el cuerpo que el servidor devuelva.

Alguna herramienta para crear y ver XML o recursos JSON. Se puede descargar los esquemas FHIR desde la especificación (hay un link dentro de los recursos del curso).

Dependiendo del entorno, y de su sistema operativo etc. hay diferentes opciones para hacer estas actividades.

Para hacer las cosas simples les digo lo que usaremos para ejemplificar. Una extensión de Chrome (PostMan) para hacer las consultas http, y el editor XML Oxygen para trabajar con JSON y XML.

Puede elegir la que más le sirva. También se puede usar notepad++, (busque en google notepad plus plus y encontrará el sitio, es gratis). Y si no tiene chance de instalar ninguna herramienta puede realizar estas actividades a través del sitio web en el cual está realizando el curso.

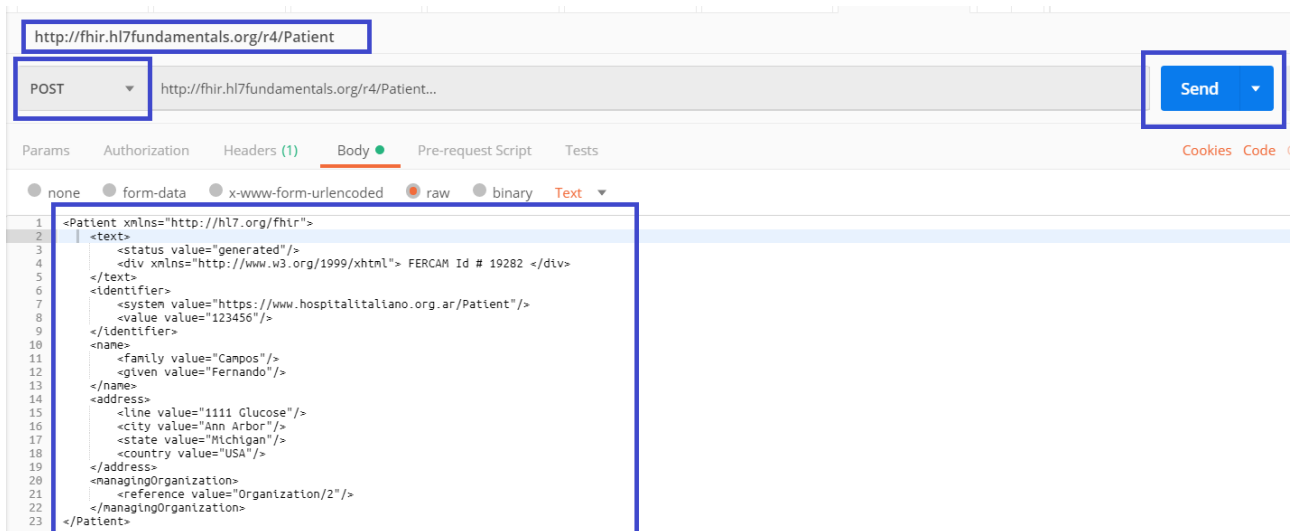
Crear un nuevo recurso.

Ya ha creado un recurso Paciente en la primera unidad y lo que haremos ahora será enviar ese recurso a un servidor FHIR.

Esto es muy simple y lo vamos a hacer ahora:

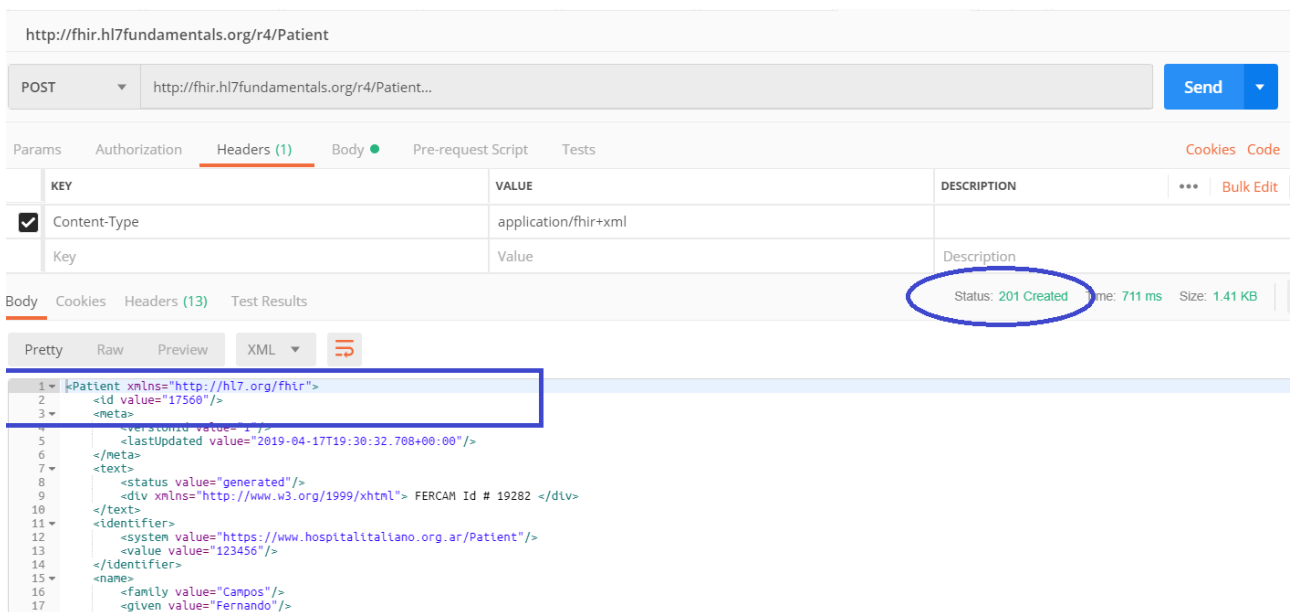
Deberá usar el método POST y enviar el recurso que crearon a la raíz del recurso en el servidor (puede usar cualquier servidor de prueba incluso el nuestro)

Lo verán simple en la siguiente captura de pantalla.



Si quiere el recurso en formato JSON, necesitará cambiar en el header, el content-type a application/JSON en lugar de application/XML.

El servidor va a procesar el pedido y va a responder. En esta ocasión el Status Code será de 201 indicando que el nuevo recurso fue creado y que le fue asignado un ID en el servidor al recurso.



Recuerde el ID ya que lo usaremos para recuperar el recurso.

El actual identificador para el recurso estará ubicado en el content-Location.

POST ▼ http://fhir.hl7fundamentals.org/r4/Patient... Send ▼

Params Authorization **Headers (1)** Body ● Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/fhir+xml	
Key	Value	Description

Body Cookies **Headers (13)** Test Results Status: 201 Created Time: 711 ms Size: 1.41 KB

Date → Wed, 17 Apr 2019 19:30:33 GMT

X-Powered-By → HAPI FHIR 3.7.0-SNAPSHOT REST Server (FHIR Server, FHIR 4.0.0/R4)

ETag → W/"1"

Content-Location → http://fhir.hl7fundamentals.org/r4/Patient/17560/_history/1

Last-Modified → Wed, 17 Apr 2019 19:30:32 GMT

Location → http://fhir.hl7fundamentals.org/r4/Patient/17560/_history/1

Content-Type → application/fhir+xml; charset=utf-8

¿Puede un cliente crear un ID además del servidor?

Es posible en algunos servidores que permiten la asignación de ID. Si el cliente ejecuta un PUT de un recurso y no hay un recurso con ese ID, el server puede crear el recurso y asignárselo. En general (y en nuestro caso), el servidor no permite que uno le asigne el ID por posibles problemas de colisiones.

Leyendo un recurso existente.

Para leer un recurso existente se debe conocer el identificador de ese recurso. Debe usar GET como método.

En este caso usaremos un servidor público. Como cualquiera puede actuar sobre él, las respuestas pueden ser diferentes, pero aquí mostramos como hacerlo usando PostMan.

GET ▼ http://fhir.hl7fundamentals.org/r4/Patient/17560 Send ▼

Params Authorization Headers Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (12) Test Results Status: 200 OK Time: 705 ms Size: 1.28 KB

Pretty Raw Preview JSON ≡

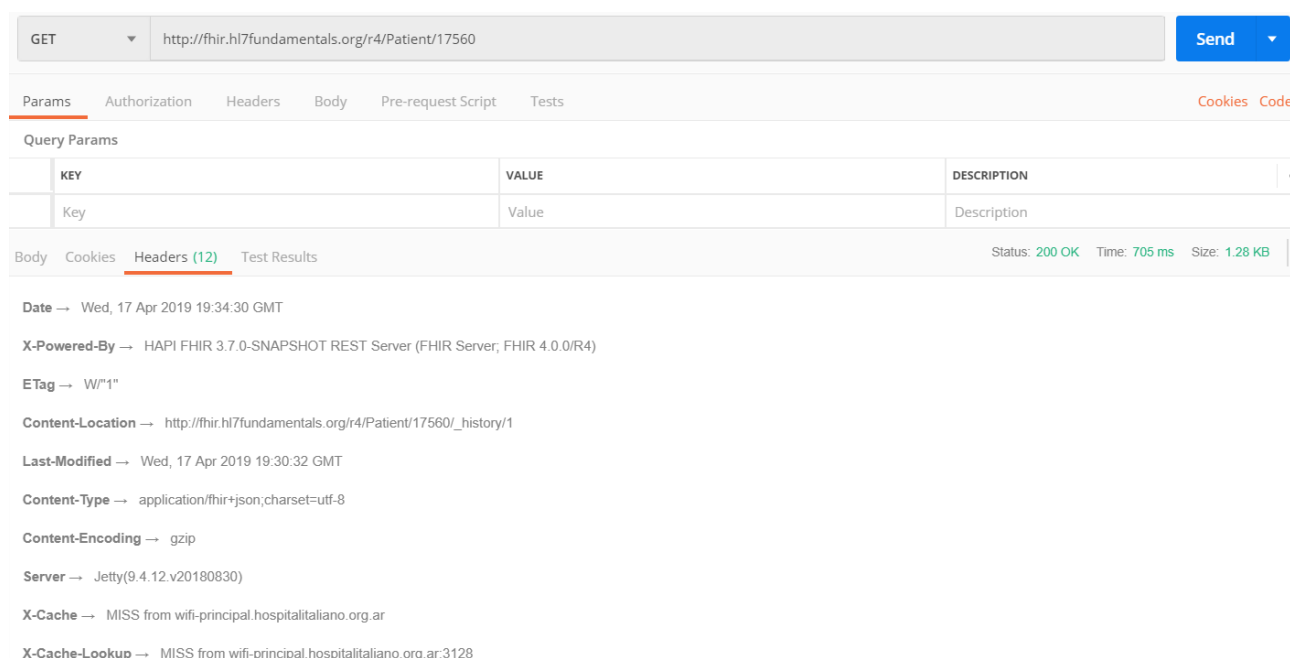
```

1 {
2   "resourceType": "Patient",
3   "id": "17560",
4   "meta": {
5     "versionId": "1",
6     "lastUpdated": "2019-04-17T19:30:32.708+00:00"
7   },
8   "text": {
9     "status": "generated",
10    "div": "<div xmlns='http://www.w3.org/1999/xhtml'> FERCAM Id # 19282 </div>"
11  },
12  "identifier": [
13    {
14      "system": "https://www.hospitalitaliano.org.ar/Patient",
15      "value": "123456"
16    }
17  ]
18 }
```

Note que:

- La ubicación completa del recurso en el servidor es :
<http://fhir.hl7fundamentals.org/r4/Patient/17560>
- Podría también usar lo que recibió en el Content-Location cuando lo creó. Pruebe !!!
- Le pedimos al servidor que devuelva el recurso en formato XML, poniendo en el encabezado del pedido `accept to application/xml+fhir`.
- Podría haber seteado este valor como `accept to header application JSON +FHIR` para recuperar el recurso en formato JSON. Pruébenlo.
- El código de retorno fue 200, lo que significa que estuvo todo bien.

Usando PostMan también pueden mirar la respuesta del headers.



GET <http://fhir.hl7fundamentals.org/r4/Patient/17560> Send

Params Authorization Headers Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (12) Test Results Status: 200 OK Time: 705 ms Size: 1.28 KB

Date → Wed, 17 Apr 2019 19:34:30 GMT

X-Powered-By → HAPI FHIR 3.7.0-SNAPSHOT REST Server (FHIR Server, FHIR 4.0.0/R4)

ETag → W/"1"

Content-Location → http://fhir.hl7fundamentals.org/r4/Patient/17560/_history/1

Last-Modified → Wed, 17 Apr 2019 19:30:32 GMT

Content-Type → application/fhir+json;charset=utf-8

Content-Encoding → gzip

Server → Jetty(9.4.12.v20180830)

X-Cache → MISS from wifi-principal.hospitalitaliano.org.ar

X-Cache-Lookup → MISS from wifi-principal.hospitalitaliano.org.ar:3128

Notas

- El content-type de la cabecera nos dice que el formato es FHIR+JSON.
- Hay un montón de otros elementos de información útil en la cabecera pero no entraremos en detalles en este momento.

Actualizando un recurso

Actualicemos un recurso y veamos qué pasa. Haremos una copia de este recurso el cual recuperamos del servidor en formato XML. Lo copiamos en nuestro editor XML, efectuamos algunos cambios en los elementos y lo volvemos a poner en PostMan. Luego ejecutamos el PUT sobre el servidor.

Siempre es recomendable validar la actualización del recurso utilizando el esquema de FHIR antes de enviarlo al servidor.

Vamos con un PASO a PASO

Paso 1 copie el recurso que acaba de bajar.

Paso 2 Pegue el recurso en su editor XML.

Paso 3 cambie el archivo por ejemplo, podría agregar un nuevo nombre o cambiar por sus datos personales.

```
1    ...
2    <!-- Original Name -->
3    <name>
4        <use value="official"/>
5        <family value="Fernando"/>
6        <given value="Campos"/>
7    </name>
8    <!-- New Name -->
9    <name>
10       <use value="usual"/>
11       <family value="Fercam"/>
12       <given value="Campos on FHIR"/>
13    </name>
14    ...
```

Paso 4 copie el recurso que ha cambiado.

Paso 5 en PostMan cambie el método GET a PUT y pegue el recurso que ha actualizado en el cuerpo de la consulta donde dice el **raw** es el mejor lugar.

Paso 6 presione el botón Send.

Si todo fue bien el servidor debiera haber aceptado los cambios del recurso y debe haber retornado como respuesta un código 200. En el content location, en la cabecera debería mostrar que la versión se ha incrementado en uno.

Esto es lo que tendremos.

PUT `http://fhir.hl7fundamentals.org/r4/Patient/17560` Send

```

9   "status": "generated",
10  "div": "<div xmlns='http://www.w3.org/1999/xhtml'> FERCAM Id # 19282 </div>"
11
12  },
13  "identifier": [
14    {
15      "system": "https://www.hospitalitaliano.org.ar/Patient",
16      "value": "123456"
17    }
18  ],
19  "name": [
20    {
21      "family": "Canpos",
22      "given": [
23        "Fernando Andres"
24      ]
25    }
26  ],
27  "address": [
28

```

Body Cookies Headers (12) Test Results Status: 200 OK Time: 706 ms Size: 1.28 KB

Pretty Raw Preview JSON

```

1  {
2    "resourceType": "Patient",
3    "id": "17560",
4    "meta": {
5      "versionId": "2",
6      "lastUpdated": "2019-04-17T19:41:27.124+00:00"
7    },
8    "text": {
9      "status": "generated",
10     "div": "<div xmlns='http://www.w3.org/1999/xhtml'> FERCAM Id # 19282 </div>"
11   },
12   "identifier": [

```

Notas:

- La especificación no requiere que la versión sea exactamente secuencial a pesar de que en general los servidores lo hacen.
- Otros usan algoritmos diferentes. En realidad lo que importa es que siempre haya una nueva versión del recurso.
- Nosotros usamos PUT y no POST, porque conocemos la ubicación exacta, el ID del recurso y el servidor en el cual está. Si hubiésemos hecho POST estaríamos diciendo que queríamos crear un nuevo recurso como ya hicimos.
- Si hacemos un PUT de un recurso único y no existía ese recurso en el servidor, el código de estado a devolver sería 201 en lugar de 200.
- Siempre puede recuperar una versión específica, usando la ubicación exacta desde la última actualización del recurso y usando el comando History.

`http://fhir_test_server/fhir/Patient/2/_history/3`

- También puede obtener una lista de todas las versiones de un recurso ya existente agregando al final término History, por ejemplo.

`http://fhir_test_server/fhir/Patient/2/history`

- Note que esto va a devolver una lista de recursos en un bundle. Cubriremos bundle cuando hagamos consultas.

Practicamos como leer, crear y actualizar un recurso paciente. Tienen que estar bastante entrenados para realizar la tarea, así que pueden trabajar con los recursos que han creado en su editor y practicar. También pueden trabajar con otro tipo de recurso.

Borrar un recurso.

No vamos a practicar como borrar un recurso pero simplemente tiene que usar el comando DELETE.

```
DELETE <host>/<resourceType>/<id>
```

Asumiendo que el borrado fue correcto, el server devolverá un StatusCode 204. Si el borrado no pudo ejecutarse le devolverá un 405 (que significa que el método no está permitido), 409 que hay algún conflicto y 404 si no fue encontrado.

El borrado significa que el server no va a retornar más ese recurso con un simple GET. Si alguien solicita el recurso el server le devolverá un status code 410, ya no está. Fíjese que esto es diferente a pedir un recurso que nunca existió. En ese caso el código de retorno es un 404.

Las versiones previas del recurso no fueron borradas y pueden recuperarse usando la versión específica en una consulta.

5. Búsquedas en FHIR.

En esta parte nos enfocaremos en cómo buscar usando el paradigma de REST en FHIR. Es un tópico muy largo y ciertamente no lo vamos a cubrir en profundidad, pero la intención es proveerles el detalle suficiente como para al menos puedan empezar. Para detalles completos de cómo hacer consultas pueden referirse a la especificación.

Nos enfocaremos en como buscar un recurso simple, en este caso encontrar pacientes. Como usar parámetros y entender como FHIR devuelve los resultados. Luego queda que practique en cualquier servidor público y que complete las actividades en el servidor del curso.

Cada recurso tiene una lista de parámetros de búsqueda predefinidos para ese recurso. En esta página incluimos como ejemplo todos los parámetros de búsqueda del recurso paciente.

5.1.9 Search Parameters

Search parameters for this resource. The [common parameters](#) also apply. See [Searching](#) for more information about searching in REST, messaging, and services.

Name	Type	Description	Paths
active	token	Whether the patient record is active	Patient.active
address	string	An address in any kind of address/part of the patient	Patient.address
address-city	string	A city specified in an address	Patient.address.city
address-country	string	A country specified in an address	Patient.address.country
address-postalcode	string	A postalCode specified in an address	Patient.address.postalCode
address-state	string	A state specified in an address	Patient.address.state
address-use	token	A use code specified in an address	Patient.address.use
animal-breed	token	The breed for animal patients	Patient.animal.breed
animal-species	token	The species for animal patients	Patient.animal.species
birthdate	date	The patient's date of birth	Patient.birthDate
careprovider	reference	Patient's nominated care provider, could be a care manager, not the organization that manages the record	Patient.careProvider (Organization , Practitioner)
deathdate	date	The date of death has been provided and satisfies this search value	Patient.deceasedDateTime
deceased	token	This patient has been marked as deceased, or as a death date entered	Patient.deceased[x]
email	token	A value in an email contact	Patient.telecom(system=email)
family	string	A portion of the family name of the patient	Patient.name.family
gender	token	Gender of the patient	Patient.gender
given	string	A portion of the given name of the patient	Patient.name.given
identifier	token	A patient identifier	Patient.identifier
language	token	Language code (irrespective of use value)	Patient.communication.language
link	reference	All patients linked to the given patient	Patient.link.other (Patient)
name	string	A portion of either family or given name of the patient	Patient.name
organization	reference	The organization at which this person is a patient	Patient.managingOrganization (Organization)
phone	token	A value in a phone contact	Patient.telecom(system=phone)
phonetic	string	A portion of either family or given name using some kind of phonetic matching algorithm	Patient.name
telecom	token	The value in any kind of telecom details of the patient	Patient.telecom

Tenga en cuenta que no hay un requerimiento en FHIR que determine que todos los servidores necesitan soportar todos estos parámetros de búsqueda, y por otro lado, el servidor también es capaz de definir sus propias búsquedas. Hay una lista de parámetros que el comité de cada recurso define y cree que son los más comunes o los más apropiados para métodos de búsqueda.

Todo servidor FHIR debería ser capaz de indicar cuales parámetros soporta para cada recurso en el recurso de conformance. El formato para una búsqueda básica siempre toma el método GET y básicamente es como se ejemplifica ahora.

```
GET [base]/[type]?[parameters] &_format=[mime-type]
```

O

```
GET [base]/[type]/_search?[parameters] &_format=[mime-type]
```

Dónde:

- Base es la URL raíz del servidor
- Type es el tipo de recurso, por ejemplo paciente.
- Parameters son los distintos tipos de búsqueda que usted quiere utilizar.
- Formato es el resultado que usted espera recibir XML/JSON.

Los parámetros están expresados como pares clave/valor.

<http://fhir.hl7fundamentals.org/r4/Patient?name=eve>

Esto debería devolver todos los pacientes cuyo nombre contenga la cadena 'eve'.

GET

Params **Authorization** Headers Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	eve	
Key	Value	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 662 ms Size: 17.65 KB

Pretty Raw Preview JSON

```

1 {
2   "resourceType": "Bundle",
3   "id": "4ef9f80c-eb0f-413c-b2a7-402a6c7eb32c",
4   "meta": {
5     "lastUpdated": "2019-04-17T19:45:48.378+00:00"
6   },
7   "type": "searchset",
8   "total": 9,
9   "link": [
10    {
11      "relation": "self",
12      "url": "http://fhir.hl7fundamentals.org/r4/Patient?name=eve"
13    }
14  ]
15 }
```

<http://fhir.hl7fundamentals.org/r4/Patient?name=eve&format=xml>

En la misma búsqueda, pero especificando que el recurso debe ser formato XML. Esto lo podemos indicar cambiándolo aquí en lugar del formato default que es JSON.

Cada parámetro de búsqueda tiene un tipo que define como se comporta, por ejemplo: si es un texto, un número, parte de un código o un token. En la mayoría de los casos, es razonablemente sencillo y a pesar de que hay algunas sutilezas en algunos de los casos. Uno que causa alguna dificultad al principio es cuando uno tiene que buscar por parámetros referenciando un recurso. Por ejemplo si quiere encontrar todos los problemas de un paciente en particular debería ejecutar la siguiente consulta

<http://fhir.hl7fundamentals.org/r4/Condition?subject=5>

Esta consulta retornará cada problema del sujeto con este ID el cual es un paciente. Se supone que uno ya buscó previamente en pacientes para localizar ese ID.

Es posible obtener consultas más complejas que esta, por ejemplo podría concatenar distintos parámetros para obtener todos los problemas de un paciente cuyo nombre es Sam.

Vaya en detalle más a la especificación si quiere obtener o implementar un caso en particular.

GET <http://fhir.hl7fundamentals.org/r4/Condition?subject=5> Send

Params Authorization Headers Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> subject	5	
Key	Value	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 334 ms Size: 7.62 KB

Pretty Raw Preview JSON

```

1 {
2   "resourceType": "Bundle",
3   "id": "2cd242e0-ca83-4183-8d23-1eea6e8524da",
4   "meta": {
5     "lastUpdated": "2019-04-17T19:49:22.951+00:00"
6   },
7   "type": "searchset",
8   "total": 5,
9   "link": [
10    {
11      "relation": "self",
12      "url": "http://fhir.hl7fundamentals.org/r4/Condition?subject=5"
13    }
14  ],

```

Recuperar un recurso

Para recuperar un recurso basado en sus ID ya lo ejecutamos debemos seguir el siguiente patrón.

Dónde:

- **Host** es el nombre del servidor.
- **Resource** es recurso como por ejemplo Paciente u observación
- **ID** es el identificador dl recurso.

Por ejemplo:

GET <host>/<resourceType>/<id>

Esto retornará el recurso Paciente cuyo ID es 1 desde el servidor de test.

El Status de la respuesta es importante:

- 200 significa que el recurso fue encontrado.

- 410 significa que existió uno con ese número pero fue borrado.
- 404 significa que no fue encontrado.

El servidor también les devolverá información en la cabecera. Esto incluye

Content-Location, donde se especifica ubicación completa del recurso incluyendo si hubiese alguna versión y **Content-Type** que será el formato del recurso XML/JSON. También hay otro tipo de información.

Se podría especificar el formato que el servidor debería retornarle los recursos de dos maneras distintas:

- 1) En el GET, poniendo en el header Accept. y el parámetro application/JSON+FHIR y cambiando estos parámetros por JSON/XML respectivamente o sino también
- 2) En la URL usando el comando `_format` para especificar en qué formato queremos la respuesta.

Cualquiera de estas opciones es correcta para aquellos clientes que quieren establecer estos parámetros de obtener el recurso en diferentes formatos. En general siempre va a obtener el ID de un recurso como resultado de búsqueda.

Parámetros de búsqueda Standar

Parámetros que aplican a todos los recursos.

Los siguientes parámetros aplican a todos los recursos:

`_content`, `_id`, `_lastUpdated`, `_profile`, `_query`, `_security`, `_tag`, `_text`.

Además el parámetro de búsqueda Text y filter también aplica para todos los recursos como parte de los resultados de búsqueda.

El parámetro de búsqueda ID siempre se refiere al ID lógico del recurso.

```
GET [base]/Patient?_id=23
```

Esta búsqueda encuentra al recurso Paciente del ID ingresado, solo puede haber un recurso con ese ID.

```
GET [base]/Patient/23
```

Sin embargo si buscamos con el parámetro ID, no vamos a obtener directamente el recurso, sino que vamos a recibir un bundle como respuesta.

Hasta ahora vimos cómo recuperar recursos del servidor, en base a la coincidencia de algún valor pasado por parámetro, y como encontrar basándose en la referencia a otros recursos. Pero no hemos hablado acerca del formato del resultado. Como en muchos casos puede haber más de una coincidencia. En FHIR a este conjunto de resultados se los agrupa en un “bundle”.

Los bundles, se usan en muchas partes de FHIR. Es la estructura básica de los documentos y de los mensajes, como también tienen la posibilidad de agrupar muchos comandos en una sola transacción.

Resumiendo, una búsqueda en FHIR es simple. Hay que enviar una petición GET con los parámetros deseados agrupados en pares de clave/valor en la URL.

También se pueden incluir parámetros adicionales que brindan más funcionalidad a la búsqueda, como por ejemplo, `_lastUpdated`, `_count`.

`_lastUpdate`: Lo usaremos para recuperar recursos basados en la última fecha de actualización en el servidor.

`_count`: Lo usaremos para determinar cuántos productos queremos recuperar por página.

Ejemplo:

```
GET [base]/Observation?_lastUpdated=>2010-10-01
```

Tipos de Parámetros de búsqueda.

Cada parámetro de búsqueda tiene definido un tipo de dato que determina cómo se comporta.

Los tipos son:

- **Number**: el parámetro de búsqueda tiene que ser un número (puede ser un entero o un decimal)
- **Date**: Este parámetro es un valor fecha/hora en el formato estándar XLM.

- **String:** Es simplemente una cadena de texto, como la parte de un nombre. Distingue entre mayúsculas y minúsculas, la coincidencia puede ser con solo el inicio de esa cadena y puede contener espacios.
- **Token:** Este parámetro es para elementos codificados o identificadores.
- **Reference:** Es una referencia a otro recurso.
- **Composite:** Este combina la búsqueda con dos valores juntos.
- **Quantity:** Busca en base a una cantidad.
- **Uri:** Busca en base al ID del recurso en el servidor.

En la búsqueda también se pueden incluir modificadores que controlan el comportamiento.

Estos modificadores varían dependiendo del parámetro que se use.

Modificadores

Para todos los parámetros:

missing por ejemplo buscar los pacientes que no tienen el valor Género. Ej: gender: missing=true (o false).

Para Cadenas:

Exact. La coincidencia tiene que ser exacta sin coincidencias parciales.

Contains: es para especificar coincidencia parcial al inicio o al final de la cadena.

Para Token

:text busca en parte del concepto codificado o una parte del código.

Para Reference:

[Type] donde type es el nombre del tipo de recurso.

Si al consultar un servidor no soporta alguno de los parámetros o modificadores va a rechazar la consulta.

Prefijos

Para los tipos de Parámetro Number, Date y Quantity es posible agregar un prefijo que determina el tipo de coincidencia.

eq	El parámetro del recurso tiene que ser igual al valor consultado.
ne	El parámetro del recurso no tiene que ser igual al valor consultado.
gt	El parámetro del recurso tiene que ser mayor al valor consultado.
lt	El parámetro del recurso tiene que ser menor al valor consultado.
ge	El parámetro del recurso tiene que ser mayor o igual al valor consultado.
le	El parámetro del recurso tiene que ser menor o igual al valor consultado.
sa	El parámetro del recurso comienza después del valor consultado.
eb	El parámetro del recurso termina después del valor consultado.
ap	El parámetro del recurso aproximadamente el mismo que el valor consultado.

Si no se especifica el prefijo se asume el eq.

Veamos muchos ejemplos que seguramente servirán más que seguir con teorías. Queda en cada uno empezar a profundizar, combinar estos parámetros en búsqueda de otros resultados o ir afinando búsquedas.

.

Ejemplos

[parameter]=lt100	Valores que son menores a 100
[parameter]=le100	Valores que son menores o igual a 100
[parameter]=gt100	Valores que son mayores a 100
[parameter]=ge100	Valores que son mayores o iguales a 100
[parameter]=ne100	Valores que son distintos a 100

GET [base]/Encounter?length=gt20	Busca todos los encuentros mayores que 20 días
GET [base]/ImmunizationRecommendation?deo-number=2	Busca por una recomendación de vacunas como segunda dosis
[parameter]=eq2013-01-14	Coincide con 2013-01-14T00:00 (obviamente) y también con 2013-01-14T10:00 pero no 2013-01-15T00:00
[parameter]=ne2013-01-14	Coincide con 2013-01-15T00:00 pero no 2013-01-14T00:00 o 2013-01-14T10:00
[parameter]=lt2013-01-14T10:00	Incluye la fecha 2013-01-14, porque se incluye parte de 14-Jan 2013 antes de 10am
[parameter]=gt2013-01-14T10:00	Incluye la fecha 2013-01-14, porque se incluye parte de 14-Jan 2013 después 10am
[parameter]=ge2013-03-14	Es un período desde 21-Jan 2013 en adelante",
[base]/Patient?name=eve	Todos los pacientes cuyo nombre contienen la palabra "eve" al inicio del nombre. Esto incluye también a las pacientes con nombre "Eve", "Evelyn"
[base]/Patient?name:contains=eve	Todos los pacientes cuyo nombre contienen la palabra "eve" en cualquier posición. Esto incluye también a las pacientes con nombre "Eve", "Evelyn" y también "Severine".
[base]/Patient?name:exact=Eve	Todos los pacientes cuyo nombre es exactamente "Eve". Esto no incluye a las pacientes con nombre "eve", "Evelyn" y tampoco "Severine".
GET [base]/Patient?identifier=http://hl7fundamentals.org/patient 12345	Busca todos los pacientes cuyo identificador sea 12345 en el servidor http://hl7fundamentals/patient"
GET [base]/Patient?gender=female	Busca cualquier paciente con género femenino
GET	Busca cualquier paciente cuyo género no

[base]/Patient?gender:not=female	sea femenino
GET [base]/Patient?active=true	Busca por pacientes activos
GET [base]/Condition?code=http://hl7fundamentals.org/conditions/codes ha543	Busca por condition con código "ha543" y que el sistema de codificación sea http://hl7fundamentals.org/conditions/codes"
GET [base]/Condition?code=ha542	Busca por condition con código "ha542, sin especificar el sistema de codificación
GET [base]/Condition?code:text=headache	Busca por problema donde el texto coincide con "headache"

Búsquedas a través de recursos.

Este tipo de búsquedas en la actualidad es bastante complejo, pero un escenario muy común es por ejemplo, buscar todos los problemas que tiene un paciente.

/Patient/example recuperaríamos el recurso paciente.

/Condition?subject._id=example devolverá todos los problemas asociados al paciente example.

Paginamiento

Si el server tiene la capacidad de paginar los resultados debe cumplir con la especificación de enviar los links de las páginas siguientes o previas.

Un ejemplo.

```
<Bundle xmlns="http://hl7.org/fhir">
  <link>
    <relation value="self">
      <url value="http://example.org/Patient?name=peter&stateid=23&page=3"/>
    </link>
    <!-- 4 links for navigation in the search. All of these are optional, but recommended -->

    <link>
      <relation value="first"/>
      <url value="http://example.org/Patient?name=peter&stateid=23&page=1"/>
    </link>
  </link>
</Bundle>
```

```
</link>
<link>
  <relation value="previous"/>
  <url value="http://example.org/Patient?name=peter&stateid=23&page=2"/>
</link>
<link>
  <relation value="next"/>
  <url value="http://example.org/Patient?name=peter&stateid=23&page=4"/>
</link>
<link>
  <relation value="last"/>
  <url value="http://example.org/Patient?name=peter&stateid=23&page=26"/>
</link>

<!-- then the search results... -->
</Bundle>
```

El servidor le puede informar la cantidad de resultados por página.

6. Resumen de la unidad y conclusiones.

En esta unidad aprendimos como interactuar con un servidor FHIR. En las próximas unidades profundizaremos los conceptos de profiles, conformance, y consideraciones a tener en cuenta en la arquitectura e implementación de FHIR.