**Universidad de Concepción**
**Facultad de Ingeniería**
**Departamento de Ingeniería Informática y Ciencias de la Computación**

# ARQUITECTURA PARA TRANSMISIÓN DE DATOS DE SALUD DESDE DISPOSITIVOS MÓVILES A FICHAS CLÍNICAS

**Hugo Contreras Román**

Memoria presentada para la obtención del título de Ingeniero Civil Informático

**Profesor Patrocinante**: Gonzalo Rojas Durán
**Profesora Co-patrocinante:** Jacqueline Sepúlveda Carreño

Noviembre de 2023
Concepción, Chile

**Resumen:**

La adherencia a tratamientos médicos de pacientes es un factor muy relevante en la salud de estos mismos y en el progreso que tengan en cierta patología. Como tal, es importante para el personal médico tener un registro claro del porcentaje o índice de adherencia que estos pacientes mantienen durante el transcurso de su tratamiento farmacológico. Esta información se suele transmitir de forma verbal de paciente a médico durante controles médicos, pero esta suele ser una manera de transmitirla poco fidedigna y sujeta a errores. Para dar un registro más verídico y estructurado al personal médico y ayudarlo a tomar mejores decisiones en el tratamiento de pacientes, se plantea la alternativa de proveer un mecanismo digital que permita a pacientes registrar las ingestas de medicamento mediante una aplicación móvil (anteriormente AFAM 1.0) y enviarlas a través de la red a sistemas de ficha clínica, donde el personal médico de interés puede llevar un seguimiento del comportamiento de ingestas del tratamiento. Esta memoria se enfoca en proveer una alternativa de arquitectura que permite la transmisión de estos datos, mediante un bus genérico de interoperabilidad que utiliza estándares de datos médicos actuales, tales como FHIR y SNOMED, y facilita la conexión entre aplicaciones móviles y sistema de fichas clínicas para esta tarea.

# Índice

# 1. Introducción

En el cuidado de la salud de las personas, es fundamental contar con información fidedigna sobre su estado por parte del personal médico que les atiende. Así, los sistemas de ficha clínica permiten registrar datos sobre diagnóstico, tratamiento, controles y otros sobre los pacientes. Sin embargo, en el caso de pacientes con patologías crónicas sometidos a tratamiento farmacológico, la real adherencia a dichos tratamientos es una información que normalmente no es compartida de forma fidedigna con los profesionales.

La adherencia a tratamiento farmacológico es un factor fundamental en el tratamiento y mejoramiento de la calidad de vida de los pacientes, por lo que tener datos objetivos de esta constituye un aporte significativo a su tratamiento. La adherencia puede definirse como el grado en el que el comportamiento de una persona, como tomar medicamentos, seguir una dieta y/o hacer cambios en su estilo de vida, se corresponde con las recomendaciones acordadas por un proveedor de atención médica **[1]** y, en este caso, haciendo especial énfasis en la toma de medicamentos.

Dentro del marco del proyecto FONDEF IT21I0097, "Seguimiento remoto e integración interoperable a ficha clínica de la adherencia a tratamiento antihipertensivo de personas mayores", se propone el desarrollo de un sistema software que apoye el proceso de recolección, análisis y visualización de datos que caracterizan la adherencia al tratamiento farmacológico de pacientes crónicos. El objetivo principal de este sistema es proveer al personal sanitario, a través de los sistemas de fichas clínicas que utilizan habitualmente, de un índice de adherencia actualizado, en el momento en que el/la paciente está siendo atendido/a, y de visualizaciones interactivas sobre detalles del cumplimiento del tratamiento prescrito (tomas de medicamentos, signos vitales, automedicación, entre otros). En un principio, el sistema sólo debe contar con compatibilidad con la patología de hipertensión arterial.

Para el profesional sanitario, este sistema permitirá contar con datos objetivos sobre el tratamiento prescrito, y tomar decisiones que consideren este importante factor, previniendo complicaciones y hospitalizaciones innecesarias. Para el paciente, este sistema permitirá llevar una trazabilidad de tratamiento y así mantener informado a los profesionales sanitarios sobre su real adherencia, con lo cual aportaría a lo descrito anteriormente.

Para lograr este objetivo, el sistema cuenta con las siguientes prestaciones:

1. Proveer al personal sanitario de índices de adherencia al tratamiento farmacológico de cada paciente, permanentemente actualizado, a partir de los datos que ha suministrado sobre su ingesta de los fármacos prescritos.
2. Proveer al personal sanitario de visualizaciones de datos interactivas, para conocer la adherencia en la toma de medicamentos, más aspectos de estilos de vida y otros datos relevantes para su tratamiento.

3. Permitir al personal sanitario acceder a una vista detallada de datos de adherencia al tratamiento farmacológico, separados por afección y medicamento, al momento de acceder a la información del/la paciente a través del sistema de ficha clínica.

Para proveer estas prestaciones, se requiere la evaluación de una arquitectura que permita interoperar un sistema de ficha clínica con una aplicación móvil que capture los datos de ingestas de medicamentos de un tratamiento en particular, suministrados por los pacientes o cuidadores. En concordancia con estándares internacionales de gestión de datos de salud, se plantea inicialmente la implementación de un bus de interoperabilidad que utilice el estándar FHIR de HL7, que considere la transmisión de datos sobre adherencia a tratamiento y su despliegue en un sistema de ficha clínica.

Dicho esto, la presente Memoria de Título se enfoca en proponer una opción de esta arquitectura de interoperabilidad, la cual sea compatible con prototipos de aplicaciones móviles de alerta, despliegue y captura de datos de tratamiento, y con un prototipo de sistema web de ficha clínica, que permite acceder a la información del paciente correspondiente permanentemente actualizada.

## 1.1. Objetivo General

Apoyar la comunicación de datos objetivos de tratamiento farmacológico a personal sanitario, desde aplicaciones móviles de pacientes a sistemas de ficha clínica, mediante el desarrollo de un prototipo basado en estándares de interoperabilidad en salud que permita la conexión entre ambos actores.

## 1.2. Objetivos Específicos

1. Proponer una arquitectura de software que favorezca la interoperabilidad entre aplicaciones móviles y sistemas de ficha clínica, para el envío y despliegue de datos de adherencia a tratamiento farmacológico.

2. Evaluar la factibilidad de la interoperabilidad de la aplicación móvil con distintos sistemas de ficha clínica, mediante el análisis de las tecnologías existentes sobre estos sistemas y protocolos de comunicación específicos del dominio de salud.

3. Desarrollo del bus de interoperabilidad que permita la transmisión de datos de aplicación móvil a ficha clínica, en base a la arquitectura propuesta.

4. Establecer la conexión entre el bus de interoperabilidad y aplicaciones móviles encargadas de capturar y transmitir los datos de adherencia a tratamientos farmacológicos de pacientes, desarrollando los adaptadores necesarios en el contexto de la compatibilidad de las aplicaciones con los estándares de salud.

5. Establecer la conexión entre el bus de interoperabilidad y sistemas de ficha clínica, utilizando sistemas de ficha clínica existentes o un prototipo que privilegie el acceso a los datos de adherencia a tratamiento de distintos pacientes, desarrollando los adaptadores necesarios en el contexto de la compatibilidad de estos sistemas con los estándares de salud.

## 1.3. Metodología

Se adoptó un método de desarrollo iterativo e incremental, con adaptaciones de propuestas ágiles a un entorno de desarrollo que consta con un equipo de trabajo encargado de ver el alcance del proyecto completo, lo cual incluye: desarrollo de aplicación móvil de captura de datos de adherencia de pacientes, propuesta de arquitectura para el sistema macro, el desarrollo del bus de interoperabilidad, y sistema de visualización de índices de adherencia de pacientes.

Se realizaron reuniones semanales con el equipo del proyecto con el fin de ir progresando en el desarrollo de los distintos frentes del sistema completo. Se mantuvo registro del trabajo mediante la plataforma de Trello [2] y Slack [3]**.** Durante las reuniones se discutía lo avanzado en cada aspecto, y las consideraciones del sistema general como el modelo de datos a utilizar, las herramientas a utilizar, etc. y de manera incremental se fue creando la compatibilidad entre la arquitectura de interoperabilidad con los 2 actores (aplicación móvil y ficha clínica o módulo de visualización). Finalmente, la arquitectura fue probada por ambos actores y obtuvo los resultados esperados, logrando la interoperabilidad con estándares de salud.

El desarrollo específico del bus de interoperabilidad se basa en un análisis de tecnologías existentes en el área de sistemas de salud que sean pertinentes a los objetivos de la Memoria.

El desarrollo de esta Memoria puede ser descompuesto en tres etapas de acuerdo a los objetivos específicos: **propuesta de arquitecturas, evaluación y selección de arquitectura, y desarrollo de arquitectura.** Las dos primeras etapas corresponden a la definición de un modelo de arquitectura que satisface los objetivos, y la última como el desarrollo iterativo de un prototipo representativo de las principales funcionalidades de esta arquitectura.

Dicho esto, durante las etapas se desarrollaron las siguientes tareas:

**1. Proposición de arquitectura:**

- Obtener requerimientos del sistema iniciales, tanto funcionales como no funcionales.
- Investigar arquitecturas de sistemas médicos existentes para propósitos similares.
- Investigar formato de datos en sistemas existentes y el estándar FHIR HL7.
- Investigar stack de tecnologías que permitan el desarrollo de la arquitectura, y las herramientas necesarias que permitan satisfacer los requerimientos de usuario, guiándose por la arquitectura de otros sistemas si existen. Se consideraron protocolos de comunicación, bases de datos, persistencia de datos, servidores, APIs, lenguajes de programación, etc.

- Verificar que las herramientas investigadas sean capaces de satisfacer los requerimientos de usuario, sean compatibles con los estándares necesarios y/o sigan los protocolos de sistemas previos.
- Generar opciones de arquitecturas en base a la información recolectada para su posterior evaluación.
- Realizar diagramas de arquitectura con los componentes principales.
- Formar un marco teórico con la información recabada.

**2. Evaluación y selección de arquitectura:**

- Establecer repositorio para el proyecto.
- Desarrollar prototipos en estado de alfa o concepto de prueba, es decir, que incorporen sólo las ideas principales de la arquitectura, como los mecanismos de interoperabilidad, sin mayores elementos visuales o complejidades estéticas.
- Validar compatibilidad de los prototipos de arquitectura con los protocolos de interoperabilidad necesarios y los requerimientos de usuario.
- Evaluar el desempeño de las arquitecturas y realizar ajustes si lo ameritan, de lo contrario descartarlas.
- Seleccionar la opción de arquitectura mejor evaluada, con la cual avanzar en el posterior desarrollo.

**3. Desarrollo de arquitectura:**

- Utilizar requerimientos previos para establecer historias de usuario a concretar en el desarrollo del prototipo de sistema final.
- Continuar el desarrollo de prototipo de bus de interoperabilidad definido anteriormente, considerando las conexiones a los sistemas con los cuales debe interactuar. Realizar este desarrollo en forma de ciclo iterativo ágil, con avances incrementales y metas definidas. Esto incluye definir historias de usuario para cada sprint o entrega, y organizar reuniones para presentar resultados y obtener retroalimentación. Repetir este ciclo hasta finalizar el prototipo de arquitectura.
- Trabajar prototipos de aplicación móvil y ficha clínica para validar interoperabilidad.
- Solicitar información sobre detalles del sistema, tales como los campos de información que debe trabajar el bus de interoperabilidad según los requerimientos funcionales, ergo, cuánta información del paciente es requerida transmitir, cuánto detalle sobre los medicamentos ingeridos se deben transmitir, etc.
- Utilizar validador de formato para los recursos FHIR que se trabajen en el bus de interoperabilidad.
- Considerar mecanismos de seguridad para el uso confidencial de datos de pacientes y control de acceso a estos por parte del personal sanitario.
- Obtener información de los sistemas existentes sobre los cuales operará la arquitectura, ya sea acceso a los sistemas médicos, prototipos de aplicación móvil, algunas vistas de interfaces,

información de cómo operan, o código fuente si es posible. Esto para considerarlos en la compatibilidad de la arquitectura.

- Trabajar en los wrappers o la compatibilidad del bus de interoperabilidad con aplicación móvil y sistema web de ficha clínica, y así considerar casos de estas plataformas tanto FHIR nativas como no nativas.
- Desarrollo de servicio que permite calcular el porcentaje o índice de adherencia de tratamiento farmacológico de pacientes durante un periodo determinado, que forma parte de una capa de análitica posteriormente desarrollable. Además de permitir consultas de pacientes individuales, se plantea la posibilidad de utilizarlo para obtener la adherencia de pacientes global.

## 1.4. Estructura del informe

Luego de esta introducción, se establece un marco teórico que establece el contexto de la solución, y que contiene los materiales y métodos necesarios para la conformación de la arquitectura. Dentro de este marco teórico se dan a conocer los estándares de salud considerados, trabajos previos relacionados a la temática, aplicaciones móvil y de ficha clínica que proveen cierta guía para continuar con la propuesta de solución. También se presentan las herramientas utilizadas en los diferentes componentes de la solución.

Luego de esto, se presenta una descripción de las propuestas de arquitectura, que consiste en las diferentes iteraciones por las cuales pasó la propuesta original de arquitectura. Después, se entrega el detalle de la arquitectura propuesta, considerando todos los aspectos que le conciernen.

Se prosigue con una evaluación de la arquitectura que mide tanto la compatibilidad con los distintos actores participantes, como el desempeño que presenta al realizar la tarea de interoperabilidad.

Finalmente, se entregan las conclusiones pertinentes acerca de la factibilidad de esta arquitectura de interoperabilidad para la transmisión de datos de salud.

# 2. Marco Teórico

Esta sección contiene la información que da contexto a la posibilidad de crear esta arquitectura de interoperabilidad. Se informa sobre los estándares de salud más relevantes, trabajo previo que está relacionado en ciertas funcionalidades con la arquitectura propuesta, aplicaciones móviles que permiten al usuario ingresar ingestas de medicamentos y sistemas de ficha clínica que dan información de pacientes a personal sanitario.

## 2.1 Estándares de salud

El planteamiento de este proyecto propone utilizar el estándar FHIR como estándar de interoperabilidad de datos de salud, por lo que se detalla en qué consiste este estándar y su importancia en el contexto actual de salud. Pero primero, cabe mencionar que FHIR es la versión más reciente de una serie de especificaciones que ha evolucionado a lo largo de los años de HL7 (Health Level 7) [4], la cual es una organización internacional sin fines de lucro dedicada a conformar estándares de salud informáticos. Se describen de manera breve versiones pasadas del estándar para dar contexto a la importancia de FHIR.

### 2.1.1 Mensajes HL7

Los mensajes HL7 (o HL7 Messaging) son un conjunto de estándares diseñados para transmitir información médica y administrativa entre distintos sistemas médicos. Definen un formato común para asegurarse que los distintos sistemas puedan procesar la información de manera consistente. Entre los estándares de Mensajes HL7 más importantes se encuentran la versión HL7 v2 y HL7 v3.

#### 2.1.1.1 HL7 v2

El estándar HL7 v2 es probablemente el estándar de salud más implementado alrededor del mundo, teniendo un nivel de uso del 95% en los sistemas de Estados Unidos, y más de 35 países con implementaciones disponibles [5]. Lanzado originalmente en el año 1987, su última versión corresponde a la versión 2.9, pero entre las más utilizadas se encuentran las versiones 2.5, 2.6 y 2.7 [5]. Debido a su popularidad, se sigue entregando soporte y extensiones a esta versión. Propone utilizar mensajes en formato ASCII y consisten en bloques de texto divididos en segmentos (separados por un salto de línea) para expresar los distintos componentes del mensaje.

```
MSH|^~\&|EPIC|EPICADT|iFW|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.5|
PID||0493575^^^2^ID 1|454721||DOE^JOHN^^^^|DOE^JOHN^^^^|19480203|M||B|254 MYSTREET AVE^^MYTOWN^OH^4
4123^USA||(216)123-4567|||M|NON|400003403~1129086|
NK1||ROE^MARIE^^^^|SPO||(216)123-4567||EC|||||||||||||||||||||||||
PV1||O|168 ~219~C~PMA^^^^^^^^^^|||277^ALLEN MYLASTNAME^BONNIE^^^^|||||||||| ||2688684|||||||||||||
||||||||||199912271408||||||002376853
```

*Figura 1: Mensaje HL7 v2 con 4 segmentos, obtenido de [6]*

Existen más de cien tipos de segmentos, identificados por el código inicial de 3 caracteres (**MSH, PID, NK1, y PV1** en Figura 1) en cada línea de segmento, y cada segmento contiene múltiples campos que representan distinto tipo de información, separados por *pipes* (|). A su vez, los campos pueden contener múltiples subcampos en su interior, los cuales se separan con un carácter (^). A este enfoque de separación o delimitación se le conoce como “pipe and hat” [7].
El primer segmento suele ser el **MSH,** que indica metadatos del mensaje. Existen múltiples tipos de mensajes que describen diferentes eventos de salud, siendo algunos de los más importantes **ADT**

para gestión de pacientes, **ORM** para órdenes médicas y **ORU** para resultados. El mensaje de Figura 1 es de tipo **ADT^A04**, lo que indica que es un mensaje para el registro de un paciente. Contiene segmentos de **PID** para información de paciente, **NK1** para información de familiares del paciente y **PV1** para información de la estadía del paciente en el hospital.

Este estándar es extensible, pudiendo añadir segmentos personalizables cuando el estándar no contiene la definición de información que se desea transmitir, mediante un carácter Z. Por ejemplo, **ZPD** para agregar información de un paciente que no se encuentre oficialmente en la especificación. El procesamiento depende del conocimiento que el sistema tenga de la extensión.

### 2.1.1.2 HL7 v3

A modo de cubrir algunas de las debilidades del estándar anterior, se lanza el estándar HL7 v3 en el año 2003, que busca ser más formal en la representación de la información. Este estándar es un enfoque dirigido por modelos, y utiliza el Modelo de información de Referencia (**RIM**), el cual es un modelo estático de diferentes conceptos y relaciones de carácter médico desarrollado durante finales de la década de 1990 **[8],** y su información es utilizada en la elaboración de los mensajes. En este sentido, se enfoca en la interoperabilidad semántica de la información, presentando la información con un contexto clínico completo respaldado en el modelo, y que el significado sea el mismo para emisor y receptor.

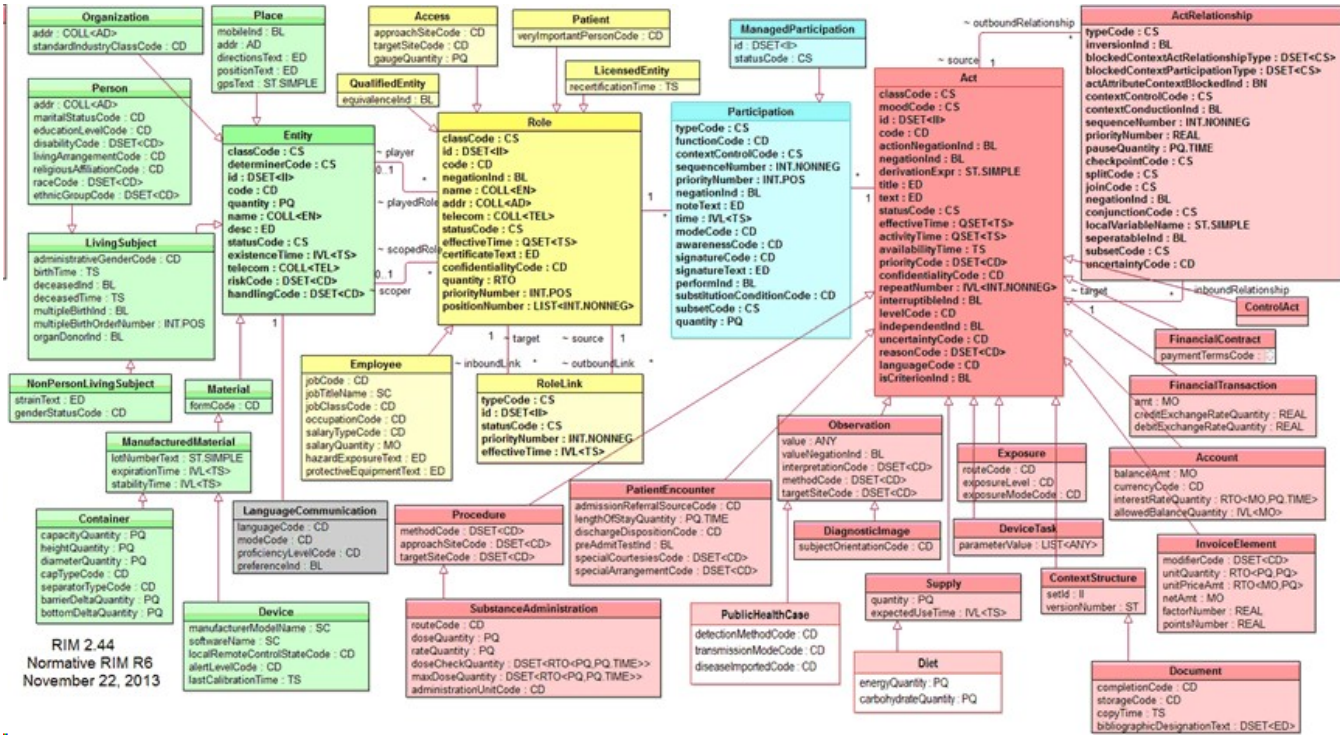
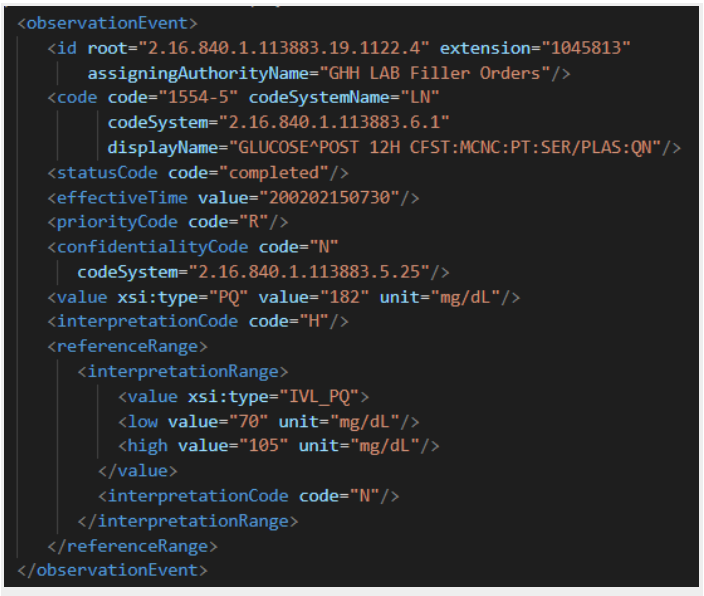



*Figura 2: Ejemplo de RIM HL7, obtenido de [7]*

A diferencia de la HL7 v2, el estándar v3 se rige estrictamente por este modelo, y necesita que todas las implementaciones de proveedores de este estándar cuenten con el modelo completo

implementado, lo que lo hace más complejo de mantener y escalar. Por otro lado, debido a la amplia cobertura de conceptos y relaciones que ofrece el modelo, se tiene libertad para modelar una plétora de casos de uso médicos, lo que provoca que el estándar tenga una curva de aprendizaje más pronunciada.



```
<observationEvent>
  <id root="2.16.840.1.113883.19.1122.4" extension="1045813"
      assigningAuthorityName="GHH LAB Filler Orders"/>
  <code code="1554-5" codeSystemName="LN"
      codeSystem="2.16.840.1.113883.6.1"
      displayName="GLUCOSE^POST 12H CFST:MCNC:PT:SER/PLAS:QN"/>
  <statusCode code="completed"/>
  <effectiveTime value="200202150730"/>
  <priorityCode code="R"/>
  <confidentialityCode code="N"
    codeSystem="2.16.840.1.113883.5.25"/>
  <value xsi:type="PQ" value="182" unit="mg/dL"/>
  <interpretationCode code="H"/>
  <referenceRange>
    <interpretationRange>
      <value xsi:type="IVL_PQ">
      <low value="70" unit="mg/dL"/>
      <high value="105" unit="mg/dL"/>
      </value>
      <interpretationCode code="N"/>
    </interpretationRange>
  </referenceRange>
</observationEvent>
```

*Figura 3: Mensaje HL7 v3 que describe una observación registrada de un paciente, obtenido de [9]*

Los mensajes HL7 v3 están conformados por 3 partes [10]: **Capa de Transporte,** que contiene información de transporte, como remitente y destinatario. **Capa de Evento,** que contiene información del evento que produjo el mensaje e información del mismo. **Capa de Contenido,** que contiene la información principal o contenido del mensaje.

El mensaje de Figura 3 es una sección de un mensaje completo HL7 v3, que corresponde a un mensaje de tipo ***observationEvent*** y presenta la observación del nivel de glucosa de una paciente realizada durante el 15 de febrero de 2002 a las 07:30, la cual es de 182 mg/dL y se encuentra en un nivel alto. A simple vista es un formato de mensaje mucho más legible que su predecesor. HL7 v3 no fue tan ampliamente extendido como HL7 v2 debido a la popularidad de este y a la complejidad de instaurar un nuevo modelo médico por sobre uno establecido. En ciertas partes del mundo tiene mayor relevancia, tales como en Reino Unido (con más de 2 millones de transacciones), Canadá y Países Bajos. [8]

En cuanto al formato de los mensajes y el RIM, son factores fundamentales en cuánto a lo que sería la nueva versión del estándar: FHIR, que reutiliza muchos componentes de estas previas versiones, como el uso de un **Modelo de Información de Referencia** y los **tipos de datos** definidos previamente, con las ventajas de las interfaces modernas web de transmisión de datos.

### 2.1.2 FHIR (Fast Healthcare Interoperability Resources)

FHIR, o Fast Healthcare Interoperability Resources, es el estándar definido para las nuevas generaciones de sistemas informáticos de salud, ya que principalmente provee soporte para las tecnologías web modernas, como llamadas HTTP y uso de APIs REST, además de ser compatible con formatos electrónicos de transmisión de datos populares en la actualidad como JSON y XML. Por otro lado, este estándar muestra un fuerte compromiso con el paradigma "Open Source", por lo que se encuentra disponible gratuitamente para su estudio y utilización, a diferencia de sus antecesores. [11]

Creado en 2011, el estándar ha evolucionado con el pasar de los años y ha tenido múltiples entregas, siendo la más reciente la entrega 5 o R5. Sin embargo, la que tiene mayor nivel de implementaciones a nivel global es la entrega 4, o R4. El trabajo investigativo y de desarrollo de esta Memoria se enfocó en la versión R4. Este estándar se basa en el intercambio de información médica con un formato liviano y su objetivo es el poder modelar todo caso de uso médico imaginable, de manera más intuitiva y facilidad que sus antecesores, tomando lo mejor de ellos (mensajes, documentos, RIM).
Consiste de 2 componentes principales:

- Un **modelo de contenidos o información**, similar a RIM en HL7 v3, pero esta vez en forma de "recursos", disponiendo de más de 140 recursos para casos de uso.
- Una **especificación para el intercambio** de estos recursos mediante interfaces RESTful, así como a través de mensajes y documentos, con un enfoque principal en el primero.

Un **recurso** es una entidad que representa un concepto médico, y sirve para intercambiar o almacenar información acerca de estos. FHIR dispone recursos para modelar cualquier caso de uso médico imaginable. La especificación de FHIR se encuentra disponible gratuitamente e incluye todo el detalle del estándar [11], junto con ejemplos para cada tipo de recurso. Durante este informe se refiere a los recursos con letra cursiva: *Resource.*

El paradigma REST con métodos HTTP es el protocolo principal que utiliza FHIR, el cual proporciona soporte para manipular los recursos del modelo siguiendo un esquema **CRUD (Create, Read, Update, Delete),** y entrega soporte a llamadas HTTP del tipo **POST, GET, PUT, DELETE** [12]. Además de éste, el estándar proporciona otros paradigmas para la transmisión de información médica:

- **Mensajes HL7**, asemejando las versiones v2 y v3 del estándar, son una colección de recursos (agrupados en un recurso *Bundle*), y que contienen un Header (con un recurso *MessageHeader*) que simulan ser los segmentos de mensajes HL7, y son gatillados mediante eventos en el flujo clínico.
- **Documentos,** se asemeja al estándar **CDA** (una extensión de HL7 v3 para documentos médicos), y utiliza un recurso *DocumentReference*, con el cual se puede incluir metadatos, y Header (con recurso *Composition*), además de un binario o link a un documento CDA real. Alternativamente, se pueden estructurar recursos en un *Bundle* para tener el formato CDA.
- **Servicios,** pueden verse como una forma más liviana de hacer mensajería, ya que no necesita incluir cabezales. El estándar permite elaborar servicios personalizados para el intercambio de

información, tanto simples o complejos, para requerimientos particulares. La única restricción es que los recursos sean efectivamente transferidos entre sistemas.

- **Almacenamiento,** la información puede ser almacenada en formato de recurso en bases de datos, lo que permite su rápido acceso para transmisión de datos, y compatibilidad con sistemas de bases de datos relacionales y no relacionales.

El modelo de recursos que presenta FHIR puede ser descompuesto en 5 módulos [13], que agrupan conceptos relacionados entre sí en el ambiente médico, además de características de la especificación.

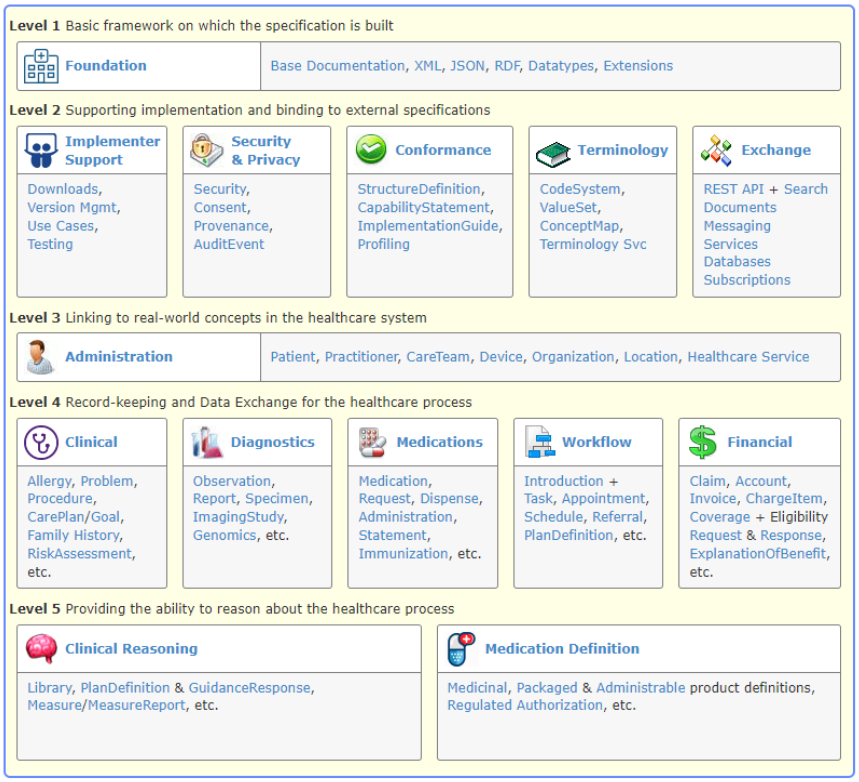



*Figura 4: 5 niveles de organización de recursos de FHIR, obtenido de [13]*

- **Nivel 1:** El framework básico de donde se construye el resto de la especificación. Aquí se encuentra la documentación como tal, definiciones de XML y JSON para recursos, los tipos de datos existentes y la definición de extensiones para el estándar.
- **Nivel 2:** Información de soporte para implementaciones, como los mecanismos de seguridad y privacidad del estándar, recursos asociados a las declaraciones de conformación de una implementación (como *CapabilityStatement*), recursos asociados con terminologías utilizadas en el estándar, y los paradigmas de intercambio de datos.
- **Nivel 3:** Recursos relacionados con los conceptos reales utilizados en los sistemas de salud. Es la información base que luego se enlaza con los niveles superiores para representar casos de uso de salud. Se pueden encontrar recursos como *Patient, Practitioner, Device, Encounter, Organization,* entre otros.

- **Nivel 4:** Se tienen los recursos que guardan y transmiten información de registros médicos de distinta índole, tales como información clínica de pacientes (*Condition, CarePlan, Procedure),* información de diagnósticos (*Observation, DiagnosticReport, ImagingStudy*), de medicamentos (*Medication, MedicationAdministration, MedicationRequest*), de flujo de trabajo clínico (*Task, Appointment, Schedule)* y de información relacionada a financiamiento (*Account, Coverage, PaymentNotice)*.
- **Nivel 5:** Se agrupan recursos relacionados al razonamiento clínico, en los cuales se puede representar artefactos de conocimiento médico, tales como protocolos clínicos y medidas de calidad de salud. Recursos como *Library, Measure, PlanDefinition.*

FHIR cuenta con una comunidad muy activa debido a su naturaleza Open Source, y existe un chat para discusión de implementaciones, dudas, y donde los mismos administradores y creadores del estándar participan activamente con los usuarios. Este chat puede ser encontrado en **Anexos.**

#### 2.1.2.1 Recursos FHIR

Un recurso en FHIR está estructurado de múltiples elementos, los cuales se detallan a continuación. En Figura 5 se puede encontrar la estructura básica de un recurso *Patient* en formato XML.


```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

*Figura 5: Estructura de un recurso de tipo Patient en formato XML, obtenido de [7]*

#### 2.1.2.1.1 Elementos

Los diferentes campos que componen un recurso en FHIR se conocen como **elementos.** Estos pueden ser de distintos tipos de datos, incluyendo:

- **Tipos de datos básicos o primitivos**, como strings, enteros, fechas, booleanos, etc.
- **Tipos de datos complejos**, los cuales son tipos de datos compuestos por más elementos, por ejemplo **Period***,* que describe un período compuesto de una fecha inicial y una fecha final.
- **Elementos estructurales o *Backbone Elements,*** que son propios de cada recurso. Son tipos de datos complejos, pero a diferencia de los anteriores se definen dentro de cada recurso en el que se encuentran, por ejemplo el elemento **contact** de *Patient.*

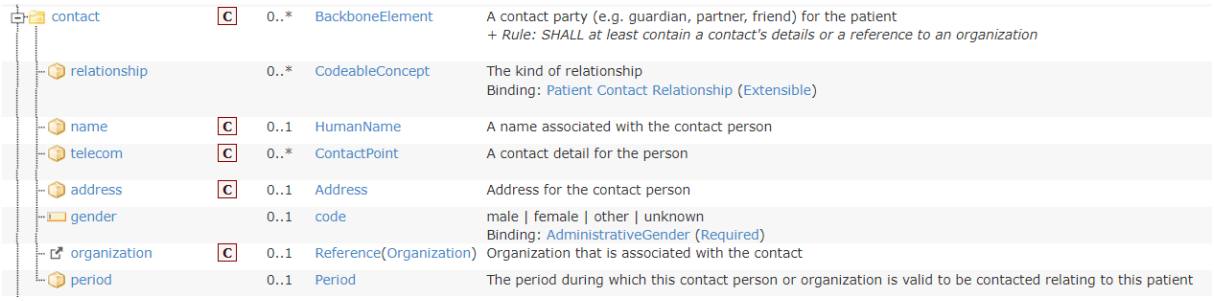



*Figura 6: Elemento contact de Patient, el cual es un BackboneElement, obtenido de la especificación FHIR*

Hay elementos que son compartidos por todos los recursos (o su mayoría) de la especificación, como son:

- **id,** que representa el identificador lógico del recurso (ver Referencias).
- **meta,** que contiene metadatos del recurso.
- **implicitRules,** las cuales son reglas que debe seguir el recurso (ver Restricciones en **Anexos**).
- **language,** que indica el lenguaje de la información contenida en el recurso.
- **text,** que contiene un resumen del recurso para interpretación humana.
- **contained,** posee recursos adicionales definidos in-line en un recurso (ver Referencias).
- **extension,** que posee extensiones de información para cubrir casos de uso no definidos en la especificación (ver Extensiones y perfiles en **Anexos**).

Los elementos pueden tener distintas cardinalidades para asignar múltiples valores a un elemento. Por ejemplo, el elemento **dosageInstruction** de *MedicationRequest*, que indica las instrucciones de ingesta o posología, puede representar múltiples instrucciones.


*Figura 7: Elemento dosageInstruction puede aparecer múltiples veces en MedicationRequest, obtenido de la especificación FHIR*

Los elementos tienen un ícono que indica su tipo de elemento. También se pueden asociar con *flags,* que indican características de uso del elemento y cómo afectan una implementación. Una lista completa de estos íconos y *flags* se puede encontrar en **Anexos.** En Figura 8 se ve el elemento **status** de *MedicationRequest* , con un ícono de tipo de dato básico y con dos ejemplos de *flags*:

- **?! -** Es un elemento modificador que cambia el significado que se le da al recurso y a otros elementos que lo compongan.
- **Σ -** El elemento debe ser incluido en el resumen del recurso cuando se solicita mediante la operación de búsqueda _summary. (ver Parámetros de búsqueda en **Anexos**).

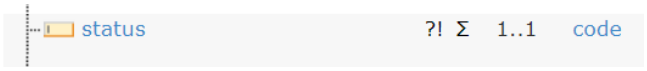



*Figura 8: Elemento status de MedicationRequest, obtenido de la especificación FHIR*

Algunos elementos pueden tomar uno de varios tipos de datos. En Figura 9 el elemento **effective[x]** de *MedicationAdministration,* que indica fecha y hora de ingesta, puede ser de tipo **dateTime** (ingerido en un timestamp específico) o **Period** (ingerido a lo largo de un periodo).


*Figura 9: Elemento effective de MedicationAdministration, obtenido de la especificación FHIR*

#### 2.1.2.1.2 Referencias

El modelo de recursos de FHIR establece relaciones entre estos mediante sus elementos que son de tipo **Reference.** Estas referencias se dan siempre en un sólo sentido, desde un recurso (fuente) hacia otro (objetivo)**.** Existen distintos tipos de referencias en el estándar FHIR.

- **Referencias literales:** Corresponde a una dirección URL del recurso, la cual puede tomar la forma de 3 tipos de referencias distintas:
  - **Absoluta:** URL absoluta de la ubicación del recurso.
  - **Relativa:** URL relativa al servidor BASE de la ubicación del recurso. Es el tipo de referencia preferida para los recursos relevantes a los objetivos de la Memoria, según los ejemplos que entrega la especificación para estos recursos.
  - **Interna:** URL que hace referencia a un recurso dentro del elemento **contained** del recurso base. Puede ser una referencia relativa o absoluta, pero incluye un carácter # para indicar que es una referencia interna. Se recomienda guardar recursos en **contained** sólo si no hay registro formal del recurso en el servidor.
- **Referencias lógicas:** Corresponde a un valor alfanumérico para identificar al recurso. Todos los recursos tienen una por defecto, presente en su elemento **id.**
- **Referencias canónicas:** Específicas de algunos recursos importantes de la estructura de la especificación, y es una URL que identifica al recurso en todo contexto de uso. Se encuentra en el elemento **uri** de estos recursos (como *CapabilityStatement, StructureDefinition,* entre otros), y se sugiere referenciarlos de esta manera.

```
-Literales:
   -Absoluta: http://fhir.hl7.org/svc/StructureDefinition/c8973a22-2b5b-4e76-9c66-00639c99e61b
   -Relativa: Patient/034AB16
   -Interna: Id del recurso en contained es med0316, entonces la referencia interna es #med0316

-Lógica: 32

-Canónica: http://healthit.gov/nhin/purposeofuse
```

*Figura 10: Ejemplos de las distintas referencias, elaboración propia*

#### 2.1.2.1.3 CodeSystems y ValueSets

FHIR al igual que en HL7 v3, utiliza sistemas de códigos y terminologías para representar gran parte de la información médica en los recursos. Los sistemas de código o **Code Systems** definen cuáles códigos existen y cómo son entendidos. Por lo general, son sistemas externos muy utilizados en áreas de salud que se integran con el estándar para tener mayor interoperabilidad. Algunos de estos sistemas son: SNOMED CT, LOINC, RxNorm.

#### 2.1.2.1.4 Restricciones

Gran parte de los recursos de la especificación se encuentran adheridos a restricciones necesarias en los elementos que los componen para ser conformes con la especificación.

#### 2.1.2.1.5 Parámetros de búsqueda

El estándar define mecanismos de navegación para trabajar con los recursos del modelo, siendo uno de los principales la búsqueda. Al operar con el esquema REST, estas búsquedas se realizan mediante la llamada HTTP GET y con diversos parámetros definidos en la especificación.

#### 2.1.2.1.6 Extensiones y perfiles

En los múltiples ecosistemas médicos a nivel mundial es muy difícil llegar a un consenso general en toda la información que debería soportar un estándar de datos de salud. Por esta razón, es comprensible permitir la implementación de **extensiones** de los recursos para representar información más específica que actualmente no se encuentra conformada en la versión del estándar utilizada, o que es muy situacional para ser incluida en el estándar. Adicionalmente, es posible definir **perfiles** que son versiones modificadas de los recursos bases, utilizando extensiones u otras restricciones, para atender un requerimiento más específico.

Se encuentra mayor detalle y ejemplos sobre CodeSystems, restricciones, parámetros de búsqueda, extensiones y perfiles en **Anexos.**

### 2.1.2.2 SMART on FHIR

SMART o Substitutable Medical Applications and Reusable Technologies, es una plataforma que permite la creación de aplicaciones que funcionan de manera fluida y segura en el ecosistema de sistemas médicos [14]. Estas aplicaciones se alimentan de la información obtenida de registros

electrónicos de salud (EHR) u otras bases de datos que soportan el estándar SMART, y su propósito es ayudar al personal sanitario a realizar funciones para casos clínicos, investigación y salud en general. FHIR tiene un rol protagónico en cuanto a esta plataforma, ya que SMART define una capa de datos de salud construida sobre la especificación FHIR, mediante el sistema de APIs y recursos del estándar. Además, SMART provee de un modelo robusto de autorización a las aplicaciones basándose en el estándar **OAuth 2.0** con **OpenID Connect.** El estándar OAuth 2.0 es un mecanismo de seguridad muy utilizado en la actualidad para autorización, y el cual permite acceder a los recursos sin tener que compartir identidad con la aplicación como tal. Este mecanismo es utilizado ampliamente por compañías como Google, Facebook, Microsoft, Twitter, etc. [15]. FHIR provee material explicativo en su especificación para implementaciones de SMART on FHIR [16].

SMART cuenta con una galería de aplicaciones pública para que desarrolladores puedan compartir sus aplicaciones y para que el personal sanitario busque aquellas de utilidad. [17] Teniendo en cuenta los otros módulos del sistema del proyecto al cual pertenece esta Memoria, se considera interesante la posibilidad de desarrollar el módulo de visualización de adherencia tal que siga el protocolo SMART on FHIR. De esta manera, los profesionales de salud que utilicen la plataforma pueden acceder a ella con las credenciales utilizadas en su sistema de ficha clínica o EHR si es que tiene compatibilidad con SMART, y podría ser exportada a la galería de aplicaciones. Algo similar podría ser el caso para la aplicación móvil, en el cual el paciente pueda utilizar el mecanismo de SMART para ser autorizado con las credenciales disponibles de antemano en los sistemas de ficha clínica o EHR en los cuales se encuentra registrado, y acceder a los recursos médicos que le competen.

#### 2.1.2.3 Implementaciones de FHIR

Existen múltiples implementaciones a nivel global para facilitar el acceso a este estándar. Se hizo una revisión sobre algunas de las alternativas existentes. Entre estas se encuentran principalmente:

- Proveedores de sistemas de registro clínico (EHR) tales como Epic, Cerner, y Allscripts, los cuales se han adaptado a este nuevo estándar en su hardware y software comercial. No son implementaciones como tal pero se menciona su compatibilidad con el estándar.
- Múltiples compañías de tecnología (Google, Microsoft, Apple [18]) ofrecen implementaciones en la nube o aplicaciones con material complementario para facilitar el uso y aprendizaje de este estándar, para construcción de soluciones específicas.
- Soluciones AD-HOC comerciales utilizando el estándar, tal es el caso de Mirth Connect de NextGen [19] y parte de las funcionalidades de IRIS for Health de InterSystems [20], que incluye herramientas para desarrollar y conectar soluciones FHIR.
- Implementaciones Open Source para desarrollar soluciones FHIR específicas, tal es el caso de HAPI FHIR [21].

Se describen brevemente a continuación algunas de las herramientas y plataformas de desarrollo mencionadas.

#### 2.1.2.3.1 Google Cloud

Google ofrece la plataforma Google Cloud HealthCare API **[22],** la cual permite almacenar, integrar y procesar información médica en la nube a través de APIs REST con estándares de salud como HL7 v2 y FHIR (DSTU2, STU3, R4) y garantiza seguridad con su sistema IAM *(Identity and Access Management)*, alto rendimiento y disponibilidad del 99.999999999%. Es una alternativa comercial y su detalle de precios se puede encontrar en **Anexos.**

#### 2.1.2.3.2 Microsoft Azure Cloud

Microsoft ofrece Azure Health Data Services **[23],** los cuales son un conjunto de servicios APIs que soportan estándares de salud como FHIR y DICOM **[24].** Puede transformar datos médicos de legado e IoT en FHIR, además prioriza FHIR por sobre otros estándares. La información médica se puede conectar con la Suite de Microsoft y aplicaciones SMART on FHIR, y cumple con requisitos de seguridad pertinentes de información de salud protegida (PHI). Es una opción comercial y su detalle de precios se puede encontrar en **Anexos.**

#### 2.1.2.3.3 Apple

Apple ofrece soluciones de salud a través de Apple Healthcare **[25],** una línea de productos de salud que incluye el framework Open Source ResearchKit **[26]** para aplicaciones con fines investigativos y CareKit **[26]** para aplicaciones de seguimiento médico. Dentro de esta línea se encuentra Apple Health Records **[27],** una aplicación móvil que permite a los pacientes ver y compartir sus registros médicos electrónicos en sistemas EHR. Está disponible en Estados Unidos, Canadá y Reino Unido, y se rige por el estándar SMART on FHIR. Aunque es gratuita, no está disponible en Chile y sólo es compatible con dispositivos Apple.

#### 2.1.2.3.4 IBM

IBM ofrece IBM Integration Bus **[28],** un software de integración que se utiliza para conectar aplicaciones con diferentes esquemas de información en diversas áreas, incluida la salud. Ofrece el Healthcare Pack v4.0 **[29]** como un complemento a este bus de integración para conectar aplicaciones clínicas, dispositivos médicos y sistemas de imagenología, y es compatible con estándares como HL7 v2, FHIR y DICOM. Proporciona patrones reutilizables para la conversión entre estándares. Es parte de un programa de licencias de software para empresas y socios Passport Advantage de IBM, por lo que no cuenta con información de tarifas fijas.

#### 2.1.2.3.5 Firely

Firely ofrece una serie de productos relacionados con el estándar FHIR **[30]**, incluyendo el **servidor FHIR Firely** construido en .NET, el cual proporciona todas las operaciones CRUD que una implementación FHIR necesita y posee opciones de almacenamiento. Es escalable, personalizable y compatible con SMART on FHIR. Es una alternativa comercial y sus precios varían, pero ofrece una prueba gratuita de una semana. También ofrece **Simplifier.net**, una plataforma de colaboración para crear y validar especificaciones FHIR personalizadas, que incluye la herramienta Forge para desarrollar perfiles y extensiones de FHIR, lo cual es muy útil en el contexto de la Memoria. Esta

herramienta es de uso gratuito de forma limitada para un solo proyecto, y su página de tarifas puede encontrarse en **Anexos**.

Además, Firely proporciona un **SDK de código abierto para .NET** que facilita la construcción de soluciones basadas en FHIR, que provee librerías para interactuar con servidores FHIR mediante el esquema API REST y modelos de datos para adherirse al estándar. Cuenta con herramientas de serialización, validación de recursos y soporte para múltiples versiones de FHIR. De forma adicional se destaca el canal de Gino Canessa **[31],** el cual es un desarrollador de Microsoft enfocado en soluciones de salud en FHIR, que provee tutoriales introductorios para desarrollar aplicativos FHIR en .NET.

#### 2.1.2.3.6 Mirth Connect

NextGen Healthcare ofrece Mirth Connect, un motor de integración de sistemas médicos que facilita la interoperabilidad de datos en varios formatos, incluyendo HL7 v2, v3, DICOM y FHIR **[19]**. Cuenta con una versión gratuita con funcionalidades básicas y soporte comunitario, pero que no provee soporte nativo con FHIR, ya que este se encuentra restringido a una versión Premium que ofrece características más avanzadas y compatibilidad con FHIR, la cual se encuentra sujeta a una tarifa flexible dependiendo de la necesidad. Al no contar con compatibilidad FHIR en su versión gratuita, se descarta su uso en relación a esta Memoria, pero podría ser útil en una integración de datos más compleja.

#### 2.1.2.3.7 IRIS for Health

InterSystems es una compañía de soluciones software que incluye la línea IRIS, centrada en administración de datos, analíticas y manejo de grandes volúmenes de datos **[20]**. Dentro de esta línea, se encuentra IRIS for Health, la cual es una variante enfocada en aplicaciones de salud, compatible con estándares como FHIR, HL7 y DICOM. Ofrece un repositorio FHIR y APIs REST (similar a un servidor FHIR), así como analíticas de datos de salud. Puede desplegarse en servicios en la nube populares y es compatible con varios lenguajes de programación. Es una alternativa comercial con tarifas flexibles y pruebas gratuitas disponibles **[32]**. Aunque se descarta como una opción para los objetivos de la Memoria debido a la existencia de soluciones Open Source, se considera como una posible opción a futuro para necesidades escalables de interoperabilidad. Un documento más detallado con las características claves de IRIS for Health puede encontrarse en **Anexos.**

Otro producto de esta compañía es el sistema médico TrakCare, que se detalla más en la sección TrakCare.

#### 2.1.2.3.8 HAPI FHIR

Smile CDR (Clinical Data Repository) es una compañía de software de salud que se centra en el uso de estándares de salud abiertos, como FHIR, con el objetivo de proporcionar soluciones personalizadas para el procesamiento de datos, interoperabilidad de sistemas y cumplimiento de estándares en el campo de la salud **[33]**. Uno de sus productos destacados es HAPI FHIR, una implementación completa del estándar FHIR en Java, de carácter Open Source **[21]**. HAPI FHIR ofrece una forma flexible de agregar capacidades FHIR a una aplicación, mediante un ecosistema

compuesto de múltiples herramientas como se ve en la Figura 11. Éstas incluyen un serializador para conversión entre modelo de datos a recursos FHIR, un cliente para transmitir datos hacia un servidor FHIR externo, un servidor para procesar datos y convertirlos a recursos FHIR, y un servidor FHIR con arquitectura JPA que contiene un módulo de persistencia o base de datos integrada.

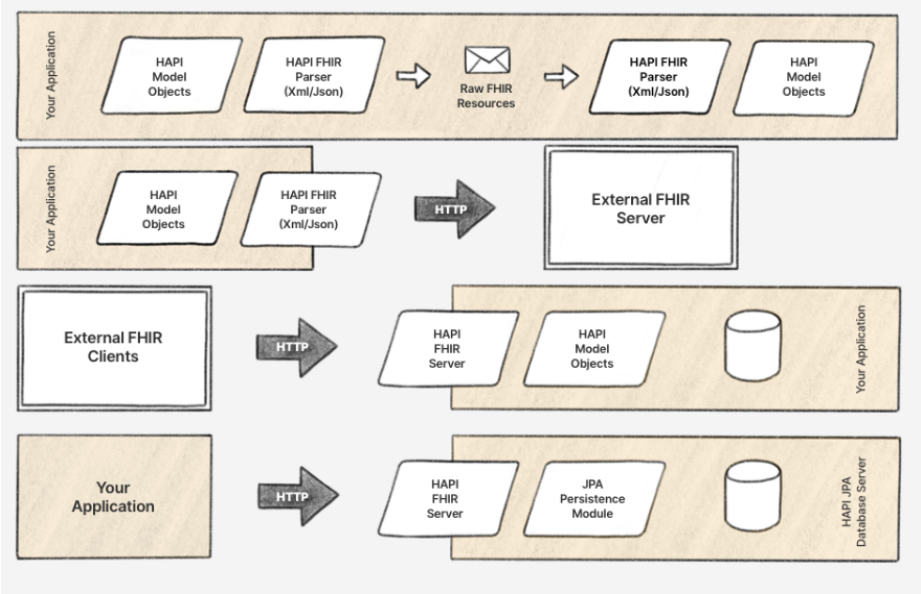



*Figura 11: Ejemplos de uso de HAPI FHIR, obtenido de [21]*

En particular, se destaca esta última implementación HAPI FHIR JPA que provee persistencia JPA (Java Persistence API), y consta de todas las capas necesarias para gestionar las llamadas HTTP FHIR desde aplicaciones externas, proporcionar operaciones CRUD para recursos FHIR, y gestionar la persistencia de estos recursos en una base de datos de tipo Apache Derby, aunque es compatible con otros gestores de BDs (ej: Postgres). También posee un validador de recursos FHIR interno y compatibilidad con perfiles y extensiones. El proyecto HAPI FHIR JPA se encuentra de manera gratuita en un repositorio en GitHub [34] y se puede personalizar según los requerimientos del usuario. En la Figura 12 se ofrece un breve esquema de la arquitectura de HAPI FHIR JPA. El servidor recibe llamadas HTTP FHIR externas, y utiliza los *Resource Providers,* encargados de dar operaciones CRUD a cada recurso FHIR, para procesarlas. Luego pasan por una capa DAO (Data Access Object) encargada de separar capa de negocio con capa de almacenamiento, y provee la lógica de almacenamiento y recuperación de recursos FHIR. Esta capa se enlaza con la base de datos como tal (por defecto Derby).

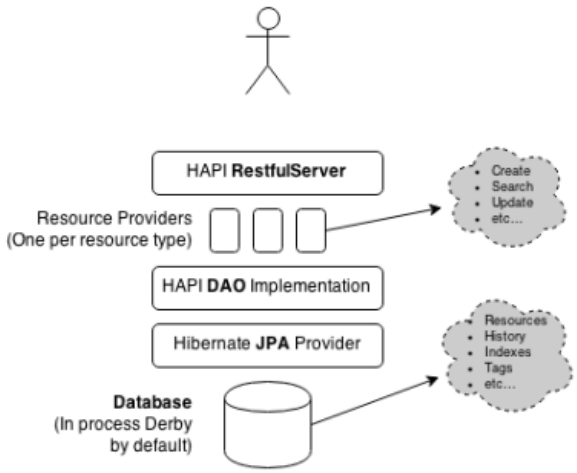



*Figura 12: Esquema de arquitectura de servidor HAPI FHIR JPA, obtenido de [21]*

El esquema de la base de datos del servidor JPA contiene múltiples tablas para gestionar índices y optimizar consultas, siendo las tablas principales **HFJ_RESOURCE** y **HFJ_RES_VER** para almacenar información de recursos FHIR y sus versiones. En la Figura 13, se puede ver un esquema de estas tablas principales, en conjunto a **HFJ_FORCED_ID** que da soporte a IDs alternativos de recursos. La tabla **HFJ_RES_VER** guarda el contenido de los recursos como tal y sus versiones. Puede encontrarse más detalle sobre este modelo en **Anexos.**

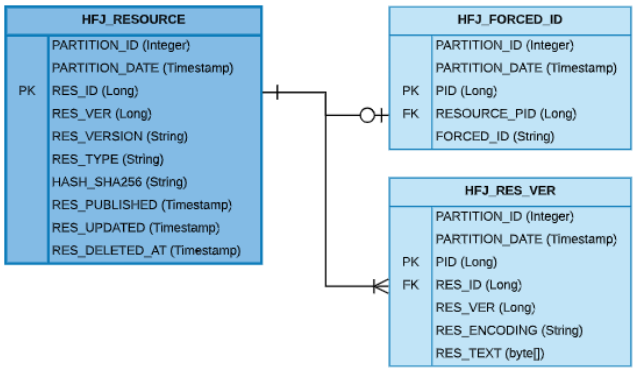



*Figura 13: Entidad HFJ_RESOURCE y su relación con HFJ_FORCED_ID y HFJ_RES_VER, parte del modelo de datos de HAPI FHIR JPA, obtenido de [21]*

En el contexto de la salud en Chile, el CENS (Centro Nacional en Sistemas de Información de Salud) utiliza herramientas basadas en HAPI FHIR en su infraestructura de testing [35]. En resumen, el servidor HAPI FHIR JPA es una herramienta valiosa para el procesamiento y transmisión de datos de salud en formato FHIR, el cual es gratuito y se actualiza constantemente para cumplir con los últimos avances del estándar, brindando una solución práctica y concisa para el procesamiento de datos de salud y se alinea con los objetivos de la Memoria.

### 2.1.3 SNOMED CT

SNOMED CT (Clinical Terminology) es una extensa terminología de salud multilenguaje utilizada para estandarizar conceptos médicos en sistemas de información de salud. Administrada por SNOMED International [36], se actualiza anualmente y consta de más de 350,000 conceptos médicos. Proporciona un sistema de códigos único para representar información clínica de manera precisa y consistente, además de poseer mapeo de conceptos con otras terminologías. Su modelo lógico incluye **conceptos** (representados por un código numérico único procesable por máquina, sin significado humano)**, descripciones** (texto legible por humanos que representa el significado del concepto) y **relaciones** (asociaciones entre dos conceptos para ser procesables por máquina) para capturar y expresar información médica.

Este estándar se utiliza en más de 80 países y es requerido o recomendado en sistemas de salud, incluyendo Estados Unidos, Reino Unido y Australia. Ofrece ediciones internacionales (a la fecha la última versión en inglés del 31 de julio de 2023 y en español 30 de abril de 2023 [37]) y locales para adaptarse a las necesidades de cada país [36]. Los países miembros tienen acceso gratuito a la última versión de la terminología, mientras que los países no miembros deben pagar una tarifa de afiliado. Chile se encuentra dentro de los países miembros desde 2013 [38] y la gestión de licencias se encuentra a cargo del MINSAL [39]**,** que además ha definido el uso de SNOMED CT como sistema de terminología oficial en sistema de salud nacional, tanto público como privado. Los beneficios de su uso incluyen la representación consistente y sin ambigüedad de información clínica, la interoperabilidad global de datos de salud, la eliminación de barreras idiomáticas y reducción de costos al evitar duplicaciones de información y errores.

En la Figura 14 se muestra un ejemplo de concepto para el medicamento **Metroprolol de 25 mg en formato de comprimidos.** Contiene su código identificador, su descripción con múltiples sinónimos y el cuadro de la derecha presenta las relaciones con otros conceptos.

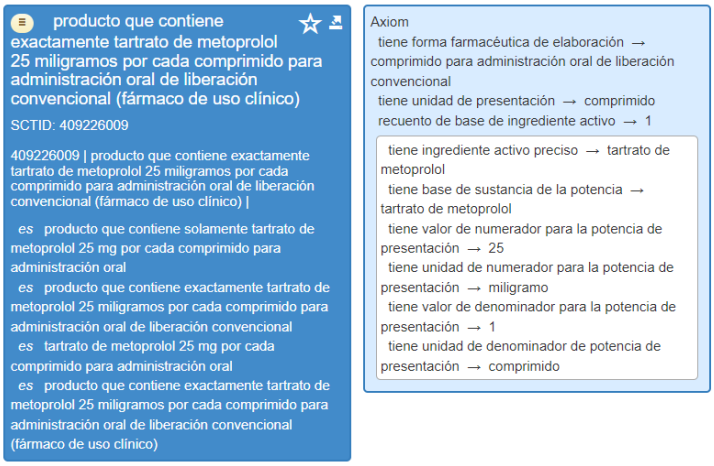



*Figura 14: Información de concepto de un producto de medicación Metroprolol de 25 mg en formato de comprimidos, obtenido de SNOMED Browser*

En el contexto de esta Memoria, FHIR hace múltiples referencias a sistemas de código SNOMED CT, especialmente en la representación codificada de medicamentos. Para una mejor gestión de estos códigos, se solicitó una licencia en Chile para su uso mediante archivos RF2 (Release Format 2) que permiten contar con la terminología completa en una implementación de sistema de salud, sin embargo no se obtuvo respuesta de las autoridades encargadas de estas licencias. Debido a esto, se trabajó con el navegador de SNOMED CT, el cual permite utilizar las funciones de SNOMED CT sin tener licencia, para obtener estos códigos de conceptos médicos manualmente.

## 2.2 Trabajo relacionado previo

En esta sección, se hizo una breve revisión de soluciones propuestas que sirven para cumplir requerimientos de usuario similares, es decir, proveer de una interoperabilidad entre dispositivos móviles y sistemas de fichas clínica o que permitan enviar información del paciente a sistemas de registro clínico (EHR), con el fin de obtener ideas de cómo orquestar la propuesta de arquitectura. Luego, se describe la arquitectura de un paper relacionado, el cual propone un bus de interoperabilidad basado en mensajería HL7, que sirve como inspiración para la proposición de arquitectura descrita en esta Memoria. Debido a esta solución existente, no se hizo una revisión más profunda sobre otras alternativas.

### 2.2.1 Soluciones relacionadas

En una encuesta realizada por la Asociación Estadounidense de Hospitales **[40]** entre 2012 y 2017, se destaca un creciente esfuerzo por mejorar la interoperabilidad entre los sistemas de registro médico internos y externos de hospitales en EE.UU. La interoperabilidad dentro de los hospitales aumentó del 30% en 2012 al 74% en 2017. Sin embargo, la integración de información externa en sistemas internos ha progresado más lentamente, pasando del 25% al 29% entre 2014 y 207, con hospitales más grandes mostrando una mayor interoperabilidad. Varias barreras persisten en este desafío, como la dificultad en el uso debido al formato de información, la integración directa con EHRs, la identificación de pacientes entre sistemas y la disponibilidad de datos.

Entre las soluciones revisadas que permiten una comunicación entre dispositivos móviles y EHRs, se destacan aplicaciones móviles como MyChart **[41]** de Epic, que permite a los pacientes acceder a su información de salud registrada en EHRs, compartir registros médicos con otros sistemas informáticos, opciones de telemedicina y programar citas. Apple Health Records también permite a los usuarios acceder y compartir información de registros médicos desde dispositivos Apple, y en 2018 Apple liberó una API para utilizar los registros médicos obtenidos de Apple Health Records **[42]** en aplicaciones de salud personalizadas. Medisafe es una de las primeras aplicaciones en integrarse con Apple Health Records **[43]** y ofrece una funcionalidad precisa de importación de prescripciones médicas, lo cual se quiere replicar para los objetivos de esta Memoria. Además, en 2016 Medisafe ya había desarrollado mecanismos para obtener las prescripciones médicas de EHRs de Epic y Cerner sin necesidad de usar la API de Apple, mediante el uso del estándar FHIR **[44]**. Dicho eso, no se descarta la utilidad de Apple Health Records por su acuerdo con más proveedores de salud

(alrededor de 40) y que facilita la obtención de los permisos necesarios para acceder a esta información médica.

La transmisión de datos en el ámbito clínico generalmente está dirigida desde registros médicos a aplicaciones de pacientes, pero no en sentido inverso, lo que limita la participación activa de los pacientes en su atención médica. Es crucial desarrollar soluciones que aprovechen las herramientas e implementaciones FHIR descritas anteriormente para permitir a los pacientes involucrarse más en su salud, en tareas como registrar la trazabilidad de ingesta medicamentos, planes nutricionales y gestión de enfermedades crónicas, lo que contribuirá a evaluar la eficacia de los tratamientos.

### 2.2.2 Bus de interoperabilidad HL7 para sistemas de legado (Paper)

Se presenta el paper ***Using a Health Level 7 Interoperability Bus to Support Legacy Systems in the Health Domain [45]***, en el cual se habla sobre un bus de interoperabilidad HL7 utilizado como middleware para facilitar la transmisión de información entre distintos sistemas de bases de datos, dispositivos médicos, y sistemas de información de salud.
El principal objetivo del trabajo realizado en este paper, es crear un bus de interoperabilidad que sirva como un centro de datos y de comunicación para todos los sistemas de información de salud involucrados, mediante el estándar HL7 v3. Este bus provee la conexión para que los clientes H7 (aplicativos de cualquier tipo que utilicen el estándar), los sistemas de legado y dispositivos médicos puedan enviar mensajes basados en el estándar HL7 v3 e interactuar con la información de EHRs.

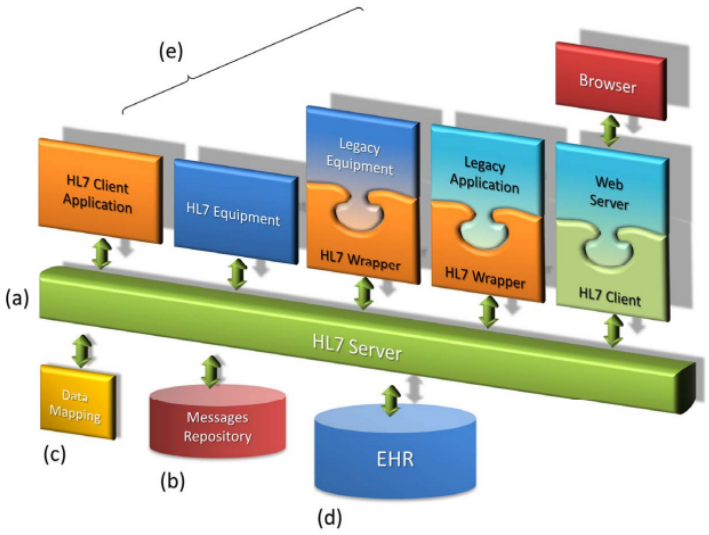



*Figura 15: Arquitectura de bus de interoperabilidad HL7, obtenida del paper descrito*

En Figura 15, se puede ver un esquema que describe las comunicaciones procesadas por este middleware de servidor HL7. Los componentes son:

**a) Servidor HL7:** Capa de abstracción entre un cliente y una base de datos, que corresponde al bus de interoperabilidad como tal. Es responsable de recibir e interpretar mensajes H7, realizar operaciones sobre la base de datos y enviar respuestas a los clientes como mensajes H7. Este servidor es el único capaz de realizar operaciones sobre la BD. Se usa una tabla de mapeo **(c)** para identificar la estructura de la información, y realizar el procedimiento almacenado asociado a esa estructura **(b)**. Si es necesario retornar algo al cliente, ocurre un mapeo de los datos a un mensaje HL7 incluyendo la información en la plantilla correspondiente en el repositorio de mensajes **(b).**

**b) Repositorio de mensajes:** Contiene plantillas de mensajes para los mensajes soportados por el servidor HL7. Estas plantillas son documentos XML con comentarios especiales que indican la información a rellenar en cada mensaje HL7, y se asocian con procedimientos almacenados que realizan acciones sobre la base de datos según la categoría del mensaje.

**c) Tabla de mapeo de datos:** Módulo que utiliza documentos XML para realizar el mapeo entre los elementos de los mensajes HL7 a campos y procedimientos almacenados de la base de datos, es decir, es el encargado de parsear los mensajes HL7. Utilizado en el envío y recepción de mensajes HL7.

**d) EHR:** Sistema de registros médicos. En el trabajo se utilizó un EHR web llamado STT/SC**,** que comparte una base de datos con un sistema PACS**,** utilizado en varios hospitales de Brasil**.** Funciona como la base de datos de la información médica.

**e) Clientes HL7:** Cualquier sistema que implementa mensajes de HL7 v3. Pueden ser clientes nativos del estándar u otros sistemas de legado no compatibles que se pueden conectar con adaptadores o wrappers HL7, que traducen la salida de los sistemas de legado a una sintaxis adecuada HL7 para enviar al servidor HL7 **(a).** Son implementados de 2 formas, dependiendo del nivel de acceso a los sistemas legado que existe: **wrappers de extensión,** mediante librerías HL7 si hay acceso al código fuente y **wrappers externos** que monitorean los archivos de bases de datos locales y los traducen a mensajes HL7 para comunicar con el bus, cuando no hay acceso al código fuente.

Para evaluar esta arquitectura se realizaron pruebas con respecto al desempeño y costo de integración, con distintos anchos de banda y tipos de composición de datos solicitados a la base de datos, principalmente imágenes médicas (tomografías computarizadas, medicina nuclear, electrocardiogramas y endoscopías). En estas pruebas se obtuvo que el desempeño del bus HL7 para procesar las peticiones de los clientes es mejor cuando se realizan solicitudes de grandes volúmenes de datos y cuando el ancho de banda del cliente es menor que el ancho de banda entre el bus HL7 y la base de datos. Por ejemplo, en casos de 100 kbps, 400 kbps y 1 Mbps, el servidor HL7 tiene mejor desempeño para realizar las operaciones en la base de datos que un acceso directo del cliente. Esto debido a la forma en que se codifica la información de las imágenes dentro del servidor HL7, que optimiza el tiempo de procesamiento.

El costo de integración se midió en personas/horas (prs/h) necesarias para desarrollar los wrappers. En las pruebas realizadas, se utilizaron 182 prs/h para crear **wrappers externos** y se estima que 150 prs/h pueden ser reutilizadas en futuras integraciones. En contraste, el desarrollo de **wrappers de extensión** requirió solo 46 prs/h, lo que lo hace considerablemente más económico. A largo plazo, el costo de desarrollo y despliegue se considera bajo, ya que el wrapper externo es un cliente HL7 que facilita la lectura de documentos XML de sistemas antiguos, lo que simplifica las futuras integraciones. Descontando las 150 prs/h involucradas para el desarrollo del cliente, sólo se necesitarían 32 prs/h adicionales para codificar plantillas de mensajes HL7 para otros servicios.

Dicha arquitectura sirve de inspiración para tomar ciertos componentes de la solución y desarrollar un bus de interoperabilidad en el estándar FHIR. La idea principal que se puede obtener de este paper es extrapolar la arquitectura de un servidor HL7 con plantillas HL7 y wrappers para sistemas no compatibles con el estándar, a una arquitectura similar pero que utilice el estándar FHIR y con el servicio de seguimiento de la adherencia a tratamiento farmacológico de pacientes (ver Propuestas de arquitectura)

## 2.3 Aplicaciones móviles relevantes

En esta sección se describen brevemente las dos aplicaciones que fueron consideradas diseñando la arquitectura. Estas fueron Medisafe, por su popularidad y larga trayectoria como una aplicación que permite a pacientes el registro de ingestas de sus medicamentos, y AFAM 1.0, la cual corresponde a la primera versión de una aplicación que también permite registro de ingestas, pero con comunicación ante un sistema de ficha clínica que habilita al personal sanitario la visualización del índice de adherencia de un paciente a su tratamiento, y fue diseñada dentro del marco de un proyecto FONDEF.

### 2.3.1 Medisafe

Medisafe es una aplicación de gestión de medicamentos ampliamente utilizada a nivel mundial, disponible en dispositivos móviles y con plataforma web, que ha sido uno de los pocos proveedores de este tipo de solución a un nivel tan amplio **[46]**. Ofrece una variedad de funciones:

- Gestión detallada de medicamentos del paciente, en cuanto a su dosificación y horarios de ingesta, y compatibilidad para medicamentos ingresados por usuario o seleccionados desde base de datos, y notificaciones para su debida ingesta.
- Formación de grupos familiares para seguimiento de adherencia
- Más de 90 rastreadores de salud referentes a la salud del paciente.
- Alertas sobre contraindicaciones entre medicamentos (más de 200.000).
- Recordatorios de reabastecimiento de medicamentos y citas médicas.
- Acceso a recursos educativos relacionados con los medicamentos ingresados.

La aplicación también permite a los usuarios importar listas de prescripciones desde EHRs, según lo visto anteriormente. Algunas de las funcionalidades de esta aplicación consideradas en la implementación de la arquitectura de esta Memoria son:

- Ingreso de medicamentos por parte del paciente, con opción de escoger medicamentos registrados en la base de datos.
- Selección de horarios de dosificación, y registro de ingestas por parte del usuario.
- Obtener lista de prescripciones registradas en el sistema de ficha clínica.

### 2.3.2 AFAM 1.0

AFAM Salud 1.0, acrónimo de Adherencia Farmacológica Adultos Mayores, es una aplicación desarrollada dentro del marco del proyecto Fondef ID16AM0007 “Programa psico-educativo transmedial para mejorar adherencia farmacológica del tratamiento antihipertensivo en adultos mayores: Proyecto Piloto” **[47]**. Esta aplicación se enfoca en adultos mayores, por lo que busca entregarles una experiencia más amigable, y les permite registrar la ingesta de medicamentos en horarios predefinidos por personal clínico, además de ofrecer características como un chat comunitario para conversaciones entre pacientes, recursos educativos sobre la HTA y un sistema de seguimiento del estado de ánimo. No utiliza estándares de salud ya que cuenta con su propio modelo de datos. En Anexos se puede encontrar un informe técnico más detallado de la aplicación.

Paralelamente, se encuentra en desarrollo la versión 2.0 de esta aplicación bajo el proyecto FONDEF asociado. Algunos de los aspectos de esta aplicación que se consideran en la implementación de arquitectura de esta Memoria son:

- Ingreso de medicamentos y dosificación por parte del personal clínico, lo que puede incluir horarios específicos.
- Ingreso de estado de ánimo diario por parte del paciente.

## 2.4 Sistemas de ficha clínica relevantes

En esta sección se describe brevemente el sistema de ficha clínica que fue considerado diseñando la arquitectura. Este sistema es TrakCare de InterSystems, el cual es un sistema informático de salud que integra múltiples operaciones del apartado clínico en un solo sistema y provee las funcionalidades de un EHR.

### 2.4.1 TrakCare

TrakCare es un sistema informático de salud desarrollado por InterSystems y utilizado en 29 países para interconectar sistemas clínicos y facilitar la comunicación entre distintos sistemas de salud, además de contar con un EHR interno. Ofrece una amplia gama de conectividad, con estándares de salud como HL7, DICOM, FHIR y SMART on FHIR **[48]**. Para esto, cuenta con múltiples APIs y un registro unificado de pacientes, con más de 400 millones de registros alrededor del mundo. Además permite la integración con otros productos de software de salud, como IRIS for Health de la misma

compañía. Es un sistema comercial con licencias ajustables según las necesidades de hospitales y clínicas, y su acceso en Chile está gestionado por PRONOVA SALUD [49].

En el contexto del proyecto FONDEF asociado a esta Memoria, TrakCare se encuentra operativo en el CESFAM Hualpencillo de Hualpén, Bío Bío, lugar dónde se implementará a modo piloto la propuesta de arquitectura de esta Memoria. Algunos aspectos del sistema TrakCare que se consideran para la implementación son:

- Registro de prescripciones médicas por personal sanitario, lo que incluye los medicamentos y dosificación en un formato estructurado.
- Compatibilidad con distintos modelos de datos y con estándar FHIR.
- No posee la funcionalidad de llevar registro y visualizar adherencia a un tratamiento farmacológico de un paciente, por lo que se necesita la conexión entre los datos del sistema TrakCare y el módulo de visualización de adherencia mediante la arquitectura propuesta.

# 3. Propuestas iterativas de arquitectura

Esta sección contiene la información de las propuestas de arquitectura consideradas durante la ideación de una arquitectura que satisfaga los requerimientos de usuario pertinentes a los objetivos de la Memoria. Las arquitecturas consisten en desarrollos iterativos de una arquitectura base que inicia como un prototipo y se va actualizando con el tiempo para adecuarse o captar nuevos requerimientos surgidos durante el desarrollo de la Memoria. Se describen de manera breve las iteraciones iniciales y se da detalle sobre la arquitectura final.

## 3.1 Requerimientos de usuario

La arquitectura final debe proveer las capacidades para satisfacer los requerimientos de los actores que interactúan con el bus. Dicho esto, se tienen requerimientos funcionales para el paciente mediante la aplicación móvil y para el personal sanitario mediante sistema de ficha clínica y módulo de visualización de adherencia. Y requerimientos no funcionales que abordan aspectos más técnicos de la arquitectura.

**Requerimientos funcionales:**

**Paciente:**

- Paciente puede iniciar sesión en la aplicación para acceder a sus funcionalidades. Puede ser con su rut y una contraseña.
- Paciente puede ver lista de sus prescripciones médicas. Esto incluye datos del medicamento, la frecuencia diaria y dosificación.
- Paciente puede ingresar horarios para recibir alertas de sus prescripciones médicas, según la frecuencia diaria del medicamento.

- Paciente puede registrar ingesta de los medicamentos registrados en su aplicación.
- Paciente puede ingresar medicamentos con dosificación y horarios, que no se encuentren necesariamente en su lista de prescripciones.
- Paciente puede registrar su estado de ánimo una vez al día.

**Personal sanitario:**

- Personal puede ingresar prescripciones médicas que serán visibles para el paciente.
- Personal puede ver información del tratamiento farmacológico de un paciente en específico, como controles médicos, prescripciones y su nivel de adherencia reflejado en distintos gráficos, dentro del módulo de visualización.
- Personal puede obtener y ver el índice de adherencia de un paciente durante un periodo señalado, dentro del módulo de visualización.

**Requerimientos no funcionales:**

**Idoneidad funcional:**

- Arquitectura debe permitir la transmisión de datos entre ambos actores utilizando el estándar FHIR o el modelo de datos especificado.

**Confiabilidad:**

- La arquitectura debería contar con mecanismos para no perder la información referente a tratamiento farmacológico y su adherencia.
- La arquitectura debe soportar la interacción de múltiples usuarios de manera simultánea es decir, debe soportar la carga estimada de pacientes y personal sanitario.

**Desempeño:**

- Transmisión de datos debe ser eficiente en términos de tiempo, aprovechando el formato ligero del estándar FHIR.

**Usabilidad:**

- Adaptación de arquitectura a nuevos sistemas de ficha clínica o aplicaciones no debe implicar mayor dificultad.

**Seguridad:**

- Información sensible de tratamientos farmacológicos debe ser protegida mediante mecanismos de seguridad (autorización).
- Información de pacientes debe ser mostrada sólo al paciente correspondiente (autenticación) y personal sanitario autorizado.

**Compatibilidad:**

- Compatible con aquellos sistemas que cuenten con los mecanismos de comunicación definidos para la arquitectura, tanto aplicaciones móviles como sistemas de ficha clínica.

**Mantenibilidad:**

- Componentes de la arquitectura deben ser diseñados considerando modularidad, para mantener el código de unidades independientes de software.
- La arquitectura debe contar con un repositorio con historial de versiones para facilitar desarrollo y edición para posteriores modificaciones.

- La arquitectura inicialmente debe soportar información de patología de hipertensión arterial (HTA), pero debe ser diseñada para ser adaptada a otras patologías.

**Portabilidad:**

- Arquitectura debe poder hospedarse en entornos locales y remotos.
- La lógica subyacente de la arquitectura debe ser portable con dificultades mínimas ante un posible cambio de plataforma de desarrollo (Ejemplo: IRIS Health).

## 3.2 Descripción de primera arquitectura

La idea inicial de una arquitectura es construida a partir del paper ***Using a Health Level 7 Interoperability Bus to Support Legacy Systems in the Health Domain*** [45]. La arquitectura de este paper cuenta con un servidor centralizado que opera como bus de interoperabilidad entre los actores interesados, los cuales son clientes HL7, tanto nativos como sistemas de legado adaptados con wrappers, y un registro de datos como un EHR o sistema de ficha clínica. Para este caso, se adapta la arquitectura al estándar FHIR y se toman la mayoría de componentes de la arquitectura anterior, como se muestra en Figura 16.

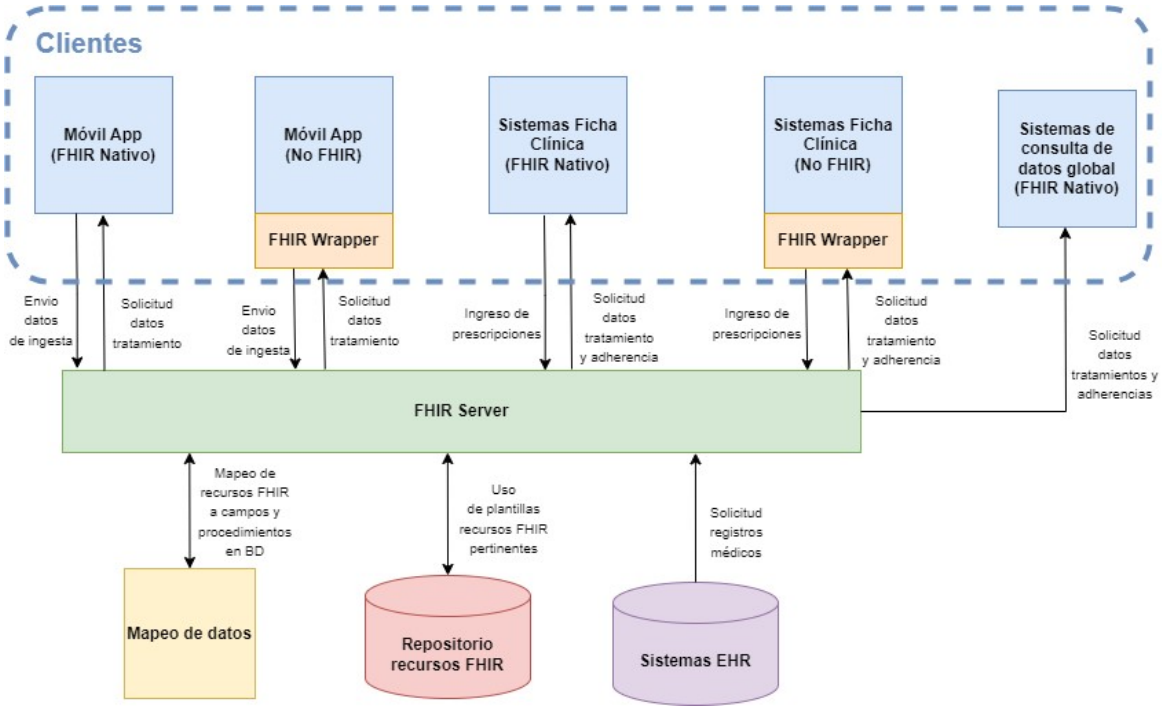



*Figura 16: Arquitectura inicial basada en bus HL7, elaboración propia*

Esta arquitectura cuenta con 3 componentes principales:

- El bus de interoperabilidad o **FHIR Server** que conecta a **clientes** con una base de datos, en este caso un **EHR**. Para esto procesa solicitudes con la especificación de APIs en FHIR, realizando transacciones a través de recursos FHIR. Cuenta con estructuras auxiliares, un módulo de **mapeo de datos** que convierte los recursos FHIR a los campos y/o

procedimientos almacenados en el sistema EHR, y un **repositorio de recursos FHIR** que contiene las estructuras de datos para construir todos los recursos de la especificación R4.

- Los **clientes FHIR**, que son cualquier sistema o aplicación que implemente el estándar FHIR y se comunique con el bus de interoperabilidad. Incluyen aplicativos nativos de FHIR que transmiten su información mediante recursos FHIR, y aplicativos no nativos de FHIR, que se conectan mediante un **wrapper FHIR** que transforma el modelo de datos en recursos FHIR. Los clientes en este caso son:
  - **Aplicaciones móviles,** que envían información de ingestas de medicamentos y solicitan datos de tratamiento, como prescripciones activas.
  - **Sistemas de ficha clínica,** los cuales solicitan información de tratamiento y adherencia para utilizarlos en el **módulo de visualización específico** y **módulo de consulta de datos y visualización global,** que se plantea como un módulo a desarrollar en el transcurso del proyecto macro. Se considera obtener la información de prescripciones médicas asociadas a través de los sistemas de ficha clínica, o desde el EHR directamente.
- Una base de datos o **EHR,** que contiene la información clínica relevante para los objetivos de la Memoria, como datos del paciente, tratamiento de HTA y prescripciones activas.

La arquitectura propuesta se delineó tempranamente en el proyecto, y es sólo una interpretación básica de la arquitectura del paper al estándar FHIR. Por lo tanto, no se han revisado a fondo aún las diferencias arquitectónicas del estándar HL7 v3 y FHIR. Dicho eso, ambas arquitecturas cumplen objetivos similares pero diferentes. La arquitectura del paper integra sistemas de legados y dispositivos médicos sin interfaces de redes al estándar HL7 v3, mientras que la arquitectura propuesta integra sistemas compatibles con protocolos modernos como HTTP y API REST al estándar FHIR. Otra consideración es el lugar de almacenamiento de la información de adherencia, pudiendo contar con un repositorio para esto externo al EHR, debido a la variabilidad entre distintos EHRs, y a la posibilidad de no contar con permisos para ingresar información de ingesta de medicamentos sobre estos. Según lo visto en la sección de soluciones relacionadas, en general las aplicaciones disponibles no permiten al usuario ingresar esta información en un EHR de manera directa, por lo que se avala más la decisión de contar con una base de datos propia para información de adherencia. En esta etapa aún no se realizan implementaciones y sirve como base para las siguientes iteraciones.

## 3.3 Descripción de segunda arquitectura

Esta siguiente iteración surge gracias a la experimentación con algunas de las implementaciones FHIR mencionadas anteriormente e investigación de documentación y videos del estándar FHIR, para construcción de prototipos de transmisión de información mediante el estándar. Se describe la arquitectura en la Figura 17.

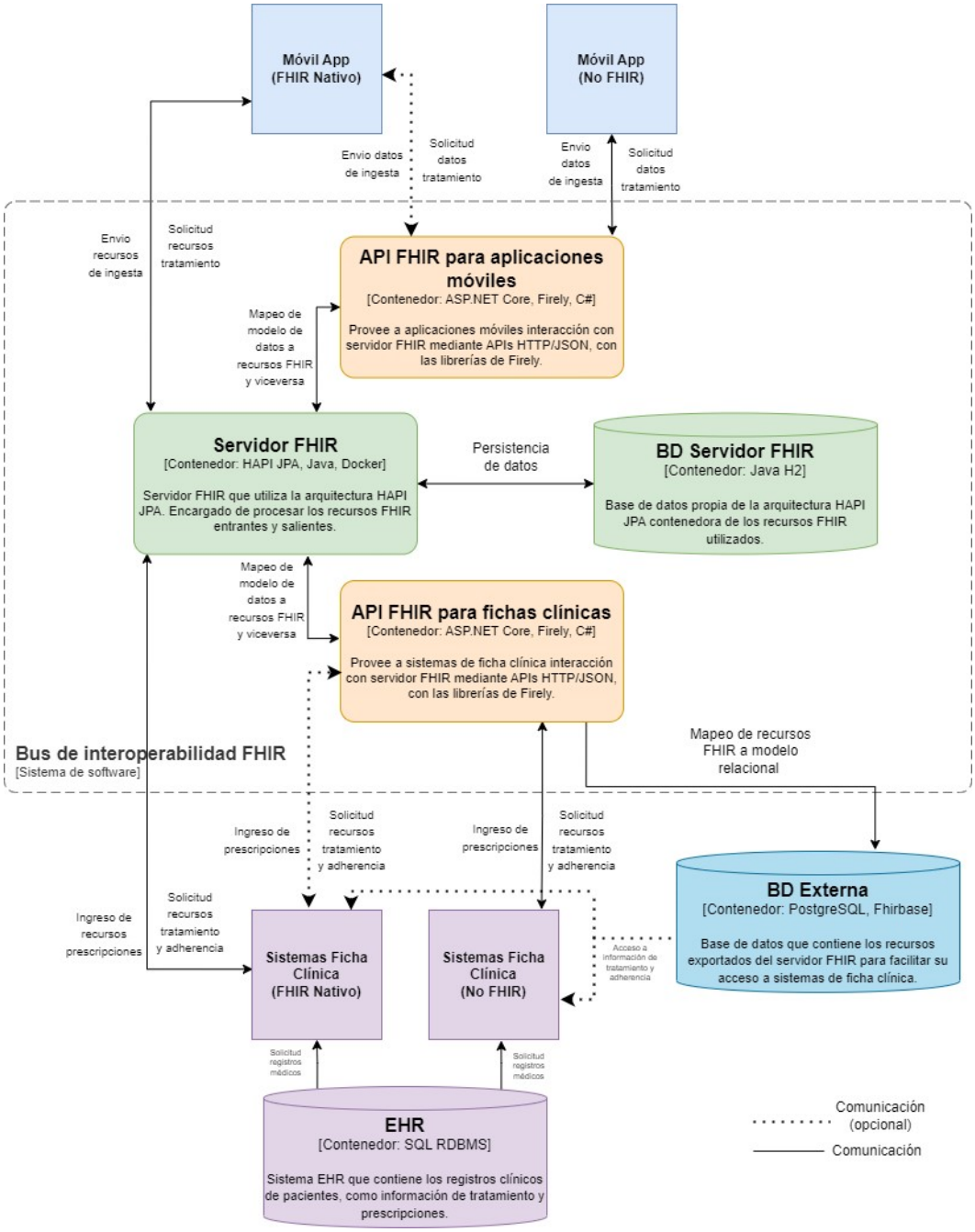



*Figura 17: Diagrama C4 de contenedores correspondiente a segunda arquitectura, elaboración propia*

Esta arquitectura consiste en un **bus de interoperabilidad** que contempla 2 componentes principales, haciendo un paralelo con el servidor FHIR y los módulos auxiliares de la arquitectura anterior:

- Un **servidor FHIR** que sirve para procesar las solicitudes de los clientes FHIR respectivos, y que cuenta con una **base de datos** que almacena la información relevante de tratamiento farmacológico y adherencia en formato de recursos FHIR, lo que cubre la necesidad de la

arquitectura anterior sin necesidad de operar sobre EHR. Los aplicativos nativos de FHIR pueden interactuar directamente con el servidor mediante la especificación API REST de FHIR.

- **2 conjuntos de APIs Wrapper** para aplicaciones móviles y sistemas de ficha clínica no nativos del estándar FHIR, las cuales se utilizan para mapear un modelo de datos a recursos FHIR, en este caso utilizando el modelo de datos propuesto. Las operaciones implementadas por estas APIs tratan con todos los requerimientos de usuario funcionales, tanto para pacientes como para personal sanitario, y por lo tanto trabajan con recursos de prescripciones, recursos de ingestas de medicamento, recursos de información de pacientes, recursos de controles médicos y tratamientos, entre otros (véase Arquitectura de datos FHIR subyacente).

Por otro lado, los **clientes** se mantienen de la arquitectura anterior. Es decir, aplicaciones móviles y sistemas de ficha clínica, tanto nativos de FHIR como no nativos, que se comunican entre sí a través del bus descrito anteriormente. En ese sentido, los módulos de visualización, tanto específico como global, pueden ser categorizados como sistemas de ficha clínica.

También se mantiene el **EHR**, que en esta etapa se presenta como un proveedor de información de pacientes y tratamientos a los sistemas de ficha clínica. Se consideró una **base de datos externa,** que simplemente replica la información contenida en el servidor FHIR pero con un esquema de datos similar al modelo de datos propuesto, para facilidad de acceso a clientes de ficha clínica. Se planteaba transformar los recursos FHIR a tuplas de la BD mediante un software llamado **Fhirbase**, pero se cuestiona la real utilidad de este módulo ya que los sistemas de ficha clínica pueden acceder a la información que replica con los mecanismos de conexión mencionados.

Esta arquitectura puede satisfacer gran parte de los requerimientos de usuario, tanto funcionales como no funcionales, a excepción de aspectos de seguridad, que no se consideran aún en esta etapa. Se define la arquitectura para operar con sistemas no nativos mediante el modelo de datos propuesto pero se podría adaptar en un futuro para el mapeo de otros modelos de datos. Utiliza comunicación API REST principalmente por el paradigma de comunicación de FHIR, y la compatibilidad vista anteriormente en aplicaciones populares como Medisafe y TrakCare.

# 4. Detalle de la propuesta de arquitectura final

En esta sección se detalla toda la información referida a la arquitectura final propuesta y la implementación de software realizada para su utilización en el piloto del CESFAM Hualpencillo.
Se da una descripción general de la arquitectura con algunos detalles de implementación, en conjunto con la arquitectura global del sistema (que incluye la aplicación móvil AFAM 2.0, sistema de ficha clínica y EHR TrakCare, y módulo de visualización), y el modelo de datos formulado para satisfacer los requerimientos. Finalmente, se ven detalles de la implementación piloto.

## 4.1 Descripción de arquitectura

La arquitectura está compuesta principalmente de los mismos componentes de la arquitectura anterior, con algunas adiciones de seguridad y diferencias en las conexiones establecidas con los actores de ficha clínica. En este punto de madurez se distingue claramente la diferencia entre sistemas de ficha clínica y **registros de salud electrónicos (EHR)**, por lo que se hace la separación de las funcionalidades de estos sistemas, con el fin de aterrizar sus propósitos en la arquitectura. Se describe la arquitectura en Figura 18.

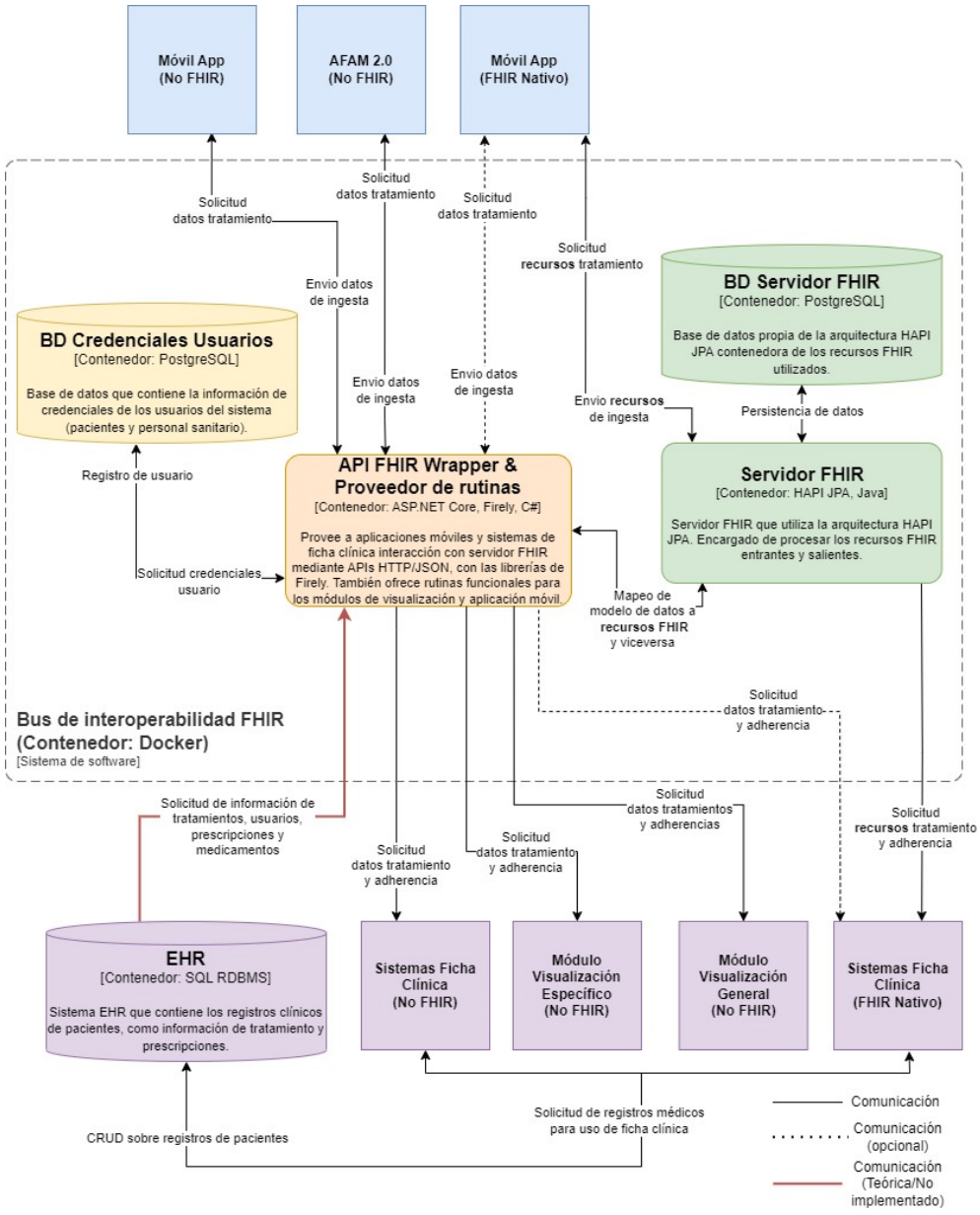



*Figura 18: Diagrama C4 de contenedores correspondiente a arquitectura final, elaboración propia*

La arquitectura es muy similar a la anterior. con un bus de interoperabilidad que contempla 2 componentes principales:

- Un **servidor FHIR** para procesar las solicitudes de los clientes FHIR respectivos y con un repositorio de recursos FHIR que sirve como único mecanismo de almacenamiento para guardar información relevante al tratamiento farmacológico y adherencia del paciente (**paradigma de almacenamiento FHIR**). Está construido en base al software Open Source de HAPI FHIR JPA, en específico con el proyecto *starter,* que provee de base todas las capacidades de un servidor FHIR y soporte para todos los recursos de la especificación R4. La implementación JPA cuenta con una base de datos interna, la cual se encarga de guardar la información de los recursos FHIR. Finalmente se descarta la **base de datos externa** de la propuesta anterior, ya que la información es accesible directamente por clientes nativos o mediante el **módulo de APIs Wrapper FHIR** descrito a continuación para clientes no nativos, y no es necesario replicar de nuevo información que se encuentra en el servidor y además en el **EHR** al que se conecta.

- Un **módulo de APIs Wrapper FHIR y de rutinas,** que toma la idea de la arquitectura anterior de proveer un conjunto de APIs para aplicaciones móviles y sistemas de ficha clínica no nativos del estándar FHIR, y que mapean el modelo de datos propuesto a recursos FHIR, y viceversa, para cubrir los requerimientos funcionales. La diferencia en este caso es que se unifican los proveedores de API a un único proveedor que pueden utilizar ambos clientes para evitar redundancia, y que provee de rutinas adicionales como **cálculo de adherencia de un paciente** o **inicio de sesión.** Estas APIs fueron construidas de manera propia utilizando las librerías SDK de Firely, que dan soporte a operaciones de recursos con servidores FHIR en el framework de desarrollo ASP.NET Core. En el aspecto de seguridad, este módulo ofrece mecanismos de autenticación y autorización mediante una base de datos adicional PostgreSQL encargada de almacenar la información de credenciales de los usuarios, y que entrega los token de autorización adecuados según el tipo de usuario (véase Mecanismos de seguridad)

Los **clientes** continúan siendo aplicaciones móviles y sistemas de ficha clínica, tanto nativos de FHIR como no nativos, que interactúan con el bus de interoperabilidad y trabajan con los mismos recursos FHIR de la arquitectura anterior. Los clientes nativos de FHIR interactúan directamente con el servidor, realizando las llamadas API definidas en la especificación de FHIR para realizar operaciones CRUD y búsqueda sobre los recursos contenidos. Mientras que los clientes no nativos deben tercerizar su interacción mediante los módulos wrapper de APIs definidos. De manera opcional, los clientes nativos también pueden utilizar el conjunto de APIs wrapper. Se incluyen como casos específicos de clientes la **aplicación móvil AFAM 2.0** y los **módulos de visualización de tratamiento específico y general.** En ese sentido, estos aplicativos se han desarrollado como no nativos de FHIR y que pueden utilizar el **módulo de APIs wrapper** con el modelo de datos propuesto.

El **EHR** es el proveedor de registros médicos de pacientes, lo que incluye información de tratamiento e historial médico, construidos en base a gestores de BDs relacionales (**RDBMS**) y cuya implementación depende mucho del proveedor de salud. Mantiene su rol de la arquitectura anterior, pero cambia su posición y vías de comunicación. En esta iteración se plantea una conexión directa entre el EHR y el componente proveedor de APIs y rutinas, el cual solicita la información referente a los pacientes, tratamientos y prescripciones, para su mapeo al estándar FHIR y posterior uso en todas las funcionalidades provistas por la arquitectura. Es decir, las prescripciones se obtienen desde este sistema y no directamente de los sistemas de ficha clínica como en la arquitectura anterior.

Con respecto a la interoperabilidad entre distintos sistemas, si bien la arquitectura busca proveer interoperabilidad, se enfoca en la comunicación entre aplicaciones móviles y sistemas de ficha clínica, más que la variabilidad de los sistemas en particular. Dicho esto, es necesario realizar ajustes según la variabilidad del modelo de datos de las distintas aplicaciones y sistemas de ficha clínica no nativos de FHIR, pero sirve como una base o prueba de concepto para elaborar sistemas más complejos. En cuanto a los sistemas nativos de FHIR, la arquitectura provee una interoperabilidad sin problemas, pues adhiere correctamente a la especificación del estándar en la representación de los datos médicos. La arquitectura fue diseñada desde un principio considerando la diferencia entre funcionalidades para el caso de un paciente (**caso específico**) o múltiples pacientes (**caso global**). Las funcionalidades provistas por la arquitectura tratan sólo con el primer caso, y se plantea como trabajo a futuro del proyecto FONDEF el caso global. Dicho eso, para el caso de sistemas nativos de FHIR se dispone de las operaciones de búsqueda inherentes del estándar FHIR que permiten solicitar información de múltiples individuos ingresando los parámetros de búsqueda adecuados y sólo habría que trabajar el caso del módulo wrapper para aplicativos no nativos.

Es necesario hacer la distinción entre **EHRs** (registros de salud electrónicos) y **sistemas de ficha clínica**. Los EHRs son los encargados, como su nombre señala, de almacenar los registros médicos de los pacientes de un proveedor de salud. Esto incluye toda la información relacionada al historial médico de un paciente, ya sea tratamientos, prescripciones, exámenes, patologías, información de muestras e imagenología, entre otras cosas. Mientras que los sistemas de ficha clínica pueden considerarse una denominación genérica para cualquier sistema informático de salud que utilice la información contenida en los EHRs para algún propósito, particularmente relacionado con la elaboración y visualización de fichas clínicas. La diferencia entre ambos sistemas queda clara en este punto y ha sido motivo a lo largo del desarrollo de la Memoria de algunas inconsistencias reflejadas durante la ideación e implementación de la arquitectura, además de no tener interacción directa con un sistema de ficha clínica real o EHR, por lo que sólo se han podido realizar simulaciones (véase Escenarios de compatibilidad). Por esto, la conexión directa descrita en Figura 18 entre EHR y el bus de interoperabilidad se indica como **teórica**, ya que no ha sido implementada, y toda información obtenible a partir de EHR como **pacientes**, **prescripciones** y **medicamentos** ha sido ingresada de forma manual en el servidor FHIR. Sin embargo, esta implementación futura involucraría reutilizar los servicios del componente Wrapper FHIR y de rutinas, y cambiarlos para involucrar conexiones SQL con el gestor de base de datos específico del EHR sólo con permisos de lectura y así obtener

toda la información requerida para la trazabilidad de la adherencia. Así, el bus de interoperabilidad no modificaría nada en el EHR real y se mantendría como un sistema independiente integrable con sistemas médicos sin preocuparse de alterar información crítica de pacientes.

La capa de seguridad mencionada en el **componente de rutinas** consiste de un sistema de registro e inicio de sesión, que valida la identidad de los usuarios según su rol (paciente o médico) y autoriza el acceso a los recursos FHIR pertinentes. Cabe mencionar que esta capa de seguridad se encuentra implementada sólo para la interacción con el servidor proporcionada por la **capa wrapper**, en cambio no hay medidas de seguridad implementadas para el acceso directo al servidor. Es necesario que se implemente una capa de seguridad para este caso, lo que requiere agregar modificaciones al código fuente del servidor HAPI utilizado, probablemente mediante el uso de **interceptores de autorización** como lo recomienda la documentación del mismo, y con la misma base de datos de credenciales. Esta implementación debe ser considerada antes de finalizar el proyecto. Sin embargo, para la etapa piloto no es necesaria tal implementación inmediata ya que sólo se utilizará la capa wrapper para interactuar con el servidor, debido a la naturaleza no nativa de FHIR de los clientes (AFAM 2.0 y módulos de visualización). Los usuarios registrados en el sistema de seguridad se han agregado de forma manual, considerando que el registro será parte del proyecto AFAM mismo, y las credenciales se almacenan de forma encriptada. Se considera utilizar las credenciales preexistentes dentro del EHR para pacientes y personal sanitario, pero eso debería validarse con la contraparte y el uso de la información confidencial de esta índole para este propósito. Otra alternativa similar es integrar el bus de interoperabilidad con SMART on FHIR, pero se deja como potencial trabajo a futuro.

## 4.2 Arquitectura del sistema AFAM 2.0

La arquitectura del sistema AFAM 2.0 define las relaciones y funcionalidades que tienen los distintos componentes del sistema, donde el bus de interoperabilidad de la arquitectura propuesta en esta Memoria cobra gran importancia al ser el Backend principal de los clientes. Este es el sistema que se planea implementar como parte del plan piloto en el CESFAM Hualpencillo, y que es compatible con los requerimientos de usuario descritos anteriormente. Se describe esta arquitectura en la Figura 19, la cual ha sido elaborada principalmente por el jefe de proyecto **Diego Muñoz**.

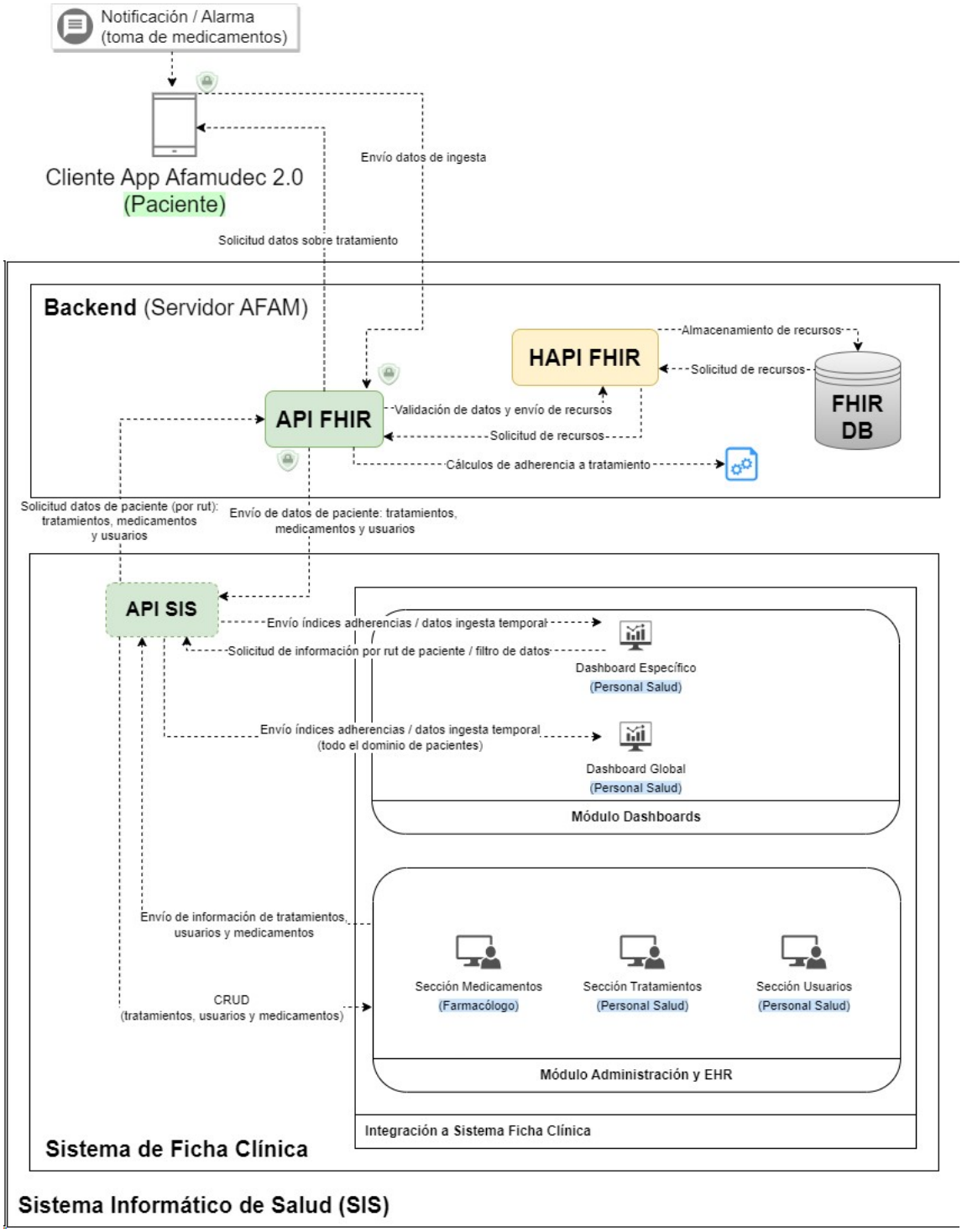



*Figura 19: Arquitectura macro de sistema AFAM 2.0, elaborada por el ingeniero a cargo del proyecto y disponible en [Enlace Arquitectura AFAM 2.0]*

Está compuesta por tres grandes componentes:

- **Aplicación móvil AFAM 2.0:** Es la aplicación principal que utiliza el usuario paciente para participar en el seguimiento de su adherencia. Con ella envía información de ingesta de medicamentos, y solicita la información referente a su tratamiento, en particular las prescripciones de medicamentos realizadas por personal sanitario. Cuenta con un sistema de alertas y recordatorios, y permite al paciente informar de su estado de ánimo diariamente. El desarrollo completo de esta aplicación se encuentra a cargo del integrante **Vicente Schultz** y la completitud de su trabajo será vista en su Memoria correspondiente.

- **Backend (Servidor AFAM):** Corresponde a la arquitectura propuesta en esta Memoria. En el diagrama se describe a la **API FHIR** como punto de interconexión entre la aplicación móvil y el sistema de ficha clínica. Esta API FHIR corresponde al componente wrapper y de rutinas de la arquitectura, y se comunica con el servidor **HAPI FHIR** para realizar las **operaciones** con recursos correspondientes según las solicitudes de los clientes. A su vez, el servidor FHIR procesa las solicitudes interactuando con su base de datos/repositorio de recursos interno, realizando las operaciones CRUD necesarias para satisfacer las solicitudes. Se describe la rutina del cálculo de adherencia realizada por la API FHIR, que solicita los recursos al servidor FHIR sobre las ingestas registradas por el paciente y el número de ingestas prescritas total para realizar el cálculo, y se entrega como respuesta el índice de adherencia a petición del cliente, pero no se almacena directamente en la base de datos, ya que es un indicador en constante cambio. De forma reciente se integra a este componente cambios en los endpoint wrapper para soportar la **carga anticolinérgica** de los medicamentos como parte de su información y conversión a recursos FHIR, de lo cual se encarga el integrante **Aníbal Ibaceta** y lo referente a este tema será visto en su Memoria correspondiente.

- **Sistema de Ficha Clínica:** Corresponde al conjunto de sistemas software que componen un sistema de ficha clínica, para la visualización de información de paciente por parte del usuario personal sanitario o médico. En el diagrama se encuentra la **API SIS,** la cual es la interfaz de software de entrada al sistema de ficha clínica. Esta API recibe solicitudes del dashboard específico de información del paciente mediante su rut, como su tratamiento e índice de adherencia, y solicita al Backend estos datos. De la misma forma, el dashboard global puede solicitar a la API SIS por la adherencia de todo el dominio de pacientes, y ésta deriva la consulta al Backend. Por otro lado, la API SIS se comunica con el módulo de administración y EHR descrito en el diagrama, el cual consta de secciones para medicamentos, tratamientos, y usuarios. La API SIS obtiene la información requerida por el Backend (información de paciente, tratamiento, prescripciones, etc.) a través de estos módulos. Este módulo de administración obtiene la información del EHR subyacente del sistema de ficha clínica. El dashboard específico ha sido construido por el integrante **Claudio Rain**, y el detalle completo puede verse en su Memoria. De la misma manera, el dashboard global está siendo construido por el integrante **Mauricio Furniel**, y el detalle completo será visto en su Memoria.

## 4.3 Modelo de datos

El modelo de datos construido en conjunto con el equipo para esta arquitectura está compuesto por las entidades necesarias para satisfacer todos los requerimientos de usuario, y se ha elaborado en estrecha relación con el estándar FHIR y la estructura de sus recursos. En Figura 20 se muestra un diagrama del modelo de datos, en conjunto con los recursos FHIR pertinentes a cada entidad, de los cuales se entrega más detalle en Arquitectura de datos FHIR subyacente.

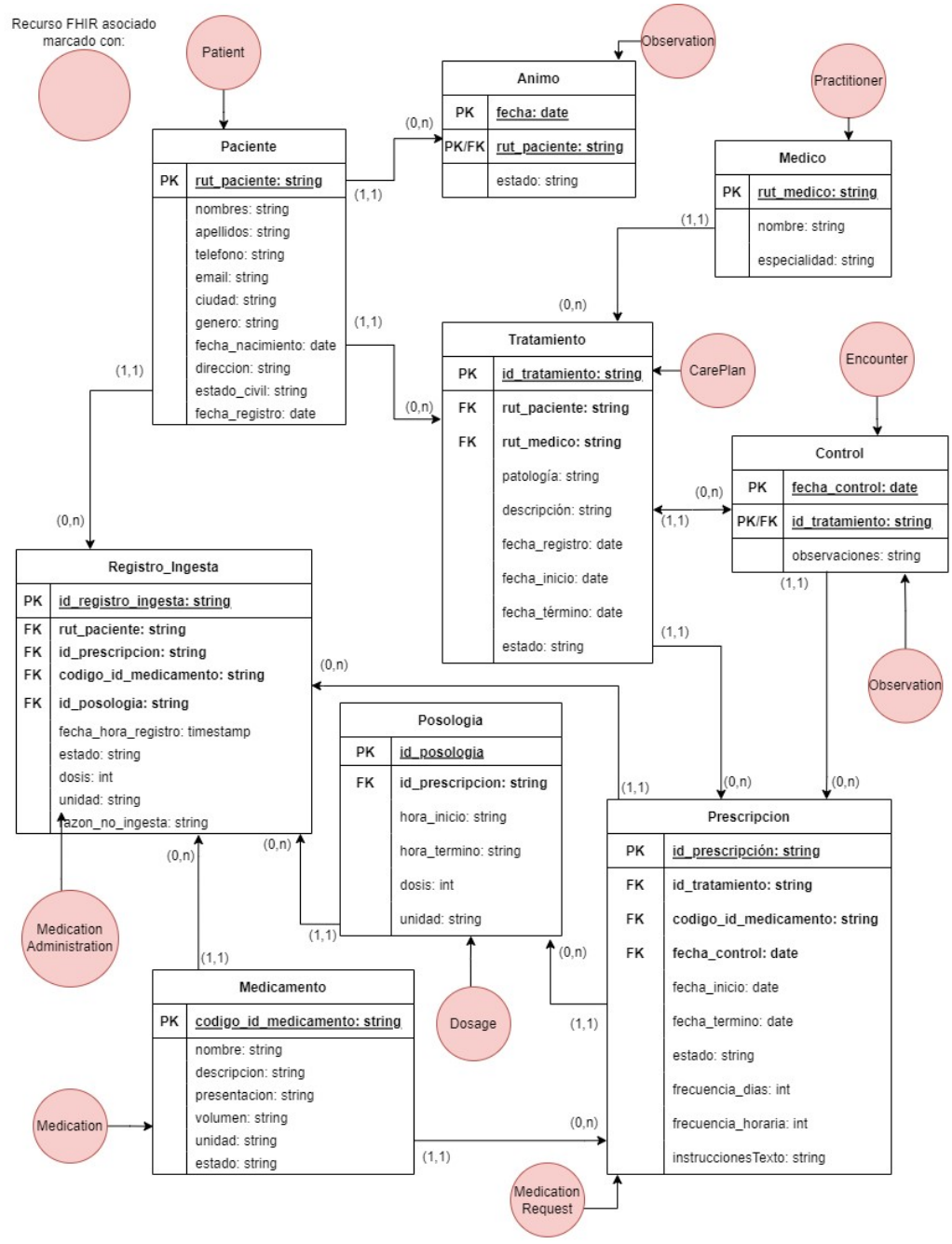



*Figura 20: Modelo de datos utilizado por sistema completo, y su mapeo a los recursos FHIR pertinentes. Disponible en Enlace modelo de datos AFAM 2.0*

A continuación se explica cada una de las entidades y su recurso o elemento FHIR asociado.

- **Paciente:** Contiene los datos personales del paciente como tal. Su *primary key* es el rut del paciente, y tiene campos como nombres, apellidos, teléfono, email, ciudad de residencia, género, fecha de nacimiento, dirección y estado civil. También se tiene la fecha de registro del paciente en el sistema AFAM 2.0. Puede contar con múltiples entradas de **Ánimo,** tener múltiples entradas de **Tratamiento** y registrar múltiples **Registros de Ingesta.** Se encuentra relacionado con el recurso FHIR **Patient.**
- **Ánimo:** Contiene el estado de ánimo registrado una vez al día por el paciente en la aplicación móvil. Sus *primary keys* son el rut del paciente al cual está asociado y la fecha en la que se registra, y tiene un campo para registrar el estado de ánimo (feliz, triste, enojado, etc). Cada entrada de Ánimo pertenece a un único **Paciente.** Se encuentra relacionado con el recurso FHIR **Observation**, que permite registrar observaciones de todo tipo sobre un paciente.
- **Médico:** Contiene información muy básica de un especialista o personal de salud. Su *primary key* es el rut del médico, y tiene campos para su nombre y especialidad. Puede tener múltiples **Tratamientos** de los cuales está a cargo. Se encuentra relacionado con el recurso FHIR **Practitioner.**
- **Tratamiento:** Contiene la información de un tratamiento asignado a un paciente. Su *primary key* es un identificador interno de tratamiento, y posee campos para registrar la patología que cubre el tratamiento, descripción en texto del tratamiento, fechas de registro, inicio y término del tratamiento, y el estado en el cual se encuentra el tratamiento. Un tratamiento pertenece sólo a un paciente con su rut, y está asignado sólo a un médico con su rut. Un tratamiento tiene asignadas múltiples entradas de **Prescripción** y de **Control.** Se encuentra relacionado con el recurso FHIR **CarePlan.**
- **Control:** Contiene una observación registrada durante un control médico del paciente como parte de un tratamiento. Sus *primary keys* son el identificador interno del tratamiento asociado y la fecha del control, ya que se asume que sólo puede tener un control al día para ese tratamiento y posee un campo para registrar la observación. Un control pertenece sólo a una entrada de **Tratamiento,** y se asocia con múltiples entradas de **Prescripción,** que se hayan prescrito durante el control médico. Se encuentra relacionado con los recursos FHIR **Encounter** y **Observation**, ya que es necesario utilizar ambos para registrar la información del encuentro cuando ocurrió el control, y la observación derivada de ese control.
- **Prescripción:** Contiene la información referente a la prescripción de un medicamento durante un tratamiento a paciente. Su *primary key* es un identificador interno de prescripción, y posee campos para las fechas de inicio y término de la prescripción, que dictan la duración de la prescripción. Un campo para el estado de la prescripción (si se encuentra activa o ya se ha completado), dos campos para la frecuencia de días (cada cuántos días se debe seguir posología) y horaria (cuántas veces en un día se debe ingerir) en la que se deben seguir las entradas asociadas de **Posología** y un campo de texto libre para las instrucciones. Una prescripción se asocia con un solo **Tratamiento** a la vez con el identificador interno del tratamiento y sólo se puede haber dado en un solo **Control,** al cual se asocia con la fecha del control. A su vez, la prescripción está hecha para un solo **Medicamento,** al cual se asocia con el código identificador del medicamento. La prescripción puede contener múltiples entradas de

**Posología,** que corresponden a las distintas instrucciones de dosificación en un día. Se encuentra relacionado con el recurso FHIR **MedicationRequest.**

- **Medicamento:** Contiene información de presentación de un medicamento. Su *primary key* es un código identificador de medicamento, el cual se acordó definir como el código SNOMED relacionado con el medicamento en el caso de tener previamente registrado el medicamento en el sistema, o un código identificador único si es ingresado por el usuario. Posee campos para su nombre, una descripción textual del medicamento, el formato de presentación del medicamento (por ejemplo comprimidos), el volumen, que corresponde a la cantidad de principio activo en el medicamento, la unidad de medida asociada con ese volumen, y el estado del medicamento, el cual se refiere a si está en uso dentro del sistema. Un medicamento puede ser parte de múltiples entradas de **Prescripción** y **Registros de Ingesta.** Se encuentra relacionado con el recurso FHIR **Medication.**
- **Posología:** Contiene la parcialidad o el total de las instrucciones de dosificación de una prescripción, según la frecuencia horaria que presente la prescripción. Su *primary key* es un código identificador interno de posología, y posee campos para la hora de inicio y término que determinan el rango de horario en el que se debe ingerir el medicamento, y campos para la dosis a ingerir y la unidad de medida de la dosis. Existen un número de entradas de posología para una prescripción igual a su valor de frecuencia horaria. Una posología se asocia con una única **Prescripción** con el identificador interno de esa prescripción**,** y puede tener múltiples **Registros de Ingesta** asociados. Se encuentra relacionado con el elemento complejo **Dosage** del recurso **MedicationRequest.**
- **Registro_Ingesta:** Contiene la información de una ingesta de medicamento realizada por un paciente. Su *primary key* es un identificador interno de registro de ingesta, y posee campos para guardar la fecha y hora cuando se realizó la ingesta, el estado de la ingesta (si fue completada, de manera parcial, o no se ha ingerido), la dosis del medicamento ingerido y la unidad de esa dosis, y un campo que registra el motivo por el cual no se ha ingerido el medicamento dado el caso. Un registro de ingesta pertenece sólo a un **Paciente** mediante su rut, se asocia solo a una entrada de **Posología** con el identificador interno de posología (aquella que se encuentra dentro del rango horario cuando se realizó la ingesta), y tiene solo un **Medicamento** asociado con el código identificador de medicamento. Se encuentra relacionado con el recurso FHIR **MedicationAdministration.**

## 4.4 Implementación de prototipo de arquitectura

En esta sección se detallan los aspectos relevantes a la implementación de la arquitectura realizada con el propósito de utilizarla en el CESFAM Hualpencillo en el contexto del proyecto FONDEF asociado. Se describen las tecnologías utilizadas en la construcción de los prototipos, la arquitectura de FHIR subyacente a la implementación, el detalle de los componentes individuales de la arquitectura, los mecanismos de seguridad implementados y finalmente se lista el trabajo pendiente al tiempo de la escritura de este informe de MT y que se debe finalizar en el periodo restante del proyecto FONDEF.

### 4.4.1 Herramientas y tecnologías utilizadas

Durante el desarrollo de la implementación de la arquitectura se utilizaron diversas herramientas para sus componentes, incluyendo tecnologías de desarrollo para la arquitectura principal, tecnologías auxiliares y tecnologías para los prototipos de ficha clínica y aplicación móvil (ver Escenarios de compatibilidad).

1. **ASP.NET CORE 6:** Framework de desarrollo de aplicaciones web de alto rendimiento que utiliza el lenguaje de programación C#, perteneciente a Microsoft. Se ha utilizado para construir por completo el componente de API FHIR Wrapper y rutinas. Sigue el paradigma de MVC (Modelo Vista Controlador), aunque para este caso no se usaron herramientas relacionadas con vistas (al ser un componente completamente de Backend), y solo se han usado características de modelo para representar el modelo de datos y controladores que realizan las acciones de mapeo de datos y las rutinas de clientes.
2. **Firely SDK:** Librerías de .NET que entregan soporte completo para trabajar con el estándar FHIR en este framework. Para mayor detalle, se describe el SDK en Firely. Se ha utilizado en la construcción del componente de API FHIR Wrapper y rutinas, utilizando librerías para construcción de recursos FHIR e interacción con el servidor HAPI FHIR.
3. **HAPI JPA Server:** Implementación de Java de un servidor FHIR que soporta las operaciones HTTP REST descritas en la especificación FHIR y cuenta con una base de datos interna que funciona como repositorio de recursos FHIR. Para mayor detalle, se describe esta implementación en HAPI FHIR. Corresponde al servidor FHIR de la arquitectura, y se ha utilizado la implementación *starter* disponible en Open Source realizando ciertos cambios en los archivos de configuración disponibles **[51]**, pero sin modificaciones al núcleo del servidor. Algunos de los cambios realizados incluyen el gestor de bases de datos utilizado, validación de recursos FHIR entrantes, sistema de caché de consultas, y número de conexiones simultáneas permitidas a la base de datos.
4. **PostgreSQL:** Es una tecnología de gestión de base de datos relacionales (RDBMS) de índole Open Source, que opera con comandos SQL para el manejo de la información, y sigue el paradigma ACID (Atomicidad, Consistencia, Isolation/Aislamiento, Durabilidad). Posee mecanismos de control de concurrencia para transacciones simultáneas en una base de datos. Se utiliza como tecnología de base de datos para la base de datos interna del servidor FHIR y para la base de datos de credenciales.
5. **JSON Web Token (JWT):** Es un método ligero y compacto para representar afirmaciones o claims entre dos contrapartes de manera segura, utilizado para otorgar autorización y autenticación en aplicaciones web o APIs. Construye un token con una duración determinada, que se compone por 3 secciones, la cabecera, la carga o payload, y la firma. La cabecera contiene el tipo de token y el algoritmo utilizado para encriptar la firma, el payload consiste en las credenciales del usuario solicitante e información adicional y la firma se construye a partir de la cabecera, el payload y un secreto, con el algoritmo de encriptación designado. Tanto la cabecera como el payload se codifican en base64. La firma sirve para verificar que el mensaje proviene del usuario autorizado y que no se ha modificado en el camino. En la implementación

de la arquitectura se ha utilizado JWT en el componente API FHIR Wrapper y rutinas para establecer seguridad al acceso de los endpoints de los controladores de API.

6. **Docker:** Es una tecnología utilizada para la gestión, creación, despliegue y ejecución de contenedores, los cuales son paquetes livianos y ejecutables de software utilizados para empaquetar aplicaciones, que incluyen sólo lo necesario para ejecutar el software, como código, dependencias y comandos de ejecución. Construye estos contenedores a través de archivos Dockerfile, que especifican todas las dependencias que debe llevar el contenedor, la imagen que se usa como base para crear el software del contenedor (por ejemplo, un contenedor de una aplicación en ASP.NET Core 6 utiliza una imagen base de .NET 6) y los comandos a ejecutar en forma de cascada. Si necesita persistencia de datos, utilizan unidades de software conocidas como volúmenes para persistir esta información a través de distintas ejecuciones del contenedor. Son utilizados principalmente en paradigmas de microservicios y distribución de aplicaciones para distintos ambientes. Para la implementación se ha utilizado Docker como una forma de integrar todos los componentes de software independientes en un contenedor y así poder distribuirlo de manera simple y ejecutable.
7. **Visual Studio Code:** Es un editor de código liviano desarrollado por Microsoft, altamente extensible gracias a un marketplace de extensiones desarrolladas por la comunidad y compañías, que proveen de funcionalidades adicionales para el desarrollo de código, como autocompletado o código boilerplate. Tiene funcionalidades para debugging de código en vivo y ejecución de comandos en consola en la misma interfaz. En la implementación se utilizó únicamente esta herramienta para el desarrollo del código de todos los componentes de software.
8. **Forge:** Herramienta de la suite de Simplifier.net, que facilita realizar operaciones sobre definiciones de recursos FHIR, y en la creación y gestión de perfiles y extensiones FHIR. Para mayor detalle, se describe la herramienta en Firely. Se utilizó esta herramienta para crear dos definiciones importantes que se asocian con la hora de término que tiene un paciente para ingerir un medicamento según la posología establecida. Estas definiciones estructurales corresponden a una extensión que agrega el elemento para representar la hora de término, y un perfil para el recurso **MedicationRequest** que utiliza esta extensión. Se detalla más sobre estas dos definiciones en la siguiente sección.
9. **Postman:** Es una herramienta de desarrollo y testing, que actúa como cliente de APIs y provee una interfaz amigable para interactuar con estos mecanismos. Permite realizar llamadas HTTP al endpoint ingresado y configurar parámetros de autorización, cabeceras, y el cuerpo o carga de la solicitud. Despliega la respuesta recibida, con el código y tiempo de respuesta. Fue altamente utilizado a lo largo de todo el desarrollo para realizar las pruebas correspondientes a la funcionalidad de cada endpoint de API implementado del componente API FHIR Wrapper y rutinas, y probar la conexión directa con el servidor FHIR. Con esto, se buscaba asegurar que cualquier cliente con capacidad de llamadas HTTP pueda utilizar la arquitectura propuesta según lo detallado en secciones anteriores.
10. **React:** Framework de desarrollo de aplicaciones web perteneciente a Facebook y basado en el lenguaje de programación JavaScript. Se encarga principalmente del apartado Frontend de las aplicaciones. Se utilizó para la construcción de un prototipo de ficha clínica con funcionalidades

de comunicación y despliegue de información muy básico, que pudiese utilizar las APIs y rutinas del componente FHIR API FHIR Wrapper correspondientes al personal sanitario. Se decidió utilizar este framework para el prototipo ya que el módulo de visualización del proyecto se desarrolló con el mismo y es una alternativa popular en el desarrollo de los componentes visuales de aplicaciones web.

11. **Flutter:** Framework de desarrollo de aplicaciones móviles perteneciente a Google que utiliza el lenguaje de programación Dart. Permite crear aplicaciones de alta calidad y rendimiento, y se enfoca en la interfaz de usuario y en tener una única base de código para aplicaciones móviles. Se utilizó para la construcción de un prototipo de aplicación móvil con funcionalidades de comunicación y despliegue de información muy básico, que pudiese utilizar las APIs y rutinas del componente FHIR API FHIR Wrapper correspondientes al paciente. Se decidió utilizar este framework para el prototipo ya que la aplicación móvil AFAM 2.0 del proyecto se encuentra en desarrollo con el mismo, y es una alternativa recientemente popular en la construcción de aplicaciones móviles.
12. **Git:** Sistema de control de versiones distribuido que sirve para mantener un historial de los cambios y las distintas versiones en el desarrollo de un proyecto de software, de manera local y remota, lo que habilita el trabajo colaborativo. Hay múltiples repositorios disponibles para mantener bases de código de manera remota con esta tecnología, siendo uno de los más populares GitHub. Git se utilizó durante todo el desarrollo de los componentes de la arquitectura propuesta para mantener una base de código disponible de manera local y en GitHub. Luego se integró con la base de código del proyecto completo, que incluye la base de código de los módulos de visualización y la aplicación móvil. Este repositorio se puede encontrar en **Anexos.**
13. **Apache JMeter:** Es una aplicación hecha en Java y de índole Open Source para realizar pruebas y mediciones de desempeño de aplicaciones web y servicios APIs [52]. Permite simular múltiples clientes realizando peticiones al servidor o servicio objetivo, y así medir tiempos de respuesta, y medir la capacidad del sistema para una carga específica. Soporta varios protocolos, con especial interés en las llamadas HTTP. Posee la funcionalidad de generar reportes y gráficos con la información de desempeño recabada. Se utilizó en la fase de evaluación de la arquitectura simulando peticiones de múltiples clientes, con el fin de tener una estimación de la carga soportada e información de desempeño del componente API FHIR Wrapper y rutinas, y del servidor FHIR de manera directa.

### 4.4.2 Arquitectura de datos FHIR subyacente

En esta sección se describe la arquitectura de datos FHIR que se estableció para representar la información referente a la adherencia de tratamiento farmacológico. Dado que FHIR no requiere la implementación y uso de todos los recursos disponibles en la especificación, se utilizan sólo los recursos FHIR asociados al modelo de datos. Se describe la estructura de cada recurso utilizado, lo que incluye los elementos a rellenar en cada recurso, ya que en FHIR la mayoría de elementos son opcionales, y las relaciones existentes entre estos recursos. Al haber trabajado con el estándar en su versión R4, algunos elementos pueden haber cambiado en versiones posteriores.

Como se había señalado en los elementos de FHIR, existen algunos elementos en común compartidos por todos los recursos, que corresponden a **id, meta y text.** El **id** refleja el identificador único del recurso en el servidor, el cual es el principal mecanismo de referencia entre recursos dentro del servidor (referencias lógicas). **meta** contiene los siguientes metadatos: **versionId** (versión del recurso), **lastUpdated** (última actualización del recurso) y **source** (código alfanumérico indicando el origen del recurso). Y **text** contiene una narrativa describiendo el contenido del recurso.

Los elementos de tipo **CodeableConcept** se repiten en múltiples recursos y representan códigos provenientes de un sistema de códigos. Se componen principalmente de un elemento **system** (que representa el sistema de origen)**,** un **code** (el código como tal) y un **display** (una representación en texto del código). Por otro lado, los elementos de tipo **Reference** son el principal mecanismo de referencia entre recursos, y están compuestos por elementos **reference** (en minúsculas, es la referencia literal del recurso) y **display** (una representación en texto de la referencia).

En las siguientes tablas se enumeran los elementos de cada recurso FHIR utilizados. En **Anexos** se puede encontrar el detalle más específico de la implementación y mapeo de cada elemento de recurso.

| **Recurso** | **Patient** | **Observation (Ánimo)** | **Observation (Control)** |
|---|---|---|---|
| **Representa** | Paciente | Ánimo | Control |
| **Elementos** | 1. identifier<br>2. active<br>3. name<br>4. gender<br>5. birthDate<br>6. address<br>7. maritalStatus | 1. status<br>2. code<br>3. subject<br>4. performer<br>5. value | 1. status<br>2. code<br>3. subject<br>4. encounter<br>5. performer<br>6. dataAbsentReason<br>7. note |

| **Recurso** | **Medication** | **MedicationRequest** | **MedicationAdministration** |
|---|---|---|---|
| **Representa** | Medicamento | Prescripción y Posología | Registro_Ingesta |
| **Elementos** | 1. identifier<br>2. code<br>3. status<br>4. form<br>5. ingredient | 1. meta.profile<br>2. status<br>3. intent<br>4. medication<br>5. subject<br>6. encounter<br>7. authoredOn<br>8. requester<br>9. basedOn<br>10. dosageInstruction<br>11. substitution | 1. status<br>2. medication<br>3. subject<br>4. effective<br>5. request<br>6. dosage |

| **Recurso** | **Practitioner** | **CarePlan** | **Encounter** |
|---|---|---|---|
| **Representa** | Médico | Tratamiento | Control |
| **Elementos** | 1. identifier<br>2. active<br>3. name<br>4. qualification | 1. identifier<br>2. status<br>3. intent<br>4. category<br>5. title<br>6. description<br>7. subject<br>8. encounter<br>9. period<br>10. created<br>11. author<br>12. addresses<br>13. activity | 1. status<br>2. class<br>3. serviceType<br>4. subject<br>5. participant<br>6. period<br>7. diagnosis |

### 4.4.3 Componentes de arquitectura

En esta sección se describen los aspectos de implementación específicos de cada componente de la arquitectura propuesta. Cabe mencionar que todos los componentes se han empaquetado en un contenedor Docker, de forma de tener los 4 componentes (servidor FHIR, base de datos de servidor FHIR, módulo de APIs wrapper y rutinas, y base de datos de credenciales) como una unidad de software ejecutable de manera sencilla, lo que incluye dos volúmenes para la persistencia de información de las dos bases de datos. Se espera hospedar en un servidor en línea para su acceso remoto por los distintos clientes.

#### 4.4.3.1 Servidor FHIR

La implementación consiste en el código fuente del proyecto “starter” de la implementación HAPI FHIR JPA, con algunas modificaciones leves de configuración y el cambio en la base de datos interna H2 que incluye por defecto. No se necesitaron mayores cambios de la implementación disponible ya que provee todas las funcionalidades base para almacenar y realizar interacciones mediante llamadas HTTP con recursos FHIR, lo cual es lo justo y necesario para esta solución. La totalidad de operaciones HTTP disponibles según la especificación R4 se encuentra en **Anexos**, que incluyen operaciones CRUD y de búsqueda.

Debido al enfoque de FHIR de utilizar referencias entre recursos como la principal forma de relacionarlos entre sí, es necesario tener algunos recursos preexistentes en el sistema para poder referenciarlos en recursos relevantes con la adherencia a tratamiento farmacológico. Estos recursos son **Medication**, **Patient** y **Practitioner.** Para ello se pueden ingresar de forma manual utilizando los mecanismos de creación de recursos disponibles, o utilizar la conexión con EHR descrita anteriormente para obtener la información de estos, transformarlos a recursos y almacenarlos en la base de datos. En un principio se ingresará la información de forma manual en recursos FHIR.

| PRINCIPIO ACTIVO MEDICAMENTO | FORMATO DE PRESENTACIÓN |
|---|---|
| AMLODIPINO | COMPRIMIDO 5MG ; 10MG |
| ATENOLOL | COMPRIMIDO 50MG |
| ATORVASTATINA | COMPRIMIDO 10MG ; 20MG |
| CAPTOPRIL | COMPRIMIDO 25MG |
| CARVEDILOL | COMPRIMIDO 6.25MG ; 12.5MG ; 25MG |
| ENALAPRIL | COMPRIMIDO 10MG ; 20MG |
| ESPIRONOLACTONA | COMPRIMIDO 25MG |
| FUROSEMIDA | COMPRIMIDO 40MG |
| HIDRALAZINA | COMPRIMIDO 50MG |
| HIDROCLOROTIAZIDA | COMPRIMIDO 25MG |
| ISOSORBIDE | COMPRIMIDO 10MG |
| LOSARTAN | COMPRIMIDO 50MG RANURADO |
| METILDOPA | COMPRIMIDO 250MG |
| NIFEDIPINO | COMPRIMIDO 20MG ACCIÓN RETARDADA |
| PROPRANOLOL | COMPRIMIDO 40MG |

*Figura 21: Medicamentos relacionados con tratamiento de HTA provistos por Dra. Claudia Sáez, ingresados inicialmente como recursos en servidor FHIR*

Los anteriores recursos pueden considerarse fijos ya que no tendrán mayores cambios en el tiempo dentro del servidor. Pero los siguientes recursos necesitarán la conexión con EHR en un futuro para ser actualizados en tiempo real con los eventos que rodean el tratamiento del paciente: **CarePlan**, **Encounter**, **Observation** (para los controles médicos, no para ánimo) y **MedicationRequest**. Es decir, si ocurren cambios en el tratamiento, nuevos controles, o cambios en las prescripciones activas del paciente, será necesario que el servidor consulte primero con EHR sobre estos recursos y si siguen siendo idénticos a los disponibles dentro del servidor, mantenerlos o de lo contrario, actualizarlos acordemente. Como no se ha tenido conexión con EHR, de momento estos recursos se han ingresado de forma manual con tratamientos farmacológicos de prueba existentes para realizar las pruebas de evaluación necesarias. El recurso **MedicationAdministration** no presenta estas dificultades ya que se ingresa directamente por parte del cliente de aplicación móvil.

Las otras dos adiciones importantes al servidor corresponden a los recursos **StructureDefinition** de la extensión **TimingHourWindows** y el perfil **MedicationRequestHoursWindowProfile**. Si se fuera a conectar este servidor de manera directa con un cliente FHIR, es necesario que el cliente tenga conocimiento de estas dos definiciones para poder interpretar los recursos extendidos. Se pueden añadir como parte del elemento **contained** de los recursos **MedicationRequest** que lo utilicen, pero depende mucho de la implementación FHIR propia del cliente el cómo los procesaría.

Las modificaciones de configuración corresponden a:

- **Activar validación de recursos entrantes al servidor.** Esto fuerza que cualquier cliente FHIR que intente crear recursos FHIR en el servidor, lo haga siguiendo las restricciones impuestas por el estándar para cada tipo de recurso. De lo contrario, notifica con un error indicando los elementos erróneos del recurso.
- **Cambiar tiempo de caché de consultas.** Por defecto, el servidor almacena resultados de búsqueda por 1 minuto, por lo que si se ingresa nueva información durante ese periodo y se busca, no se verá reflejada. Por aquella limitación, se cambia el tiempo de caché a 1 segundo.
- **Aumentar el número de conexiones concurrentes de la base de datos.** La cantidad de conexiones activas o en pausa por defecto se encuentra en 10, y se aumentó a 30 para

considerar un mayor número de pacientes o sistemas de ficha clínicas concurrentes, y así reducir las colas de espera al llegar al límite de conexiones.

- **Cambio de gestor de base de datos:** Se cambia la base de datos H2 por una base de datos postgreSQL, ya que la primera se encontraba en un archivo temporal que se reiniciaba cada vez que se publicaba el servidor con nuevos cambios.

#### 4.4.3.2 FHIR Wrapper y Rutinas

La implementación consiste en una aplicación web API en ASP.NET Core 6, utilizando las librerías del Firely SDK para la construcción de recursos (**Hl7.Fhir.Model**) y llamadas HTTP (**Hl7.Fhir.Rest**) para interactuar con el servidor FHIR. Este componente se puede considerar un proveedor de endpoints de APIs, que se dividen en 2 grupos: aquellos que mapean la información entre el modelo de datos y recursos FHIR, y aquellos que proveen algunas rutinas o servicios más específicos a los clientes.

El proceso de inicialización de esta aplicación web de APIs configura el middleware necesario que se utiliza durante el manejo de peticiones entrantes. Algunos de estos middleware son:

- Servicio de autenticación y autorización mediante JSON Web Token.
- Conexión con la base de datos de credenciales.
- Políticas de CORS (Cross-Origin Resource Sharing) para permitir a otros dominios acceder a la aplicación web API.
- Conexión al servidor FHIR mediante un cliente FHIR que es utilizado en todos los controladores de la web API mediante inyección de dependencias.

La aplicación se ejecuta en la dirección IP de la máquina local (no localhost), lo que permite que otros clientes en la misma red puedan conectarse. Cuenta con un módulo de funciones auxiliares. Este módulo en un principio cuenta con solo una función de conversión de fechas, usada para adaptar fechas al formato requerido por los recursos FHIR (YYYY-mm-dd).

##### 4.4.3.2.1 Conjunto de APIs Wrapper

La sección Wrapper de la aplicación está compuesta por **controladores** y **modelos,** siguiendo el esquema MVC de ASP.NET Core 6. Los controladores realizan toda la lógica de realizar las operaciones CRUD básicas y mapeo sobre el servidor FHIR en respuesta a peticiones entrantes, mientras que los modelos contienen las clases que reflejan el modelo de datos descrito anteriormente. Se han construido controladores y modelos para cada uno de los recursos FHIR participantes del sistema.

- **PatientController,** utiliza el modelo **PatientClass**.
- **ObservationController**, utiliza el modelo **ObservationControlClass** para observaciones de controles y el modelo **ObservationAnimoClass** las observaciones de ánimo.
- **PractitionerController,** utiliza el modelo **PractitionerClass.**
- **CarePlanController,** utiliza el modelo **CarePlanClass.**
- **EncounterController,** utiliza el modelo **EncounterClass.**

- **MedicationRequestController,** utiliza los modelos **MedicationRequestClass** para las prescripciones y **DosageItem** para las posologías.
- **MedicationAdministrationController,** utiliza el modelo **MedicationAdministrationClass.**
- **MedicationController,** utiliza el modelo **MedicationClassSNOMED** para medicamentos con código SNOMED disponible, y **MedicationClassExt** para medicamentos sin código SNOMED y que en su lugar utilizan un identificador externo.

Todos estos controladores soportan las operaciones CRUD (Create, Read, Update, Delete) para operar sobre los recursos respectivos. Cada operación es un endpoint diferente, y aquellos que requieren información del cliente la reciben como un JSON del modelo respectivo a ese controlador en el body de la petición HTTP (p. ej: para crear un paciente, el endpoint de creación de **PatientController** solicita un JSON con los campos del modelo **PatientClass**).

- Las operaciones de **crear** traducen el modelo de datos a un recurso FHIR.
- Las operaciones de **lectura** obtienen el recurso FHIR con la primary key respectiva de la entidad, pero devuelven el recurso FHIR como tal y no lo traducen al modelo de datos (para eso están las APIs más específicas de la otra sección).
- Las operaciones de **actualizar** obtienen el recurso FHIR con la primary key respectiva de la entidad y le aplica las modificaciones ingresadas en el JSON recibido.
- Las operaciones de **eliminar** utilizan la primary key respectiva de la entidad y eliminan el recurso del servidor asociado con ella.

#### 4.4.3.2.2 Conjunto de APIs de rutinas

La sección de rutinas de la aplicación también se compone de **controladores** y **modelos,** que son utilizados para procesar las solicitudes de requerimientos de usuario más específicos.

Se dispone sólo de tres controladores de momento:

- **AuthenticationController** que maneja la autenticación de usuario y entrega los permisos necesarios para utilizar los endpoint de la aplicación. Utiliza el modelo **Usuario.** (más detalle en Mecanismos de Seguridad)
- **MobileController y VisualizationController,** encargados de los requerimientos específicos de estos clientes. Estos requerimientos se han recabado en el transcurso del proyecto. Los endpoints para estos requerimientos que se encuentran implementados o planeados a implementar son:
    - **Prescripciones de un paciente con información de los medicamentos.** Utilizado por la aplicación móvil y el módulo de visualización específico, obtiene los recursos **MedicationRequest** y **Medication** asociados al tratamiento HTA del paciente y los mapea al modelo de datos del sistema.
    - **Registros de ingesta de medicamentos con la prescripción y horario de posología asociados, y que incluye información de los medicamentos.** Utilizado por el módulo de visualización específico, obtiene los recursos de **MedicationAdministration, MedicationRequest y Medication** asociados al tratamiento HTA del paciente y los mapea al modelo de datos del sistema.

- **Cálculo de índice de adherencia a tratamiento dentro de un periodo especificado.** Utilizado por el módulo de visualización específico, se solicitan los recursos **MedicationRequest y MedicationAdministration** asociados al tratamiento HTA del paciente en un periodo determinado, y los procesa para realizar un cálculo de $\frac{Cantidad\ total\ ingerida\ en\ periodo}{Cantidad\ total\ prescrita\ de\ periodo}$.

Si es necesario añadir más endpoints de requerimientos en el futuro, se pueden utilizar estos controladores para ello.

#### 4.4.3.3 Bases de datos

La implementación de las dos bases de datos (servidor FHIR y credenciales) se ha realizado con el gestor postgreSQL. Se utiliza este gestor por la compatibilidad con el esquema de datos de la arquitectura JPA del servidor, a diferencia de otra alternativa popular como MySQL, la cual se encuentra obsoleta en esa arquitectura. La decisión de tener 2 bases de datos independientes se debe a que HAPI FHIR necesita de una base de datos completa en la que ejecuta su algoritmo inicial para implementar el esquema relacional de la arquitectura JPA.

El esquema relacional que sigue la base de datos del servidor FHIR puede encontrarse en **Anexos.** Por otro lado, el esquema relacional que sigue la base de datos de credenciales consiste de una única tabla **users** con campos **rut, password** y **role,** con el último indicando si el usuario tiene rol de paciente o médico.

### 4.4.4 Mecanismos de seguridad

El mecanismo principal de seguridad para la arquitectura se basa en el uso de JSON Web Token (JWT) como mecanismo ligero de autenticación y autorización de usuario, que se alinea muy bien con el uso de una aplicación web API como la construida en el componente Wrapper y de rutinas.

La implementación de este mecanismo se ha hecho mediante librerías de seguridad .NET encargadas de la construcción del token y validación del mismo. Al construirse la aplicación web API y configurar el middleware relacionado con el JWT, se establece el esquema a seguir por los tokens y sus 3 secciones, incluyendo un secreto definido por la aplicación a utilizar en la verificación de firmas.

El controlador de **AuthenticationController** se utiliza para los inicios de sesión del usuario. Este capta las credenciales de **rut** y **password,** y las valida con la base de datos. Cabe mencionar que el campo **password** dentro de la base de datos se encuentra encriptado con un algoritmo simétrico **Blowfish**, por lo que al ingresar las credenciales en el controlador se aplica este mismo algoritmo a la **password** y se comparan los valores encriptados. Nunca se revela la información de la contraseña. Al momento de validar la identidad se construye el token respectivo, al cual se agregan los permisos de médico o paciente, según el **role** del usuario.

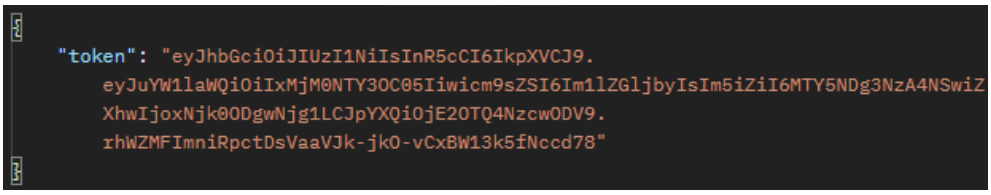



*Figura 22: Ejemplo de token producido por AuthenticationController*

Luego, se solicita ingresar este token como parte del header de autorización en las solicitudes HTTP realizadas a los endpoints de la aplicación web API. Parte del middleware de la aplicación incluye mecanismos de **autenticación** y **autorización** que utilizan el JWT para conceder los permisos, y se declaran específicamente en los endpoints que se desea resguardar. Por lo tanto, se ha establecido seguridad en todos los controladores de la aplicación, con especial énfasis en aquellos que involucran información del paciente. Si se intenta acceder a estos endpoints sin el JWT validado, la API devuelve un error de **401 no autorizado.**

Cada endpoint de controlador solicita un JWT con permisos de **paciente** (en caso de ser operaciones que puede realizar el paciente) o permisos de **médico** en caso de ser acciones más administrativas y que el paciente no debería realizar (p. ej: Cambiar información sobre un control, cambiar prescripción ingresada por personal sanitario, etc.). Si no cuenta con el rol respectivo, la API devuelve un error de **403 prohibido.**

### 4.4.5 Trabajo restante

Esta sección comenta sobre aquellos aspectos faltantes sobre la implementación disponible al momento de la redacción de este informe de MT. Estos aspectos se deben desarrollar antes de la finalización del proyecto FONDEF con el fin de cumplir con todas las expectativas de funcionalidad e interoperabilidad que propone la arquitectura.

Hasta el momento, se cuenta con el siguiente listado:

- **Mecanismo para obtener información desde EHR.** Se trata de la conexión teórica mencionada en la descripción de arquitectura. Es necesaria para obtener las últimas actualizaciones con respecto a información de tratamientos, controles y prescripciones de pacientes, y su conversión a recursos FHIR.
- **APIs para estados de ánimo.** Falta agregar endpoints para realizar operaciones CRUD sobre recursos **Observation** relacionados con el estado de ánimo del paciente.
- **Modificar APIs Wrapper de prescripciones y posología para considerar sólo frecuencia.** Actualmente, el endpoint que crea recursos **MedicationRequest** (relacionados con prescripciones y posología) lo hace considerando que se cuenta con información de horarios de ingesta. Esto en realidad no ocurre ya que se espera que sólo se ingrese la frecuencia diaria de

medicamento por parte de personal sanitario, y que el paciente ingrese los horarios de alerta por su cuenta.

- **Agregar los endpoints de requerimientos específicos faltantes.** Actualmente falta implementar el cálculo de índice de adherencia, y la solicitud de registros de ingesta con prescripción y posología asociada. Esto también considera nuevos requerimientos que podrían surgir en lo que queda del proyecto, como aquellos relacionados con el módulo de visualización global.
- **API para registro de usuario.** Actualmente los usuarios se registran de forma manual en la base de datos de credenciales. Falta un endpoint que realice el registro del paciente y que podría estar asociado con la aplicación móvil, pero todo depende de la decisión que se tome para gestionar este registro de usuarios en el sistema.
- **Seguridad en conexión directa con servidor FHIR.** Punto muy importante que concierne la seguridad de la información, ya que si bien el acceso mediante el componente web API Wrapper está resguardado, la información es libremente accesible al realizar solicitudes HTTP directamente al servidor FHIR. Es necesario implementar un mecanismo de seguridad en este ámbito.
- **Pulir APIs de componente FHIR Wrapper y rutinas.** En resumen, refactorizar el código de todos los controladores para seguir un estándar en común y que sea más mantenible con el tiempo.

# 5. Evaluación

Para la evaluación de la arquitectura propuesta se realizaron pruebas con ciertos escenarios de compatibilidad que involucran sistemas de prototipo propios y de miembros del equipo de proyecto FONDEF. Por otro lado, se evaluó el rendimiento de la arquitectura en términos de velocidad de respuesta y espacio en memoria utilizado, y su comportamiento al ser sometida a un test de carga con la herramienta **Apache Jmeter.**

## 5.1 Escenarios de compatibilidad

En esta sección se describen los escenarios que demuestran la compatibilidad de la arquitectura propuesta con clientes no nativos y nativos de FHIR.

### 5.1.1 FHIR no nativo

La arquitectura es compatible con sistemas no nativos de FHIR, gracias al mapeo de datos que realiza con su componente de API FHIR Wrapper. Para demostrar la compatibilidad con ambos extremos de la comunicación se crearon 2 prototipos de sistemas:

- Prototipo de ficha clínica, que solicita información sobre las ingestas de medicamentos realizadas por el paciente.
- Prototipo de aplicación móvil, que permite al usuario iniciar sesión e ingresar una ingesta de medicamento.

En conjunto a los prototipos mencionados, el bus de interoperabilidad fue distribuido a integrantes del equipo encargados de la aplicación móvil AFAM 2.0 y módulo de visualización específico en ficha clínica con el fin de verificar compatibilidad y dar el puntapié inicial a la integración entre estos sistemas.

#### 5.1.1.1 Escenario con aplicación móvil

Se construyó una aplicación móvil con el framework de desarrollo Flutter que utiliza algunos de los endpoints del componente Wrapper y rutinas, a modo de simular las funcionalidades de la futura aplicación AFAM 2.0. Las funcionalidades de este prototipo consisten en inicio de sesión del usuario, e ingreso de registro de ingesta de un medicamento disponible previamente en el servidor, con la dosis y hora de ingesta.

Los endpoint que fueron probados con esta aplicación fueron:

- Endpoint del controlador **AuthenticationController** de validación de usuario.
- Endpoint del controlador **PatientController** de lectura de paciente según su rut.
- Endpoint del control **MedicationController** de lectura de todos los medicamentos dentro del servidor.
- Endpoint del controlador **MedicationAdministrationController** de ingresar registro de ingesta de medicamento.

La aplicación realiza la petición HTTP a la API de autenticación al iniciar sesión, y al validarse la identidad del usuario realiza las peticiones a las APIs de paciente y medicamentos, para cargar esta información. Luego se abre una pantalla con un formulario para ingresar la información relevante a la ingesta de medicamento (medicamento, dosis, horario) como se ve en Figura 23 y al presionar el botón de envío se llama la API de creación de registro de ingesta.


*Figura 23: Pantalla de registro de ingesta en prototipo de aplicación móvil no FHIR, elaboración propia*

Por otro lado, el integrante del equipo Vicente Schultz se encuentra realizando la aplicación oficial AFAM 2.0. Se integró esta aplicación con el bus de interoperabilidad y se probaron los siguientes endpoints:

- Endpoint del controlador **AuthenticationController** de validación de usuario.
- Endpoint del controlador **PatientController** de lectura de paciente según su rut.
- Endpoint del controlador **MobileController** de lectura de prescripciones y sus posologías con información de medicamento, del paciente.
- Endpoint del controlador **MedicationController** de ingreso de medicamento externo al servidor (sin código SNOMED)

La aplicación realiza la petición HTTP a la API de autenticación al iniciar sesión, y al validarse la identidad del usuario realiza las peticiones a las APIs de paciente y de prescripciones con medicamentos, para cargar esta información. Esta información es utilizada en una pantalla que lista todas las ingestas pendientes por realizar según los horarios guardados en los recursos MedicationRequest del servidor, como se ve en Figura 24, y el paciente puede marcarlas como realizadas una vez ha ingerido el medicamento. También puede agregar nuevos medicamentos y horarios de ingesta según lo necesite, para lo cual llama la API de creación de medicamento externo.


*Figura 24: Pantalla de lista de ingestas pendientes en prototipo de app móvil AFAM 2.0, elaborada por Vicente Schultz*

Con todo esto dicho, se verifica que los prototipos son capaces de utilizar las APIs Wrapper para interactuar con el servidor FHIR de manera exitosa, con especial énfasis en los recursos que son de interés para el paciente como sus prescripciones y el poder registrar ingestas de estas prescripciones.

#### 5.1.1.2 Escenario con ficha clínica y módulo de visualización

Se construyó una aplicación web con el framework de desarrollo React que utiliza algunos de los endpoints del componente Wrapper y rutinas, a modo de simular las funcionalidades de un módulo de visualización en un sistema de ficha clínica. Las funcionalidades de este prototipo consisten en ingreso de un paciente a consultar y lectura de información de ingestas del paciente ingresado.

Los endpoint que fueron probados con esta aplicación fueron:

- Endpoint del controlador **PatientController** de lectura de paciente según su rut.
- Endpoint del controlador **VisualizationController** de lectura de registros de ingestas de un paciente con información del medicamento asociado.

La aplicación realiza las peticiones HTTP a las API de paciente y de registros de ingesta con medicamentos al ingresar un rut en el buscador, lo que carga la información para utilizarla en la pantalla principal. Esta pantalla muestra a modo de ficha clínica la información del paciente y una tabla con los registros de ingesta realizados, con el nombre de medicamento, dosis y horario de ingesta, como se ve en Figura 25.


***Figura 25: Pantalla de información de paciente con sus registros de ingestas en prototipo de ficha clínica no FHIR, elaboración propia***

Por otro lado, el integrante del equipo Claudio Rain se encargó de desarrollar el módulo de visualización específico oficial de AFAM 2.0. Se integró este módulo con el bus de interoperabilidad y se probaron múltiples endpoints, dentro de los cuales se encuentran:

- Endpoint del controlador **PatientController** de lectura de paciente según su rut.
- Endpoint del controlador **VisualizationController** de lectura de registros de ingestas de un paciente con información del medicamento asociado, y la prescripción y horario de posología asociada.

- Endpoint del controlador **VisualizationController** de lectura de prescripciones y sus posologías con información de medicamento, del paciente.

Este módulo utiliza el endpoint relacionado con la solicitud de índice de adherencia, pero aún está por implementarse.

El módulo realiza las peticiones HTTP a las API de paciente, de registros de ingesta y de prescripciones al ingresar un rut en el buscador, lo que carga la información para utilizarla en la pantalla principal. Esta pantalla consiste en un dashboard que muestra la información del paciente, su nivel de adherencia con respecto a todas sus prescripciones en forma de porcentaje, múltiples gráficos con la información de los registros de ingesta y secciones con información de las prescripciones en periodos determinados, como se ve en Figura 26.


***Figura 26: Sección de prescripciones con información de medicamento de un paciente en módulo de visualización específica, elaborado por Claudio Rain***

Con todo esto dicho, se verifica que el prototipo y módulo de visualización específica son capaces de utilizar las APIs Wrapper para interactuar con el servidor FHIR de manera exitosa, con especial énfasis en los recursos que son de interés para el personal sanitario como son información del paciente, registros de ingesta, las prescripciones, e índice de adherencia.

### 5.1.2 FHIR nativo

La arquitectura es compatible con sistemas nativos de FHIR, gracias a las operaciones API REST que provee el servidor HAPI FHIR. En este caso no se realizaron escenarios de prueba pero se listan las peticiones HTTP que son capaces de interactuar con los recursos relevantes a los clientes de aplicación móvil y módulo de visualización. Estas llamadas pueden ser realizadas por sistemas nativos de FHIR y estos pueden procesar la respuesta contenedora de recursos FHIR sin problemas. Los recursos ingresados en las peticiones deben utilizar los elementos descritos en la arquitectura FHIR subyacente, según el recurso específico. Estas llamadas específicas pueden encontrarse en **Anexos.**

## 5.2 Rendimiento

En esta sección se describen aspectos de rendimiento de los endpoints de API de la arquitectura. Se describe el espacio en memoria utilizado por el bus de interoperabilidad, para lo cual se realizaron pruebas de carga mediante llamadas concurrentes a modo de simulación de múltiples clientes y se obtuvo los tiempos de respuesta en algunos de los endpoints más importantes, y el nivel de estrés permitido por el servidor.

### 5.2.1 Espacio en memoria

Se considera el espacio en RAM que utiliza el conjunto de componentes de la arquitectura. Para ello, se puede ver las estadísticas del Docker que contiene el bus de interoperabilidad. Este provee estadísticas del uso de recursos de los 4 componentes. Se obtuvieron los valores de la memoria utilizada cuando los servicios se encuentran en standby y cuando procesan solicitudes entrantes. Todas las consideraciones de uso de memoria se realizaron de manera local con mi equipo personal que dispone de 12.28 GB de RAM, por lo que la cantidad asignada puede variar en dispositivos con capacidades distintas.

- El servidor FHIR utiliza 2 GB de RAM cuando se encuentra en standby y al procesar solicitudes.
- El componente FHIR Wrapper y de rutinas utiliza entre 520 MB de RAM en standby y cerca de 600 MB al procesar solicitudes.
- La base de datos del servidor FHIR utiliza entre 250 MB y 300 MB de RAM al estar en standby y al procesar solicitudes. Adicionalmente, se considera el espacio en disco que utiliza su volumen de persistencia Docker, el cual utiliza registra un tamaño de 506 MB con más de 50000 recursos en el servidor FHIR, al momento de realizar las pruebas.
- La base de datos de credenciales utiliza entre 20 MB y 30 MB de RAM al estar en standby y procesar solicitudes. Adicionalmente, se considera el espacio en disco que utiliza su volumen de persistencia Docker, el cual utiliza registra un tamaño de 47 MB con 3 tuplas de usuarios, al momento de realizar las pruebas.

El Docker contenedor de los 4 componentes indica un espacio en memoria total en el rango de 5.12 y 5.39 GB de RAM cuando se encuentra inactivo, pero esto considera un overhead del uso de Docker que involucra 2.57 GB de RAM, por lo que el espacio total en memoria real varía entre 2.55 GB y 2.82 GB de RAM. Se aprecia la mayor exigencia de recursos que necesita el servidor FHIR, lo cual es lógico ya que es el procesador principal de los recursos FHIR utilizados en todas las peticiones. Mientras que la base de datos es el componente menos exigente ya que procesa solicitudes de autenticación con una tabla de esquema muy simple, y contiene muy pocos registros.

### 5.2.2 Test de carga y desempeño

Con la herramienta de **JMeter** se configuró un plan de pruebas a ejecutar sobre las endpoints de la arquitectura. En específico, se realizaron pruebas sobre la webapi Wrapper y de rutinas, ya que estas tienen más importancia por su uso real de los actores en el proyecto FONDEF.

En específico, se diseñó un plan de pruebas inicial que simula el acceso concurrente por 100 pacientes a la aplicación móvil, y por 50 funcionarios al módulo de visualización específico.

En el caso de la simulación de pacientes, estos ejecutan peticiones HTTP relacionadas a un workflow común que se dará en la aplicación, de iniciar sesión en la aplicación, obtener su información de paciente en conjunto con las prescripciones registradas en el sistema, y luego de eso registran una ingesta de medicamento. Cada llamada de este workflow se realizó 3 veces por cada paciente, los cuales ejecutan con 1 segundo de diferencia, obteniendo un total de 1200 peticiones HTTP.

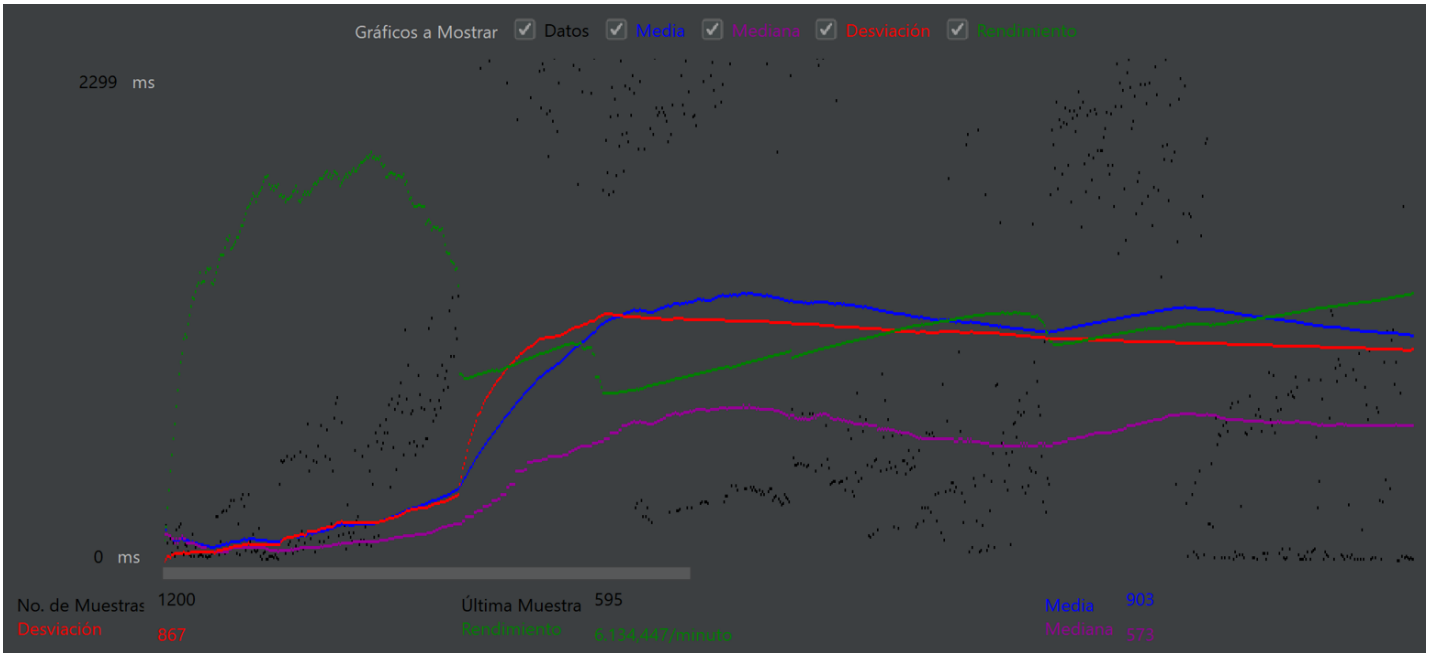



*Figura 27: Gráfico que describe el comportamiento de las 1200 llamadas HTTP realizadas por pacientes, elaborado por Jmeter*

Según los datos obtenidos con las pruebas, el servidor es capaz de soportar **6.134,447 peticiones por minuto** o **102,2 peticiones por segundo**, relacionadas con este workflow de paciente, con una latencia mínima de **12 ms** y máxima de **4621 ms.**

En el caso de la simulación de funcionarios médicos, estos ejecutan peticiones HTTP relacionadas al workflow común que se dará en la aplicación, de buscar información de un paciente, y recibir la información de prescripciones, e ingestas realizadas por el paciente. Esto también incluye el cálculo de porcentaje de adherencia, pero aún no se encuentra implementado. Cada llamada de este workflow se realizó 3 veces por cada funcionario, los cuales ejecutan con 1 segundo de diferencia, obteniendo un total de 600 peticiones HTTP.

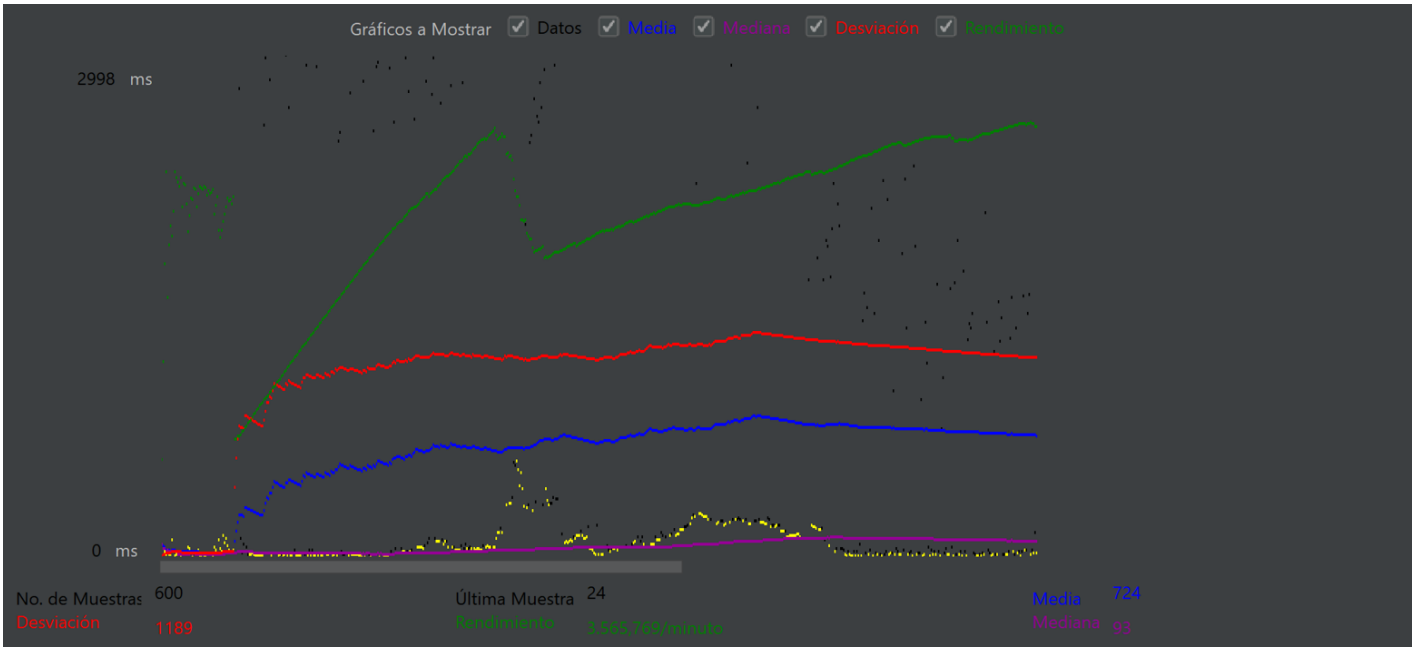



*Figura 28: Gráfico que describe el comportamiento de las 600 llamadas HTTP realizadas por funcionarios médicos, elaborado por Jmeter*

Según los datos obtenidos con las pruebas, el servidor es capaz de soportar **3.565,769 peticiones por minuto**, relacionadas con este workflow de paciente, con una latencia mínima de **10 ms** y máxima de **1364 ms.** Un detalle a considerar fue que la API de lectura de ingestas del paciente falló en todas sus llamadas debido a una sobrecarga de recursos **MedicationAdministration** de más de 50.000 en el servidor FHIR, que fueron creados al realizar pruebas con esta herramienta. Considerando que este inconveniente será corregido y que el funcionamiento de las APIs está comprobado, se puede decir con certeza que esta prueba obtendría resultados más cercanos a aquellos del workflow del paciente.

El bus de interoperabilidad es capaz de soportar las 1800 llamadas concurrentes sin problemas, presentando resultados muy buenos de latencia y tiempo de respuesta. Por lo mismo, se decidió

someter a más estrés al servidor aumentando el número de simulaciones, por lo que se definió simular 500 pacientes y 250 funcionarios, cada uno ejecutando las llamadas del workflow 10 veces (30.000 llamadas HTTP en total). En este caso el servidor dejó de responder y no fue capaz de procesar este tamaño de solicitudes, con una congelación por parte del cliente JMeter.

Este nivel de estrés permitido por el servidor es aceptable, ya que la simulación de llamadas simula más usuarios de los que realísticamente usarán el sistema. Pero se debe considerar que el plan de pruebas se ejecutó de forma local y los resultados podrían variar en un entorno remoto.

# 6. Conclusiones

La adherencia de un paciente a su tratamiento farmacológico es un factor fundamental en el bienestar de su salud. Este aspecto tiene repercusiones importantes en la gestión del tratamiento, cambios en las recetas prescritas y decisión por parte del personal sanitario. Sin embargo, se ha mostrado que esta información no suele llegar de la forma más fidedigna a estos últimos.

El soporte informático en esta área es limitado. Si bien existen soluciones informáticas que permiten al paciente traer sus registros médicos desde sistemas de salud y llevar un seguimiento local de ingestas de medicamentos, no hay una integración directa en la dirección del paciente hacia los sistemas de salud que permita a los médicos beneficiarse de la información actualizada en tiempo real y precisa para la toma de decisiones. Por otro lado, los estándares de salud suelen enfocarse en la interoperabilidad intra e inter sistemas informáticos de proveedores de salud (centros médicos, hospitales, etc.) por lo que hay implementaciones que utilizan los estándares para transmitir la información de los sistemas dentro de la misma organización o hacia organizaciones externas. Algo tan específico como permitir al paciente tener una mayor participación en la información de su tratamiento, no se ha implementado como una solución formal anteriormente.

En ese sentido, se han cumplido efectivamente los objetivos propuestos de esta MT, la cual propone una gran contribución al sentar las bases de un bus de interoperabilidad en base al estándar FHIR que será de vital importancia para las comunicaciones dentro del sistema AFAM 2.0. Y no sólo en este sistema, ya que funciona como bloque fundamental para construir soluciones con sistemas que necesiten de este tipo de integración para una comunicación entre paciente y módulos de ficha clínica.

La construcción de la arquitectura sólo utilizando herramientas Open Source es un hito importante para la comunicación entre paciente y personal sanitario, ya que es una alternativa gratuita que provee lo justo y necesario para satisfacer la comunicación con posibilidades de escalabilidad, en comparación con soluciones comerciales que posiblemente provean capas adicionales de herramientas pero tienen un costo importante asociado. En el contexto del proyecto FONDEF, se ha discutido la opción de migrar el Backend a un entorno de IRIS for Health, debido a la implementación existente de EHR y ficha clínica con el sistema TrakCare de la misma compañía

InterSystems. Considerando eso, el contenido de esta MT sigue siendo de utilidad para establecer las bases de cómo configurar los componentes y la arquitectura FHIR subyacente en el sistema IRIS for Health.

Además, la estructura del componente web API Wrapper y de rutinas provee una mecanismo extensible para la integración de nuevos requerimientos de usuario. En ese sentido, la arquitectura establece un framework inicial de trabajo con el estándar FHIR y que puede desarrollar mayor compatibilidad con distintos sistemas de ficha clínica que utilicen otro tipo de estándar de salud, como aquellos de legado HL7 v2, v3 o CDA.

Si bien cuenta con puntos positivos al permitir la interoperabilidad de la información y la comunicación entre pacientes y médicos mediante herramientas informáticas, la arquitectura no se niega a la posibilidad de evolucionar y mejorar con el tiempo, al tener más experiencia "hands-on" con los sistemas informáticos de salud como tal (sistema de ficha clínica y EHR). Por ejemplo, en el soporte de un mayor número de usuarios al cual el servidor es incapaz de responder según el test de carga.

Finalmente, la arquitectura satisface los requerimientos de usuario propuestos, tanto los referentes a las funcionalidades que poseen los pacientes y el personal sanitario, como los no funcionales que incluyen la idoneidad funcional de poder transmitir los datos entre sistemas nativos y no nativos de FHIR, la confiabilidad de contar con mecanismos de persistencia de información y de soporte para múltiples usuarios concurrentes en el sistema, el desempeño de una transmisión eficiente de datos, la usabilidad de para adaptar la arquitectura a nuevos sistemas sin mayor dificultad, la seguridad y acceso a la información sólo a usuarios autorizados, la compatibilidad con los sistemas que cuentan con las características de comunicación HTTP que admite la arquitectura, la mantenibilidad de diseñar los componentes para favorecer la escalabilidad de nuevas funcionalidades, y la portabilidad para poder hospedar el bus de interoperabilidad en entornos remotos y poder trasladar la arquitectura FHIR subyacente a otros sistemas de desarrollo (como IRIS for Health).

## 6.1 Trabajo futuro

A partir de la arquitectura propuesta en esta MT, se pueden abordar múltiples aristas que están fuera del alcance del proyecto FONDEF, pero que sin dudas serían características interesantes de explorar si se decide continuar con la extensión de este bus de interoperabilidad para trazabilidad de adherencia a tratamiento farmacológico. Estos aspectos incluyen:

- **Compatibilidad con XML**: XML es un formato de transmisión de información muy popular, predecesor de JSON pero no por eso menos importante. FHIR es compatible con este formato pero la implementación actual sólo trabaja con recursos FHIR en formato JSON. Sería interesante ver la posibilidad de construir recursos en este formato para mayor interoperabilidad.

- **Transferir arquitectura FHIR a otro entorno de desarrollo:** Como se ha mencionado, existe la posibilidad de utilizar el entorno de desarrollo provisto por IRIS Health para portar la arquitectura construida en esta MT. Si se da la oportunidad de tener una licencia para esos propósitos, la idea suena interesante por las posibles capacidades que este entorno provea, al ser una alternativa comercial de renombre, y el diseño de la arquitectura FHIR para ser porteada. Aún no es seguro si esto está dentro del alcance del proyecto pero se incluye en esta sección.
- **Integración con sistemas de legado:** Aquellos sistemas que utilizan estándares HL7 v2, v3 o CDA, los cuales siguen siendo muy populares a pesar de su antigüedad, debido a su temprana adopción en varios sistemas de salud. Sería interesante explorar la posibilidad de mapear recursos FHIR a esos estándares y viceversa.
- **Integración con SMART on FHIR:** SMART on FHIR suena como una opción idónea para la autenticación y autorización de usuarios para utilización de recursos FHIR, ya que en teoría puede solicitar tokens de autorización directamente al sistema EHR que contiene credenciales de usuario, y así no contar con una base de datos de usuarios independiente. En esta ocasión no se pudo verificar su funcionalidad por temas de tiempo de desarrollo y por falta de interacción con EHRs y sistemas de ficha clínica reales, pero sigue siendo una alternativa de seguridad al JWT implementado muy ad-hoc al caso FHIR.
- **Compatibilidad con más patologías y polifarmacia:** La consideración inicial era desarrollar una arquitectura para registrar la adherencia a tratamiento farmacológico de HTA, para lo cual se tuvo acceso al repertorio de medicamentos para tratar esta patología. Durante el transcurso del proyecto, se comentó el requerimiento de compatibilizar la arquitectura con más patologías y un mayor rango de medicamentos. Esto implicaría modificar la lógica de construcción de recursos FHIR de la arquitectura y una mayor cantidad de códigos SNOMED para incluir todos los conceptos médicos relacionados con estas patologías y medicamentos nuevos, pero es una consideración a tener luego de la etapa piloto del proyecto.

# 7. Referencias

**[1] Sabate E. World Health Organization (2003).** ***Adherence to Long Term Therapies: Evidence for Action*****,** ***Geneva: World Health Organization*****. Available at: https://drive.google.com/file/d/1nO01wxn2vjT2sCyPyOguksCRNWFUDv8G/view?usp=sharing (Accessed: August 2022).**

**[2] Atlassian (2011)** ***Trello*****. Available at: https://trello.com/es (Accessed: March 2023).**

**[3] Slack (2009)** ***Slack Es Tu Plataforma de Productividad*****,** ***Slack*****. Available at: https://slack.com/intl/es-cl (Accessed: March 2023).**

**[4] HL7 (2007) *Health Level Seven international FAQs*, *HL7 International*. Available at: https://www.hl7.org/about/FAQs/index.cfm?ref=nav (Accessed: September 2022).**

**[5] HL7 (2007) *HL7 Version 2 Product Suite*, *HL7 International*. Available at: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185 (Accessed: August 2023).**

**[6] iNTERFACEWARE Inc. (2021) *HL7 Message Structure*, iNTERFACEWARE Inc. *HL7 Resource*. Available at: https://www.interfaceware.com/hl7-message-structure (Accessed: August 2023).**

**[7] Torres, P.M. (2018) *Los 5 estándares HL7 fundamentales*, *Caduceus Connecting eHealth*. Available at: https://www.caduceus.es/estandares-hl7-fundamentales/ (Accessed: August 2023).**

**[8] HL7 (2007) *HL7 Version 3 Product Suite*, *HL7 International*. Available at: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186 (Accessed: August 2023).**

**[9] Spronk (2011) *An explanation of HL7 version 3 in terms of HL7 version 2*, *Ringholm*. Available at: https://ringholm.com/docs/01200_en_HL7v3_using_HL7v2_terms.htm (Accessed: August 2023).**

**[10] Vedmed, S. (2023) *FHIR vs HL7: Key Differences and Which Is a Better Choice?*, *Kodjin*. Available at: https://kodjin.com/blog/fhir-vs-hl7-key-differences-and-which-is-a-better-choice/ (Accessed: August 2023).**

**[11] HL7 (2019) *Welcome to FHIR*, *HL7 FHIR Release 4*. Available at: https://hl7.org/fhir/R4/index.html (Accessed: September 2022).**

**[12] Goyal, K. (2023) *Unleashing the power of FHIR PARADIGMS: A global perspective on implementation and best practices*, *Medblocks*. Available at: https://blog.medblocks.org/unleashing-the-power-of-fhir-paradigms-a-global-perspective-on-implementation-and-best-practices/ (Accessed: August 2023).**

**[13] HL7 (2019) *FHIR Overview - Architects*, *HL7 FHIR Release 4*. Available at: https://hl7.org/fhir/R4/overview-arch.html (Accessed: September 2022).**

**[14] Computational Health Informatics Program (ed.) (2018) *What is SMART?*, *SMART Health IT*. Available at: https://smarthealthit.org/an-app-platform-for-healthcare/about/ (Accessed: October 2022).**

**[15] López Magaña, L.M. (2020) *Qué es OAuth 2*, *OpenWebinars*. Available at: https://openwebinars.net/blog/que-es-oauth2/ (Accessed: August 2023).**

**[16] HL7 (2021) *SMART App Launch Overview*, *HL7 FHIR*. Available at: https://www.hl7.org/fhir/smart-app-launch/ (Accessed: October 2022).**

**[17] Computational Health Informatics Program (ed.) (2018) *SMART App Gallery, SMART Health IT*. Available at: https://apps.smarthealthit.org/apps/featured (Accessed: October 2022).**

**[18] HL7 (2007) *FHIR® (HL7 Fast Healthcare Interoperability Resources), HL7 International*. Available at: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=491 (Accessed: August 2023).**

**[19] NextGen Healthcare (ed.) *Healthcare Integration Engine: Mirth® Connect, NextGen Healthcare*. Available at: https://www.nextgen.com/solutions/interoperability/mirth-integration-engine (Accessed: August 2023).**

**[20] InterSystems, *InterSystems IRIS for Health, InterSystems*. Available at: https://www.intersystems.com/cl/intersystems-iris-for-health/ (Accessed: August 2023).**

**[21] Smile CDR (ed.) *HAPI FHIR - The Open Source FHIR API for Java, HAPI FHIR.* Available at: https://hapifhir.io/ (Accessed: October 2022).**

**[22] Google Cloud (2020) *Overview of the Cloud Healthcare API, Google Cloud Healthcare API.* Available at: https://cloud.google.com/healthcare-api/docs/introduction (Accessed: August 2023).**

**[23] Microsoft Azure (2019) *What is the FHIR Service?, Microsoft Azure Health Data Services*. Available at: https://learn.microsoft.com/en-us/azure/healthcare-apis/fhir/overview (Accessed: August 2023).**

**[24] Medical Imaging Technology Association (MITA) (ed.) *DICOM Standard, DICOM.* Available at: https://www.dicomstandard.org/ (Accessed: August 2023).**

**[25] Apple (2014) *Apple Healthcare Overview, Apple Healthcare.* Available at: https://www.apple.com/healthcare/ (Accessed: August 2023).**

**[26] Apple (2015) *Apple ResearchKit y CareKit, Apple Healthcare*. Available at: https://www.apple.com/la/researchkit/ (Accessed: August 2023).**

**[27] Apple (2018) *Apple Health Records, Apple Healthcare*. Available at: https://www.apple.com/healthcare/health-records/ (Accessed: August 2023).**

**[28] IBM Corporation (2015) *IBM Integration Bus Introduction, IBM*. Available at: https://www.ibm.com/docs/en/integration-bus/10.0?topic=overview-integration-bus-introduction (Accessed: August 2023).**

**[29] IBM Corporation (2015) *IBM Integration Bus Healthcare Pack V4.0 delivers improved interoperability of healthcare systems through support for additional standards and IHE profiles, IBM*. Available at: https://www.ibm.com/docs/en/announcement_archive/ENUS215-165/ENUS215-165.PDF (Accessed: August 2023).**

**[30] Firely (2015) *Firely Server, Firely Products. Available at:* https://fire.ly/products/firely-server/ (Accessed: October 2022).**

**[31] Canessa, G. (2020) *FHIR for Developers: FHIR in C#, Youtube.* https://www.youtube.com/playlist?list=PLsR-zcO--dypP688ilpL3rAiYjWnZbubA (Accessed: October 2022).**

**[32] InterSystems, *Pruebe gratis InterSystems IRIS Data Platform, InterSystems*. Available at: https://www.intersystems.com/cl/pruebe-intersystems-iris/ (Accessed: June 2023).**

**[33] Smile CDR (ed.) *Smile Digital Health: A complete HL7 FHIR-based Clinical Data Repository, Smile CDR*. Available at: https://www.smiledigitalhealth.com/ (Accessed: August 2023).**

**[34] Hapifhir (2019) *HAPI-FHIR Starter Project, GitHub*. Available at: https://github.com/hapifhir/hapi-fhir-jpaserver-starter (Accessed: October 2022).**

**[35] *Centro Nacional en Sistemas de Información en Salud*, *Página principal, CENS*. Available at: https://cens.cl/ (Accessed: August 2023).**

**[36] SNOMED International (2007) *Home Page, SNOMED*. Available at: https://www.snomed.org/ (Accessed: May 2023).**

**[37] SNOMED International (2007) *SNOMED CT Browser, SNOMED CT.* Available at: https://browser.ihtsdotools.org/ (Accessed: May 2023).**

**[38] SNOMED International (2007) *Chile, SNOMED*. Available at: https://www.snomed.org/member/chile (Accessed: May 2023).**

**[39] Ministerio de Salud (2021) *Guía Clínica de Servicios Terminológicos, MINSAL: Salud Digital*. Available at: https://interconsulta.minsal.cl/img/guias/clinica.pdf (Accessed: May 2023).**

**[40] American Hospital Association (2018)** ***Sharing Health Information for Treatment, American Hospital Association Annual Survey IT Supplement Brief #2*****. Available at: https://www.aha.org/system/files/2018-03/sharing-health-information.pdf (Accessed: September 2022).**

**[41] Epic (2000)** ***MyChart: Your secure online health connection, MyChart*****. Available at: https://www.mychart.org/ (Accessed: August 2023).**

**[42] Apple (2018)** ***Apple opens Health Records API to developers, Apple Newsroom*****. Available at: https://www.apple.com/newsroom/2018/06/apple-opens-health-records-api-to-developers/ (Accessed: September 2022).**

**[43] Comstock, J. (2018)** ***Inside Apple's integration with Medisafe, the first test of the Apple Health Records API, MobiHealthNews*****. Available at: https://www.mobihealthnews.com/content/inside-apples-integration-medisafe-first-test-apple-health-records-api (Accessed: September 2022).**

**[44] Medisafe (2016)** ***Medisafe, in Collaboration with Cerner and Epic, First to Bring Interoperable Medication Lists to Patients through Electronic Health Records, Medisafe News & Events*****. Available at: https://www.medisafe.com/medisafe-in-collaboration-with-cerner-and-epic-first-to-bring-interoperable-medication-lists-to-patients-through-electronic-health-records/ (Accessed: September 2022).**

**[45] Andrade, R., Wangenheim A., Savaris A., Petry K. (2013)** ***Using a Health Level 7 Interoperability Bus to Support Legacy Systems in the Health Domain, E-Health Telecommunication Systems and Networks*****. Available at: https://www.scirp.org/pdf/ETSN_2013122713484447.pdf (Accessed: August 2022).**

**[46] Medisafe (2012)** ***Home Page, Medisafe*****. Available at: https://medisafeapp.com/ (Accessed: September 2022).**

**[47] Equipo AFAM (2017)** ***Informe técnico AFAM Salud, AFAM Salud*****. Available at: https://drive.google.com/file/d/1AlHkSkA0qsko23piyeu50GHHOYVCfRHD/view (Accessed: August 2023).**

**[48] InterSystems,** ***TrakCare, InterSystems*****. Available at: https://www.intersystems.com/cl/trakcare/ (Accessed: June 2023).**

**[49] PronovaSalud (2014)** ***Productos, PronovaSalud.*** **Available at: https://pronovasalud.com/productos/ (Accessed: August 2023).**

**[50] Ramesh, S. (2020) *The Definitive FHIR Playlist, Youtube.* Available at: https://www.youtube.com/playlist?list=PLUr-PTsPYKV4SHhszovqJ3v4hEhRezqzo (Accessed: October 2022).**

**[51] Smile CDR (ed.) *JPA Server Configuration, HAPI FHIR*. Available at: https://hapifhir.io/hapi-fhir/docs/server_jpa/configuration.html (Accessed: May 2023).**

**[52] The Apache Software Foundation (1998) *Home Page, Apache JMeter*. Available at: https://jmeter.apache.org/ (Accessed: August 2023).**

**[53] HL7 (2019) *Search, HL7 FHIR Release 4*. Available at: https://hl7.org/fhir/r4/search.html (Accessed: October 2022).**

**[54] McKenzie, L. (2020) *Change Requests, Confluence HL7*. Available at: https://confluence.hl7.org/display/FHIR/Change+Requests (Accessed: July 2023).**

# 8. Anexos

## 8.1 Información adicional FHIR

### 8.1.1 CodeSystems y ValueSets

En FHIR, se utiliza el recurso *CodeSystem* para representar información de un sistema de códigos. Los sistemas de códigos se identifican por su **referencia canónica**, es decir, un URL único que los identifica en cualquier contexto**.** Cabe destacar que FHIR no se hace cargo de la mantención de los sistemas, sólo se utiliza este recurso para publicar las propiedades, los filtros de búsqueda y los contenidos de estos sistemas para ser utilizados en el ecosistema FHIR, ya que la mayoria de sistemas cuenta con su propio sistema de mantención y formato de distribución.

Siguiendo con la analogía a HL7 v3, el estándar también utiliza conjuntos de valores o **Value Sets,** los cuales son subconjuntos de códigos tomados de un sistema de códigos en específico. Son ampliamente utilizados a lo largo de la especificación, para definir los valores permitidos en ciertos elementos de recursos.

Existen los Value Sets definidos por la especificación para aplicar restricciones sobre elementos en concreto, o el implementador puede definir sus propios conjuntos de valores mediante el recurso *ValueSet*, para otros casos de uso. Por ejemplo, el Value Set de *administrative-gender* es utilizado en los elementos de género de los recursos administrativos referentes a personas en el workflow de salud (recursos *Patient, Practitioner, Person,* entre otros). Este ValueSet toma sus valores de un sistema de códigos interno a la especificación (sistema de códigos también llamado

*AdministrativeGender*), y contiene la totalidad de códigos definidos en ese sistema. Estos códigos representan los géneros de masculino, femenino, otro o desconocido.

| Code | System | Display | Definition |
|---|---|---|---|
| male | http://hl7.org/fhir/administrative-gender | Male | Male. |
| female | http://hl7.org/fhir/administrative-gender | Female | Female. |
| other | http://hl7.org/fhir/administrative-gender | Other | Other. |
| unknown | http://hl7.org/fhir/administrative-gender | Unknown | Unknown. |

***Figura 29: Códigos de ValueSet administrative-gender, obtenido de la especificación FHIR***
***De izquierda a derecha, el código, sistema de códigos, texto para legibilidad humana, y definición del código.***

Por otro lado, se encuentran Value Sets de sistemas de códigos externos a la especificación. Un ejemplo se presenta en los Value Sets que heredan códigos de SNOMED CT (ver SNOMED CT). En el recurso *Medication,* en su elemento **code** se espera proporcionar un código de un medicamento y la referencia a su sistema de códigos. La especificación sugiere un Value Set correspondiente a un conjunto de códigos de medicamentos de SNOMED CT, denominado *medication-codes* y contiene alrededor de 3000 códigos. El sistema de códigos de SNOMED CT es referenciado por el URL canónico http://snomed.info/sct.

| | |
|---|---|
| 1336006 | Deoxycortone |
| 1355006 | Coagulation factor IX Oxford 3 variant |
| 1368003 | Iodine 131 |
| 1381005 | ^126^Iodine |
| 1450002 | Methylpentynol |
| 1476002 | Codeine sulfate |
| 1536005 | Pargyline hydrochloride |
| 1575001 | Maltose tetrapalmitate |
| 1603001 | Cobalt isotope |
| 1668008 | Ceforanide |
| 1914001 | von Willebrand factor antibody |
| 1944003 | Coagulation factor X Patient variant |
| 1956002 | Buclizine hydrochloride |
| 1971003 | Loxapine hydrochloride |
| 2029004 | Fibrinogen Oslo II |
| 2125008 | Betazole |

***Figura 30: Fragmento de códigos de ValueSet medication-codes, obtenido de la especificación FHIR***
***De izquierda a derecha, el código y texto para legibilidad humana.***

El estándar solicita que la mayoría de elementos que utilizan códigos se encuentren vinculados a un Value Set especificado en la definición del recurso, y este vínculo puede tener distintos niveles de fuerza.

- **required:** El elemento debe obligatoriamente tomar valores de este Value Set para ser conforme con la especificación.
- **extensible:** El elemento debe siempre tomar valores del Value Set para ser conforme con la especificación, pero de darse el caso que el Value Set no contenga el valor necesario para el caso particular a modelar, se puede tomar un codificado alternativo y el elemento sigue siendo conforme con la especificación.
- **preferred:** Se prefiere e incentiva que los valores provengan del Value Set por motivos de interoperabilidad, pero no es obligatorio para ser conforme con la especificación.
- **example:** El elemento debe tomar valores de la misma índole que se encuentran en este Value Set, pero no es necesario ni se incentiva que los tome de ese Value Set en particular.

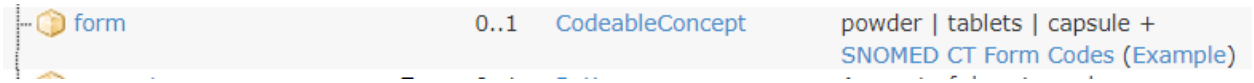



*Figura 31: Elemento form de Medication, que se adhiere al Value Set de form-codes de SNOMED CT, con un nivel de vínculo de example, obtenido de la especificación FHIR*

### 8.1.2 Restricciones

Gran parte de los recursos de la especificación se encuentran adheridos a restricciones, las cuales tienen distintos niveles de severidad pero la más importante es la de categoría **regla,** que indica que las restricciones impuestas de este tipo deben seguirse o el validador de recursos (dependiendo de la implementación) lo marcará como error.

Las reglas o restricciones se definen en el elemento **implicitRules** del recurso en particular, y también pueden encontrarse en la página de contenido del recurso. Una de las reglas que heredan todos los recursos es la de incluir narrativas en su elemento **text** que describan el contenido del recurso en un párrafo de texto**.** El estándar incita a considerar la parte de legibilidad humana imponiendo esta regla en sus recursos.

| id | Level | Location | Description | Expression |
|---|---|---|---|---|
| mad-1 | Rule | MedicationAdministration.dosage | SHALL have at least one of dosage.dose or dosage.rate[x] | `dose.exists() or rate.exists()` |

*Figura 32: Regla sobre elemento dosage de recurso MedicationAdministration, obtenido de la especificación FHIR De izquierda a derecha, se encuentra el id de la restricción, restricción de tipo regla, el elemento al cual se aplica, la descripción de la regla, y la expresión utilizada a nivel interno por la implementación*

Por ejemplo, en Figura 32 se define una regla para el elemento **dosage** de *MedicationAdministration,* la cual indica que el elemento (el cual es un BackBone Element) debe contener por lo menos un valor para **dose** o **rate**. En palabras más simples, indica que el elemento **dosage** que agrupa la información de cómo se ingirió el medicamento, debe incluir por lo menos la dosis que se ingirió (dose), o la dosis ingerida por unidad de tiempo (rate).

### 8.1.3 Parámetros de búsqueda

Las búsquedas se realizan llamando a una URL con el método GET indicando valores para los parámetros especificados (como se muestra en Figura 33) y así obtener una respuesta con los recursos que satisfagan esta petición.

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

*Figura 33: Llamada HTTP GET para obtener un paciente de id lógica 23, obtenido de la especificación FHIR*

La mayoría de recursos comparten un conjunto común de parámetros de búsqueda, apreciable en Figura 34. Además, cada recurso del estándar posee un conjunto de parámetros específicos referentes a los elementos del recurso; por ejemplo, en Figura 35 se encuentran los parámetros de búsqueda correspondientes al recurso *Medication*, por lo que se puede solicitar recursos que posean cierto **code,** o **identifier**, o una lista de todos los medicamentos que estén compuestos por un mismo **ingredient**.

Los parámetros tienen sus propios tipos de datos, que pueden ser revisados en la lista completa de parámetros. **[53]**

| Search Parameter Types | Parameters for all resources | Search result parameters |
|---|---|---|
| Number<br>Date/DateTime<br>String<br>Token<br>Reference<br>Composite<br>Quantity<br>URI<br>Special | _id<br>_lastUpdated<br>_tag<br>_profile<br>_security<br>_text<br>_content<br>_list<br>_has<br>_type | _sort<br>_count<br>_include<br>_revinclude<br>_summary<br>_total<br>_elements<br>_contained<br>_containedType |

*Figura 34: Tabla que contiene información de los parámetros de búsqueda en común de todos los recursos, obtenido de [53]*

*De izquierda a derecha, los tipos de datos que pueden tomar, los parámetros como tal y parámetros especiales que sirven para manipular los recursos recibidos.*

| Name | Type | Description | Expression |
|---|---|---|---|
| code | token | Returns medications for a specific code | Medication.code |
| expiration-date | date | Returns medications in a batch with this expiration date | Medication.batch.expirationDate |
| form | token | Returns medications for a specific dose form | Medication.form |
| identifier | token | Returns medications with this external identifier | Medication.identifier |
| ingredient | reference | Returns medications for this ingredient reference | (Medication.ingredient.item as Reference) (Medication, Substance) |
| ingredient-code | token | Returns medications for this ingredient code | (Medication.ingredient.item as CodeableConcept) |
| lot-number | token | Returns medications in a batch with this lot number | Medication.batch.lotNumber |
| manufacturer | reference | Returns medications made or sold for this manufacturer | Medication.manufacturer (Organization) |
| status | token | Returns medications for this status | Medication.status |

*Figura 35: Tabla que contiene información de los parámetros de búsqueda del recurso Medication, obtenido de la especificación FHIR*
*De izquierda a derecha, el nombre del parámetro, tipo de dato del parámetro, descripción del parámetro y la expresión utilizada a nivel interno por la implementación.*

Con esta especificación de búsqueda es posible relegar la responsabilidad del sistema de búsquedas al estándar y simplemente utilizarlo para obtener los recursos necesarios de peticiones simples o complejas para casos de usos específicos.

### 8.1.4 Extensiones y perfiles

Las **extensiones** en FHIR corresponden a elementos adicionales dentro de un recurso. Estos pueden ser de tipo primitivo o complejos, por lo que es posible definir extensiones anidadas. Para esto se utiliza el elemento **extension** que todos los recursos heredan, y que a su vez es de tipo complejo llamado **Extension.** Este tipo contiene 2 elementos, una **url** con la cual se identifica la extensión y suele ser una referencia canónica (cuando es una extensión anidada se utiliza una referencia relativa), y el valor **value[x]** que puede tomar la extensión, el cual puede ser definido como cualquier tipo de dato del estándar, como se ve en Figura 36, indicado en su tipo con un carácter *.

**Structure**

| Name | Flags | Card. | Type | Description & Constraints |
|---|---|---|---|---|
| Extension | N | | Element | Optional Extensions Element<br>*+ Rule: Must have either extensions or value[x], not both*<br><br>Elements defined in Ancestors: id, extension |
| url | | 1..1 | uri | identifies the meaning of the extension |
| value[x] | C | 0..1 | * | Value of extension |

*Figura 36: Elemento complejo Extension, obtenido de la especificación FHIR*
*De arriba hacia abajo, contiene el url de la extensión (referencia canónica) y el valor que toma.*

Existen dos tipos de extensiones:

- Extensiones validadas por el estándar y que se encuentran disponibles en línea de manera nativa, por lo que su uso debiera ser entendido por todas las implementaciones del estándar en

la versión de FHIR correspondiente. Estas extensiones pueden ser utilizadas simplemente mediante su URL canónica.

- Extensiones definidas de manera personalizada por el implementador. Para que una implementación de FHIR soporte estas extensiones, es necesario que además de tener su URL canónica, tenga acceso a su definición de estructura en un recurso *StructureDefinition*, el cual describe el recurso y elemento donde se aplica la extensión, además de descripciones, cardinalidad de la extensión y el tipo de datos que soporta. Por lo general esta definición puede almacenarse en un servidor FHIR, o enviar una solicitud de cambio al estándar para integrar la extensión de manera nativa [54]. Dado que es definida a nivel local por un implementador, si se planea lograr la interoperabilidad con otros sistemas, es necesario que tengan conocimiento de la definición de esta extensión para poder procesarlo.

Por ejemplo, en Figura 37 se da la definición de una extensión integrada al estándar que representa a un paciente animal, agregando elementos que no se encuentran en el recurso *Patient* de manera nativa. Esta extensión **patient-animal** es de tipo complejo, por lo que tiene múltiples extensiones anidadas; **species** para la especie del animal, **breed** para la raza del animal, y **genderStatus** para el estado de su aparato reproductor (esterilización), cada una con su **url** y su **value,** que en este caso son de tipo CodeableConcept, adhiriéndose a los ValueSet respectivos.

| Name | Flags | Card. | Type | Description & Constraints |
|---|---|---|---|---|
| Extension | | 0..1 | Extension | This patient is known to be an animal (non-human) |
| extension:species | | 1..1 | Extension | The animal species. E.g. Dog, Cow. |
| ~~extension~~ | | ~~0..0~~ | | |
| url | | 1..1 | uri | "species" |
| value[x] | | 1..1 | CodeableConcept | Value of extension<br>**Binding:** AnimalSpecies (example): The species of an animal. |
| extension:breed | | 0..1 | Extension | The animal breed. E.g. Poodle, Angus. |
| ~~extension~~ | | ~~0..0~~ | | |
| url | | 1..1 | uri | "breed" |
| value[x] | | 1..1 | CodeableConcept | Value of extension<br>**Binding:** AnimalBreeds (example): The breed of an animal. |
| extension:genderStatus | | 0..1 | Extension | The status of the animal's reproductive parts. E.g. Neutered, Intact. |
| ~~extension~~ | | ~~0..0~~ | | |
| url | | 1..1 | uri | "genderStatus" |
| value[x] | | 1..1 | CodeableConcept | Value of extension<br>**Binding:** GenderStatus (example): The state of the animal's reproductive organs. |
| url | | 1..1 | uri | "http://hl7.org/fhir/StructureDefinition/patient-animal" |
| ~~value[x]~~ | | ~~0..0~~ | | |

*Figura 37: Extensión compleja patient-animal, obtenido de la especificación FHIR*
*De arriba hacia abajo, contiene extensiones anidadas de tipo CodeableConcept que describen a un paciente animal, como especie, raza y estado de esterilización.*

Los **perfiles** son versiones modificadas de recursos. Añaden aún más personalización en la definición de éstos, permitiendo modificación de cardinalidades de los elementos, agregando nuevas reglas a los recursos, y la opción de añadir extensiones específicas como elementos que son parte del recurso. Básicamente permite configurar un recurso para adherirse mejor a un caso de uso específico.

Del mismo modo que las extensiones, existen perfiles validados por el estándar disponibles de manera pública, y es posible definir nuevos perfiles por el implementador. Para estos últimos también es necesario tener acceso a la estructura del perfil con un recurso *StructureDefinition*, y que los sistemas que operen con este perfil tengan conocimiento de esta estructura para procesarlo de manera correcta.

Para indicar que un recurso se adhiere a la estructura de un perfil, es necesario que en su elemento **meta.profile,** se declare la URL canónica de la definición del perfil, de esta manera se es conforme con la especificación y el validador de una implementación FHIR puede comprobar si un recurso de este perfil efectivamente sigue su estructura.


```
"meta": {
  "versionId": "1",
  "lastUpdated": "2023-08-01T02:15:01.399+00:00",
  "source": "#QjDqPty7wq6TAZLu",
  "profile": [ "https://example.org/fhir/StructureDefinition/MedicationRequestHoursWindow" ]
},
```


*Figura 38: Elemento meta de recurso MedicationRequest, que se adhiere a un perfil llamado MedicationRequestHoursWindow, elaboración propia.*

Es recomendable siempre verificar si se encuentra disponible una extensión en la especificación para representar un caso de uso antes de crearla como una extensión nueva, para no perder el propósito del estándar y crear muchas diferencias entre las implementaciones con extensiones que cumplen un propósito similar. La misma recomendación es aplicable para los perfiles.

## 8.2 Detalle de arquitectura de datos subyacente FHIR

En el siguiente listado, se refiere con **campos**, a los campos del modelo de datos de la entidad correspondiente, y con **elementos,** a los atributos de los recursos FHIR.
A continuación se describen el resto de elementos particulares a cada recurso:

1. **Patient:** Recurso utilizado para representar información de un paciente (entidad **Paciente**). Sus elementos son:
    a. **identifier:** Identificador de negocio del paciente (diferente de identificador interno **id**). Se utiliza el campo **rut_paciente** en este elemento.
    b. **active:** Indica si el registro del paciente se encuentra en uso. Por defecto, este elemento queda con un valor de activo.
    c. **name:** Nombre completo del paciente. Elemento de tipo **HumanName,** compuesto por elementos **family** (para los apellidos) y **given** (para los nombres). Se utilizan los campos **apellidos** y **nombres** respectivamente.
    d. **telecom:** Información de contacto del paciente. Elemento de tipo **ContactPoint,** compuesto por elementos **system** (forma de contacto) y **value** (string con el contacto). Se crean dos registros de ContactPoint para ambos tipos de contacto, y se utilizan los campos **teléfono** y **email** respectivamente.

e. **gender:** Género del paciente. Debe tomar uno de los siguientes valores: **male, female, other** o **unknown**. Se utiliza el campo **género** para decidir.
f. **birthDate:** Fecha de nacimiento del paciente. Se utiliza el campo **fecha_nacimiento** en este elemento.
g. **address:** Dirección del paciente. Elemento de tipo **Address,** compuesto por elemento **text** (dirección en texto). Se utiliza una concatenación de los campos **ciudad** y **dirección.**
h. **maritalStatus:** Estado civil del paciente. Elemento de tipo **CodeableConcept,** que toma códigos de un sistema de códigos de estados civiles propio de HL7. Se utiliza el campo **estado_civil** para decidir.

2. **Observation:** Recurso utilizado para registrar una observación (entidades **Ánimo** y **Control).**
   Los elementos para representar **Ánimo** son:
   a. **status:** El estado de la observación registrada. Por defecto siempre lleva el valor "final".
   b. **code:** El código que representa lo observado. Elemento de tipo **CodeableConcept.** Por defecto se usa un código referente al estado de ánimo (o mood) proveniente del sistema SNOMED.
   c. **subject:** El sujeto sobre el cual se hace la observación. Elemento de tipo **Reference,** que hace referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el **rut_paciente.**
   d. **performer:** El responsable de la observación. Elemento de tipo **Reference,** que hace referencia al mismo paciente que registra su estado de ánimo mediante el **rut_paciente.**
   e. **value:** El valor como tal obtenido en la observación. Elemento de tipo **CodeableConcept,** que toma el código de estado de ánimo (feliz, triste, entre otros por decidir con equipo del proyecto) del sistema de códigos SNOMED. Se utiliza el campo **estado** para decidir.

   Los elementos para representar la observación de **Control** son:
   a. **status:** El estado de la observación registrada. Por defecto siempre lleva el valor "final".
   b. **code:** El código que representa lo observado. Elemento de tipo **CodeableConcept.** Por defecto se usa un código genérico referente a trastorno de HTA proveniente del sistema SNOMED.
   c. **subject:** El sujeto sobre el cual se hace la observación. Elemento de tipo **Reference,** que hace referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el **rut_paciente** del tratamiento asociado con **id_tratamiento**.
   d. **encounter:** El encuentro o control donde ocurrió la observación. Elemento de tipo **Reference,** que hace referencia al control mediante su **id** interna. Esta id interna se obtiene con **fecha_control.**
   e. **performer:** El responsable de la observación. Elemento de tipo **Reference,** que hace referencia al médico encargado mediante el **rut_medico** del tratamiento asociado con **id_tratamiento.**
   f. **dataAbsentReason:** Razón de por qué los datos de observación no están disponibles. El recurso Observation tiene un elemento **value** para agregar valores de observaciones más específicas (una medición de nivel de glucosa por ejemplo). En este caso, no se usa ese

elemento por lo que se manifiesta en el elemento presente de tipo **CodeableConcept** un código de **"Unsupported"** proveniente de un sistema de códigos propio de HL7.

g. **note:** Contenido y comentarios de la observación. De tipo **Annotation**, compuesto por elementos **time** (cuándo se hizo la observación) y **text** (la observación como tal). Se utilizan los campos **fecha_control** y **observaciones** respectivamente.

3. **Practitioner:** Recurso utilizado para representar información de un médico, especialista o personal sanitario (entidad **Médico**). Sus elementos son:
   a. **identifier:** Identificador de negocio del médico (diferente de identificador interno **id**). Se utiliza el campo **rut_medico** en este elemento.
   b. **active:** Indica si el registro del médico se encuentra en uso. Por defecto, este elemento queda con un valor de activo.
   c. **name:** Nombre del médico. Elemento de tipo **HumanName,** compuesto por el elemento **given** (para el nombre). Se utiliza el campo **nombre** en este elemento.
   d. **qualification:** Calificaciones o certificaciones del médico relacionadas al área médica. Elemento estructural compuesto por un **code,** el cual es un elemento de tipo **CodeableConcept** que representa la calificación como tal y toma el valor genérico de "Doctor en Medicina" proveniente de un sistema de códigos de calificaciones propio de HL7. Se incluye en el elemento **text** del **code** el campo **especialidad** para declarar esta calificación del médico.

4. **CarePlan:** Recurso utilizado para representar un tratamiento médico (entidad **Tratamiento**). Sus elementos son:
   a. **identifier:** Identificador de negocio del tratamiento (diferente de identificador interno **id**). Se utiliza el campo **id_tratamiento** en este elemento.
   b. **status:** El estado del tratamiento. Toma valores de un ValueSet de estados, y se utiliza el campo **estado** para decidir.
   c. **intent:** El nivel de intencionalidad o autoridad del tratamiento. Toma valores de un ValueSet de intencionalidades, y por defecto se deja en un valor de "Order" para especificar que representa una solicitud y está autorizado por un médico.
   d. **category:** El tipo de plan o tratamiento. De tipo **CodeableConcept,** que toma códigos de un sistema de códigos de categorías de tratamientos propio de HL7. Por defecto se usa un código genérico referente a plan de manejo clínico de hipertensión, proveniente del sistema SNOMED.
   e. **title:** Nombre o título del tratamiento. Por defecto se estableció titular todos los tratamientos como "Tratamiento HTA".
   f. **description:** Resumen o descripción del tratamiento. Se utiliza el campo **descripción** en este elemento.
   g. **subject:** El sujeto al cual se hace el tratamiento. Elemento de tipo **Reference,** que hace referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el campo **rut_paciente.**

h. **encounter:** El encuentro donde fue creado el tratamiento. Elemento de tipo **Reference**, que hace referencia a un encuentro mediante su **id** interna. El control debe existir previamente y la id interna se obtiene con los campos **rut_paciente** y **fecha_registro,** para encontrar el control de ese paciente en la fecha que se registró el tratamiento.
i. **period:** Periodo cubierto por el tratamiento. Elemento de tipo **Period,** compuesto por elementos **start** (inicio del periodo) y **end** (final del periodo). Se utilizan los campos **fecha_inicio** y **fecha_término** respectivamente. Si no se especifica fecha_termino, sólo se llena el elemento start.
j. **created:** Fecha de creación del tratamiento. Se utiliza el campo **fecha_registro** en este elemento.
k. **author:** Responsable del tratamiento. Elemento de tipo **Reference,** que hace referencia al médico autor mediante su id interna. Esta id interna se obtiene mediante el **rut_medico**.
l. **addresses:** Patologías o condiciones que cubre el tratamiento. Elemento de tipo **Reference**, que hace referencia a un recurso **Condition** mediante su id interna. Sin embargo, esta implementación no trabaja con esos recursos de momento porque sólo opera con la condición de HTA, por lo que en este elemento, por defecto se referencia un código genérico del sistema SNOMED para referirse al trastorno hipertensivo arterial sistémico.
m. **activity:** Acciones a ocurrir como parte de un tratamiento. Elemento estructural compuesto por elementos **Reference**, que pueden hacer referencia a múltiples recursos de actividades. Se inicializa vacío al crear el tratamiento pero al agregar prescripciones (**MedicationRequest**) referentes al tratamiento se añaden referencias a esas prescripciones.

5. **Encounter:** Recurso utilizado para representar ocurrencia de un control (entidad **Control**). Sus elementos son:
   a. **status:** Estado del encuentro. Toma valores de un ValueSet de estados, y por defecto se asigna como “finalizado”, ya que estos recursos se crean al finalizar un control médico.
   b. **class:** Clase del control médico. Toma valores de un ValueSet de clasificaciones de encuentros, y por defecto se asigna como “ambulatorio” por la naturaleza de estos controles médicos que se darán durante el proyecto.
   c. **serviceType:** Tipo de servicio que se realiza en el control médico. De tipo **CodeableConcept,** toma códigos de un sistema de códigos de tipos de servicio propio de HL7. Por defecto se usa un código genérico referente a la hipertensión.
   d. **subject:** El sujeto presente en el encuentro. Elemento de tipo **Reference,** que hace referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el **rut_paciente** del tratamiento asociado con **id_tratamiento**.
   e. **participant:** Lista de participantes involucrados en el encuentro. Es un elemento estructural compuesto por elementos **individual,** que son de tipo **Reference**, y referencian a un médico mediante su **id** interna. Esta id interna se obtiene mediante el **rut_medico** del tratamiento asociado con **id_tratamiento**.

f. **period:** Periodo cubierto por el encuentro. Elemento de tipo **Period,** compuesto por elementos **start** (inicio del periodo) y **end** (final del periodo). Se utiliza el campo **fecha_control** para declarar que el control inició y terminó en la misma fecha.
g. **diagnosis:** Lista de diagnósticos relevantes para el encuentro. Elemento estructural compuesto por **condition** (condición relevante del paciente) y **use** (el rol del diagnóstico en el encuentro). El elemento **condition** es de tipo **Reference** para referenciar un recurso **Condition,** sin embargo, aquí solo se ha referenciado el concepto de trastorno hipertensivo arterial sistémico en SNOMED. El elemento **use** es de tipo **CodeableConcept** que toma valores de un sistema de códigos de roles de diagnóstico propio de HL7, y por defecto se asigna su valor en "diagnóstico de admisión".

6. **MedicationRequest:** Recurso utilizado para representar solicitudes de medicamento de un personal de salud a un paciente. En otras palabras, representa las prescripciones. (entidades **Prescripción** y **Posología**).
   Para este recurso se han realizado dos definiciones estructurales mediante el recurso **StructureDefinition**, correspondientes a una extensión llamada **TimingHourWindows,** la cual agrega la hora de término de un periodo para ingerir un medicamento y un perfil de MedicationRequest llamado **MedicationRequestHoursWindowProfile,** que utiliza la extensión anterior. Se crean estas definiciones estructurales ya que FHIR de manera nativa no soporta un elemento para guardar la hora de término de un paciente para ingerir un medicamento, lo cual fue confirmado con uno de los administradores de FHIR en el chat comunitario de FHIR (disponible en **Anexos**). Sus elementos son:
   a. **meta.profile:** Perfil al cual se adhiere el recurso. Es un elemento que contiene una referencia canónica a un recurso **StructureDefinition**, el cual contiene la estructura del perfil. Este elemento puede contener la referencia al perfil **MedicationRequestHoursWindowProfile** si es que el recurso contiene horarios de ingesta (campos **hora_ingesta** y **hora_termino** en **Posologia**). En caso contrario, este elemento se deja vacío.
   b. **status:** El estado de la prescripción. Toma valores de un ValueSet de estados, y se utiliza el campo **estado** para decidir.
   c. **intent:** El nivel de intencionalidad o autoridad de la prescripción. Toma valores de un ValueSet de intencionalidades, y por defecto se deja en un valor de "Order" para especificar que representa una solicitud y está autorizado por un médico.
   d. **medication:** El medicamento prescrito. Elemento de tipo **Reference,** que hace referencia a un medicamento mediante su **id** interna. Esta id interna se obtiene mediante el campo **codigo_id_medicamento**.
   e. **subject:** El sujeto al cual se realiza la prescripción. Elemento de tipo **Reference,** que referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el **rut_paciente** del tratamiento asociado con **id_tratamiento**.
   f. **encounter:** El encuentro donde fue creado la prescripción. Elemento de tipo **Reference**, que hace referencia a un encuentro mediante su **id** interna. La id interna se obtiene con el campo **fecha_control**.

g. **authoredOn:** Fecha cuando fue creada la prescripción. Se utiliza el campo **fecha_control** en este elemento.
h. **requester:** El sujeto que realizó la prescripción. Elemento de tipo **Reference,** que referencia a un médico mediante su **id** interna. Esta id interna se obtiene mediante **rut_medico** del tratamiento asociado con **id_tratamiento.**
i. **basedOn:** Tratamiento al cual pertenece esta prescripción. Elemento de tipo **Reference,** que referencia a un tratamiento mediante su **id** interna. Esta id interna se obtiene mediante el campo **id_tratamiento.**
j. **dosageInstruction:** Lista de instrucciones de dosificación o posología del medicamento de la prescripción. Elemento de tipo **Dosage**, y se agrega una nueva entrada a la lista por cada dosificación diferente que exista en **Posología** para la prescripción**,** es decir, si hay diferencias en los campos **dosis** de las múltiples posologías para una misma prescripción. Cada instrucción está compuesta por los siguientes elementos:
    i. **sequence:** Indica el orden de la instrucción si hay múltiples. Si todas tienen el mismo valor, indica que ocurren de manera concurrente. Se deja así por defecto, ya que las diferentes instrucciones cambian en dosis y horarios pero no indica que ocurren de manera secuencial en el orden que fueron ingresadas.
    ii. **text:** Instrucciones en texto libre. Se utiliza el campo **instruccionesTexto** en este elemento.
    iii. **timing:** Cuándo se debe ingerir el medicamento. Elemento de tipo **Timing**, el cual está compuesto por el elemento **repeat** (conjunto de reglas que indica cuándo ocurre el evento), que a su vez está compuesto por los siguientes elementos:
        1. **bounds:** Límites en los que transcurre la prescripción. Elemento de tipo **Period,** compuesto por elementos **start** (inicio del periodo) y **end** (final del periodo). Se utilizan los campos **fecha_inicio** y **fecha_término** de **Prescripción,** respectivamente. Si no se especifica fecha_termino, sólo se llena el elemento start.
        2. **frequency:** Número de veces que se repite la acción dentro del periodo **period** especificado. Se utiliza el campo **frecuencia_horaria** para este elemento. Se le suma 1 si existe otra entrada de Posología con **dosis** similar pero en otro horario.
        3. **period:** Duración de tiempo sobre el cual las repeticiones de **frequency** se realizan. Se utiliza el campo **frecuencia_dias** para este elemento.
        4. **periodUnit:** Unidad del periodo. Toma valores de un ValueSet de unidades de tiempo propio de HL7. Por defecto, se utiliza el código que representa “día”. En conjunto con los dos anteriores elementos, se puede indicar el número de repeticiones de ingesta diarias a realizar, y si las repeticiones se realizan diariamente o en intervalos de días.
        5. **timeOfDay:** Horario del día donde ocurre la acción. Utiliza el campo **hora_inicio** de **Posología.**
        6. **extension:** Elemento para incluir contenido adicional. Compuesto por elementos **url** (referencia canónica a **StructureDefinition** de extensión) y **value** (valor de la extensión). En esta implementación, si el recurso se adhiere al perfil mencionado

anteriormente, el elemento **url** usa el **StructureDefinition** de la extensión **TimingHourWindows,** y **value** corresponde al campo **hora_termino** de **Posología.**

iv. **asNeeded:** Indica si la medicina debiera tomarse "según sea necesario", en lugar de seguir los horarios especificados. Por defecto, toma un valor de falso.

v. **route:** Lugar del cuerpo por donde se ingiere la medicina. Elemento **CodeableConcept** que toma códigos del sistema SNOMED. Por defecto, toma el código del concepto "vía oral (calificador)".

vi. **doseAndRate:** Dosis del medicamento a ingerir en los horarios establecidos en el elemento **timing.** Elemento estructural compuesto por elemento **dose** (cantidad de medicamento por dosis) de tipo **Quantity**. Cuenta con los elementos **value** y **unit** para el valor y unidad de la cantidad, y se utilizan los campos respectivos **dosis** y **unidad** de Posología para estos elementos.

k. **substitution:** Restricciones sobre sustitución de medicamento. Elemento estructural compuesto por el elemento **allowed** (indica si se permiten reemplazos), que es de tipo **CodeableConcept** y toma valores de un sistema de códigos de sustitución de sustancias propio de HL7. En un principio, se asume que todos los medicamentos pueden ser sustituidos por un equivalente, por lo que se asigna un valor de "equivalente".

7. **Medication:** Recurso utilizado para representar información de un medicamento (entidad **Medicamento**). Sus elementos son:
   a. **identifier:** Identificador de negocio del medicamento (diferente de identificador interno **id**). Se utiliza el campo **codigo_id_medicamento** para este elemento.
   b. **code:** Código que identifica el medicamento. De tipo **CodeableConcept**, toma códigos de medicamentos del sistema de códigos SNOMED. Este elemento está presente si se dispone del código SNOMED del medicamento (es decir, si es que se encuentran en el servidor previamente) o se deja vacío en caso contrario (cuando un paciente ingresa un medicamento nuevo de manera manual en su aplicación).
   c. **status:** Estado que indica si el medicamento se encuentra activo en el servidor. Toma valores de un ValueSet de estados, y se utiliza el campo **estado** para decidir.
   d. **form:** Formato de presentación del medicamento. De tipo **CodeableConcept,** toma códigos de un ValueSet de formas de presentación del sistema SNOMED. Por defecto, se usa el código referente al formato de comprimidos (forma farmacéutica básica), debido a que los medicamentos del plan piloto ingresados son comprimidos. En otro caso se podría utilizar el campo **presentación** para decidir.
   e. **ingredient:** Lista de ingredientes presentes en el medicamento. Es un elemento estructural. Si bien es una lista, se decidió tener sólo el ingrediente principio activo del medicamento. Está compuesto por los siguientes elementos:
   f. **item:** De tipo **CodeableConcept**, toma códigos de sustancias y/o principios activos del sistema de códigos SNOMED. Este elemento está presente si se dispone del código SNOMED del principio activo (es decir, si es que se encuentran en el servidor previamente), o de lo contrario se ingresa el nombre del principio activo sin el código con

el campo **nombre** (cuando un paciente ingresa un medicamento nuevo de manera manual en su aplicación).

i. **isActive:** Indica si el ingrediente es un principio activo. Por defecto, se asigna un valor de verdadero.

ii. **strength:** Fracción que representa la cantidad del ingrediente presente en un medicamento. De tipo **Ratio,** compuesto por elementos **numerator** (valor del numerador) y **denominator** (valor del denominador). Ambos elementos son de tipo **Quantity,** el cual refleja una cantidad con un valor numérico, una unidad y el sistema de código de dónde proviene la unidad**.** Se ha establecido que esta fracción sea del formato "cantidad en miligramos por comprimido". Dicho eso, **numerator** toma el valor numérico de la cantidad de principio activo en miligramos, con el código del sistema **unitsofmeasure.org.** Por otro lado, **denominator** toma el valor 1 y unidad de "comprimido", con el código SNOMED. Se utilizan los campos **volumen** y **unidad** para el valor y unidad del elemento numerator.

8. **MedicationAdministration:** Recurso utilizado para representar una administración de medicamento en un paciente. En otras palabras, representa los registros de ingesta (entidad **Registro_Ingesta**). Sus elementos son:
   a. **status:** El estado de la ingesta de medicamento. Toma valores de un ValueSet de estados, y se utiliza el campo **estado** para decidir.
   b. **medication:** El medicamento ingerido. Elemento de tipo **Reference,** que hace referencia a un medicamento mediante su **id** interna. Esta id interna se obtiene mediante el campo **codigo_id_medicamento**.
   c. **subject:** El sujeto que realiza la ingesta. Elemento de tipo **Reference,** que referencia a un paciente mediante su **id** interna. Esta id interna se obtiene mediante el campo **rut_paciente**.
   d. **effective:** Fecha y horario cuando ocurrió la ingesta. Se utiliza el campo **fecha_hora_registro** para este elemento.
   e. **request:** Prescripción a la que corresponde la ingesta. Elemento de tipo **Reference,** que hace referencia a una prescripción mediante su **id** interna. Esta id interna se obtiene mediante la **id_prescripción** de la posología asociada con **id_posología.**
   f. **dosage:** Detalles de cómo se ingirió el medicamento. Elemento estructural compuesto por **route** (lugar del cuerpo por donde se ingiere) y **dose** (cantidad de medicamento ingerida). **route** es de tipo **CodeableConcept,** toma valores del sistema SNOMED. Por defecto toma el código del concepto "vía oral (calificador)". **dose** es de tipo **Quantity** y cuenta con los elementos **value** y **unit** para el valor y unidad de la cantidad. Se utilizan los campos respectivos **dosis** y **unidad** para estos elementos.

## 8.3 Peticiones HTTP para clientes nativos

### 8.3.1 Descripción de peticiones HTTP para aplicación móvil

Considerando las APIs utilizadas en los escenarios de compatibilidad de aplicación móvil no nativa de FHIR, se cuenta con las siguientes llamadas:

- **http://server_url/fhir/Patient?identifier=$rut**
  Llamada de tipo **GET**. Lee la información de un paciente mediante su **$rut**. Retorna un recurso **Bundle** que contiene un recurso de tipo **Patient**, con respuesta **200 Ok**.
- **http://server_url/fhir/Medication**
  Llamada de tipo **GET**. Lee la información de todos los medicamentos. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **Medication** disponibles en el servidor, con respuesta **200 Ok**.
- **http://server_url/fhir/MedicationAdministration**
  Llamada de tipo **POST.** Ingresa un registro de ingesta de medicamento. En el cuerpo de la petición debe ir el recurso **MedicationAdministration** que representa el registro de la ingesta. Retorna el mismo recurso ingresado con respuesta **201 Creado.**
- **http://server_url/fhir/MedicationRequest?subject=Patient/$id&_include=MedicationRequest:medication**
  Llamada de tipo **GET.** Lee la información de las prescripciones de un paciente mediante su **$id**, e incluye la información de los medicamentos asociados. La **$id** se obtiene con la petición HTTP de información de paciente mediante rut. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **MedicationRequest** del paciente en cuestión, e incluye los recursos **Medication** referenciados en las prescripciones, con respuesta **200 Ok**.

- **http://server_url/fhir/Medication**
  Llamada de tipo **POST**. Ingresa un medicamento al servidor. Se puede utilizar para ingresar un medicamento externo al servidor cuando el paciente lo desea. En el cuerpo de la petición debe ir el recurso **Medication** que representa este medicamento externo (sin código SNOMED). Retorna el mismo recurso ingresado con respuesta **201 Creado**.

### 8.3.2 Descripción de peticiones HTTP para ficha clínica

Considerando las APIs utilizadas en los escenarios de compatibilidad de ficha clínica y módulo de visualización no nativo de FHIR, se cuenta con las siguientes llamadas:

- **http://server_url/fhir/Patient?identifier=$rut**
  Llamada de tipo **GET**. Lee la información de un paciente mediante su **$rut**. Retorna un recurso **Bundle** que contiene un recurso de tipo **Patient**, con respuesta **200 Ok**.

- **http://server_url/fhir/MedicationAdministration?subject=Patient/$id&_include=MedicationAdministration:medication**
  Llamada de tipo **GET.** Lee la información de los registros de ingesta de un paciente mediante su **$id**, e incluye la información de los medicamentos ingeridos. La **$id** se obtiene con la petición HTTP de información de paciente mediante rut. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **MedicationAdministration** del paciente en cuestión, e incluye los recursos **Medication** referenciados en las prescripciones, con respuesta **200 Ok**.

- **http://server_url/fhir/MedicationAdministration?patient=Patient/$id&effective-time=ge$date1&effective-time=le$date2&_include=MedicationAdministration:medication&_include=MedicationAdministration:request**
  Llamada de tipo **GET.** Lee la información de registros de ingesta realizados entre la fecha **$date1** y la fecha **$date2**, de un paciente mediante su **$id**, e incluye la información de los medicamentos ingeridos y la prescripción a la cual se asocian. La **$id** se obtiene con la petición HTTP de información de paciente mediante rut, y las fechas se especifican por el usuario. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **MedicationAdministration** entre las fechas ingresadas, del paciente en cuestión, e incluye los recursos **Medication** de las ingestas, y los recursos **MedicationRequest** a los cuales corresponden las ingestas, con respuesta **200 Ok**.
- **http://server_url/fhir/MedicationRequest?subject=Patient/$id&_include=MedicationRequest:medication**
  Llamada de tipo **GET.** Lee la información de las prescripciones y su posología de un paciente mediante su **$id**, e incluye la información de los medicamentos asociados. La **$id** se obtiene con la petición HTTP de información de paciente mediante rut. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **MedicationRequest** del paciente en cuestión, e incluye los recursos **Medication** referenciados en las prescripciones, con respuesta **200 Ok**.

- **http://server_url/fhir/MedicationAdministration?patient=Patient/$id&effective-time=ge$date1&effective-time=le$date2&_include=MedicationAdministration:request**
  Llamada de tipo **GET.** Lee la información de registros de ingesta realizados entre la fecha **$date1** y la fecha **$date2**, de un paciente mediante su **$id**, e incluye la información de la prescripción a la cual se asocian, con el fin de calcular porcentaje de adherencia. La **$id** se obtiene con la petición HTTP de información de paciente mediante rut, y las fechas se especifican por el usuario. Retorna un recurso **Bundle** que contiene todos los recursos de tipo **MedicationAdministration** entre las fechas ingresadas, del paciente en cuestión, e incluye los recursos **MedicationRequest** a los cuales corresponden las ingestas, con respuesta **200 Ok**. La información obtenida con esta petición es la necesaria para realizar el cálculo de porcentaje de adherencia con algún servicio implementado en el cliente FHIR solicitante.

Estas son algunas de las llamadas posibles para el contexto de interactuar con recursos de adherencia a tratamiento farmacológico. La especificación RESTful de FHIR provee la base para

construir peticiones muy específicas gracias a la definición de los múltiples parámetros de búsqueda, por lo que es compatible con nuevos requerimientos entrantes que puedan surgir durante el transcurso del proyecto FONDEF.

## 8.4 Links de interés

Chat comunitario de FHIR

Estructura de tabla de recursos FHIR

Tarifas de Google Cloud Healthcare API

Tarifas de Azure Health Data Services

Tarifas de Simplifier.net

Documento con características principales de IRIS for Health

Esquema de BD HAPI FHIR JPA

Informe técnico AFAM 1.0

Enlace repositorio GitHub AFAM 2.0

Especificación RESTful FHIR

Enlace modelo de datos AFAM 2.0

Respuesta Lloyd McKenzie FHIR sobre extensión