

Trabajo Fin de Grado

Ingeniería de Telecomunicación

Servicio de adquisición y filtrado de consentimientos de pacientes en formato HL7 FHIR

Autor: Mario Martín Casanova
Tutor: Jorge Calvillo Arbizu

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022



Proyecto Fin de Grado
Ingeniería de Telecomunicación

Servicio de adquisición y filtrado de consentimientos de pacientes en formato HL7 FHIR

Autor:
Mario Martín Casanova

Tutor:
Jorge Calvillo Arbizu
Profesor Ayudante Doctor

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Servicio de adquisición y filtrado de consentimientos de pacientes en
formato HL7 FHIR

Autor: Mario Martín Casanova
Tutor: Jorge Calvillo Arbizu

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia
A mis maestros

Agradecimientos

Quisiera agradecer todo este esfuerzo que he tenido que hacer para llegar hasta este punto a varias personas.

En primer lugar, agradecer a mi familia por el apoyo incondicional que me han dado a lo largo no solo de la realización de este trabajo, sino también de la carrera.

En segundo lugar, agradecer a mi tutor Jorge por hacer que todo sea más sencillo. Desde el inicio de la carrera le dije que sería mi tutor de TFG y así fue.

En tercer lugar, agradecer a todos mis amigos tanto de Huelva como los que he hecho a lo largo de la carrera. Fran, Estrella, Miguel, Alejandro. Han sido mi soporte y no estaría donde estoy si no fuera gracias a ellos.

Por último, dejar cuatro menciones especiales que son las personas, junto con mi familia, que más me han aportado y les tengo un amor inmenso. Francisco, Cinta, Víctor y Julia. Aportáis mucho más de lo que realmente pensáis.

*Mario Martín Casanova
Sevilla, 2022*

La Regulación General de Protección de Datos (GDPR) establece que el tratamiento de datos personales por parte de las organizaciones debe estar previamente autorizado por los propios ciudadanos a los que se refieren dichos datos. En el dominio sanitario es aún más importante puesto que los datos personales incluyen información sensible como es la relacionada con la salud de la persona. Por ello, el uso que una organización sanitaria hace de los datos sanitarios de sus pacientes (por ejemplo, para análisis, investigación, docencia, etc.) debe estar autorizada por los propios pacientes.

Tradicionalmente el paciente firmaba un consentimiento en papel antes de recibir un servicio por parte de la organización. No obstante, con la revolución digital de todos los sectores de la sociedad podemos encontrarnos que los ciudadanos creen consentimientos independientemente de las organizaciones sanitarias y sean gestionados por terceros (por ejemplo, agencias gubernamentales).

Este trabajo parte de dicho escenario y se basa en la adquisición y filtrado de consentimientos por parte de una organización sanitaria desde una entidad de gestión de consentimientos. La organización sanitaria deberá así recoger solo aquellos consentimientos de sus pacientes y que sean relevantes con la información almacenada en sus sistemas de información sanitarios. De esta forma, cada corporación observará los pacientes que tiene y reunirá los consentimientos públicos para filtrarlos, recogiendo la información que se tiene sobre cada una de las personas en su base de datos privada. En el caso de que exista un consentimiento que restringe o acepta cierta acción, pero el paciente no contiene dichos de datos, se desechará ya que no se le dará un uso al consentimiento.

La idea principal es almacenar consentimientos de los pacientes que estén relacionados con la información que se tiene sobre ellos, así si la organización quiere realizar un estudio pues revisará los consentimientos de la persona específica para mirar si se ha aceptado o se ha denegado el uso.

El formato de los consentimientos adopta el estándar FHIR de Health Level 7 International (HL7®) diseñado para permitir un intercambio rápido de registros sanitarios electrónicos. FHIR trata de combinar lo mejor de cada uno de los modelos que están actualmente en uso con estándares web modernos de forma que se mejore en la medida de lo posible la implementación de los estándares de interoperabilidad.

Finalmente, se llega a la conclusión de que es necesario comprobar los consentimientos de los pacientes en el caso de que la organización sanitaria quiera realizar un estudio. Para ello, se va a usar el estándar FHIR con la definición de 4 recursos (pueden ser más). Además, se hace uso de la API que proporciona el estándar llamado HAPI-FHIR para almacenar primeramente los datos de los pacientes y su información correspondiente. Para almacenar los consentimientos se ha decidido usar una base de datos no relacional (por ejemplo, MongoDB). Se ha de remarcar que el uso de esta aplicación es muy sencillo para la organización y que en un futuro que exista una interfaz gráfica hará que sea más intuitivo, facilitando el trabajo. Finalmente, el uso de FHIR no es sencillo y sumándole la poca documentación que existe hace que el realizar un proyecto sobre dicho estándar sea un trabajo tedioso.

The General Data Protection Regulation (GDPR) states that the processing of personal data by organisations must be previously authorised by the citizens to whom the data relates. In the field of health, it is even more important since personal data include sensitive information such as that related to the health of the person. Therefore, a healthcare organisation's use of its patients' health data (e. g. for analysis, research, teaching, etc.) must be authorised by the patients themselves.

Traditionally, the patient signed a written consent before receiving a service from the organization. However, with the digital revolution in all sectors of society, we may find that citizens create consents independently of health organisations and are managed by third parties (e. g. government agencies).

This work starts from this scenario and is based on the acquisition and filtering of consents by a health organization from a consent management entity. The healthcare organisation should therefore collect only those consents of its patients that are relevant to the information stored in its healthcare information systems. In this way, each corporation will observe the patients it has and gather public consents to filter them, collecting the information it has about each individual in its private database. In the event that there is a consent that restricts or accepts a certain action, but the patient does not contain such data, it will be discarded as the consent will not be used. The main idea is to store patient consents that are related to the information that is held about them, so if the organization wants to conduct a study it will review the consents of the specific person to see if the use has been accepted or denied.

The form of consent adopts the FHIR standard of Health Level 7 International (HL7 ®) designed to enable rapid exchange of electronic health records. FHIR strives to combine the best of each of the models currently in use with modern web standards so that the implementation of interoperability standards is improved as much as possible.

Finally, it is concluded that it is necessary to check patients' consent if the health organisation wants to carry out a study. For this, we will use the FHIR standard with the definition of 4 resources (may be more). In addition, the API provided by the standard called HAPI-FHIR is used to first store patient data and related information. To store consents it has been decided to use a non-relational database (e. g. MongoDB). It should be noted that the use of this application is very simple for the organization and that in the future the existence of a graphical interface will make it more intuitive, facilitating the work. Finally, the use of FHIR is not easy and adding the little documentation that exists makes it tedious to carry out a project on this standard.

Agradecimientos	vi
Resumen	vii
Abstract	ix
Índice	xi
Índice de Tablas	xiii
Índice de Figuras	xv
Notación	xviii
1 Introducción	1
1.1 <i>Motivación y objetivos</i>	1
1.2 <i>Metodología Empleada</i>	3
1.3 <i>Plan de trabajo</i>	3
2 MATERIALES	5
2.1 <i>FHIR</i>	5
2.1.1 Tipos de datos	6
2.1.2 Recursos FHIR	9
2.2 <i>Herramientas Software</i>	22
2.2.1 Visual Studio Code	22
2.2.2 Postman	23
2.2.3 Docker	24
2.2.4 MongoDB	25
2.2.5 JSON	26
2.2.6 HAPI FHIR (Librería externa)	26
2.2.7 Diagrams.net	27
2.2.8 Maven	28
2.2.9 Swagger UI	29
3 Resultados	31
3.1 <i>Análisis del proyecto</i>	31
3.1.1 Base de datos EHR FHIR	31
3.1.2 Servicio de Importación	35
3.1.3 Servicio de Filtrado	35
3.1.4 Base de datos Consent	35
3.2 <i>Servidor FHIR</i>	36
3.2.1 Instalación	36
3.2.2 Puesta en marcha	36
3.3 <i>Implementación FHIR</i>	36
3.3.1 Estructura	36
3.3.2 Implementación código FHIR	38
3.4 <i>Base de datos</i>	45
3.4.1 Instalación	45
3.4.2 Puesta en marcha	45

3.4.3	Modelado	45
3.5	<i>Pruebas</i>	46
3.5.1	Ejecución de la aplicación	46
3.5.2	Creación de instancias Patient	47
3.5.3	Creación de instancias Consent	48
3.5.4	Creación de instancias Observation	49
3.5.5	Filtrar por código	50
3.5.6	Filtrar por capa de seguridad	52
3.5.7	Filtrar por fecha	53
3.5.8	Filtrar por referencia	54
4	Conclusiones y Líneas futuras	57
4.1	<i>Conclusiones</i>	57
4.2	<i>Líneas futuras</i>	57
	Referencias	59
	Anexo a: Manual de instalación y despliegue de la aplicación	62

ÍNDICE DE TABLAS

Tabla 1 Plan de trabajo	4
Tabla 2 Método main App	41
Tabla 3 Método pocosPacientes App	41
Tabla 4 Método muchosPacientes App	41
Tabla 5 Método pocosConsentimientos App	42
Tabla 6 Método muchosConsentimientos App	42
Tabla 7 Método importación	42
Tabla 8 Método getPacientes	43
Tabla 9 Método importarConsentimiento	43
Tabla 10 Método getObservation	43
Tabla 11 Método Filtrado	43
Tabla 12 Método filtrado con parámetros	43
Tabla 13 Método filtrar	44
Tabla 14 Método filtrarData	44
Tabla 15 Método filtrarSecurityLabel	44
Tabla 16 Método filtrarCode	44
Tabla 17 Método filtrarFecha	44
Tabla 18 Método acceso	45
Tabla 19 Método insertarDatos	46
Tabla 20 Método comprobarDatos	46
Tabla 21 Método actualizarDatos	46
Tabla 22 Método recogerConsent	46

ÍNDICE DE FIGURAS

Ilustración 1.1 Estructura del proyecto	2
Ilustración 1.1.2 Metodología incremental	3
Ilustración 2.1 Módulos FHIR	6
Ilustración 2.2 Datos primitivos	7
Ilustración 2.3 JSON primitivo	7
Ilustración 2.4 Datos complejos	8
Ilustración 2.5 Datos Complejos JSON	8
Ilustración 2.6 Datos complejos y extensión	8
Ilustración 2.7 Tipos de metadatos	9
Ilustración 2.8 Tipos de datos de propósito especial	9
Ilustración 2.9 Meta resource	10
Ilustración 2.10 Contenido del recurso básico “Resource”	10
Ilustración 2.11 Lista de recursos	11
Ilustración 2.12 Recurso Patient	12
Ilustración 2.13 Instancia de recurso Patient en formato JSON	13
Ilustración 2.14 FHIR consentimiento parte 1	14
Ilustración 2.15 FHIR consentimiento parte 2	14
Ilustración 2.16 Consentimiento JSON	15
Ilustración 2.17 Recurso Bundle	17
Ilustración 2.18 Instancia de recurso Bundle en formato JSON	18
Ilustración 2.19 Recurso Observation parte 1	19
Ilustración 2.20 Recurso Observation parte 2	20
Ilustración 2.21 Instancia de recurso Observation en formato JSON	21
Ilustración 2.22 IDE VS CODE	22
Ilustración 2.23 Visual Studio code Logo	23
Ilustración 2.24 Postman	24
Ilustración 2.25 POSTMAN Logo	24
Ilustración 2.26 Docker Desktop	25
Ilustración 2.27 Docker Desktop logo	25
Ilustración 2.28 Logo MongoDB	25
Ilustración 2.29 JSON	26
Ilustración 2.30 Base de datos HAPI FHIR pública	26
Ilustración 2.31 Base de datos HAPI FHIR privada o local	27
Ilustración 2.32 Logo HAPI FHIR	27

Ilustración 2.33 Representación de diagrams.net	28
Ilustración 2.34 Maven	29
Ilustración 3.1 POST de un consentimiento	33
Ilustración 3.2 GET al consentimiento	33
Ilustración 3.3 Swagger UI	34
Ilustración 3.4 Consent URL JSON	34
Ilustración 3.5 Estructura del proyecto	36
Ilustración 3.6 Estructura Database	37
Ilustración 3.7 Estructura Filtrado	37
Ilustración 3.8 Estructura Importacion	38
Ilustración 3.9 Diagrama de Actividad de App v1	38
Ilustración 3.10 Diagrama de Actividad del método pocosPacientes	39
Ilustración 3.11 Diagrama de Actividad del método pocosConsentimientos	40
Ilustración 3.12 Diagrama de clase Database	45
Ilustración 3.13 “exeTFG.sh”	46
Ilustración 3.14 POST patient	47
Ilustración 3.15 Body Json	47
Ilustración 3.16 POSTMAN Patient	47
Ilustración 3.17 POSTMAN 201 Created	48
Ilustración 3.18 POSTMAN 400 Bad Request	48
Ilustración 3.19 POST consent	48
Ilustración 3.20 Cuerpo del recurso consent	49
Ilustración 3.21 POST observation	50
Ilustración 3.22 Cuerpo del recurso Observation	50
Ilustración 3.23 Filtro por código	51
Ilustración 3.24 Filtro Codigo MongoDB	51
Ilustración 3.25 Filtro por capa de seguridad	52
Ilustración 3.26 Filtro capa de seguridad MongoDB	52
Ilustración 3.27 Filtro por fecha	53
Ilustración 3.28 Filtro fecha MongoDB	54
Ilustración 3.29 Filtro por referencia	55
Ilustración 3.30 Filtro Referencia MongoDB	55
Ilustración 0.1 Docker Desktop	62
Ilustración 0.2 Servidor FHIR local	63
Ilustración 0.3 Servidor FHIR-REST	63
Ilustración 0.4 CapabilityStatement Postman	64
Ilustración 0.5 Docker mongo	65
Ilustración 0.6 Comando mongo	66
Ilustración 0.7 Extensión mongoDB VScode	66

Ilustración 0.8 Conexión a mongoDB	67
Ilustración 0.9 Menu de MongoDB en VSCode	67

Notación

FHIR	Fast Health Interoperability Resources
HL7	Health Level Seven
GDPR	General Data Protection Regulation (Reglamento General de Protección de Datos)
JSON	JavaScript Object Notation
POM	Project Object Model
API	Application Programming Interface

1 INTRODUCCIÓN

En este capítulo se abordarán los temas como la motivación que ha llevado a realizar este trabajo, los objetivos que se desean conseguir, la metodología seguida en el transcurso de la operación y por último un esquema que mostrará el tiempo invertido en este proyecto.

1.1 Motivación y objetivos

Uno de los pilares fundamentales en el nuevo Reglamento Europeo de Protección de Datos (GDPR) es el consentimiento de los interesados como forma de legitimar el tratamiento de sus datos personales. Un campo en el que el consentimiento informado toma especial importancia es el sanitario puesto que trabajan no solo con datos personales habituales, sino también con lo que se conoce como información sensible. [1]

El GDPR afecta a todos los profesionales que operan en el sector sanitario y su correcta aplicación es incluso mayor que en otros ámbitos, ya que el tipo de datos que tratan son especialmente sensibles: los datos de salud.

El matiz novedoso es que también se incluye como datos de salud la información o datos relativos a la prestación de servicios de atención sanitaria que revelen información sobre el estado de salud de una persona.

Por lo trascendental que puede tener este tipo de datos para la privacidad del interesado, el GDPR otorga a este tipo de datos mayor protección, lo cual hace que se deban cumplir una serie de condiciones adicionales para su tratamiento [3]. De esta forma, existen algunos derechos listados a continuación:

- Derecho a estar informado
 - o Un sujeto tiene el derecho a conocer cómo serán sus datos recolectados, procesados, almacenados y con qué finalidad.
- Derecho al acceso
 - o Un sujeto tiene el derecho a conocer cómo los datos existentes han sido recolectados, procesados y almacenados, y con qué propósitos.
- Derecho a la corrección
 - o Un sujeto tiene el derecho a obtener la corrección de datos que estén incompletos o incorrectos.
- Derecho a la eliminación (Derecho a ser olvidado)
 - o Un sujeto tiene el derecho de tener sus datos personales eliminados de forma permanente.
- Derecho a restringir el procesamiento
 - o Un sujeto tiene el derecho de bloquear o suprimir sus datos personales en el momento en que están siendo procesados o en uso.
- Derecho a la portabilidad de datos
 - o Un sujeto tiene el derecho de mover, copiar, o transferir datos personales de un controlador a otro, de una forma segura, en un formato legible y comúnmente usado.
- Derecho a objetar el procesamiento
 - o Un sujeto tiene el derecho a objetar el hecho de ser sujeto para autoridades públicas o empresas

que procesen sus datos sin un consentimiento explícito.

- Derecho a no ser sujeto de toma de decisiones automatizadas
 - o Un sujeto tiene el derecho de solicitar intervención humana en la toma de decisiones, en lugar de ser objetivo de decisiones tomadas solamente por algoritmos.

Sin embargo, lo que se ha explicado previamente no se refleja con la realidad ya que la mayoría de las organizaciones gestionan los consentimientos de manera limitada sin observar la amplitud de derechos considerados en la GDPR.[4]

Tras lo explicado previamente, se ha decidido desarrollar un proyecto en el que se lleve a cabo la importación y filtrado de los consentimientos de los pacientes por parte de una organización sanitaria desde otras organizaciones de gestión de consentimientos. La organización sanitaria deberá así recoger solo aquellos consentimientos de sus pacientes y que sean relevantes con la información almacenada en sus sistemas de información sanitarios.

Para ello, hay que apoyarse en el estándar más conocido en el ámbito de la sanidad FHIR. En él, se define el formato y cómo se representan los datos con los que se trabajará.

Ciertamente, en este proyecto se busca realizar un análisis detallado sobre el funcionamiento y el formato de la representación de la información sobre dicho estándar. De hecho, otro objetivo es aplicar los derechos del Reglamento Europeo de Protección de Datos sobre FHIR, para poder generar un programa que permita importar los consentimientos de los pacientes de una organización sanitaria.

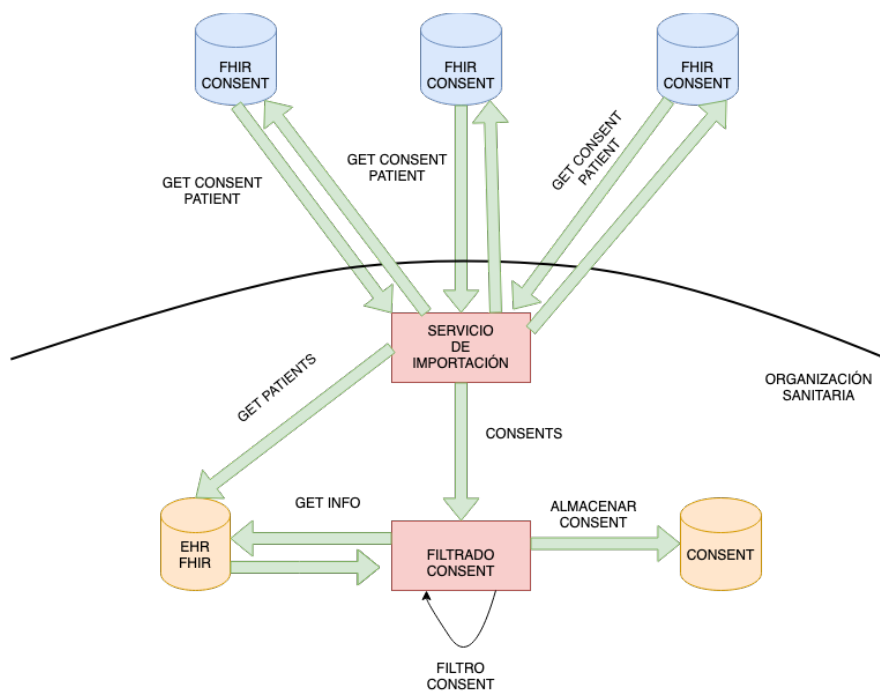


Ilustración 1.1 Estructura del proyecto

Como se puede observar en la Figura 1.1 la estructura del proyecto se basa en la división de la organización sanitaria y otras organizaciones que gestionan consentimientos de ciudadanos.

Las organizaciones sanitarias contienen una base de datos donde almacenan o reside la información de sus pacientes. Por ende, el servicio de importación se va a encargar de recoger todos los informes de dicha base de datos (contiene todos los pacientes) para posteriormente, solicitar a las entidades de gestión de consentimientos los asociados a cada uno de ellos. Cabe destacar que se ha optimizado de tal forma que, si el consentimiento ya se ha guardado previamente y no se ha modificado, no se realizará todo el proceso y se desechará inmediatamente. Dado que

se recogen datos que pertenecen a otras entidades, se necesita un filtrado para poder eliminar aquella información que no se corresponda con los datos que se tiene guardado de los pacientes. Una vez que han pasado la depuración, se almacenará en una base de datos específica de consentimientos.

1.2 Metodología Empleada

La metodología empleada para el desarrollo del trabajo está basada en el modelo incremental. Este método combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos y tiene como objetivo un crecimiento progresivo de la funcionalidad. Por tanto, el producto se irá desarrollando con cada una de las entregas previstas.[5]

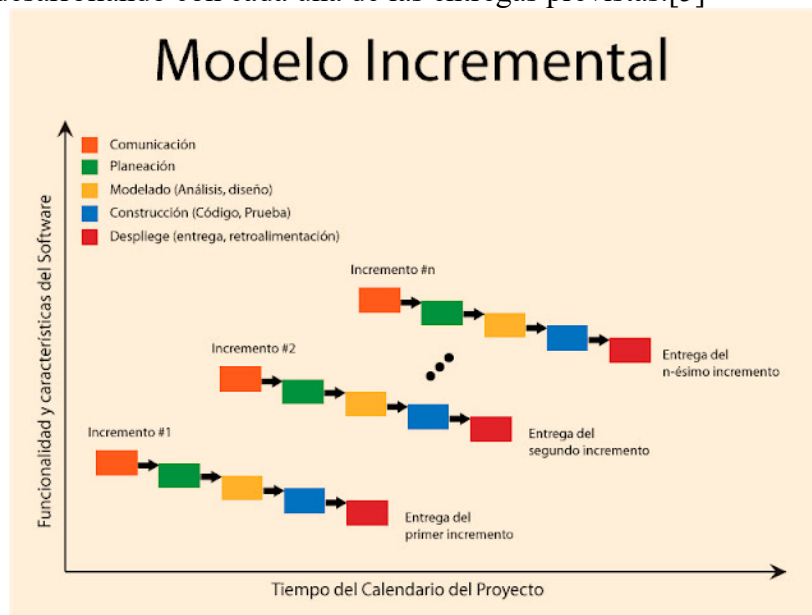


Ilustración 1.1.2 Metodología incremental

Como se puede observar en la Figura 1.2 el proyecto se desarrolla mediante iteraciones de incrementos los cuales se dividen en cinco fases. No todos los modelos incrementales siguen estas fases, son flexibles en su representación y depende de la empresa.

Cuando acaba el último periodo se produce una retroalimentación y se vuelve a empezar con el primer ciclo. Este método es muy usado en metodologías como SCRUM y AGILE.

El primer incremento representa el desarrollo del núcleo del proyecto. Cuando acaba, se produce una retroalimentación, que permite aprovechar los conocimientos y experiencia aprendidos de dicha fase para poder progresar de forma más sencilla en la siguiente iteración. El último incremento acaba cuando el cliente ya está satisfecho con su solución.

1.3 Plan de trabajo

En la Tabla adjunta se puede observar todas las tareas que se han realizado para llegar al final del proyecto.

Tabla 1 Plan de trabajo

Tareas	Horas estimadas	Horas reales
Estudio del estándar FHIR y su API	10	20
Estudio de los recursos FHIR que se van a implementar	7	15
Estudio de la base de datos HAPI-FHIR	10	20
Estudio de la librería HAPI-FHIR	30	40
Diseño e implementación de la estructura del proyecto	10	15
Implementación del código de los puntos anteriores	60	80
Pruebas y evaluación de los resultados obtenidos	40	50
Corrección de errores	80	90
Total de horas	247	330

2 MATERIALES

A continuación se va a describir la parte teórica del proyecto. En este capítulo se involucra la definición del estándar FHIR junto con su implementación y los recursos que han sido necesarios para el desarrollo del proyecto. Por último, se explica qué materiales se han usado a lo largo del proyecto, por qué y qué han aportado.

2.1 FHIR

La medicina es una ciencia de la salud dedicada a la prevención, diagnóstico, pronóstico y tratamiento de las enfermedades, lesiones y problemas de salud del ser humano que se lleva practicando desde la época de la prehistoria. Asimismo, el tiempo transcurre y se ha convertido en una de las ciencias más básicas e importantes. Con ello, otra ciencia muy importante que se ha desarrollado de forma paralela y ha llegado a cruzarse sus caminos es la tecnología.

HL7 es el ejemplo perfecto de la fusión de la tecnología y la sanidad. Este es un promotor que hace más sencillo la interoperabilidad entre todas las partes del mundo. Las siglas vienen dadas por Health Level Seven y es una organización de Desarrollo de Estándares para el ámbito de la salud. Fue fundada sin ánimo de lucro en 1987, opera a nivel internacional y su misión es proveer estándares globales para los dominios clínicos, asistencial, administrativo y logístico para conseguir una interoperabilidad entre los diferentes sistemas de información en el área de la salud.

HL7 fue quien dio forma y definió FHIR, el significado de las siglas viene dadas por Fast Healthcare Interoperability Resources y es una especificación de plataforma que describe formatos y elementos de datos, también conocidos como recursos. Estos (los recursos) son la unidad básica de interoperabilidad y la más pequeña que se puede intercambiar de manera significativa. Se pueden usar en la forma más simple o como un mensaje (los recursos se adaptan a los segmentos del mensaje), los documentos (como colecciones de recursos agrupados) o los servicios (usando uno o más recursos) también se pueden agrupar.

Todos los recursos, salvo excepciones, contienen un identificador a través de la cual puede ser registrado, localizado y recuperado. También tienen extensiones que permite a los implementadores agregar fácilmente nuevas propiedades.

Todo esto hace que los documentos físicos no sean archivados y se pierdan en el transcurso del tiempo. Ahora los datos están directamente disponibles como servicios. Por ejemplo, se puede acceder y editar datos médicos como pacientes, admisiones hospitalarias, informes de diagnóstico y medicamentos mediante URL de recursos únicos.

En la página oficial de FHIR [6] se divide en módulos la documentación para ayudar a responder ciertas preguntas a los usuarios. Cada uno de ellos representa un área funcional diferente de la especificación. Como se puede observar en la Figura 2.1, existen cinco niveles distintos:

- Primer nivel: Representa el marco básico desde donde se construye las especificaciones.
- Segundo nivel: Muestra información sobre la implementación y la vinculación a especificaciones externas.

- Tercer nivel: Relaciona los conceptos de la vida real con los de los sistemas de salud.
- Cuarto nivel: Objetos de información para el registro e intercambio de datos en el proceso sanitario.
- Quinto nivel: Recursos relacionados con la capacidad de razonar sobre el proceso de atención de la salud.

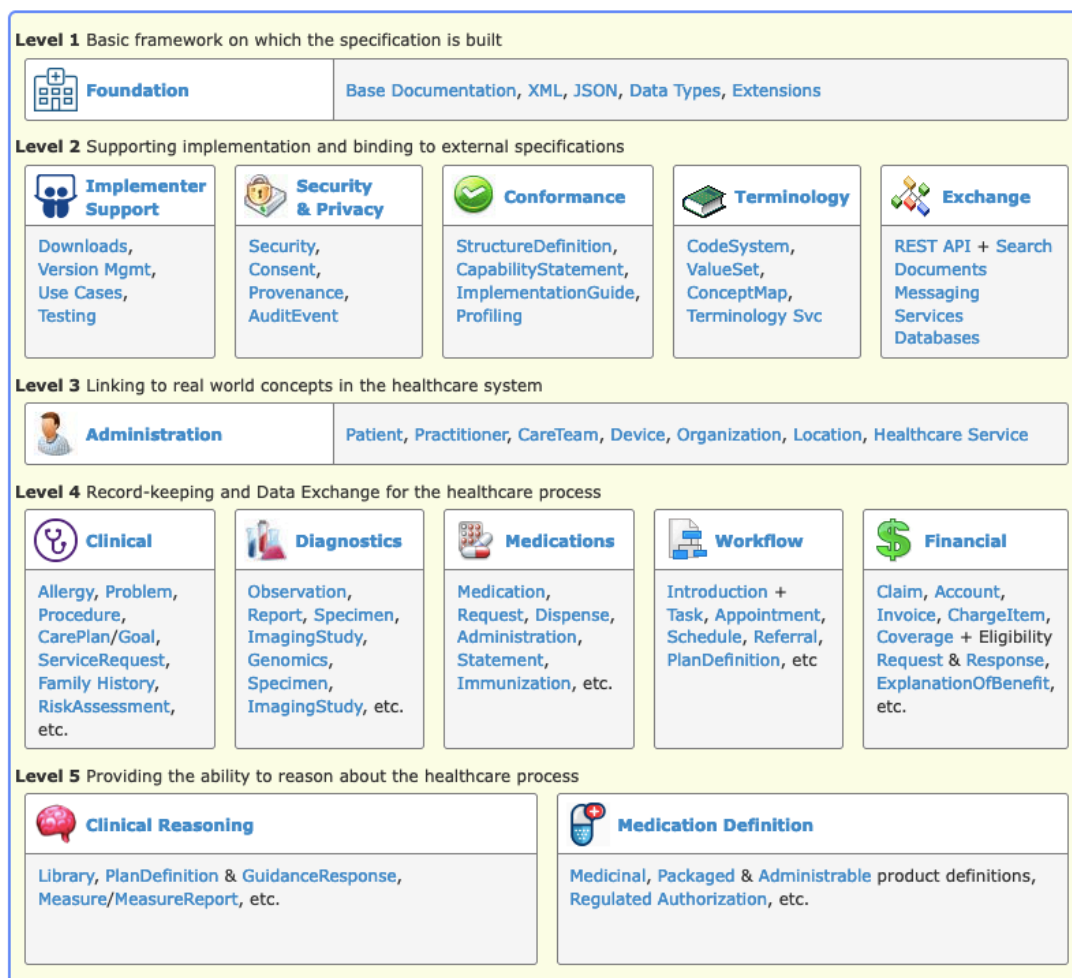


Ilustración 2.1 Módulos FHIR

2.1.1 Tipos de datos

FHIR tiene establecido y definido ciertos tipos de datos. Estos se dividen en cuatro categorías que pueden ser [9]:

- Tipos simples / primitivos:

Son elementos individuales con un valor primitivo, es decir, tienen un único valor y no tienen elementos adicionales como hijos, aunque, como todos los tipos, tiene extensiones.

2.24.0.1 Primitive Types

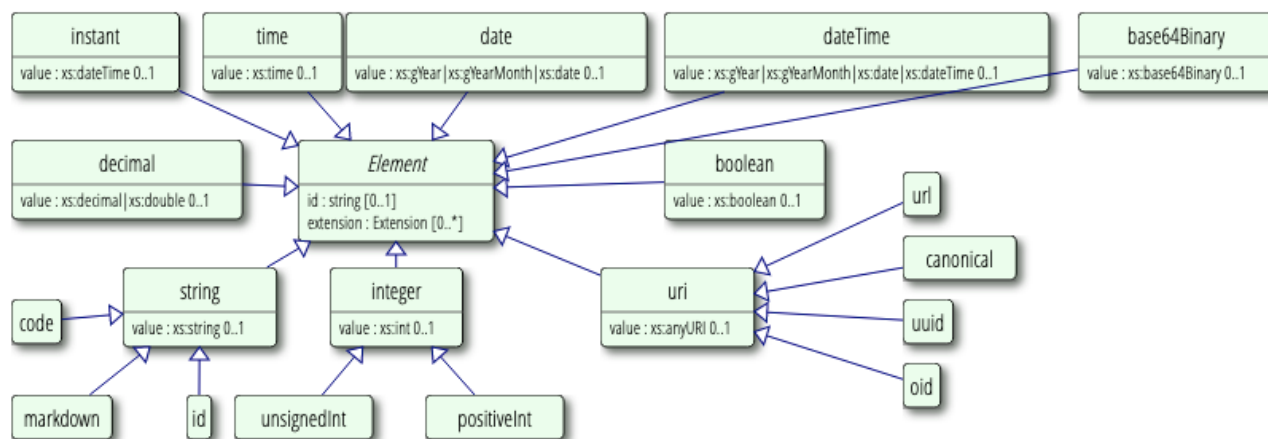


Ilustración 2.2 Datos primitivos

Se muestra a continuación un ejemplo de cómo se realiza la representación completa en el formato JSON. Se puede ver que el valor muestra la propiedad en sí misma.

```

"count" : 2
"_count" : {
  "id" : "a1",
  "extension" : [{
    "url" : "...",
    "valueXXX" : "...."
  }]
}
    
```

Ilustración 2.3 JSON primitivo

• Tipos complejos:

Se realiza un uso general y son grupos de elementos reutilizables.

UML Diagrams of the Data types

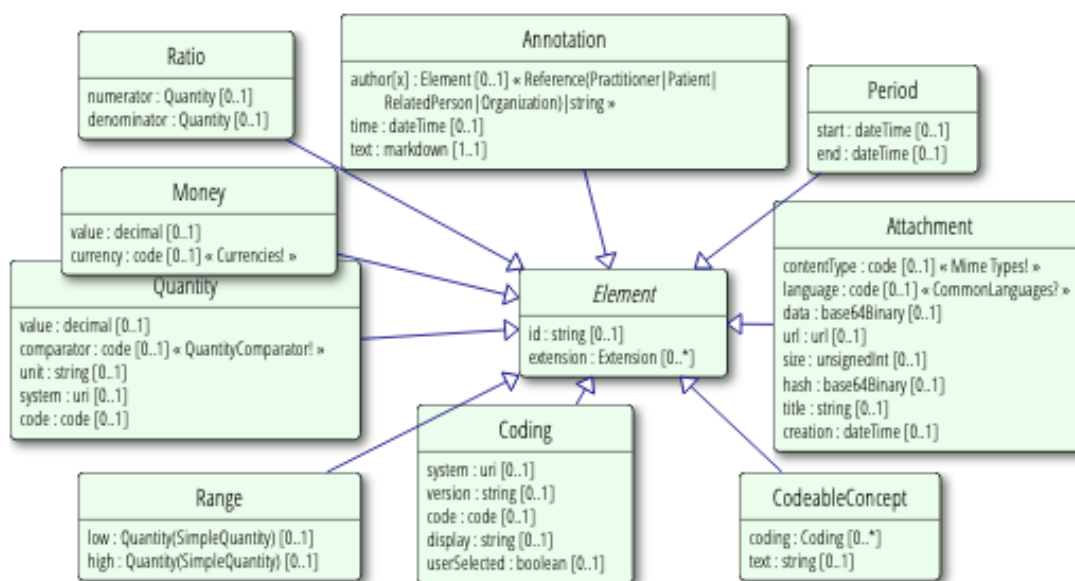


Ilustración 2.4 Datos complejos

En XML, estos tipos de datos se representan como componentes XML con elementos secundarios que reciben el nombre del elemento definido del tipo. Los nombres de estos se definen donde se utilizan los tipos. En JSON, los tipos de datos están representados por objetos con propiedades que tienen los mismos nombres que los elementos XML. Solo el primer ejemplo tiene una representación JSON explícita adicional, ya que la representación JSON es casi idéntica. Los tipos de datos complejos pueden ser “perfilados”. Con esto se quiere decir que el recurso StructureDefinition define una serie de reglas sobre qué elementos pueden tener valores y cuáles son sus respectivos valores asociados.

```
document : {
  contentType : { value : "application/pdf" },
  language : { value : "en" },
  data : { value : "/9j/4...KAP//Z"},
  title : { value : "Definition of Procedure" }
}
```

Ilustración 2.5 Datos Complejos JSON

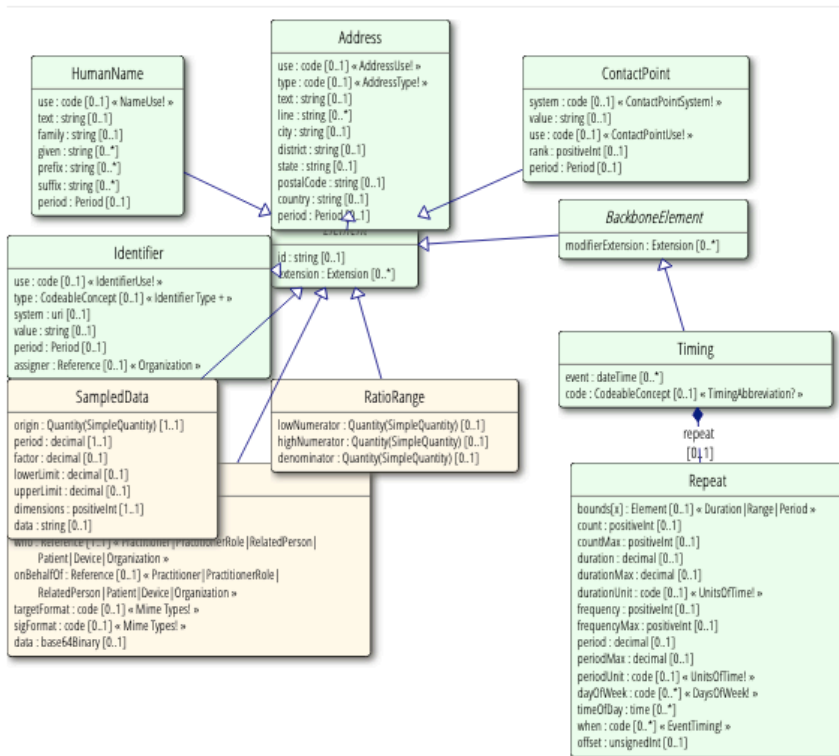


Ilustración 2.6 Datos complejos y extensión

- Tipos de metadatos:
Son un conjunto de tipos para uso con recursos de metadatos.

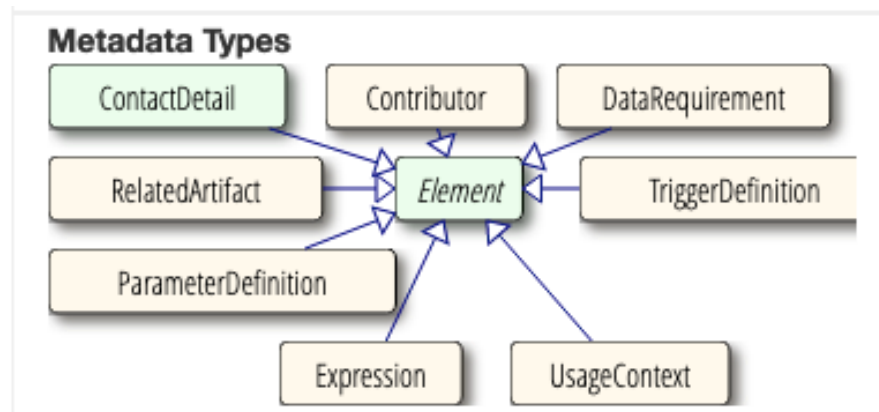


Ilustración 2.7 Tipos de metadatos

- Tipos de datos para fines especiales:

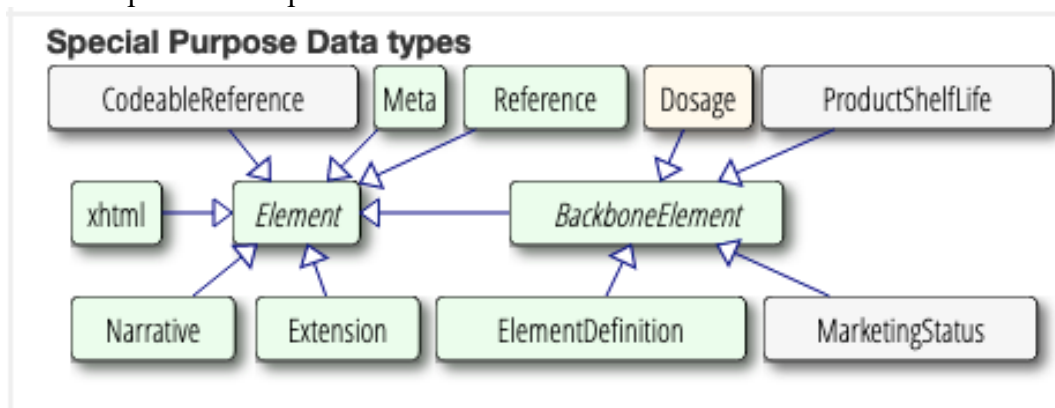


Ilustración 2.8 Tipos de datos de propósito especial

2.1.2 Recursos FHIR

En este tipo de estandarización se definen diferentes tipos de recursos. Estos se utilizan para intercambiar y/o almacenar datos para poder resolver una amplia gama de problemas relacionados con la atención médica, tanto clínicos como administrativos. Además, esta especificación define varios formatos para poder realizar un intercambio de recursos.[29]

Según se define en la página oficial de FHIR, un recurso es una entidad que (HL7 FHIR, 2022):

- tiene una identidad conocida (una URL) por la que se puede dirigir
- se identifica como uno de los tipos de recurso definidos en la presente especificación
- contiene un conjunto de elementos de datos estructurados según se describe en la definición del tipo de recurso
- tiene una versión identificada que cambia si el contenido del recurso cambia

En la Figura 2.9 se puede observar los campos que contienen todos los recursos (definidos en el recurso básico “Resource”). Está formado por:

- Un id que es de tipo “string”, cada recurso tiene un elemento “id” que es único en todo el espacio de los recursos del mismo tipo en el mismo servidor. Una vez se haya asignado, este no se verá modificado bajo ninguna circunstancia
- Un “meta” que es de tipo “Meta”, cada recurso contiene un elemento de este estilo que es un conjunto de metadatos que proporciona contexto técnico sobre el recurso. Todos los tipos de

metadatos son opcionales, aunque algunos o todos son demandados dependiendo del contexto del uso.

Name	Flags	Card.	Type	Description & Constraints
Meta	Σ [N]		Element	Metadata about a resource Elements defined in Ancestors: id , extension Version specific identifier
versionId	Σ	0..1	id	When the resource version last changed
lastUpdated	Σ	0..1	instant	Identifies where the resource comes from
source	Σ	0..1	uri	Profiles this resource claims to conform to
profile	Σ	0..*	canonical(StructureDefinition)	Security Labels applied to this resource SecurityLabels (Extensible)
security	Σ	0..*	Coding	Tags applied to this resource Common Tags (Example)
tag	Σ	0..*	Coding	

Ilustración 2.9 Meta resource

- Un “implicitRules” que es de tipo “uri”, cada recurso contiene un elemento de este estilo que es una referencia de un acuerdo complejo que describe como se está usando el recurso.
- Un “language” que es de tipo “code”, cada recurso debe contener un elemento de este tipo para especificar la base de lenguaje del contenido usando el código que se ha definido en el estándar FHIR [6] Si se ha especificado el tipo de lenguaje, se debe también especificar la forma de la narrativa.

2.26.3 Resource Content

Name	Flags	Card.	Type	Description & Constraints
Resource	«A» [N]		n/a	Base Resource
id	Σ	0..1	string	Logical id of this artifact
meta	Σ	0..1	Meta	Metadata about the resource
implicitRules	?! Σ	0..1	uri	A set of rules under which this content was created
language		0..1	code	Language of the resource content Common Languages (Preferred) but limited to AllLanguages

Ilustración 2.10 Contenido del recurso básico “Resource”

Existe una gran variedad de recursos que se definen y se van desarrollando, con diferentes grados de madurez en el que el mínimo es “0” y el máximo es “N” (“normative”). Cuando tiene un grado “N” significa que no va a sufrir más modificaciones.

Foundation	Conformance <ul style="list-style-type: none"> CapabilityStatement N StructureDefinition N ImplementationGuide 1 SearchParameter 3 MessageDefinition 1 OperationDefinition N CompartmentDefinition 1 StructureMap 2 GraphDefinition 1 ExampleScenario 0 	Terminology <ul style="list-style-type: none"> CodeSystem N ValueSet N ConceptMap 3 NamingSystem 2 TerminologyCapabilities 0 	Security <ul style="list-style-type: none"> Provenance 3 AuditEvent 3 Consent 2 	Documents <ul style="list-style-type: none"> Composition 2 DocumentManifest 2 DocumentReference 3 CatalogEntry 0 	Other <ul style="list-style-type: none"> Basic 1 Binary N Bundle N Linkage 0 MessageHeader 4 OperationOutcome N Parameters N Subscription 3 SubscriptionStatus 0 SubscriptionTopic 0 	
	Base	Individuals <ul style="list-style-type: none"> Patient N Practitioner 3 PractitionerRole 2 RelatedPerson 2 Person 2 Group 1 	Entities #1 <ul style="list-style-type: none"> Organization 3 OrganizationAffiliation 0 HealthcareService 2 Endpoint 2 Location 3 	Entities #2 <ul style="list-style-type: none"> Substance 2 BiologicallyDerivedProduct 0 Device 2 DeviceMetric 1 NutritionProduct 0 	Workflow <ul style="list-style-type: none"> Task 2 Appointment 3 AppointmentResponse 3 Schedule 3 Slot 3 VerificationResult 0 	Management <ul style="list-style-type: none"> Encounter 2 EpisodeOfCare 2 Flag 1 List 1 Library 3
		Clinical	Summary <ul style="list-style-type: none"> AllergyIntolerance 3 AdverseEvent 0 Condition (Problem) 3 Procedure 3 FamilyMemberHistory 2 ClinicalImpression 0 DetectedIssue 1 	Diagnostics <ul style="list-style-type: none"> Observation N Media 1 DiagnosticReport 3 Specimen 2 BodyStructure 1 ImagingStudy 3 QuestionnaireResponse 3 MolecularSequence 1 	Medications <ul style="list-style-type: none"> MedicationRequest 3 MedicationAdministration 2 MedicationDispense 2 MedicationStatement 3 Medication 3 MedicationKnowledge 0 Immunization 3 ImmunizationEvaluation 0 ImmunizationRecommendation 1 	Care Provision <ul style="list-style-type: none"> CarePlan 2 CareTeam 2 Goal 2 ServiceRequest 2 NutritionOrder 2 VisionPrescription 2 RiskAssessment 1 RequestGroup 2
	Financial		Support <ul style="list-style-type: none"> Coverage 2 CoverageEligibilityRequest 2 CoverageEligibilityResponse 2 EnrollmentRequest 0 EnrollmentResponse 0 	Billing <ul style="list-style-type: none"> Claim 2 ClaimResponse 2 Invoice 0 	Payment <ul style="list-style-type: none"> PaymentNotice 2 PaymentReconciliation 2 	General <ul style="list-style-type: none"> Account 2 ChargeItem 0 ChargeItemDefinition 0 Contract 1 ExplanationOfBenefit 2 InsurancePlan 0
		Specialized	Public Health & Research <ul style="list-style-type: none"> ResearchStudy 1 ResearchSubject 0 	Definitional Artifacts <ul style="list-style-type: none"> ActivityDefinition 3 DeviceDefinition 0 EventDefinition 0 ObservationDefinition 0 PlanDefinition 3 Questionnaire 3 SpecimenDefinition 0 	Evidence-Based Medicine <ul style="list-style-type: none"> Citation 0 Evidence 1 EvidenceReport 0 EvidenceVariable 1 	Quality Reporting & Testing <ul style="list-style-type: none"> Measure 3 MeasureReport 3 TestScript 2 TestReport 0

Ilustración 2.11 Lista de recursos

Se listan a continuación los recursos que se han usado en el proyecto:

2.1.2.1 Patient

Este recurso contiene datos sobre pacientes y animales involucrados en una variedad de actividades relacionadas con la salud, que incluyen: actividades curativas, cuidado de la salud mental, servicios sociales, cuidados durante el embarazo, cuidado y apoyo vital, servicio de comidas y seguimiento de datos personales de salud y ejercicio.[11]

Los recursos Patient cubren la información "quién" del paciente. Sus atributos se enfocan en la información demográfica necesaria para sustentar registros administrativos, financieros y logísticos. Las organizaciones que atienden a los pacientes suelen crear y mantener registros de pacientes. Por lo tanto, la información de un paciente o animal que recibe atención en varias organizaciones puede existir en varios recursos para pacientes.

La estructura que tiene el recurso Patient se puede observar en la Figura 2.12, en ella se pueden ver claramente los nombres de los campos que contiene con su respectiva cardinalidad, bandera, tipo y descripción.

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal receiving health care services Elements defined in Ancestors: <code>id</code> , <code>meta</code> , <code>implicitRules</code> , <code>language</code> , <code>text</code> , <code>contained</code> , <code>extension</code> , <code>modifierExtension</code>
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male female other unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	An address for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient MaritalStatus (Extensible)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend) for the patient + Rule: SHALL at least contain a contact's details or a reference to an organization
relationship		0..*	CodeableConcept	The kind of relationship Patient: Contact Relationship (Extensible)
name		0..1	HumanName	A name associated with the contact person
telecom		0..*	ContactPoint	A contact detail for the person
address		0..1	Address	Address for the contact person
gender		0..1	code	male female other unknown AdministrativeGender (Required)
organization	I	0..1	Reference(Organization)	Organization that is associated with the contact
period		0..1	Period	The period during which this contact person or organization is valid to be contacted relating to this patient
communication		0..*	BackboneElement	A language which may be used to communicate with the patient about his or her health
language		1..1	CodeableConcept	The language which can be used to communicate with the patient about his or her health Common Languages (Preferred) but limited to AllLanguages
preferred		0..1	boolean	Language preference indicator
generalPractitioner		0..*	Reference(Organization Practitioner PractitionerRole)	Patient's nominated primary care provider
managingOrganization	Σ	0..1	Reference(Organization)	Organization that is the custodian of the patient record
link	?! Σ	0..*	BackboneElement	Link to another patient resource that concerns the same actual person
other	Σ	1..1	Reference(Patient RelatedPerson)	The other patient or related person resource that the link refers to
type	Σ	1..1	code	replaced-by replaces refer seealso LinkTvoe (Required)

Ilustración 2.12 Recurso Patient

Sin embargo, esta no es la estructura para poder leer los datos. Si se desea manipular la información se trabajará con el formato de JSON o XML. En este caso, se hará con JSON y en la siguiente figura se puede observar un ejemplo que proporciona el propio estándar FHIR.

```

{
  "resourceType": "Patient",
  "id": "infant-mom",
  "text": {
    "status": "generated",
    "div": "<div xmlns=\\"http://www.w3.org/1999/xhtml\\"><p><b>Leia Solo (OFFICIAL)</b> female, D
oB: 1995-10-12</p></div>"
  },
  "name": [
    {
      "use": "official",
      "family": "Solo",
      "given": [
        "Leia"
      ]
    },
    {
      "use": "maiden",
      "family": "Organa",
      "given": [
        "Leia"
      ]
    }
  ],
  "gender": "female",
  "birthDate": "1995-10-12",
  "maritalStatus": {
    "coding": [
      {
        "system": "http://terminology.hl7.org/CodeSystem/v3-MaritalStatus",
        "code": "M",
        "display": "Married"
      }
    ]
  },
  "generalPractitioner": [
    {
      "reference": "Practitioner/21B",
      "display": "Too-Onebee"
    }
  ]
}

```

Ilustración 2.13 Instancia de recurso Patient en formato JSON

Este recurso es importante en este proyecto puesto que se almacenará toda la información en una base de datos local de la organización sanitaria. Para poder luego tramitar con todos los documentos y obtener los recursos de tipo Consent.

2.1.2.2 Consent

Este recurso es un registro de las elecciones de un consumidor de atención médica para permitir o denegar que un destinatario identificado o un rol de destinatario realice una o más acciones dentro de un contexto de política específico para un propósito y una duración exacta.

El propósito de este recurso es expresar el consentimiento sobre cuestiones de atención médica. Los recursos de consentimiento tienen cuatro usos, todos los cuales están restringidos a instrucciones para el propósito, el uso y el tratamiento por parte de un consumidor [productor] de atención médica o un representante personal de una entidad autorizada [productor].

Todos ellos son un acuerdo oral o escrito para acciones autorizadas o restringidas con Directiva de consentimiento de privacidad:

- Consentimiento para recopilar, acceder, usar o divulgar (compartir) información.
- Política de Consentimiento Médico: Consentimiento (o registro de negativa de consentimiento) para recibir ciertos tratamientos médicos.
- Política de consentimiento de investigación: se requiere consentimiento para participar en protocolos de investigación y compartir información.
- Testamentos en vida: estar de acuerdo con cualquier orden de tratamiento necesaria (como DNR).

Este recurso está diseñado para cubrir los cuatro casos de uso, pero actualmente solo se modela el caso de uso de protección de datos. El alcance de los recursos está sujeto a cambios a medida que se exploran, prueban o perfilan otras áreas potenciales. [10]

Structure

Name	Flags	Card.	Type	Description & Constraints
Consent	TU		DomainResource	A healthcare consumer's choices to permit or deny recipients or roles to perform actions for specific purposes and periods of time + Rule: <i>Either a Policy or PolicyRule</i> + Rule: <i>IF Scope=privacy, there must be a patient</i> + Rule: <i>IF Scope=research, there must be a patient</i> + Rule: <i>IF Scope=adr, there must be a patient</i> + Rule: <i>IF Scope=treatment, there must be a patient</i> Elements defined in Ancestors: id , meta , implicitRules , language , text , contained , extension , modifierExtension
Identifier		0..*	Identifier	Identifier for this record (external references)
status	?!	1..1	code	draft proposed active rejected inactive entered-in-error ConsentState (Required)
scope	?!	1..1	CodeableConcept	Which of the four areas this resource covers (extensible) Consent Scope Codes (Extensible)
category		1..*	CodeableConcept	Classification of the consent statement - for indexing/retrieval Consent Category Codes (Extensible)
patient		0..1	Reference(Patient)	Who the consent applies to
dateTime		0..1	dateTime	When this Consent was created or indexed
performer		0..*	Reference(Organization Patient Practitioner RelatedPerson PractitionerRole)	Who is agreeing to the policy and rules
organization		0..*	Reference(Organization)	Custodian of the consent
source[x]		0..1		Source from which this consent is taken
sourceAttachment			Attachment	
sourceReference			Reference(Consent DocumentReference Contract QuestionnaireResponse)	
policy		0..*	BackboneElement	Policies covered by this consent
authority	I	0..1	uri	Enforcement source for policy
uri	I	0..1	uri	Specific policy covered by this consent
policyRule	? I	0..1	CodeableConcept	Regulation that this consents to Consent PolicyRule Codes (Extensible)
verification		0..*	BackboneElement	Consent Verified by patient or family
verified		1..1	boolean	Has been verified
verifiedWith		0..1	Reference(Patient RelatedPerson)	Person who verified
verificationDate		0..1	dateTime	When consent verified

Ilustración 2.14 FHIR consentimiento parte 1

provision		0..1	BackboneElement	Constraints to the base Consent.policyRule
type		0..1	code	deny permit ConsentProvisionType (Required)
period		0..1	Period	Timeframe for this rule
actor		0..*	BackboneElement	Who what controlled by this rule (or group, by role)
role		1..1	CodeableConcept	How the actor is involved SecurityRoleType (Extensible)
reference		1..1	Reference(Device Group CareTeam Organization Patient Practitioner RelatedPerson PractitionerRole)	Resource for the actor (or group, by role)
action		0..*	CodeableConcept	Actions controlled by this rule Consent Action Codes (Example)
securityLabel		0..*	Coding	Security Labels that define affected resources SecurityLabels (Extensible)
purpose		0..*	Coding	Context of activities covered by this rule PurposeOfUse (Extensible)
class		0..*	Coding	e.g. Resource Type, Profile, CDA, etc. Consent Content Class (Extensible)
code		0..*	CodeableConcept	e.g. LOINC or SNOMED CT code, etc. in the content Consent Content Codes (Example)
dataPeriod		0..1	Period	Timeframe for data controlled by this rule
data		0..*	BackboneElement	Data controlled by this rule
meaning		1..1	code	instance related depends authoredby ConsentDataMeaning (Required)
reference		1..1	Reference(Any)	The actual data reference
provision		0..*	see provision	Nested Exception Rules

Documentation for this format

Ilustración 2.15 FHIR consentimiento parte 2


```
{
  "resourceType": "Consent",
  "id": "consent-example-basic",
  "text": {
    "status": "generated",
    "div": "<div xmlns=\\"http://www.w3.org/1999/xhtml\\">\n\t\t\t<p>\n\t\tAuthorize Normal access f
or Treatment\n\t\t\t</p>\n\t\t\t<p>\n\t\t\tPatient &quot;P. van de Heuvel&quot; wishes to have all
of the PHI collected at the Good Health Psychiatric Hospital \n\t\t\tavailable for normal treatment
use.\n\t\t\t</p>\n\t\t\t</div>"
  },
  "status": "active",
  "scope": {
    "coding": [
      {
        "system": "http://terminology.hl7.org/CodeSystem/consentscope",
        "code": "patient-privacy"
      }
    ]
  },
  "category": [
    {
      "coding": [
        {
          "system": "http://loinc.org",
          "code": "59284-0"
        }
      ]
    }
  ],
  "patient": {
    "reference": "Patient/f001",
    "display": "P. van de Heuvel"
  },
  "dateTime": "2016-05-11",
  "organization": [
    {
      "reference": "Organization/f001"
    }
  ],
  "sourceAttachment": {
    "title": "The terms of the consent in lawyer speak."
  },
  "policyRule": {
    "coding": [
      {
        "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
        "code": "OPTIN"
      }
    ]
  },
  "provision": {
    "period": {
      "start": "1964-01-01",
      "end": "2016-01-01"
    }
  }
}
```

Ilustración 2.16 Consentimiento JSON

Para este proyecto, este recurso es el más importante de todos porque indicará a las organizaciones sanitarias si los pacientes permiten o no el tratamiento de sus datos de salud para propósitos diferentes al de la asistencia sanitaria. La organización si quiere usar la información con un fin no puede tramitarlo como quiera, necesita el permiso del sujeto asociado a dicho consentimiento y por tanto será necesario acudir a este recurso para conocer si se permite o no.

2.1.2.3 Bundle

Bundle es un recurso FHIR que se encuentra dentro del grupo de “Otros” en la zona de “Fundación”. Su grado de madurez es el máximo, “Normative”. Este es un contenedor de colección de recursos. Dado que un paquete en sí mismo es un recurso, tiene los mismos metadatos comunes que todos los recursos, incluyendo afirmaciones de perfiles, etiquetas y etiquetas de seguridad. Aunque el propio recurso Bundle no contiene extensiones, a pesar de ello “link”, “entry”, “search/request/response” pueden sí contenerlo.

Un bundle suele contener una colección de recursos en una sola instancia con un contexto contenedor. En FHIR, esto se denomina "agrupación" de recursos. Estos paquetes son útiles por unas grandes variedades de razones. Por ejemplo, si se devuelve un conjunto de recursos que cumplen los criterios especificados como parte de una operación del servidor y también retorna un grupo de versiones de recursos como parte de una operación de historial en el servidor.

Además, agrupa un conjunto autónomo de recursos para servir como una colección permanente e intercambiable con integridad clínica. Por último, crea/actualiza/elimina registros de recursos en el servidor como una sola operación, incluida su realización como una sola transacción atómica y almacena colecciones de recursos. Se muestra en la siguiente Figura los campos que contiene dicho recurso y los tipos de datos.[13]

Name	Flags	Card.	Type
Bundle	N		Resource
identifier	Σ	0..1	Identifier
type	Σ	1..1	code
timestamp	Σ	0..1	instant
total	Σ I	0..1	unsignedInt
link	Σ	0..*	BackboneElement
relation	Σ	1..1	string
url	Σ	1..1	uri
entry	Σ I	0..*	BackboneElement
link	Σ	0..*	see link
fullUrl	Σ	0..1	uri
resource	Σ	0..1	Resource
search	Σ I	0..1	BackboneElement
mode	Σ	0..1	code
score	Σ	0..1	decimal
request	Σ I	0..1	BackboneElement
method	Σ	1..1	code
url	Σ	1..1	uri
ifNoneMatch	Σ	0..1	string
ifModifiedSince	Σ	0..1	instant
ifMatch	Σ	0..1	string
ifNoneExist	Σ	0..1	string
response	Σ I	0..1	BackboneElement
status	Σ	1..1	string
location	Σ	0..1	uri
etag	Σ	0..1	string
lastModified	Σ	0..1	instant
outcome	Σ	0..1	Resource
signature	Σ TU	0..1	Signature

Ilustración 2.17 Recurso Bundle

```

{
  "resourceType": "Bundle",
  // from Resource: id, meta, implicitRules, and language
  "identifier": { Identifier }, // Persistent identifier for the bundle
  "type": "<code>", // R! document | message | transaction | transaction-response | batch | batch-respo
nse | history | searchset | collection
  "timestamp": "<instant>", // When the bundle was assembled
  "total": "<unsignedInt>", // C? If search, the total number of matches
  "link": [{ // Links related to this Bundle
    "relation": "<string>", // R! See http://www.iana.org/assignments/link-relations/link-relations.xht
ml#link-relations-1
    "url": "<uri>" // R! Reference details for the link
  }],
  "entry": [{ // Entry in the bundle - will have a resource or information
    "link": [{ Content as for Bundle.link }], // Links related to this entry
    "fullurl": "<uri>", // URI for resource (Absolute URL server address or URI for UUID/OID)
    "resource": { Resource }, // A resource in the bundle
    "search": { // C? Search related information
      "mode": "<code>", // match | include | outcome - why this is in the result set
      "score": <decimal> // Search ranking (between 0 and 1)
    },
    "request": { // C? Additional execution information (transaction/batch/history)
      "method": "<code>", // R! GET | HEAD | POST | PUT | DELETE | PATCH
      "url": "<uri>", // R! URL for HTTP equivalent of this entry
      "ifNoneMatch": "<string>", // For managing cache currency
      "ifModifiedSince": "<instant>", // For managing cache currency
      "ifMatch": "<string>", // For managing update contention
      "ifNotExist": "<string>" // For conditional creates
    },
    "response": { // C? Results of execution (transaction/batch/history)
      "status": "<string>", // R! Status response code (text optional)
      "location": "<uri>", // The location (if the operation returns a location)
      "etag": "<string>", // The Etag for the resource (if relevant)
      "lastModified": "<instant>", // Server's date time modified
      "outcome": { Resource } // OperationOutcome with hints and warnings (for batch/transaction)
    }
  }],
  "signature": { Signature } // Digital Signature
}

```

Ilustración 2.18 Instancia de recurso Bundle en formato JSON

Este recurso se ha usado en el proyecto para almacenar, en formato de lista, información de todos los otros recursos.

2.1.2.4 Observation

Este recurso se encarga de las mediciones y las simples afirmaciones hechas sobre un paciente, dispositivo u otro sujeto. Además, es un recurso de eventos desde una perspectiva de flujo de trabajo FHIR. Observation es un elemento central de la asistencia sanitaria, usado para apoyar el diagnóstico, monitorizar el progreso y determinar líneas de base y patrones. También para captar características demográficas. La mayoría de estos recursos son simples aserciones de pares de nombre/valor con algunos metadatos, pero algunas observaciones agrupan otras observaciones lógicamente, o incluso son de varios componentes. Se observa que el recurso DiagnosticReport proporciona un contexto clínico o de flujo de trabajo para un conjunto de observaciones y DiagnosticReport hace referencia al recurso Observation para representar datos de laboratorio, imágenes y otros datos clínicos y diagnósticos para formar un informe completo.

Este tipo de recurso se usa para:

- Signos vitales como el peso corporal, la presión sanguínea y la temperatura
- Datos de laboratorio como sangre, glucosa y tasa de filtración glomerular estimada
- Datos de densidad de los huesos y medidas fetales
- Medidas de dispositivos como los datos EKG o los datos de pulso de oximetría

- Características personales como el color de los ojos...
- Características principales como el estado de embarazo, o una afirmación de una muerte
- Antecedentes sociales como el consumo del tabaco, el apoyo familiar o el estado cognitivo

Name	Flags	Card.	Type
Observation	N		DomainResource
identifier	Σ	0..*	Identifier
basedOn	Σ	0..*	Reference(CarePlan DeviceRequest ImmunizationRecommendation MedicationRequest NutritionOrder ServiceRequest)
partOf	Σ	0..*	Reference(MedicationAdministration MedicationDispense MedicationStatement Procedure Immunization ImagingStudy)
status	?! Σ	1..1	code
category		0..*	CodeableConcept
code	Σ	1..1	CodeableConcept
subject	Σ	0..1	Reference(Patient Group Device Location Organization Procedure Practitioner Medication Substance)
focus	Σ TU	0..*	Reference(Any)
encounter	Σ	0..1	Reference(Encounter)
effective[x]	Σ	0..1	
effectiveDateTime			dateTime
effectivePeriod			Period
effectiveTiming			Timing
effectiveInstant			instant
issued	Σ	0..1	instant
performer	Σ	0..*	Reference(Practitioner PractitionerRole Organization CareTeam Patient RelatedPerson)
value[x]	Σ I	0..1	
valueQuantity			Quantity
valueCodeableConcept			CodeableConcept
valueString			string
valueBoolean			boolean
valueInteger			integer
valueRange			Range
valueRatio			Ratio
valueSampledData			SampledData
valueTime			time
valueDateTime			dateTime
valuePeriod			Period
dataAbsentReason	I	0..1	CodeableConcept
interpretation		0..*	CodeableConcept
note		0..*	Annotation
bodySite		0..1	CodeableConcept
method		0..1	CodeableConcept
specimen		0..1	Reference(Specimen)
device		0..1	Reference(Device DeviceMetric)

Ilustración 2.19 Recurso Observation parte 1

referenceRange	I	0..*	BackboneElement
low	I	0..1	SimpleQuantity
high	I	0..1	SimpleQuantity
type		0..1	CodeableConcept
appliesTo		0..*	CodeableConcept
age		0..1	Range
text		0..1	string
hasMember	Σ	0..*	Reference(Observation QuestionnaireResponse MolecularSequence)
derivedFrom	Σ	0..*	Reference(DocumentReference ImagingStudy Media QuestionnaireResponse Observation MolecularSequence)
component	Σ	0..*	BackboneElement
code	Σ	1..1	CodeableConcept
value[x]	Σ	0..1	
valueQuantity			Quantity
valueCodeableConcept			CodeableConcept
valueString			string
valueBoolean			boolean
valueInteger			integer
valueRange			Range
valueRatio			Ratio
valueSampledData			SampledData
valueTime			time
valueDateTime			dateTime
valuePeriod			Period
dataAbsentReason	I	0..1	CodeableConcept
interpretation		0..*	CodeableConcept
referenceRange		0..*	see referenceRange

 [Documentation for this format](#)

Ilustración 2.20 Recurso Observation parte 2

```

{
  "resourceType": "Observation",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier": [{ Identifier }], // Business Identifier for observation
  "basedOn": [{ Reference(CarePlan|DeviceRequest|ImmunizationRecommendation|
    MedicationRequest|NutritionOrder|ServiceRequest) }], // Fulfills plan, proposal or order
  "partOf": [{ Reference(ImagingStudy|Immunization|MedicationAdministration|
    MedicationDispense|MedicationStatement|Procedure) }], // Part of referenced event
  "status": "<code>", // R! registered | preliminary | final | amended +
  "category": [{ CodeableConcept }], // Classification of type of observation
  "code": { CodeableConcept }, // R! Type of observation (code / type)
  "subject": { Reference(Device|Group|Location|Medication|Organization|
    Patient|Practitioner|Procedure|Substance) }, // Who and/or what the observation is about
  "focus": [{ Reference(Any) }], // What the observation is about, when it is not about the subject of r
  ecord
  "encounter": { Reference(Encounter) }, // Healthcare event during which this observation is made
  // effective[x]: Clinically relevant time/time-period for observation. One of these 4:
  "effectiveDateTime": "<dateTime>",
  "effectivePeriod": { Period },
  "effectiveTiming": { Timing },
  "effectiveInstant": "<instant>",
  "issued": "<instant>", // Date/Time this version was made available
  "performer": [{ Reference(CareTeam|Organization|Patient|Practitioner|
    PractitionerRole|RelatedPerson) }], // Who is responsible for the observation
  // value[x]: Actual result. One of these 11:
  "valueQuantity": { Quantity },
  "valueCodeableConcept": { CodeableConcept },
  "valueString": "<string>",
  "valueBoolean": <boolean>,
  "valueInteger": <integer>,
  "valueRange": { Range },
  "valueRatio": { Ratio },
  "valueSampledData": { SampledData },
  "valueTime": "<time>",
  "valueDateTime": "<dateTime>",
  "valuePeriod": { Period },
  "dataAbsentReason": { CodeableConcept }, // C? Why the result is missing
  "interpretation": [{ CodeableConcept }], // High, low, normal, etc.
  "note": [{ Annotation }], // Comments about the observation
  "bodySite": { CodeableConcept }, // Observed body part
  "method": { CodeableConcept }, // How it was done
  "specimen": { Reference(Specimen) }, // Specimen used for this observation
  "device": { Reference(Device|DeviceMetric) }, // (Measurement) Device
  "referenceRange": [{ // Provides guide for interpretation
    "low": { Quantity(SimpleQuantity) }, // C? Low Range, if relevant
    "high": { Quantity(SimpleQuantity) }, // C? High Range, if relevant
    "type": { CodeableConcept }, // Reference range qualifier
    "appliesTo": [{ CodeableConcept }], // Reference range population
    "age": { Range }, // Applicable age range, if relevant
    "text": "<string>" // Text based reference range in an observation
  }],
  "hasMember": [{ Reference(MolecularSequence|Observation|
    QuestionnaireResponse) }], // Related resource that belongs to the Observation group
  "derivedFrom": [{ Reference(DocumentReference|ImagingStudy|Media|
    MolecularSequence|Observation|QuestionnaireResponse) }], // Related measurements the observation is ma
  de from
  "component": [{ // Component results
    "code": { CodeableConcept }, // R! Type of component observation (code / type)
    // value[x]: Actual component result. One of these 11:
    "valueQuantity": { Quantity },
    "valueCodeableConcept": { CodeableConcept },
    "valueString": "<string>",
    "valueBoolean": <boolean>,
    "valueInteger": <integer>,
    "valueRange": { Range },
    "valueRatio": { Ratio },
    "valueSampledData": { SampledData },
    "valueTime": "<time>",
    "valueDateTime": "<dateTime>",
    "valuePeriod": { Period },
    "dataAbsentReason": { CodeableConcept }, // C? Why the component result is missing
    "interpretation": [{ CodeableConcept }], // High, low, normal, etc.
    "referenceRange": [{ Content as for Observation.referenceRange } // Provides guide for interpretati
    on of component result
  ]
  }
}

```

Ilustración 2.21 Instancia de recurso Observation en formato JSON

Este recurso se ha usado en el proyecto para marcar la información de los pacientes. Ya que cuando un paciente visita una organización sanitaria se genera información sobre la acción que ha realizado. Todos estos datos se guardan en este tipo de recurso para luego compararlo con los consentimientos y saber si la entidad puede realizar estudios.[12]

2.2 Herramientas Software

Para poder realizar este proyecto se ha hecho uso de programas que han ayudado al desarrollo del trabajo y son los siguientes.

2.2.1 Visual Studio Code

Microsoft ha desarrollado un editor de código fuente llamado Visual Studio Code (VS Code) basado en Electron. Es compatible con la mayoría de los lenguajes de programación, contiene multitud de extensiones para lenguajes de programación como PHP, Python, Java ... y muchos más. Además, incluye soporte para JavaScript, Node.js

Está disponible en los tres sistemas operativos más usados, Linux/GNU, Windows y macOS y es software libre.

La integración con Git es extraordinaria, contando con soporte para la depuración de código, multitud de extensiones que se pueden descargar en el propio programa, posibilidad de personalizar el diseño de las letras.[25]

Gracias a este programa, se ha desarrollado la mayoría del proyecto, además se han incluido varias extensiones como pueden ser:

- Docker, es una aplicación de contenedores que ha permitido que se pueda realizar el despliegue de dos aplicaciones que son mongoDB (base de datos no relacional) y HapiFHIR (base de datos FHIR implementada en una página web)
- MongoDB: Es una base de datos no relacional que desde VS Code ha permitido observar de forma gráfica toda la información que existe y se ha ido guardando. También se puede ver desde el terminal.
- Maven: Es una herramienta de software para la gestión y construcción de proyectos en Java. Se ha usado en este proyecto para importar librerías como “hapi fhir”, “mongo-db”, “slf4j-api” y otras.

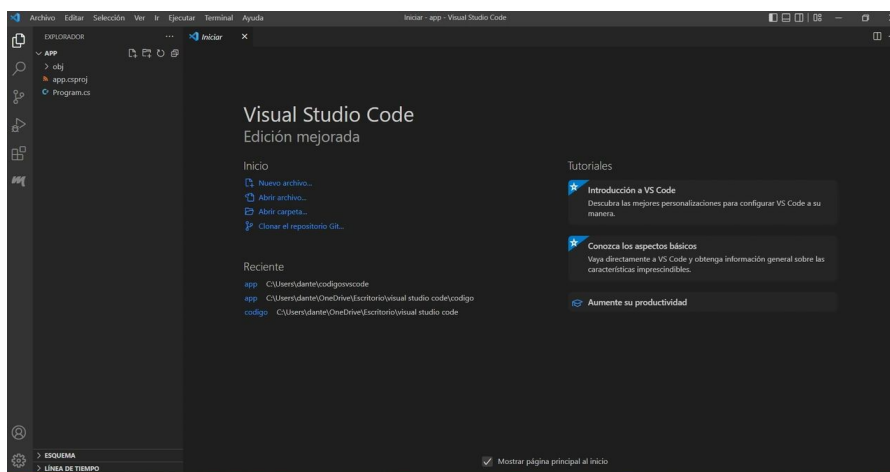


Ilustración 2.22 IDE VS CODE



Ilustración 2.23 Visual Studio code Logo

2.2.2 Postman

Postman es una herramienta que sirve para realizar peticiones RESTful y está disponible como una aplicación de escritorio para cualquier sistema operativo, GNU/Linux, Windows, macOS. [23] Tiene la estructura de cliente-servidor, siendo el cliente HTTP que realiza las peticiones al servidor. Para poder generar una solicitud hay que especificar la URL destino y seleccionar el tipo de mensaje que se desea. Los tipos de peticiones RESTful que existen en POSTMAN son:

- GET, se solicitan datos de lectura, se puede obtener un mensaje 200 (todo ha ido bien y se ha encontrado el recurso) o bien 404 (algo ha ido mal).
- POST, es solicitado para realizar la creación de un nuevo registro.
- PUT, su uso está basado en la actualización de recursos existentes.
- DELETE, este tipo de mensaje se usa para mandar una petición de borrado.
- PATCH, este método es similar a PUT porque permite actualizar registros existentes. Sin embargo, solo se desea actualizar un fragmento del registro.
- HEAD, este método se usa para obtener información sobre un determinado recurso.
- OPTIONS, este tipo de petición RESTful se usa para determinar las opciones de comunicación para el recurso de destino.

Esta aplicación se ha usado en el proyecto para realizar peticiones GET depurando y comprobando el perfecto funcionamiento. Además, se han insertado los recursos FHIR a la base de datos. Se detallará en los próximos capítulos cómo se ha realizado.[24]

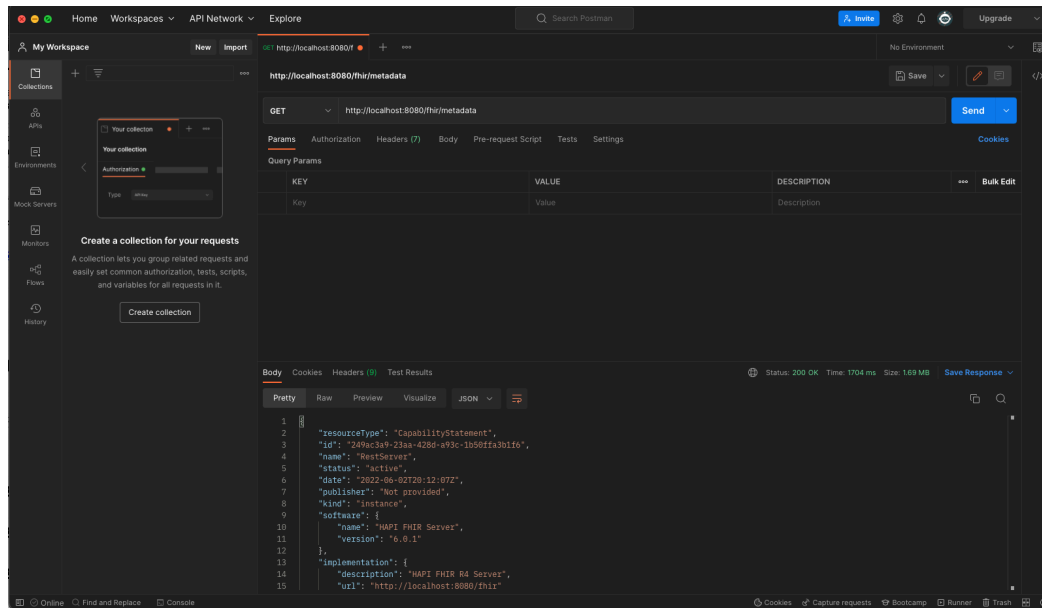


Ilustración 2.24 Postman



Ilustración 2.25 POSTMAN Logo

2.2.3 Docker

Docker es una plataforma de software que permite la creación e implementación de aplicaciones. Se caracteriza porque los procesos de instalación son bastante rápidos, esto es debido a que Docker empaqueta el software en unidades estandarizadas llamadas contenedores que incluyen las librerías, herramientas del sistema y código necesarios para que se lleve a cabo la ejecución. Proporciona comandos sencillos para crear, iniciar y parar contenedores.

Al igual que una máquina virtual, virtualiza el hardware necesario y el sistema operativo de un servidor haciendo mucho más sencillo la administración, pudiendo ajustar la escala de aplicaciones rápidamente en cualquier entorno.

En el próximo capítulo se mostrará cómo se ha realizado la instalación y puesta en marcha de los contenedores y la posterior administración. [21]

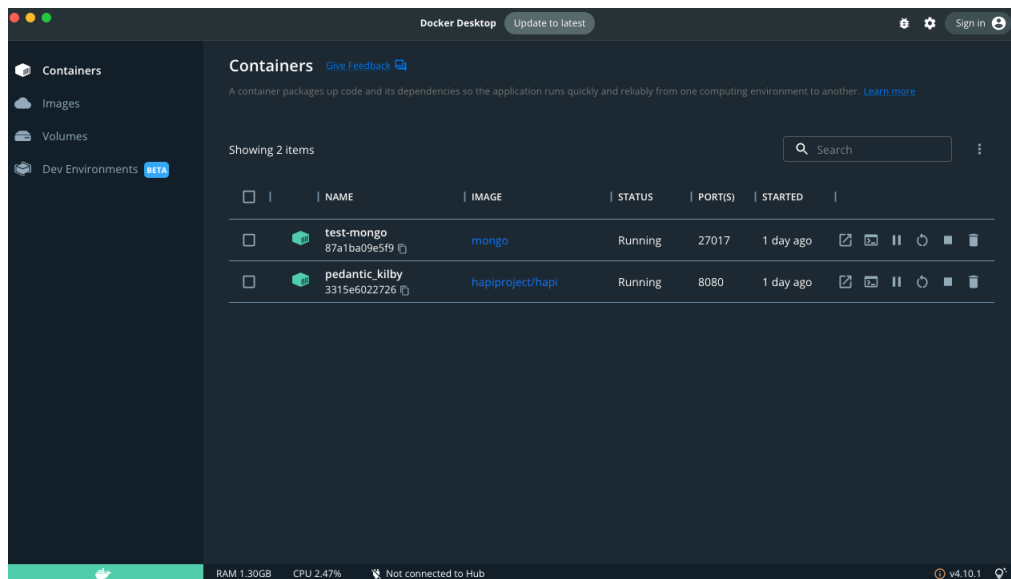


Ilustración 2.26 Docker Desktop

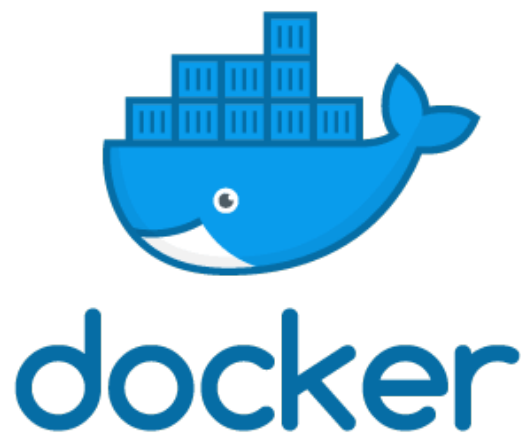


Ilustración 2.27 Docker Desktop logo

2.2.4 MongoDB

MongoDB es una base de datos no relacional, también denominada NoSQL, que guarda la información en estructuras de datos BSON (similar a JSON) y no en formatos de tablas como hacen los sistemas de bases de datos SQL. Es un software abierto, por tanto, está disponible para todos los sistemas operativos como Linux, Windows, macOS...

Por tanto, como se puede ver la gran diferencia que tiene con respecto a las bases de datos relacionales es que no es necesario seguir un esquema. La información se guarda de forma directa sin tener que crear una estructura o claves primarias, además hace la lectura más fácil. Cabe destacar que los documentos de una misma colección pueden tener esquemas diferentes.[22]



Ilustración 2.28 Logo MongoDB

2.2.5 JSON

JavaScript Object Notation (JSON) es un formato para guardar e intercambiar información que cualquier persona pueda leer y comprender fácilmente. En la Figura 2.28 se puede observar cómo se representa la información.

```
{
  "empleados": {
    "nombre": "Tom",
    "apellido": "Jackson"
  }
}
```

Ilustración 2.29 JSON

2.2.6 HAPI FHIR (Librería externa)

FHIR, como se ha comentado en los capítulos anteriores, es un estándar que sigue una normativa. Para poder realizar una implementación, en este caso con el lenguaje de programación Java, existe una librería externa que se llama HAPI FHIR.

Existe documentación que muestra cómo desplegar una base de datos hapi-fhir de forma local. Así pues, se puede introducir desde la aplicación Postman los recursos que se desean o desde un navegador web. Esta es una de las formas que se tiene para almacenar información. Otra forma es trabajar con java y crear objetos de los distintos tipos de recursos que existen en el estándar FHIR y subirlos a la base de datos.[26]

Como se puede observar en la Figura 2.31 se ha desplegado un servidor de pruebas hapi fhir en local, en él se manipulará los recursos FHIR. También existe la posibilidad de trabajar con la base de datos pública que proporciona FHIR en <https://hapi.fhir.org/>, como se puede ver en la Figura 2.30

The screenshot shows the HAPI FHIR public database interface. On the left, there are 'Options' for Encoding (JSON), Pretty (On), and Summary (none). Below that is a 'Resources' section with a list of resource types and their counts: Observation (2185535), Specimen (1875254), Composition (937608), Patient (262574), Encounter (124668), QuestionnaireResponse (84228), MedicationStatement (77022), Condition (74066), Claim (72639), and Location (58753). The main content area features the HAPI FHIR logo and a warning: 'This is not a production server! Do not store any information here that contains personal health information or any other confidential information. This server will be regularly purged and reloaded with fixed test data.' Below the warning is a table with server details: Server (HAPI FHIR Test/Demo Server R4 Endpoint), Software (HAPI FHIR Server - 6.0.0-PRE8-SNAPSHOT/3f58e277e6/2022-03-23), and FHIR Base (http://hapi.fhir.org/baseR4). At the bottom, there are 'Server Actions' including 'Conformance' and 'History' (with a 'Since' field and a 'Limit #' option).

Ilustración 2.30 Base de datos HAPI FHIR pública

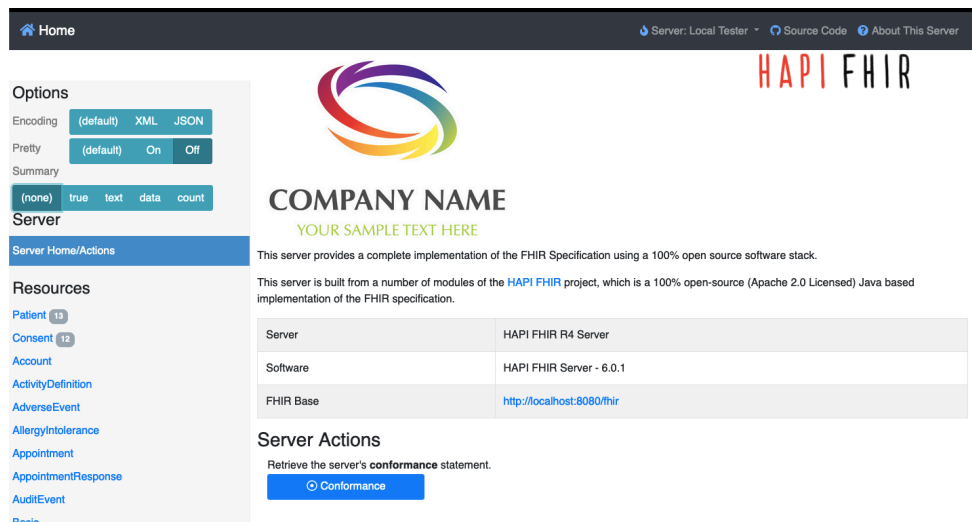


Ilustración 2.31 Base de datos HAPI FHIR privada o local



Ilustración 2.32 Logo HAPI FHIR

2.2.7 Diagrams.net

Diagrams.net es un software gratuito y de código abierto que está disponible en formato de página web y como aplicación de escritorio. Está disponible en todos los sistemas operativos como GNU/Linux, Windows, macOS. Su interfaz proporciona funciones como crear diagramas de flujo, diagramas UML, organigramas, diagramas de red y muchas más. Al igual que VS Code, su framework está basado en Electron.

Otro dato que destacar es que está disponible como plugin en páginas web como Notion, Atlassian Confluence, JIRA y más. La utilidad que ha tenido en este trabajo ha sido para la representación de la estructura del proyecto.

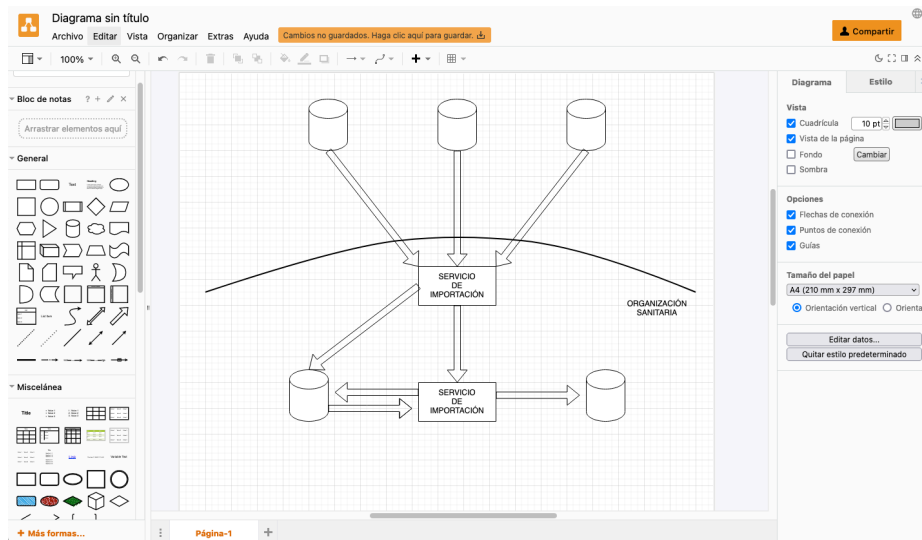


Ilustración 2.33 Representación de diagrams.net

2.2.8 Maven

Maven es una herramienta de software para la gestión de proyectos y de construcción en Java. Su modelo de configuración de construcción está basado en XML, esto hace que la lectura sea más sencilla. Además, puede administrar los informes y la documentación de un proyecto a partir de una pieza central de información, esta pieza se le denomina Project Object Model (POM) simplificando el proceso de construcción.[19][20]

En este archivo se almacenará todas las librerías o bibliotecas necesarias para poder trabajar en el proyecto añadiéndolas como “dependencias”. Se muestra a continuación su estructura.

Como se ha comentado con anterioridad, el archivo POM contiene información sobre el proyecto y detalles de la configuración, varios parámetros que son obligatorios especificar son:

- Proyecto: Es el elemento raíz del archivo pom.xml.
- modelVersion: Versión del modelo “pom” con la que se está trabajando actualmente
- groupId: Indica el id del grupo del proyecto. Es único.
- artifactId: Se usa para dar nombre al proyecto que está creando.
- Versión: Este elemento representa el número de la versión

Todos estos parámetros se pueden observar aquí:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>tfg</groupId>
  <artifactId>fhir</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
```

```
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>
```

A continuación, se muestran varias dependencias que se han añadido en el fichero.[18]

```
<dependencies>
<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>2.0.0-alpha1</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver -->
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongo-java-driver</artifactId>
  <version>3.12.11</version>
</dependency>
```



Ilustración 2.34 Maven

2.2.9 Swagger UI

Escribir APIs ha sido tradicionalmente una tarea muy compleja debido a la gran variedad de tecnologías y lenguajes de programación. Primer punto importante: Actualmente se utiliza el paradigma de programación REST para desarrollar APIs. Sitios como Google, Amazon y Twitter usan API RESTful. Las interfaces se describieron previamente en el lenguaje de descripción de servicios web WSDL. Desde un punto de vista técnico, WSDL 2.0 también se puede usar para describir las API REST, pero esto es muy difícil para los desarrolladores. Se suponía que el lenguaje de descripción de aplicaciones web (WADL) resolvería este problema, pero nunca se estandarizó debido a su estructura XML.

Por este motivo, Swagger se ha convertido rápidamente en la tecnología más popular para la documentación de API. Más bien, es la tecnología más popular para las API REST más utilizadas. Swagger fue desarrollado por Reverb, pero ahora se destaca como código abierto neutral bajo el paraguas de la Fundación Linux llamada Open API Initiative. Con el tiempo, se

cambió el nombre de Swagger a Especificación OpenAPI, pero todavía se lo conoce informalmente como Swagger.

3 RESULTADOS

Se va a detallar el desarrollo y los resultados del trabajo en el presente capítulo. Con ello primero se comentará como se ha realizado la puesta en marcha del servidor FHIR, junto con las otras posibles soluciones. Además, se mostrará un trabajo previo que ha sido necesario realizar para poder poner en marcha el proyecto. De forma análoga, se detallarán todas las partes del trabajo con su funcionalidad correspondiente. Para finalizar, se mostrará el modelado de la base de datos de los consentimientos junto con las pruebas que demuestran el buen funcionamiento de las partes que se han implementado.

3.1 Análisis del proyecto

En este apartado se va a explicar detalladamente la funcionalidad que tiene el trabajo y los distintos problemas que se han encontrado en el transcurso del proyecto.

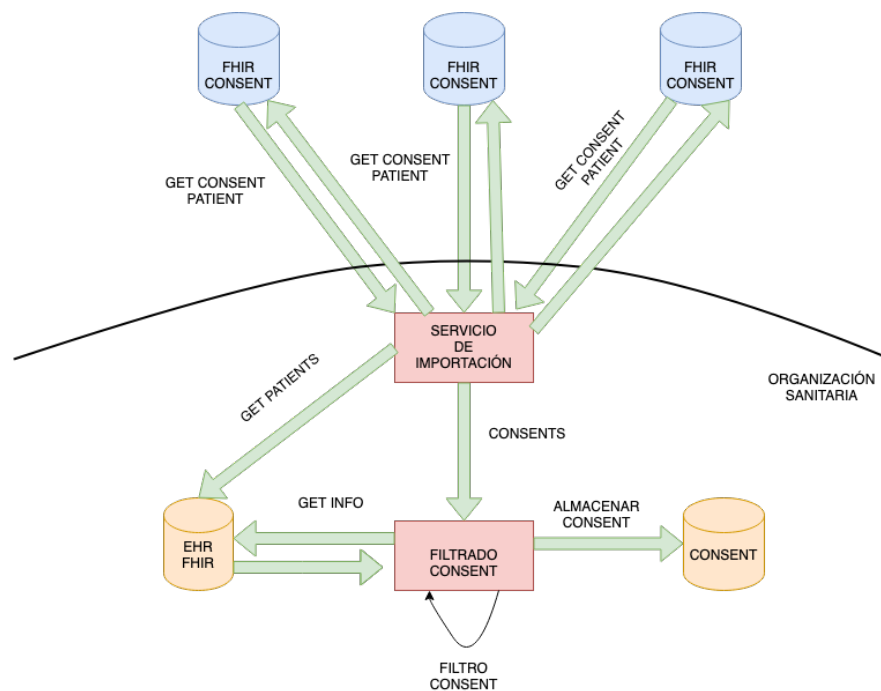


Ilustración 1.1 Estructura del proyecto

3.1.1 Base de datos EHR FHIR

Para poder realizar este proyecto es necesario montar una base de datos local HAPI-FHIR donde se tiene almacenados todos los pacientes de la organización sanitaria. Para ello, se ha decidido realizar un trabajo previo donde se almacene la información necesaria para este proyecto. Se muestra cierta información sobre ellos:

- Patient

El recurso Patient está en el último grado de madurez, esto quiere decir que no sufrirá modificaciones. La estructura de datos se almacenará en formato JSON. Es importante subir primero el recurso Patient porque más en adelante se verá que el recurso Consent y Observation dependen de este.

Entonces, generará una errata porque se subirá un consentimiento asociado con un paciente y este no existe en la base de datos. Por tanto, a pesar de que sintácticamente es correcto generará un error de que no existe dicha persona en la base de datos.

El cómo subir la información a la base de datos se explicará detenidamente más adelante. Cabe destacar que se puede crear el recurso con los campos que el usuario desee o bien puede acceder a la base de datos pública copiando uno de los que ha proporcionado la comunidad.

- Consent

Este recurso es el más importante del proyecto. Como se ha comentado en el punto anterior, para poder subir cualquier información de tipo Consent tiene que existir un paciente que este asociado a dicho consentimiento. Los campos a destacar de dicho recurso son:

- Provision
 - code: este campo indica el código que registrará el consentimiento. Puede ser de distintos tipos como LOINC o SNOMED CT.
 - data.reference: Indica la referencia actual de datos del consentimiento por el que se rige.
 - securityLabel: este campo indica las etiquetas de seguridad que definen los recursos afectados.
 - dataPeriod: este campo indica el periodo/intervalo de validación de un consentimiento.
 - type: este campo indica si el consentimiento restringe los datos o los permite.

- Observation

Este recurso también depende de Patient. Por tanto, es necesario primero que exista en el base de datos el paciente asociado al recurso Observation. Los campos a destacar son:

- Subject: campo usado para guardar la referencia de quién o de qué trata la observación.
- EffectiveDateTime: marca el tiempo relevante de la observación
- Code: se guarda la representación de un valor que es una referencia a las terminologías u ontologías como LOINC, SNOMED CT...

3.1.1.1 Postman

Se realiza una petición REST llamada "POST" en la que se insertará información, según como indica Postman en formato "raw". Después, se debe introducir la URL necesaria, en este caso <http://localhost:8080/fhir/Consent>. Es importante indicar que el tipo de dato que se va a introducir es de tipo JSON.

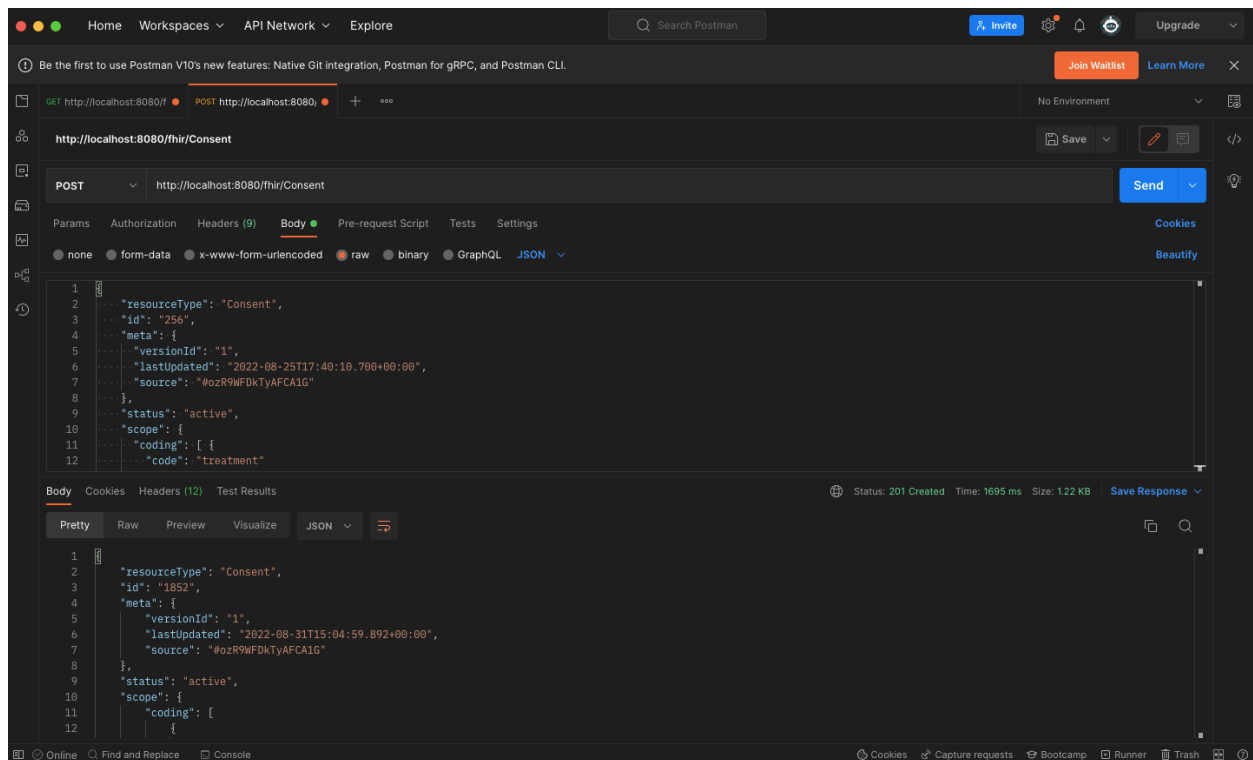


Ilustración 3.1 POST de un consentimiento

Como se puede observar en la Figura anterior, el estado de la operación es 201 por tanto se ha creado el objeto correctamente. Si se desea comprobar, se puede acceder al navegador (se detallará más en adelante) o bien se puede realizar una petición GET en POSTMAN a la URL <http://localhost:8080/consent/{id}> siendo el id 1852.

Recordar que el id se ha establecido de forma aleatoria, como ya existe un consentimiento con el id que se ha establecido en el cuerpo, se ha puesto uno al azar.

En la siguiente Figura se puede observar como el estado de la operación es un 200 OK recibándose el consentimiento que se había escrito anteriormente en el cuerpo de la petición. Por tanto, todo ha transcurrido de forma correcta.

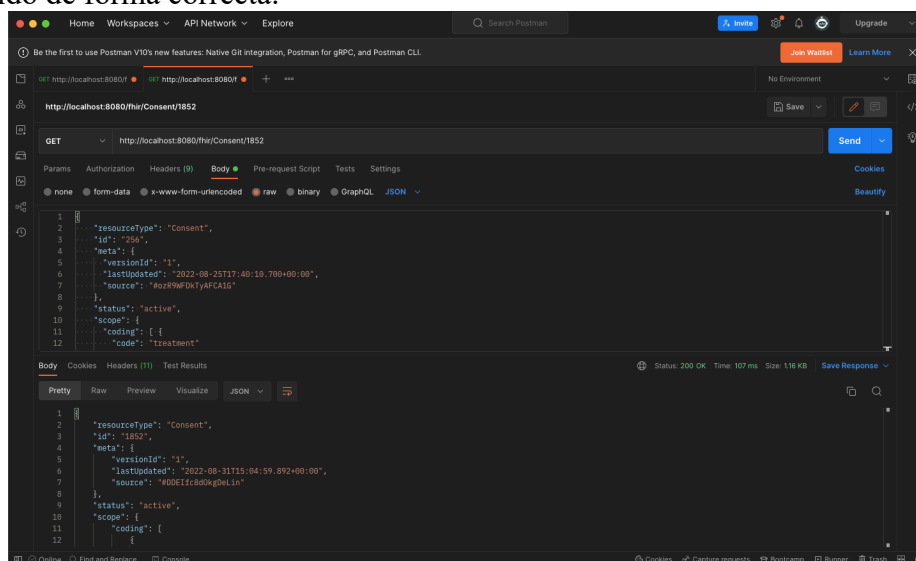


Ilustración 3.2 GET al consentimiento

3.1.1.2 Swagger UI

Por ello, cuando se accede a la URL <http://localhost:8080/fhir>, se selecciona la opción de Consent y tras ello se obtendrá una gran serie de peticiones REST que se puede aplicar de forma directa.

Para poder subir un nuevo consentimiento en la base de datos se tiene que escoger la opción de POST /Consent. En ella, se mostrará una interfaz sencilla donde se deberá insertar el cuerpo de la información en formato JSON.

Tras realizarlo, se subirá a la base de datos y en la siguiente Figura se puede observar cómo se puede recoger la información de un consentimiento a través de su id. Se puede ver que es correcto porque la respuesta que se recibe es un 200 OK

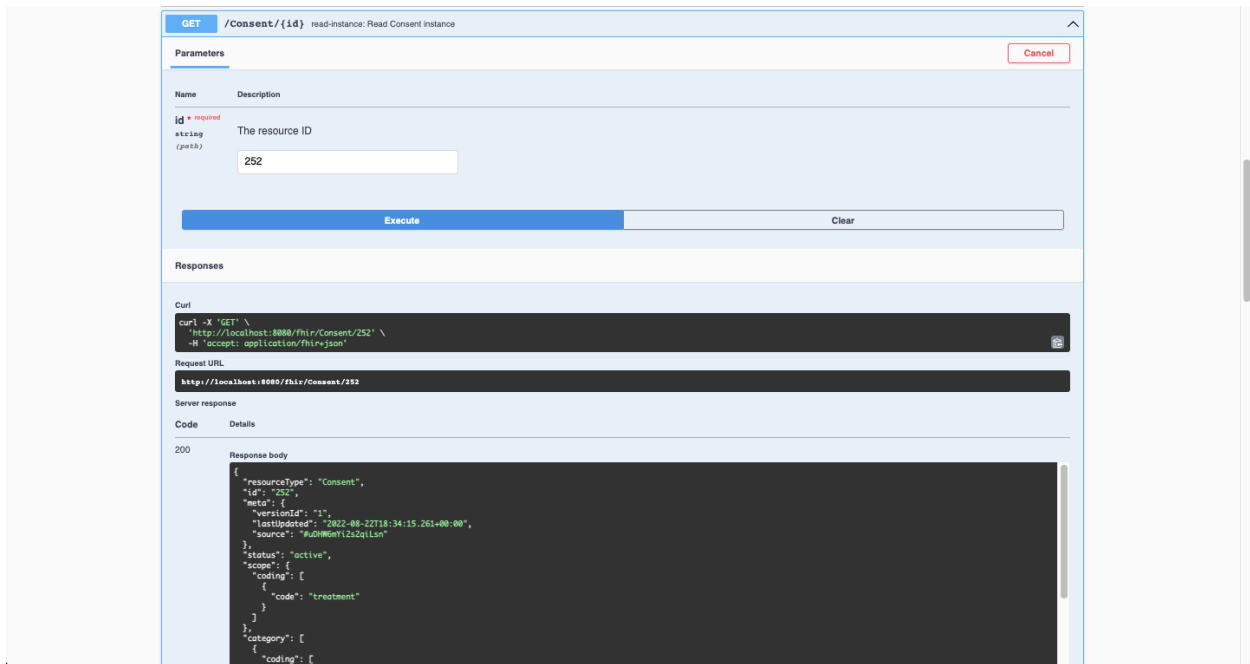


Ilustración 3.3 Swagger UI

También se proporciona la URL de la petición que si se copia y se pega en el navegador se obtiene el objeto en formato JSON.

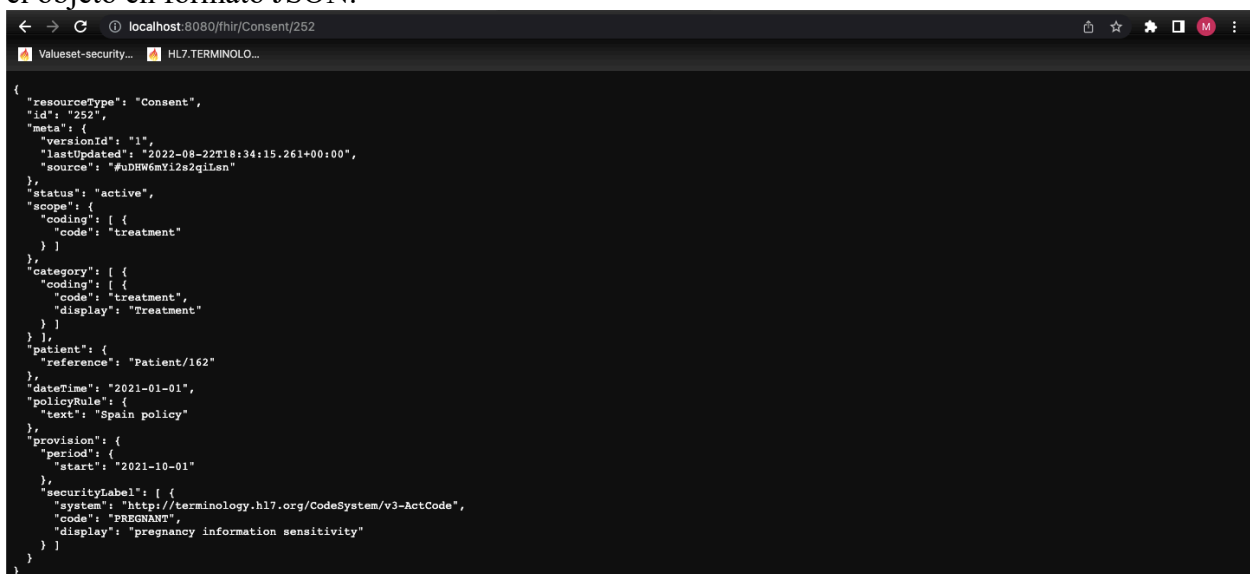


Ilustración 3.4 Consent URL JSON

Para este proyecto se ha decidido usar Swagger UI para subir la información de los recursos FHIR.

3.1.2 Servicio de Importación

El servicio de importación se va a encargar primero de recoger todos los pacientes de la base de datos en un recurso llamado 'Bundle' (agrupa o contiene hasta 20 pacientes distintos en una página). Una vez recogido los recursos de todos los pacientes, la organización sanitaria pedirá a todas las entidades existentes de gestión de consentimientos información sobre los consentimientos asociados a cada uno de los pacientes. Cabe destacar que puede existir más de un consentimiento asociado a un paciente.

Una vez que se ha realizado la importación de los consentimientos de los pacientes, se pasan al servicio de filtrado para que produzca la depuración de los consentimientos.

3.1.3 Servicio de Filtrado

Cuando se ha realizado la importación, todos los consentimientos de cada paciente deben de ser filtrados. Esto se debe a que se quieren almacenar solo aquellos consentimientos relacionados con la información que la organización sanitaria tenga de cada paciente. Por tanto, si se obtuviese un consentimiento que no estuviese relacionado con los datos que tiene la organización sobre dicho sujeto, entonces se desechará y no se guardará en la base de datos.

Este pequeño resumen se muestra en la Ilustración 1.1, se puede ver cómo el servicio de filtrado, que recibe tan solo consentimientos, solicita información a la base de datos para comparar los documentos que se tienen del paciente y lo que este permite tramitar. Es importante destacar que las organizaciones sanitarias quieren los consentimientos de todos los pacientes (que se encuentren en su base de datos) para poder controlar cómo se usa la información dentro de la organización. Es decir, si el paciente permite o no el uso de los documentos que se le imputan de forma directa.

En este proyecto se han puesto cuatro ejemplos de filtros, se nombran a continuación, pero más tarde se explicarán detalladamente.

- Filtrado por nivel de seguridad de la información.
- Filtrado por código LOINC o SNOMED de enfermedad.
- Filtrado por fecha de creación de datos.
- Filtrado por un tipo de dato concreto que en el consentimiento se señale directamente.

3.1.4 Base de datos Consent

En esta base de datos se guardará toda la información de los consentimientos que se han imputado a los pacientes con sus correspondientes referencias. Cabe destacar que en el proyecto no se ha realizado tan solo una depuración de la información y la inserción a la base de datos a posteriori.

Primero, se comprueba que el consentimiento no existe en la base de datos, en el caso de que exista este se desecha. Segundo, se comprueba si se halla un consentimiento con el mismo id, de esta forma si fuese correcto se extrae un campo que detalla la última actualización que ha tenido el consentimiento, en el caso de que sea mayor se actualiza y si es menor se desecha. Por último, si no existiese se inserta.

3.2 Servidor FHIR

El estándar FHIR no proporciona un servidor, por ende, la comunidad se ha encargado de desarrollar un servidor público que todo el mundo pueda acceder (<https://hapi.fhir.org/>). En él, todas las personas pueden subir los recursos (siempre y cuando sintácticamente sean correctos) de cualquier tipo del estándar. Sin embargo, la desventaja que ofrece para el uso de este proyecto es que es muy recurrente que dejase de funcionar durante un cierto tiempo, lo que impedía y retrasaba el desarrollo del proyecto. Así pues, se decidió implementar un servidor local FHIR.

3.2.1 Instalación

Para poder realizar la instalación del servidor se ha usado una página web donde se sube contenido de directorios, archivos, proyectos comúnmente denominados repositorios. El propio estándar FHIR ha subido un repositorio [15]

En él se encuentra un archivo que se llama “README.md” en el que se explica detalladamente como se debe realizar la instalación de forma local. A pesar de ello en el Anexo A se explica cómo se realiza el proceso, en el apartado A.1.

3.2.2 Puesta en marcha

La puesta en marcha del servicio no ha requerido ninguna modificación. Por tanto, los pasos se marcan en el Anexo, en el apartado A.2.

3.3 Implementación FHIR

En esta sección se verá cómo se ha conseguido implementar el proyecto. Se comenzará viendo la estructura del servicio junto con el único paquete que se ha desarrollado para después entrar en el código usado para lograr el correcto funcionamiento.

3.3.1 Estructura

El servicio está dividido en varios paquetes. Por tanto, cada paquete genera una funcionalidad distinta y funcionan en conjunto a través de llamadas para darle forma al proyecto.



Ilustración 3.5 Estructura del proyecto

Como se puede observar en la Figura, existe un paquete que se llama tfg que sirve para arrancar el servicio. A continuación, se explicará detalladamente la funcionalidad del resto de paquetes.

3.3.1.1 Paquete Database

En el paquete Database se tiene las clases correspondientes a la funcionalidad de la base de datos de MongoDB donde se almacenará los consentimientos.

A través de la librería de mongo se conectará a la base de datos que se ha desplegado con Docker Desktop para poder realizar la inserción de los archivos o la actualización de estos.

Esta clase será llamada por la aplicación general una vez se cumplan con las restricciones previas para poder subir el consentimiento a la base de datos.

Cabe destacar que en el caso de que el consentimiento ya esté almacenado en la base de datos este no se guardará. Además, si ha sido actualizado se observará la fecha de modificación para comprobar si se debe de subir a la base de datos o no.

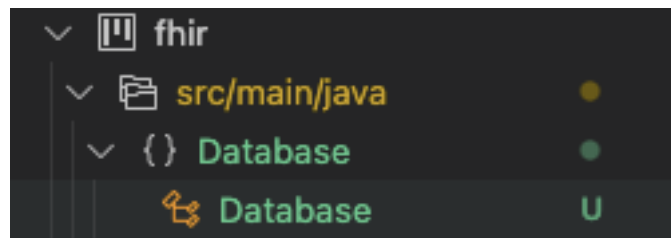


Ilustración 3.6 Estructura Database

3.3.1.2 Paquete Filtrado

En el paquete de Filtrado se tiene la clase correspondiente con el filtrado de los consentimientos que se han recibido del paquete de importación.

Esta clase contiene métodos que depuran la información, próximamente se explicará detalladamente sus funcionalidades.

Sin embargo, para poder realizarlo es necesario llamar al paquete de importación para obtener información sobre los pacientes asociados a los consentimientos. De esta forma, se llamará a los distintos métodos de filtrado según las restricciones que exige el consentimiento con las observaciones que se tiene.

Este paquete es llamado por la app general tantas veces como consentimientos se hayan recogido de los pacientes. Devolverá una lista con todos los consentimientos que hayan coincidido con la información que se almacena.

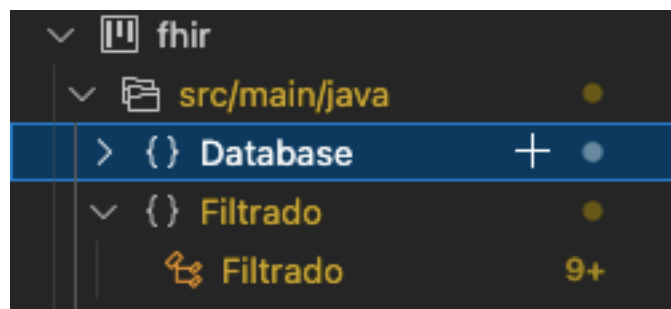


Ilustración 3.7 Estructura Filtrado

3.3.1.3 Paquete Importación

El paquete de importación se encarga de llamar a la base de datos local FHIR, con el código correspondiente de la librería de HAPI-FHIR para poder obtener un Bundle de los datos necesarios (no tiene por qué ser un recurso de tipo Bundle).

Esto es posible porque en el constructor de la clase, la aplicación general le pasa por parámetro el contexto de la base de datos además del cliente. Por tanto, se puede acceder a la librería de HAPI FHIR para recoger información, modificarla etc.

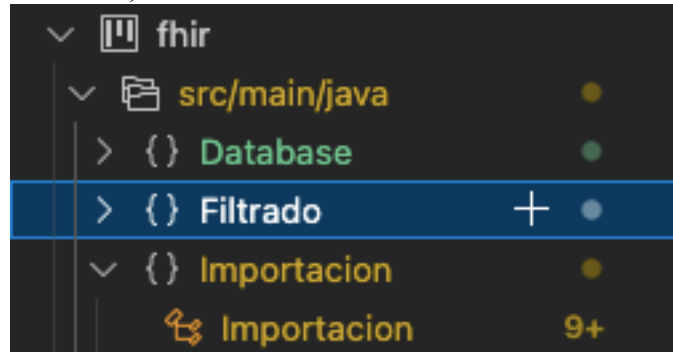


Ilustración 3.8 Estructura Importacion

3.3.2 Implementación código FHIR

3.3.2.1 App.java

A continuación, se muestran los diagramas de clase relacionados.

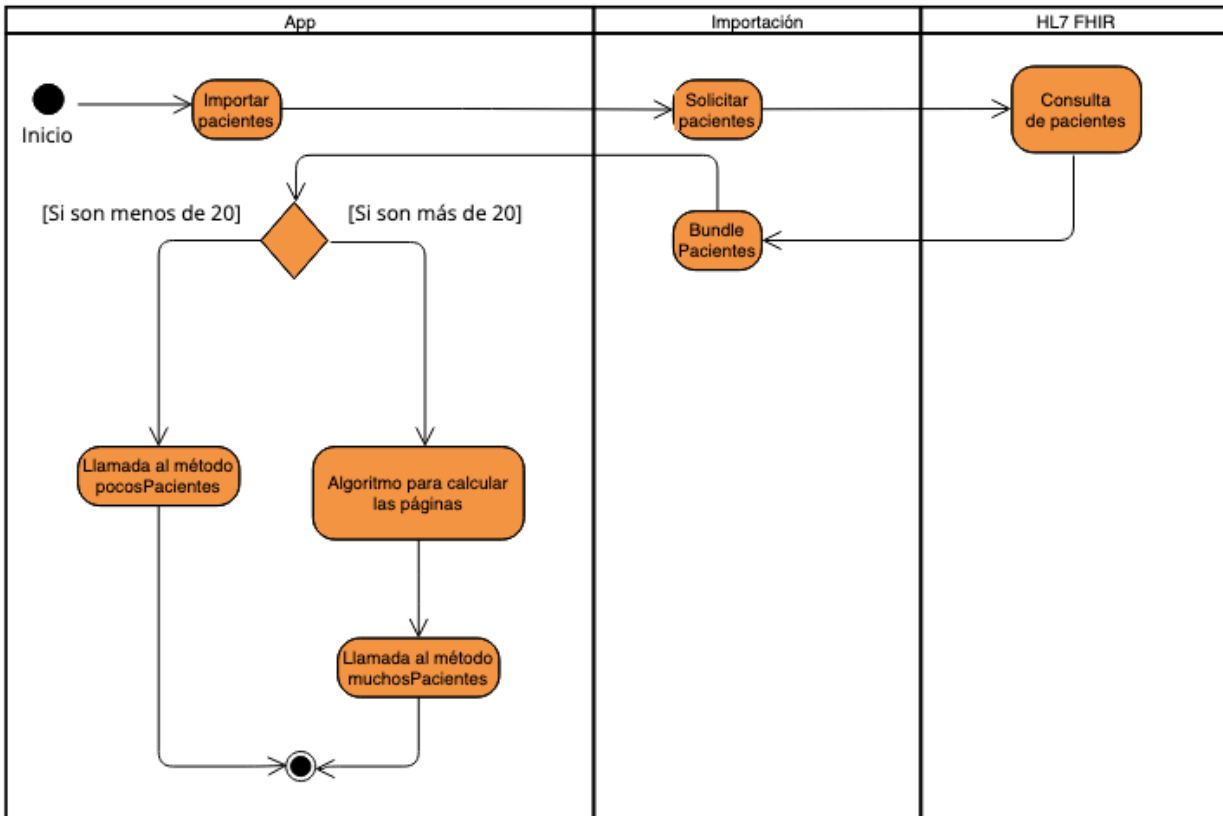


Ilustración 3.9 Diagrama de Actividad de App v1

Como se puede observar en la Ilustración 3.9, se busca primeramente recoger los pacientes que tiene la organización sanitaria. Esto se recoge en formato de recurso Bundle de la base de datos local HAPI-FHIR que se ha desplegado.

Una vez obtenido en un Bundle todos los pacientes de la organización sanitaria se comprueban si son más de veinte pacientes o menos. Según una opción u otra pues se invocará a un método u a otro.

Cabe destacar que se comprueba si son más de veinte pacinetes porque la base de datos HAPI-FHIR almacena la información de los pacientes en un recurso Bundle, en el momento que supera dicha cantidad se instancia un nuevo Bundle con veinte pacientes más.

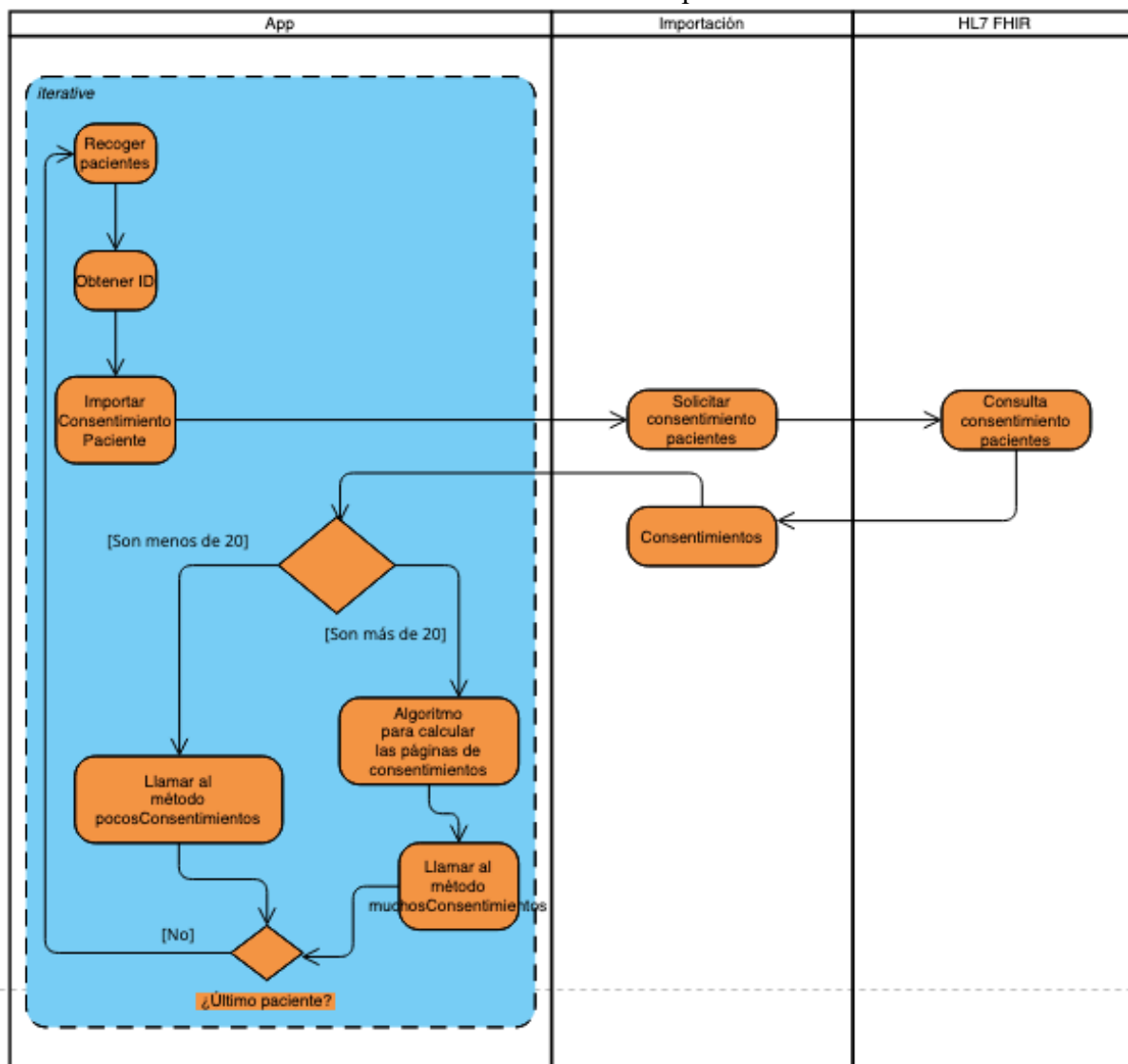


Ilustración 3.10 Diagrama de Actividad del método pocosPacientes

Como se puede observar en la Ilustración 3.10, se ha invocado un método de la clase App llamado pocosPacientes. Este se encarga de forma iterativa de recoger los recursos Patient obteniéndolo del Bundle. Después, se obtiene su id correspondiente para recoger los consentimientos asociados (también están almacenados en un recurso Bundle) de dichos pacientes.

Los consentimientos se recogerán en el formato de recurso Bundle, como se ha explicado anteriormente solo se pueden almacenar hasta veinte consentimientos en dicho recurso. Por ende, en el caso de que existan más se llamará a un método calculando previamente el número de páginas.

Cabe destacar que la base de datos HAPI-FHIR almacena la información en un recurso Bundle y cuando satura un recurso (que se está almacenando dentro del Bundle) en veinte, crea una nueva página con otra URL distinta.

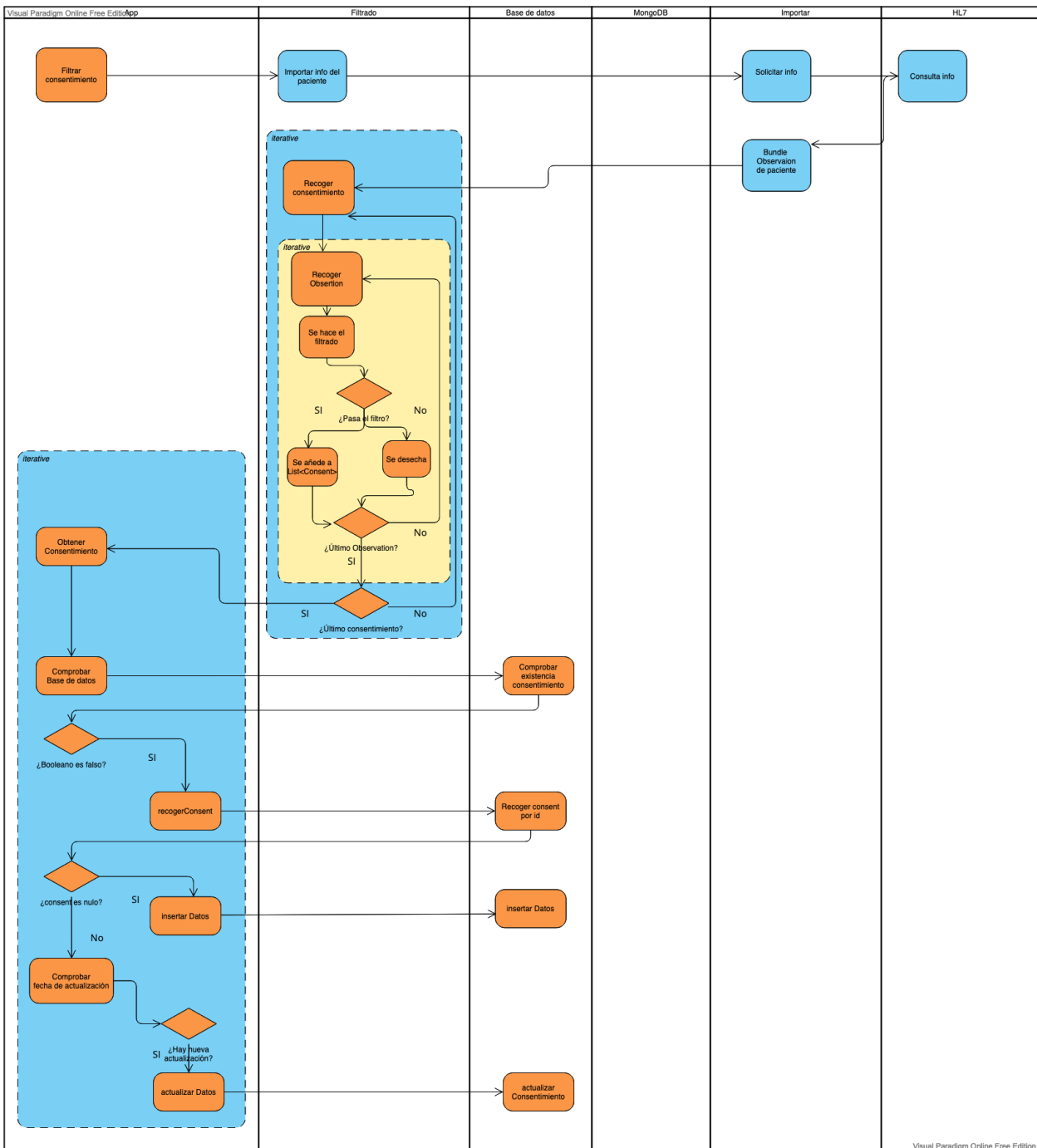


Ilustración 3.11 Diagrama de Actividad del método pocosConsentimientos

Como se puede ver en la Ilustración 3.10, existe un camino que invoca a `pocosConsentimientos`. Esta Ilustración 3.11 representa el diagrama de actividad de dicho método.

Primeramente, se llama a la clase `filtrar`, en el que se importa la información de un paciente obteniéndolo de la base de datos en formato Bundle.

La clase `filtrado` se encarga de recoger el consentimiento del Bundle, de un paciente específico, para iterar sobre los recursos `Observation`, del mismo paciente, que se han recogido de Bundle.

Se realiza el filtrado y en el caso de que pase la depuración se añade a la lista. Después, se comprueba si es el último recurso `Observation` del paciente, en caso afirmativo se comprueba si es el último consentimiento del recurso Bundle y si es así se pasa a la clase principal.

En la clase principal se obtiene el consentimiento de la lista del filtrado. Más tarde, se comprueba en la base de datos si existe ya dicha información, en caso opuesto se realiza una búsqueda por id del consentimiento, si no existe el id se insertan los datos.

Esta comprobación se debe a que es posible que exista un consentimiento con el mismo id, pero con distinta información, esto conduce a que está desactualizado. Por tanto, se comprueba el campo de lastUpdate, en el caso de que la fecha sea más actual que la del que se tiene guardado en la base de datos se almacena. En caso opuesto, se desecha.

Cabe destacar que no se muestran los diagramas de actividad del método 'muchosConsentimientos' y 'muchosPacientes' debido a que tan solo se realiza un bucle llamando a 'pocosConsentimientos' o 'muchosPacientes' pasando las páginas que se han obtenido en el algoritmo anterior.

Se muestra a continuación todos los métodos que está constituida la clase.

Tabla 2 Método main App

Método	Main
Descripción	Método que se invoca cuando se ejecuta la aplicación. Crea el contexto con la base de datos local HAPI-FHIR para llamar al paquete importar y comunicarse con pocosPacientes si ha recogido menos de 20 pacientes o muchosPacientes si ha recogido 20 o más pacientes.
Parámetros de entrada	Args (String[]) : Los posibles parámetros que se le pueden pasar por el terminal
Respuesta	Genera la respuesta del proyecto

Tabla 3 Método pocosPacientes App

Método	pocosPacientes
Descripción	Se invoca cuando se han recogido menos de 20 pacientes. Llama al paquete importar para recoger consentimientos del paciente, si son menos de 20 llama al método pocosConsentimientos y si son veinte o más llama a muchosConsentimientos.
Parámetros de entrada	Paciente (Bundle): Agrupación de recursos de pacientes
	Filtracion (Filtrado): Instancia del objeto filtrado
	BaseDeDatos(Database): Instancia de la base de datos
	importarPacientes (Importacion): Instancia de la importación
	Ctx_prework_local (FhirContext): Contexto FHIR
Client_prework_local (IGenericClient): Cliente FHIR	
Respuesta	No devuelve nada

Tabla 4 Método muchosPacientes App

Método	muchosPacientes
Descripción	Se invoca cuando se han recogido más de 20 pacientes. Llama al paquete importar para recoger consentimientos del paciente, si son menos de 20 llama al método pocosConsentimientos y si son veinte o más llama a muchosConsentimientos.
Parámetros de entrada	Paciente (Bundle): Agrupación de recursos de pacientes
	Filtracion (Filtrado): Instancia del objeto filtrado
	BaseDeDatos(Database): Instancia de la base de datos

	importarPacientes (Importacion): Instacia de la importación
	Ctx_prewrite_local (FhirContext): Contexto FHIR
	Client_prewrite_local (IGenericClient): Cliente FHIR
	Capacidad (int): Número de páginas de pacientes que tiene la base de datos HAPI-FHIR
Respuesta	No devuelve nada

Tabla 5 Método pocosConsentimientos App

Método	pocosConsentimientos
Descripción	Se invoca cuando se han recogido menos de 20 pacientes. Se tiene una lista de consentimientos de un paciente que llama al paquete de filtración. Los consentimientos que resten se suben a la base de datos si no existiesen.
Parámetros de entrada	Filtracion (Filtrado): Instancia del objeto filtrado
	Consentimiento (Bundle): Agrupación de recursos de tipo consentimiento
	ID_REAL(String): ID del paciente asociado a los consentimientos.
	Ctx_prewrite_local (FhirContext): Contexto FHIR
	BaseDeDatos(Database): Instancia de la base de datos
	idConsent (String): Id del consentimiento
Respuesta	No devuelve nada

Tabla 6 Método muchosConsentimientos App

Método	muchosConsentimientos
Descripción	Se invoca cuando se han recogido más de 20 pacientes. Se tiene una lista de consentimientos de un paciente que llama al paquete de filtración. Los consentimientos que resten se suben a la base de datos si no existiesen.
Parámetros de entrada	Filtracion (Filtrado): Instancia del objeto filtrado
	Consentimiento (Bundle): Agrupación de recursos de tipo consentimiento
	ID_REAL(String): ID del paciente asociado a los consentimientos.
	Ctx_prewrite_local (FhirContext): Contexto FHIR
	BaseDeDatos(Database): Instancia de la base de datos
	Capacidad (int): Número de páginas de consentimientos que tiene la base de datos HAPI-FHIR
	Client_prewrite_local (IGenericClient): Cliente FHIR
	idConsent (String): Id del consentimiento
Respuesta	No devuelve nada

3.3.2.2 Importación.java

La clase importación se encarga de, habiendo establecido una conexión con la base de datos local HAPI-FHIR. Importar los pacinetes devolviéndolo en formato Bundle.

Además, también importa los consentimientos y los datos clínicos que se han generado de la persona según el ID de dichos pacientes.

Tabla 7 Método importación

Método	Importacion
Descripción	Método que se invoca cuando instancian un objeto de la clase importación.
Parámetros de entrada	Client_prewrite_local (IGenericClient): Cliente FHIR
	Ctx_prewrite_local (FhirContext): Contexto FHIR

Respuesta	
-----------	--

Tabla 8 Método getPacientes

Método	getPacientes
Descripción	Método que invoca app para recoger todos los pacientes de la organización.
Parámetros de entrada	
Respuesta	Devuelve un recurso de tipo 'Bundle' con todos los pacientes.

Tabla 9 Método importarConsentimiento

Método	importarConsentimiento
Descripción	Método que se invoca cuando se desea recoger los consentimientos de un cierto paciente.
Parámetros de entrada	ID_REAL(String): ID del paciente asociado a los consentimientos.
Respuesta	Devuelve un recurso de tipo 'Bundle' con todos los consentimientos asociados a un paciente.

Tabla 10 Método getObservation

Método	getObservation
Descripción	Método que se invoca cuando se desea recoger los recursos Observation de un cierto paciente.
Parámetros de entrada	ID_REAL(String): ID del paciente asociado a los consentimientos.
Respuesta	Devuelve un recurso de tipo 'Bundle' con todas las observaciones asociadas a un paciente.

3.3.2.3 Filtrado.java

El filtrado en este proyecto es la clase más importante debido a que se encarga de hacer la depuración de los consentimientos asociados a los pacientes. Anteriormente se ha explicado cómo se realiza y qué pasos se siguen en la Ilustración 3.11, pero no se nombran los distintos tipos de filtrado que se han hecho. A continuación, se muestra un pequeño resumen de los tipos que se han modelado.

Tabla 11 Método Filtrado

Método	Filtrado
Descripción	Método que se invoca cuando se instancia un objeto de tipo Filtrado.
Parámetros de entrada	
Respuesta	Constructor de la clase sin parámetros

Tabla 12 Método filtrado con parámetros

Método	Filtrado
Descripción	Método que se invoca cuando se instancia un objeto de tipo Filtrado.

Parámetros de entrada	Client_prework_local (IGenericClient): Cliente FHIR
	Ctx_prework_local (FhirContext): Contexto FHIR
Respuesta	Constructor de la clase

Tabla 13 Método filtrar

Método	filtrar
Descripción	Método que se invoca cuando se desea filtrar los consentimientos de un paciente.
Parámetros de entrada	Consentimiento (Bundle): Agrupación de recursos de tipo consentimiento
	ID_REAL(String): ID del paciente asociado a los consentimientos.
Respuesta	Devuelve una lista con los consentimientos que han pasado el filtro.

Tabla 14 Método filtrarData

Método	filtrarData
Descripción	Método que es invocado por filtrar los consentimientos de un paciente por sus datos de referencia. En el caso de que no exista en la base de datos se desecha,
Parámetros de entrada	Consentimiento (Consent): Recursos de tipo consentimiento
	referenciaFiltro(String): la referencia del recurso Observation
Respuesta	Devuelve el consentimiento si pasa el filtro, sino se desecha.

Tabla 15 Método filtrarSecurityLabel

Método	filtrarSecurityLabel
Descripción	Método que es invocado por filtrar los consentimientos de un paciente por su nivel de seguridad. En el caso de que no exista en la base de datos se desecha.
Parámetros de entrada	Consentimiento (Consent): Recursos de tipo consentimiento
	codeFiltro(String): código del recurso Observation
	systemFiltro(String): campo sistema del recurso Observation
Respuesta	Devuelve el consentimiento si pasa el filtro, sino se desecha.

Tabla 16 Método filtrarCode

Método	filtrarCode
Descripción	Método que es invocado por filtrar los consentimientos de un paciente por su código. En el caso de que no exista en la base de datos se desecha.
Parámetros de entrada	Consentimiento (Consent): Recursos de tipo consentimiento
	systemFiltro(String): campo sistema del recurso Observation
Respuesta	Devuelve el consentimiento si pasa el filtro, sino se desecha.

Tabla 17 Método filtrarFecha

Método	filtrarFecha
Descripción	Método que es invocado por filtrar los consentimientos de un paciente por su fecha de almacenamiento. En el caso de que no exista en la base de datos se desecha.
Parámetros de entrada	Consentimiento (Consent): Recursos de tipo consentimiento
	startObserv(String): campo dateTime del recurso observación
Respuesta	Devuelve el consentimiento si pasa el filtro, sino se desecha.

3.4 Base de datos

3.4.1 Instalación

Para poder realizar la instalación de la base de datos es necesario seguir los pasos que se marcan en el Anexo.

3.4.2 Puesta en marcha

Para la puesta en marcha de la base de datos no se ha realizado ninguna modificación, por ende, se ha incluido en el Anexo con los pasos a seguir.

3.4.3 Modelado

MongoDB es una base de datos no relacional, esto quiere decir que no es necesario realizar una estructura lógica. Se almacenará información general sobre los consentimientos de la organización sanitaria que incluye en su base de datos. Aquellos datos que no se encuentre dentro de estructura no se almacenarán.

Para poder interactuar con la base de datos se ha hecho uso de la librería que proporciona MongoDB. Esto implica que no es necesario aprender el lenguaje en profundidad, con tan solo realizar la implementación de varios métodos es más que suficiente.

Se ha definido una clase donde se generan las operaciones básicas CRUD (create-read-update-delete). En la Figura 3.12 se muestra un diagrama de clase donde se han definido los distintos métodos y variables.

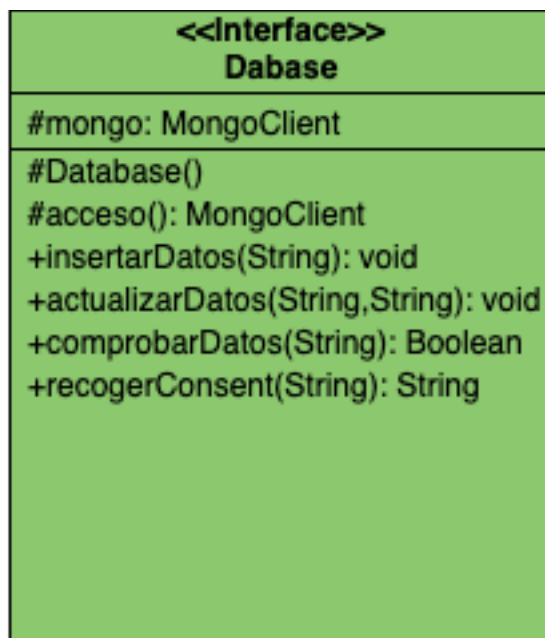


Ilustración 3.12 Diagrama de clase Database

Tabla 18 Método acceso

Método	acceso
Descripción	Método que es llamado para instanciar la base de datos MongoDB.

Parámetros de entrada	
Respuesta	Devuelve un objeto de tipo MongoClient con la conexión a la base de datos.

Tabla 19 Método insertarDatos

Método	insertarDatos
Descripción	Método que se encarga de insertar los datos a la base de datos.
Parámetros de entrada	Json(String): Consentimiento en formato cadena de texto
Respuesta	No devuelve nada.

Tabla 20 Método comprobarDatos

Método	comprobarDatos
Descripción	Método que invoca a todos los paquetes, además de crear el contexto con la base de datos local HAPI FHIR. Ejecuta la aplicación
Parámetros de entrada	Json(String): Consentimiento en formato cadena de texto
Respuesta	Devuelve 'true' si existe el consentimiento o 'false' si no existe

Tabla 21 Método actualizarDatos

Método	actualizarDatos
Descripción	Método que se encarga de actualizar los datos de la base de datos
Parámetros de entrada	Json(String): Consentimiento en formato cadena de texto id(String): Id del consentimiento en formato cadena de texto
Respuesta	No devuelve nada.

Tabla 22 Método recogerConsent

Método	recogerConsent
Descripción	Método que se encarga de recoger un consentimiento.
Parámetros de entrada	id(String): Id del consentimiento en formato cadena de texto
Respuesta	Devuelve el consentimiento si lo encuentra, sino null.

3.5 Pruebas

3.5.1 Ejecución de la aplicación

Para poder realizar la ejecución de la aplicación se necesitan los siguientes archivos que se han hecho para facilitar el proceso.

```

exeTFG.sh
1  #!/bin/sh
2  cd /Users/mariomartin/Desktop/TFG3
3  java -jar TFG3.jar

```

Ilustración 3.13 “exeTFG.sh”

Se ha tenido en cuenta que se ha realizado el proceso del Anexo A, ya que se necesita tener en marcha los dos contenedores.

3.5.2 Creación de instancias Patient

Se envía un POST a la base de datos con las características de la Figura siguiente. Se usa esta URL, ejecutándose en el servidor local en el puerto 8080. Como se va a insertar un paciente, se tiene que especificar en la URL /Patient.

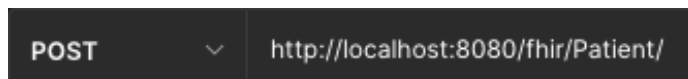


Ilustración 3.14 POST patient

Es importante escoger en el cuerpo de la petición el formato raw con la especificación de tipo JSON.

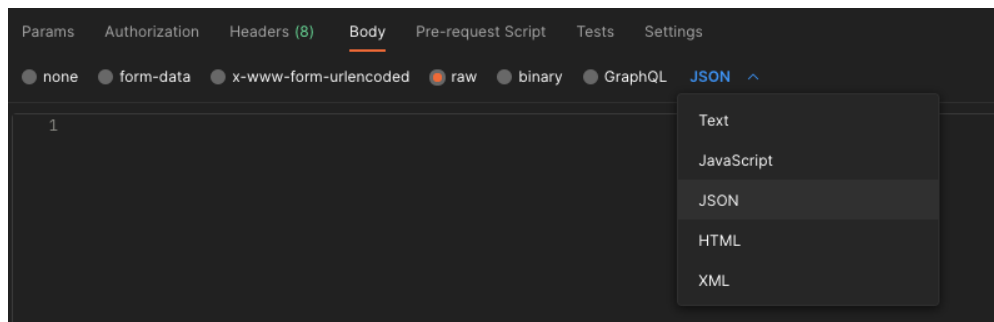


Ilustración 3.15 Body Json

Una vez realizado estas partes, se inserta el cuerpo del recurso Patient.

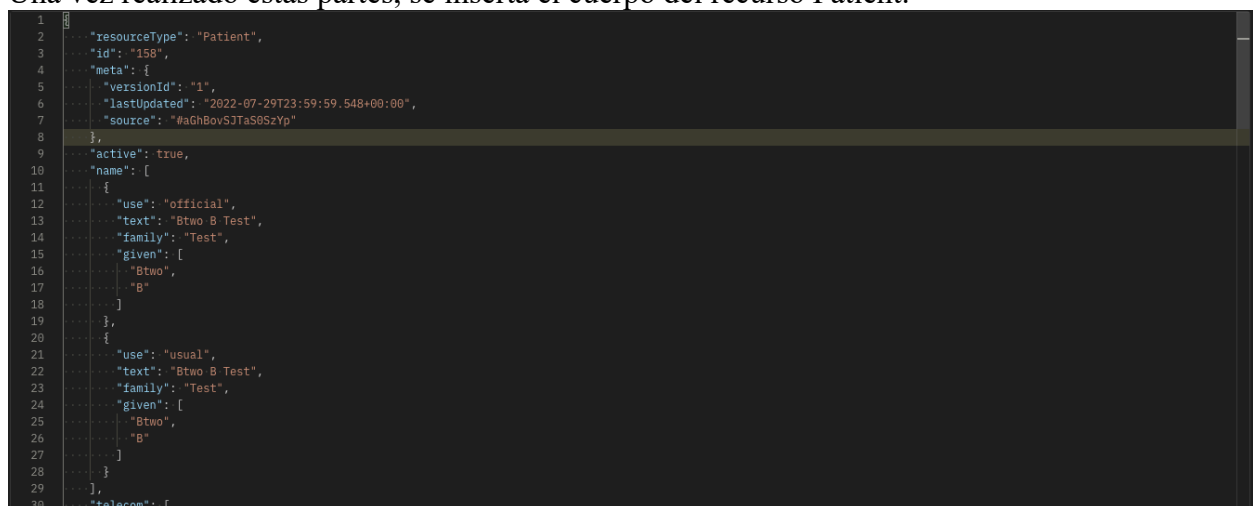


Ilustración 3.16 POSTMAN Patient

Si todo va bien, el servidor local responderá con una respuesta HTTP con código 201 Created.

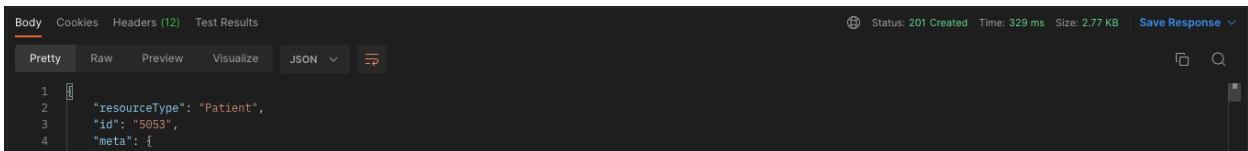


Ilustración 3.17 POSTMAN 201 Created

En el caso de que se haya producido un error sintáctico, el servidor mostrará una respuesta HTTP con código 400 Bad Request.

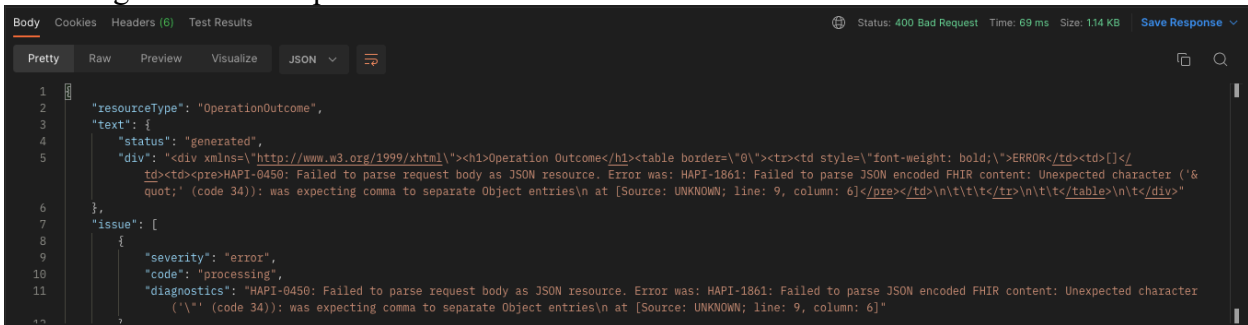


Ilustración 3.18 POSTMAN 400 Bad Request

3.5.3 Creación de instancias Consent

Para la creación de un recurso Consent es importante haber realizado un estudio de los campos que se van a necesitar de estándar. Posteriormente, una vez escrito el cuerpo del mensaje (en formato JSON) se tiene que enviar la petición a la siguiente URL. Esta petición viajará hacia el servidor local por el puerto 8080.

En la Ilustración 3.20 se muestra el cuerpo del mensaje del recurso Consent.

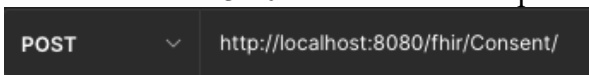


Ilustración 3.19 POST consent

```
{
  "resourceType": "Consent",
  "id": "256",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2022-08-25T17:40:10.700+00:00",
    "source": "#ozR9WFDkTyAFCA1G"
  },
  "status": "active",
  "scope": {
    "coding": [ {
      "code": "treatment"
    } ]
  },
  "category": [ {
    "coding": [ {
      "code": "treatment",
      "display": "Treatment"
    } ]
  } ],
  "patient": {
    "reference": "Patient/162"
  },
  "dateTime": "2021-01-01",
  "policyRule": {
    "text": "Spain policy"
  },
  "provision": {
    "period": {
      "start": "2021-10-01"
    },
    "code": [ {
      "coding": [ {
        "system": "http://snomed.info/sct",
        "code": "1261007",
        "display": "Fracture of multiple ribs"
      } ]
    } ]
  },
  "search": {
    "mode": "match"
  }
}
```

Ilustración 3.20 Cuerpo del recurso consent

El proceso de realizar una petición POST es el mismo que se ha aplicado en el recurso Patient. Donde se recibe una respuesta HTTP desde el servidor 201 Created si se ha creado correctamente el servidor y un 400 Bad Request en el caso de que exista un error en el cuerpo de la petición.

3.5.4 Creación de instancias Observation

Como en los casos anteriores, para poder insertar un recurso se necesita la siguiente URL <http://localhost:8080/fhir/<recurso>>

En este caso, como se desea insertar el recurso Observation pues se cumple con lo establecido anteriormente.

En la Ilustración 3.22 se puede observar los campos que se han rellenado de recurso Observation. No son estándares, es decir, no es necesarios que todos los pacientes de la organización sanitaria deban de tener los mismos campos. Esto mismo se aplica a los demás recursos

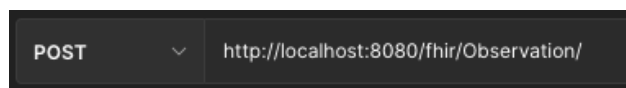


Ilustración 3.21 POST observation

```
{
  "resourceType": "Observation",
  "id": "3452",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2022-09-03T09:38:30.620+00:00",
    "source": "#mZrPPrJla6gfp0G6"
  },
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "9279-1"
      }
    ],
    "text": "Respiratory Rate"
  },
  "subject": {
    "reference": "Patient/160"
  },
  "effectiveDateTime": "2020-10-20T16:18:01-04:00",
  "valueQuantity": {
    "value": 31,
    "unit": "resp/min"
  }
}
```

Ilustración 3.22 Cuerpo del recurso Observation

Ya se ha comentado en los puntos anteriores las respuestas que se obtiene por parte del servidor en el caso de que el recurso no esté bien escrito sintácticamente y cuando no hay ningún error.

3.5.5 Filtrar por código

En los apartados anteriores se ha explicado que se hace un proceso de filtrado. Sin embargo, no se ha especificado exactamente cómo se realiza, pues se pueden implementar numerosos filtros. En este apartado y los sucesivos, describimos algunos tipos de filtros de consentimientos.

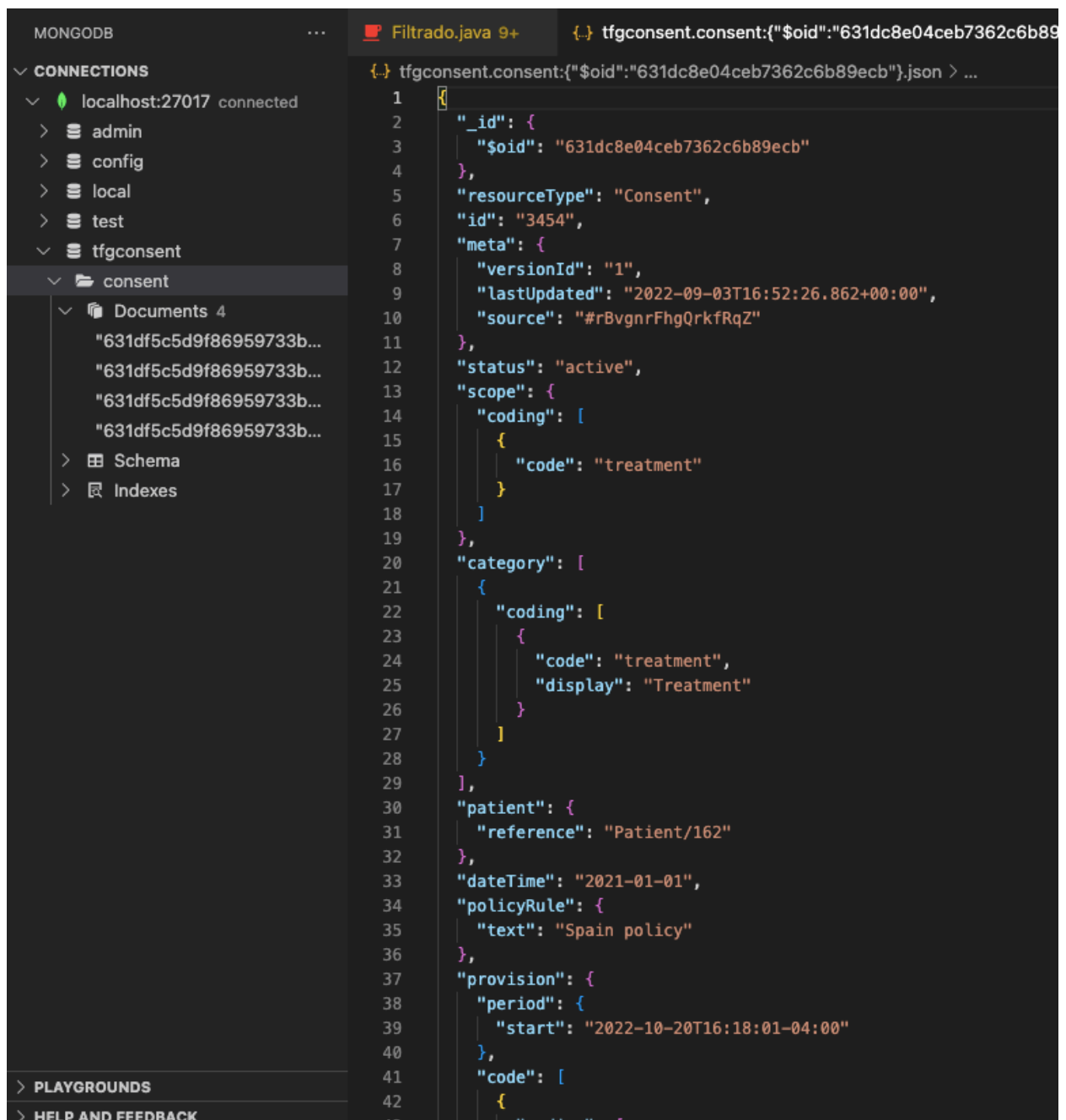
En este primer caso, la filtración se realizará comparando el campo del consentimiento “code” con el campo “code” de todos los recursos Observation asociados al mismo paciente.

En la siguiente figura se puede observar cómo ciertos consentimientos pasan el filtro. Cabe destacar que, para este caso, se ha modificado el código para que se muestra visualmente los consentimientos que pasan el filtro. Lo ideal es que no se muestre nada porque ciertas organizaciones sanitarias pueden contener grandes cantidades de pacientes y esto lo haría incomprensible.

```
mariomartin@MacBook-Air-de-Mario TFG3 % java -jar TFG3.jar
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#noProviders for further details.
El consentimiento http://localhost:8080/fhir/Consent/257/_history/1 ha pasado el filtro por código.
El consentimiento http://localhost:8080/fhir/Consent/3454/_history/1 ha pasado el filtro por código.
El consentimiento http://localhost:8080/fhir/Consent/3455/_history/1 ha pasado el filtro por código.
El consentimiento http://localhost:8080/fhir/Consent/3456/_history/1 ha pasado el filtro por código.
```

Ilustración 3.23 Filtro por código

Para comprobar que se ha subido a la base de datos, se muestra en la siguiente Figura donde uno de los consentimientos ha pasado el filtro. Es decir, la organización sanitaria tiene al menos una instancia de recurso Observation para el paciente del consentimiento, y esa instancia posee el código que indica el consentimiento.



```
1 {
2   "_id": {
3     "$oid": "631dc8e04ceb7362c6b89ecb"
4   },
5   "resourceType": "Consent",
6   "id": "3454",
7   "meta": {
8     "versionId": "1",
9     "lastUpdated": "2022-09-03T16:52:26.862+00:00",
10    "source": "#rBvgnrFhgQrkfRqZ"
11  },
12  "status": "active",
13  "scope": {
14    "coding": [
15      {
16        "code": "treatment"
17      }
18    ]
19  },
20  "category": [
21    {
22      "coding": [
23        {
24          "code": "treatment",
25          "display": "Treatment"
26        }
27      ]
28    }
29  ],
30  "patient": {
31    "reference": "Patient/162"
32  },
33  "dateTime": "2021-01-01",
34  "policyRule": {
35    "text": "Spain policy"
36  },
37  "provision": {
38    "period": {
39      "start": "2022-10-20T16:18:01-04:00"
40    },
41    "code": [
42      {
43        "coding": [
```

Ilustración 3.24 Filtro Código MongoDB

3.5.6 Filtrar por capa de seguridad

Este tipo de filtrado es muy importante. La idea principal es recoger el campo “provision.securityLabel” del recurso Consent para compararlo con la información que tiene el paciente. En este caso, se debería buscar en el campo “category” del recurso Observation.

Por tanto, en el caso de que la organización sanitaria necesite cierta información del paciente para realizar un estudio, se necesitará que exista un consentimiento en el que coincida la capa de seguridad frente a la categoría del paciente. Cabe destacar que el consentimiento puede no permitir el uso de dicha información.

En la Ilustración 3.25 se muestra como un consentimiento ha pasado el filtrado y posteriormente como se ha subido a la base de datos. Se puede ver claramente por el ID que se muestra en la Ilustración 3.25 y 3.26.

```
[mariomartin@MacBook-Air-de-Mario TFG3 % java -jar TFG3.jar
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#noProviders for further details.
El consentimiento http://localhost:8080/fhir/Consent/5056/_history/1 ha pasado el filtro por securityLabel.
```

Ilustración 3.25 Filtro por capa de seguridad

```

MONGODB
... App.java 5 tfgconsent.consent:{"$oid":"631e17216ab04e6336bea3ff"},json x Filtrado.java
CONNECTIONS
localhost:27017 connected
admin
config
local
test
tfgconsent
  consent
    Documents 5
      "631df5c5d9f86959733b..."
      "631df5c5d9f86959733b..."
      "631df5c5d9f86959733b..."
      "631df5c5d9f86959733b..."
      "631e17216ab04e6336be..."
    Schema
    Indexes
PLAYGROUNDS
HELP AND FEEDBACK

tfgconsent.consent:{"$oid":"631e17216ab04e6336bea3ff"},json > ...
1  [
2  {
3    "_id": {
4      "$oid": "631e17216ab04e6336bea3ff"
5    },
6    "resourceType": "Consent",
7    "id": "5056",
8    "meta": {
9      "versionId": "1",
10     "lastUpdated": "2022-09-11T17:12:52.711+00:00",
11     "source": "#SywHc22tKzXSMtSd"
12   },
13   "status": "active",
14   "scope": {
15     "coding": [
16       {
17         "code": "treatment"
18       }
19     ]
20   },
21   "category": [
22     {
23       "coding": [
24         {
25           "system": "http://terminology.hl7.org/CodeSystem/observation-category",
26           "code": "laboratory",
27           "display": "laboratory"
28         }
29       ]
30     }
31   ],
32   "patient": {
33     "reference": "Patient/160"
34   },
35   "dateTime": "2021-01-01",
36   "policyRule": {
37     "text": "Spain policy"
38   },
39   "provision": {
40     "period": {
41       "start": "2021-10-01"
42     }
43   },
44   "securityLabel": [
45     {
46   }
47   ]
48 }
49 ]

```

Ilustración 3.26 Filtro capa de seguridad MongoDB

3.5.7 Filtrar por fecha

Si una organización sanitaria desea recoger cierta información de un paciente, necesita acudir a los consentimientos que se tiene sobre él para poder observar si el usuario permite o restringe su uso.

Un ejemplo para este filtrado puede ser cuando se tiene información en la base de datos de un usuario de una fecha específica. El consentimiento establece que permite que se pueda hacer uso de dichos datos en un intervalo de tiempo, por tanto, en el caso de que la fecha que se tiene está dentro del rango se guarda el consentimiento. Posteriormente, se hace uso de la información del paciente porque el usuario a través del consentimiento lo ha permitido.

Para este caso, la filtración se realizará comparando el campo del consentimiento “dataPeriod” con el campo “effectiveDateTime” de todos los recursos Observation asociados al mismo paciente.

```
mariomartin@MacBook-Air-de-Mario TFG3 % java -jar TFG3.jar
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#noProviders for further details.
El consentimiento http://localhost:8080/fhir/Consent/3456/_history/1 ha pasado el filtro por fecha.
```

Ilustración 3.27 Filtro por fecha

```

MONGODB ... 9733b2e4b}.json {} tfgconsent.consent:{"$oid":"631df5c5d9f86959733b2e53"}
v CONNECTIONS {} tfgconsent.consent:{"$oid":"631df5c5d9f86959733b2e53"}.json > ...
v localhost:27017 connected 1 {}
  > admin 2
  > config 3
  > local 4
  > test 5
  > tfgconsent 6
    > consent 7
      > Documents 4 8
        "631df5c5d9f86959733b..." 9
        "631df5c5d9f86959733b..." 10
        "631df5c5d9f86959733b..." 11
        "631df5c5d9f86959733b..." 12
      > Schema 13
      > Indexes 14
    > PLAYGROUNDS 15
    > HELP AND FEEDBACK 16

17 {
18   "_id": {
19     "$oid": "631df5c5d9f86959733b2e53"
20   },
21   "resourceType": "Consent",
22   "id": "3456",
23   "meta": {
24     "versionId": "1",
25     "lastUpdated": "2022-09-03T16:57:52.844+00:00",
26     "source": "#ZrQc68FJfSpq12Ir"
27   },
28   "status": "active",
29   "scope": {
30     "coding": [
31       {
32         "code": "treatment"
33       }
34     ]
35   },
36   "category": [
37     {
38       "coding": [
39         {
40           "code": "treatment",
41           "display": "Treatment"
42         }
43       ]
44     }
45   ],
46   "patient": {
47     "reference": "Patient/162"
48   },
49   "dateTime": "2021-01-01",
50   "policyRule": {
51     "text": "Spain policy"
52   },
53   "provision": {
54     "code": [
55       {
56         "coding": [
57           {
58             "system": "http://loinc.org",
59             "code": "100052-0"
60           }
61         ]
62       }
63     ]
64   }
65 }

```

Ilustración 3.28 Filtro fecha MongoDB

3.5.8 Filtrar por referencia

Por último, existe otro tipo de filtrado que se va a realizar y es el de referencia.

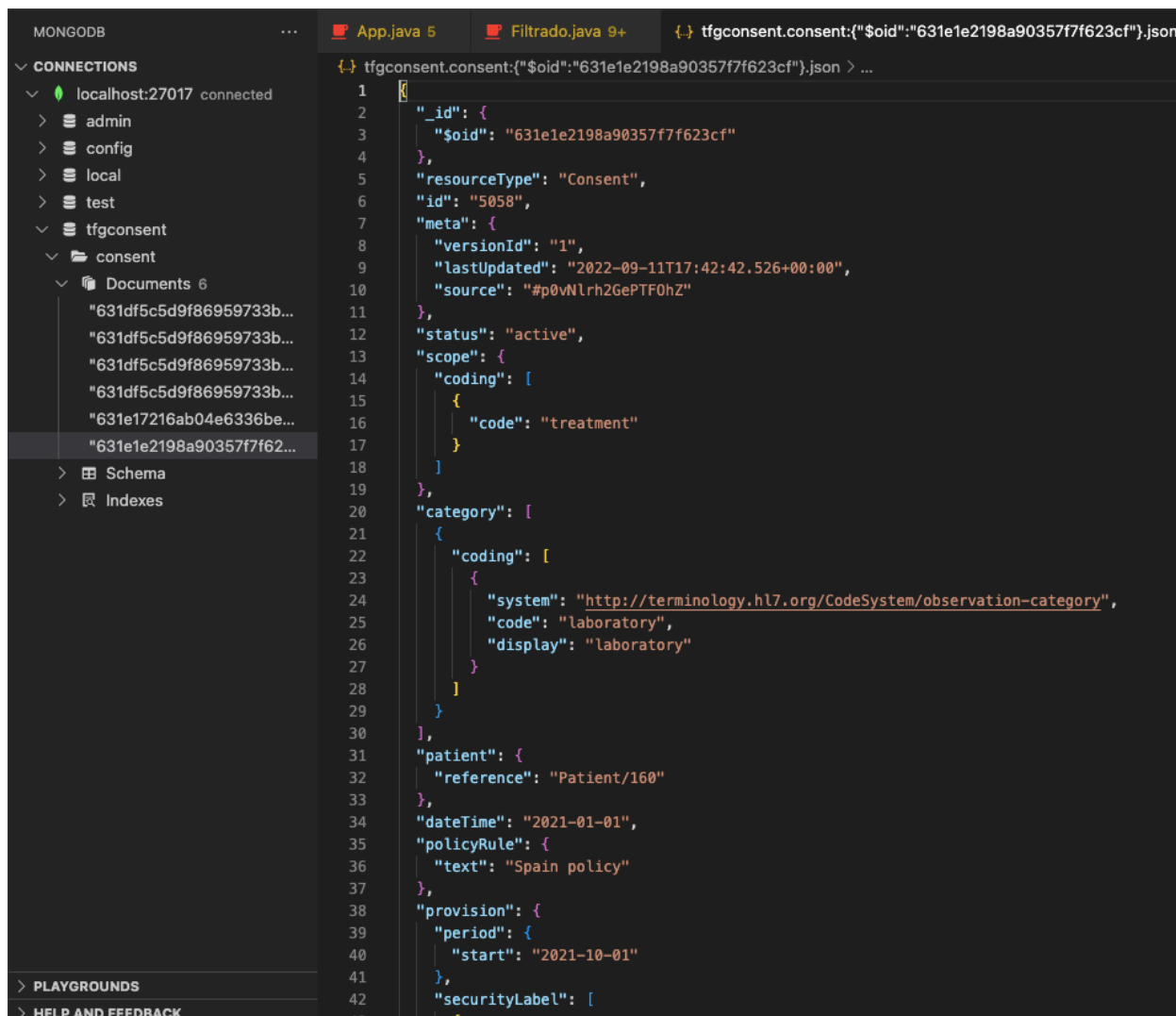
Como se ha comentado antes, se recogerán los consentimientos de un paciente para después mandar una petición a la base de datos local HAPI-FHIR sobre la información, en este caso la referencia, de dicho paciente.

Se comparará pues el campo del consentimiento “provision.data.reference” con el campo “performer” del recurso observación para saber si la información que tiene la organización sanitaria se cumple con el consentimiento asociado al paciente. En el caso de que coincida, se almacenará para el posterior uso de la organización.

En la Figura 3.29 se puede ver como un consentimiento con el ID 5058 ha pasado la depuración. Más tarde, en la Ilustración 3.30 se puede ver en la base de datos MongoDB como sí se ha almacenado correctamente con el id 5058.

```
mariomartin@MacBook-Air-de-Mario TFG3 % java -jar TFG3.jar
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#noProviders for further details.
El consentimiento http://localhost:8080/fhir/Consent/5058/_history/1 ha pasado el filtro por referencia.
```

Ilustración 3.29 Filtro por referencia



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel shows a connection to 'localhost:27017'. Under the 'tfgconsent' database, the 'consent' collection is expanded, showing a list of documents. The document with ID '631e1e2198a90357f7f623cf' is selected. The main panel displays the JSON document for this ID, which includes fields for '_id', '\$oid', 'resourceType', 'id', 'meta', 'status', 'scope', 'category', 'patient', 'dateTime', 'policyRule', 'provision', and 'securityLabel'. The 'id' field is '5058', and the 'patient' field has a 'reference' of 'Patient/160'. The 'provision' field has a 'start' date of '2021-10-01'.

```
{ "_id": {
  "$oid": "631e1e2198a90357f7f623cf"
},
  "resourceType": "Consent",
  "id": "5058",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2022-09-11T17:42:42.526+00:00",
    "source": "#p0vNlrh2GePTF0hZ"
  },
  "status": "active",
  "scope": {
    "coding": [
      {
        "code": "treatment"
      }
    ]
  },
  "category": [
    {
      "coding": [
        {
          "system": "http://terminology.hl7.org/CodeSystem/observation-category",
          "code": "laboratory",
          "display": "laboratory"
        }
      ]
    }
  ],
  "patient": {
    "reference": "Patient/160"
  },
  "dateTime": "2021-01-01",
  "policyRule": {
    "text": "Spain policy"
  },
  "provision": {
    "period": {
      "start": "2021-10-01"
    }
  },
  "securityLabel": [

```

Ilustración 3.30 Filtro Referencia MongoDB

4 CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se van a mostrar las conclusiones finales y qué puntos de mejora se pueden realizar en un futuro para la extensión de este proyecto.

4.1 Conclusiones

La realización de este proyecto ha hecho que profundice en los conocimientos que he obtenido a lo largo de la carrera. Además, he realizado un estudio sobre toda la información que aporta el estándar HL7 FHIR.

Este modelo es muy importante y conocido en el ámbito sanitario. Sin embargo, se han de destacar varios aspectos negativos. Este estándar, para aquellas personas que deseen empezar a trabajar sobre él, lo más común es que cueste entenderlo porque existe poca documentación. Esto hace que la mayoría de los proyectos sean costosos al inicio. Además, está en continuo desarrollo lo que provoca que se tengan que actualizar proyectos del pasado que usaban cierta información que en un futuro se verá modificada o incluso llegar a estar obsoleta.

Se ha trabajado por primera vez con contenedores, exactamente con Docker que empaqueta software en contenedores que incluye en ellos todo lo necesario para que se ejecute, incluyendo las librerías. Estos son más livianos porque trabajan directamente sobre el Kernel y no es necesario instalar un sistema operativo por contenedor. Se han usado para desplegar las aplicaciones necesarias para desarrollar este proyecto haciendo que se profundice los conocimientos sobre este tema.

El servidor local HAPI-FHIR se ha desplegado mediante Docker y como ya se ha comentado la información que se ha subido se ha representado en formato JSON, esto me ha permitido aprender más sobre la sintaxis de esta modalidad de texto sencilla para el intercambio de datos.

Por último, mencionar que se ha hecho un estudio sobre el nuevo formato de base de datos llamado NoSQL, donde no se impartió a lo largo del grado. Este modelo trae con él nuevas formas de guardar la información en contextos en los que no se necesite una estructura lógica. Lo más común es guardar los datos con el formato JSON, pero no es la única opción.

4.2 Líneas futuras

Este proyecto es el nacimiento de un programa que realiza, en tiempo real, la importación de los consentimientos de los pacientes de una organización sanitaria específica siempre y cuando se dote de dicha información en la base de datos de la entidad. Sin embargo, se pueden realizar muchas mejoras en el desarrollo del proyecto como:

- **Interfaz gráfica para la gestión de la aplicación:** La creación de una interfaz gráfica, ya sea una aplicación de escritorio o una página web, ayudaría al personal sanitario de la organización para realizar la importación de los consentimientos de los pacientes que tenga asociado. De esta forma, no sería necesario ejecutar un archivo ni tampoco conocer el lenguaje JSON para

subir los datos a la base de datos de la organización.

- **Incluir un mecanismo de seguridad:** La seguridad es muy importante en cualquier ámbito. Por tanto, como se está tramitando información de personas que permiten que sus consentimientos sean compartidos a otras organizaciones es muy importante que exista un mínimo de seguridad para que no se puedan manipular dichos datos. FHIR no es un protocolo de seguridad ni define ninguna funcionalidad relacionada, por ende se pueden aplicar otros protocolos como:
 - Intercambio de datos mediante SSL
 - Sistemas de autenticación y control de acceso
 - Políticas de gestión de datos
 - Uso de firmas digitales
- **Aumento de recursos:** Para este proyecto solamente se han hecho uso de cuatro recursos que se han explicado anteriormente que han sido necesario para llegar al objetivo de este proyecto. Sería interesante incluir recursos como “Practitioner”, “Organization” etc.
- **Aumento de funcionalidades:** Debido al aumento de los recursos, esto generaría un aumento de funcionalidades como un registro de fechas de importación de consentimientos. También se podrían incluir más operaciones como DELETE, PUT... En este proyecto solamente se han hecho uso de las operaciones GET, POST.
- **Base de datos remota:** El uso de una base de datos remota facilitaría el trabajo puesto que la información que se modifique para un agente (persona que hace uso del proyecto) afectaría también a otro. Sin necesidad de que este último tuviese que actualizar la información modificada.

REFERENCIAS

- [1] *La protección de datos en la UE*. Comisión Europea - European Commission. (2022). https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_es.
- [2] Ruiz, F. J. (2021). *Gestor de consentimientos para el procesamiento de información sanitaria en el marco de la GDPR y conforme al estándar HL7 FHIR* [Trabajo Fin de Grado]. Universidad de Sevilla. <https://idus.us.es/bitstream/handle/11441/126804/TFG-3662-RUIZ%20GOMEZ.pdf?sequence=1&isAllowed=y>
- [3] *8 Derechos de los Sujetos de Datos con GDPR | Blog de Seguridad Informática*. Helpsystems.com. (2022). <https://www.helpsystems.com/es/soluciones/seguridad-informatica/cumplimiento/cumplimiento-de-gdpr/8-derechos>.
- [4] *GDPR: 7 claves para cumplir con el RGPD en hospitales y clínicas*. Blog.signaturit.com. (2018). <https://blog.signaturit.com/es/gdpr-claves-para-hospitales-y-clinicas>.
- [5] Ortiz, M. *Modelo Incremental*. Isw-udistrital.blogspot.com. <http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html>.
- [6] *Index - FHIR v4.3.0*. Hl7.org. (2022). <http://hl7.org/fhir/>.
- [7] *Security - FHIR v4.3.0*. Hl7.org. (2022). <https://www.hl7.org/fhir/security.html>.
- [8] *Valueset-observation-codes - FHIR v4.3.0*. Hl7.org. (2022). <http://hl7.org/fhir/valueset-observation-codes.html>.
- [9] *Datatypes - FHIR v4.3.0*. Hl7.org. (2022). <https://www.hl7.org/fhir/datatypes.html>.
- [10] *Consent - FHIR v4.3.0*. Hl7.org. (2022). <http://hl7.org/fhir/consent.html>.
- [11] *Patient - FHIR v4.3.0*. Hl7.org. (2022). <http://hl7.org/fhir/patient.html>.
- [12] *Observation - FHIR v4.3.0*. Hl7.org. (2022). <http://hl7.org/fhir/observation.html>.
- [13] *Bundle - FHIR v4.3.0*. Hl7.org. (2022). <https://www.hl7.org/fhir/bundle.html>.
- [14] *LOINC Terminology Service using FHIR – LOINC*. LOINC. (2022). <https://loinc.org/fhir/>.
- [15] *GitHub - hapi.fhir/hapi-fhir-jpaserver-starter*. GitHub. (2022) <https://github.com/hapi.fhir/hapi-fhir-jpaserver-starter>.
- [16] *FHIR APIs | FHIR® tutorials*. Fhir-drills.github.io. (2022). <https://fhir-drills.github.io/fhir-api.html>.
- [17] *Spring makes Java simple..* Spring. (2022). <https://spring.io/>.
- [18] Rodríguez, F. (2022). *Maven Repository*. MVNrepository. <https://mvnrepository.com/>.

-
- [19] *Maven* - *Wikipedia, la enciclopedia libre*. Es.wikipedia.org. (2022). <https://es.wikipedia.org/wiki/Maven>.
- [20] Porter, B., Zyl, J., & Lamy, O. (2022). *Maven – Welcome to Apache Maven*. Maven.apache.org. <https://maven.apache.org/>.
- [21] *Docker Desktop - Docker*. Docker. (2022). <https://www.docker.com/products/docker-desktop/>.
- [22] Robledano, A. (2019). *Qué es MongoDB y características*. OpenWebinars.net. <https://openwebinars.net/blog/que-es-mongodb/>.
- [23] Postman.com. (2022). <https://www.postman.com/>.
- [24] Rosa, J. (2018). *¿Qué es REST? Conoce su potencia*. OpenWebinars.net <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>.
- [25] *Visual Studio Code - Code Editing. Redefined*. Code.visualstudio.com. (2022). <https://code.visualstudio.com/>.
- [26] *Table of Contents - HAPI FHIR Documentation*. Hapifhir.io. (2022). <https://hapifhir.io/hapi-fhir/docs/>.
- [27] Programming Knowledge. (2022). *Cómo instalar MongoDB en Mac OS X* [Video]. <https://www.youtube.com/watch?v=DX15WbKidXY>.
- [28] Verjel, F. (2020). ▷ *Guía para Instalar MongoDB en un Mac de dos maneras distintas*. ▷ Guía para Instalar MongoDB en un Mac de dos maneras distintas. <https://franyerverjel.com/blog/instalar-mongodb-en-un-mac>.
- [29] *Resource - FHIR v4.3.0*. HL7.org. (2022) <https://hl7.org/fhir/resource.html>.

ANEXO A: MANUAL DE INSTALACIÓN Y DESPLIEGUE DE LA APLICACIÓN

A.1 Instalación del servidor HAPI FHIR

Para realizar la instalación del servidor se pueden seguir dos caminos.[26]

- Vía Docker:

Primero se procede a visitar la página oficial de Docker y descargar la interfaz disponible en los S.O de Apple y de Windows. Una vez descargada, siguiendo los pasos de instalación se obtendrá algo como la Figura, pero sin los contenedores (la foto se realizó después de añadir los contenedores).

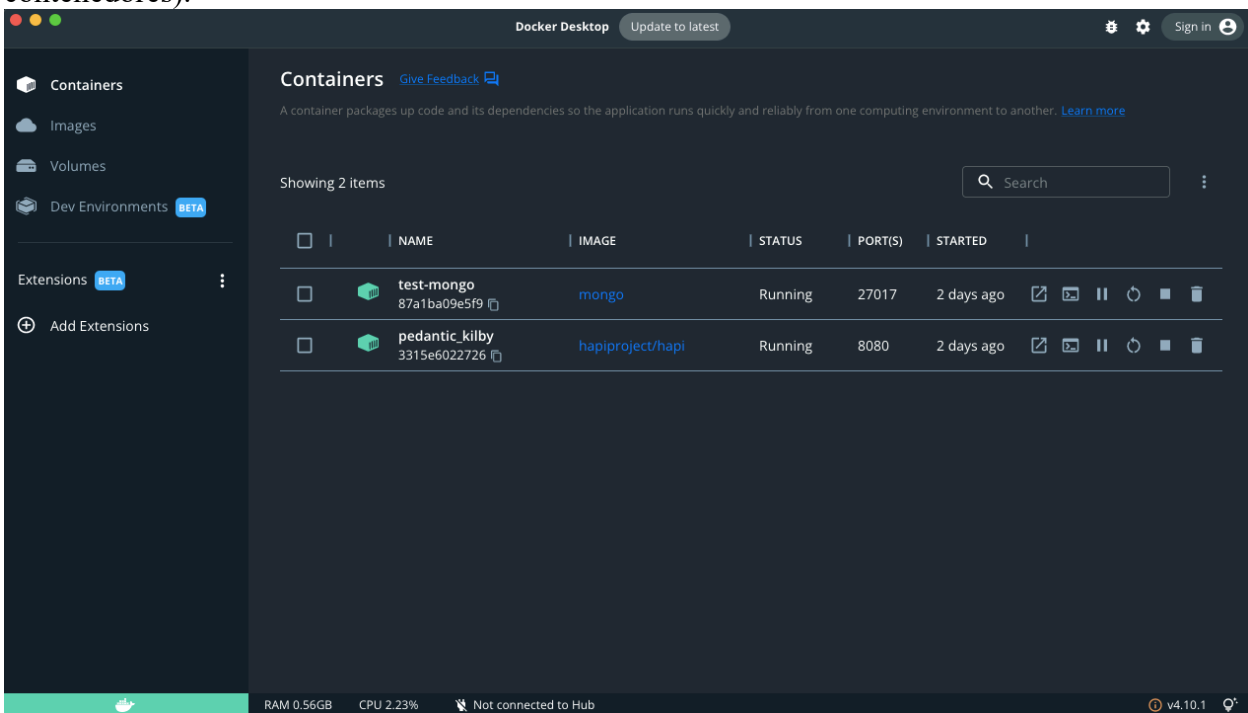


Ilustración 0.1 Docker Desktop

Una vez descargado Docker Desktop, se procede a abrir el terminal e insertar el siguiente comando:

```
docker pull hapiproject/hapi:latest
docker run -p 8080:8080 hapiproject/hapi:latest
```

Esto ejecutará la imagen de Docker con la configuración predeterminada, asignando el puerto 8080 en el contenedor al puerto 8080 en el host.

A.2 Puesta en marcha del servidor HAPI-FHIR

La puesta en marcha del servidor FHIR se rige habiendo realizado el paso anterior. Una vez se hayan ejecutado los comandos, se puede acceder a la interfaz de usuario del servidor HAPI FHIR visitando la URL <http://localhost:8080> en cualquier navegador, o puede usar <http://localhost:8080/fhir> para realizar las solicitudes REST.

Otra opción para poder iniciar el servicio es mediante Docker Desktop pulsando el botón de iniciar y tras ello, aparecerá nuevos iconos en el que se deberá pulsar sobre “Abrir en el navegador”.

Existe la opción de cambiar el puerto asignado desde la configuración de HAPI.

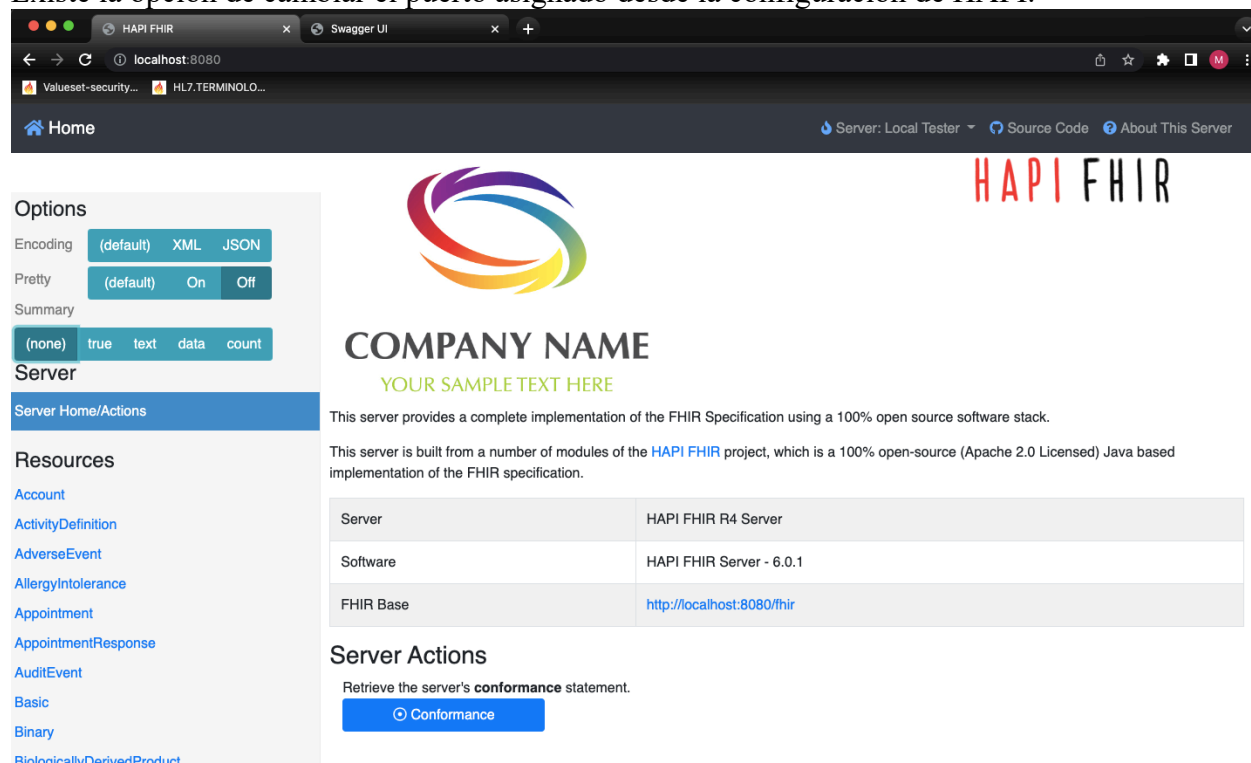


Ilustración 0.2 Servidor FHIR local

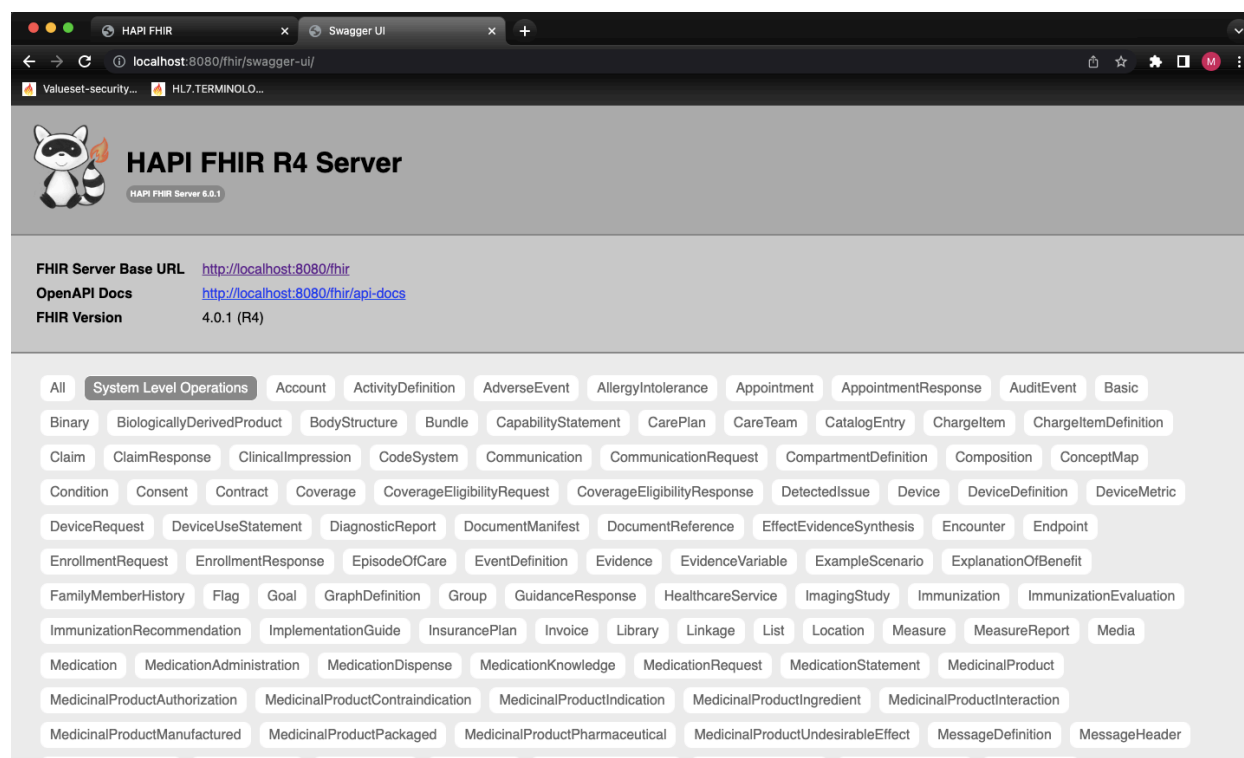


Ilustración 0.3 Servidor FHIR-REST

Además, se puede observar que el servidor funciona correctamente si en la aplicación POSTMAN se realiza una petición GET sobre la URL localhost:8080/fhir/metadata. Devolverá un recurso de tipo "CapabilityStatement" con información.

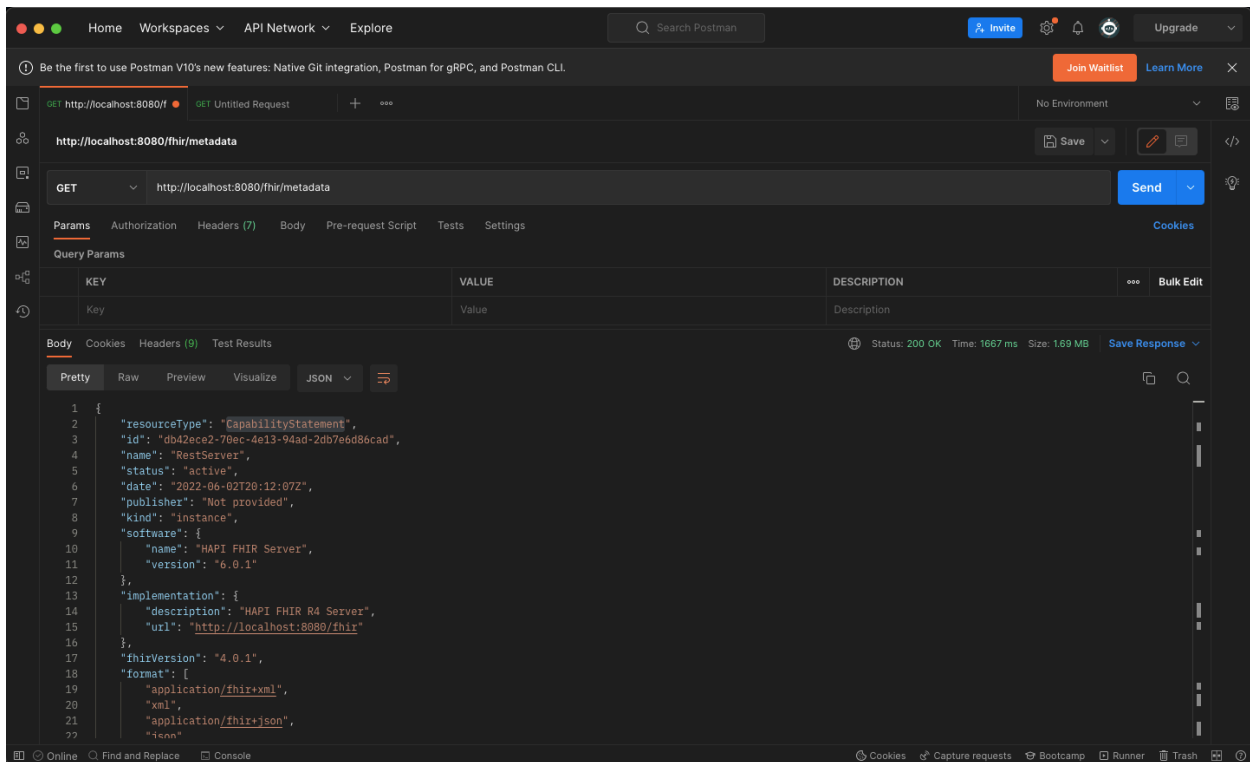


Ilustración 0.4 CapabilityStatement Postman

A.3 Instalación MongoDB

Como se ha comentado con anterioridad, los consentimientos se guardarán en una base de datos MongoDB. Por tanto, para poder realizar la instalación de dicho programa se seguirán los pasos que se detallan a continuación, aclarar que existen dos formas de realizar la instalación[28]:

⇒ Vía Docker

Los prerrequisitos que se necesitan para instalar mongodb es tener la aplicación de Docker Desktop. Una vez instalada, cualquier usuario puede comenzar con un contenedor MongoDB usando Docker con el siguiente comando en su terminal:

```
docker run --name mongodb -d mongo
```

Este comando empezará a ejecutar un servidor de mongoDB con la última versión disponible. Se recomienda que se use una etiqueta para especificar la versión para asegurar la consistencia.

Si necesita acceder al servidor desde otra aplicación que está en local, tendrá que especificar el puerto con el comando -p

```
docker run --name mongodb -d -p 27017:27017 mongo
```

Una vez haya terminado la ejecución ya se dispondrá en la interfaz gráfica de Docker desktop el contenedor de mongo. En él se puede iniciar y se estará haciendo uso de la aplicación en el puerto 27017.

⇒ Vía manual

Para poder realizar la instalación y ejecución de MongoDB de forma manual se debe ir a la sección de descargas de la página web <https://www.mongodb.com/try/download/community> y descargar la versión correcta que se adapte a las condiciones de su sistema operativo.

Tras descargar mongo, debe mover el archivo comprimido .tar (el archivo con la extensión .tgz que has descargado) al directorio donde desee instalar su base de datos no relacional. Para este caso, se dice que mongo resida en la carpeta de inicio, por lo que se abre el terminal y se expresan los siguientes comandos:

- `cd Downloads`
- `tar -zxvf mongodb-osx-x86_64-6.0.1.tgz`
- `sudo mv mongodb-macos-x86_64-6.0.1 /usr/local/mongodb`
- `cd /usr/local/mongodb/`
- `sudo mkdir -p /data/db`
- `cd /data/db`
- `sudo chown mariomartin /data/db`
- `cd`
- `open .bash_profile`

Entonces el usuario debe de copiar y pegar las dos siguientes líneas en el fichero:

```
export MONGO_PATH=/usr/local/mongodb
```

```
export PATH=$PATH:$MONGO_PATH/bin
```

Esto se hace para poder guardar las variables en el PATH y ejecutar la aplicación con un alias en vez de tener que poner toda la ruta donde se encuentra el archivo. Una vez copiada las dos líneas, se procede a ejecutar el comando “source .bash_profile” para actualizarlo. [27]

Por último, se debe de abrir un nuevo terminal para poder ejecutar el servidor con el comando `mongod`, comúnmente llamado demonio, mientras que en otro terminal se ejecutará el comando `mongo` (correspondiente al cliente).

A.4 Puesta en marcha de MongoDB

Una vez se hará ejecutado la instalación del servicio, la puesta en marcha se puede realizar de dos formas:

- Vía terminal

Para poder acceder desde el terminal se debe pulsar la opción que proporciona Docker “Abrir en un terminal” como en la Figura (en el caso de que se haya instalado la aplicación por Docker Desktop).

En él iniciará una nueva ventana en la que se tendrá que escribir el comando “mongo” para poder interactuar con la base de datos. Si se ha instalado manualmente la aplicación solo bastaría con abrir un nuevo terminal y ejecutar el comando anterior, siempre y cuando el demonio, es decir `mongod`, también esté ejecutándose.

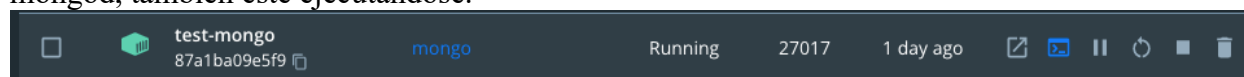


Ilustración 0.5 Docker mongo

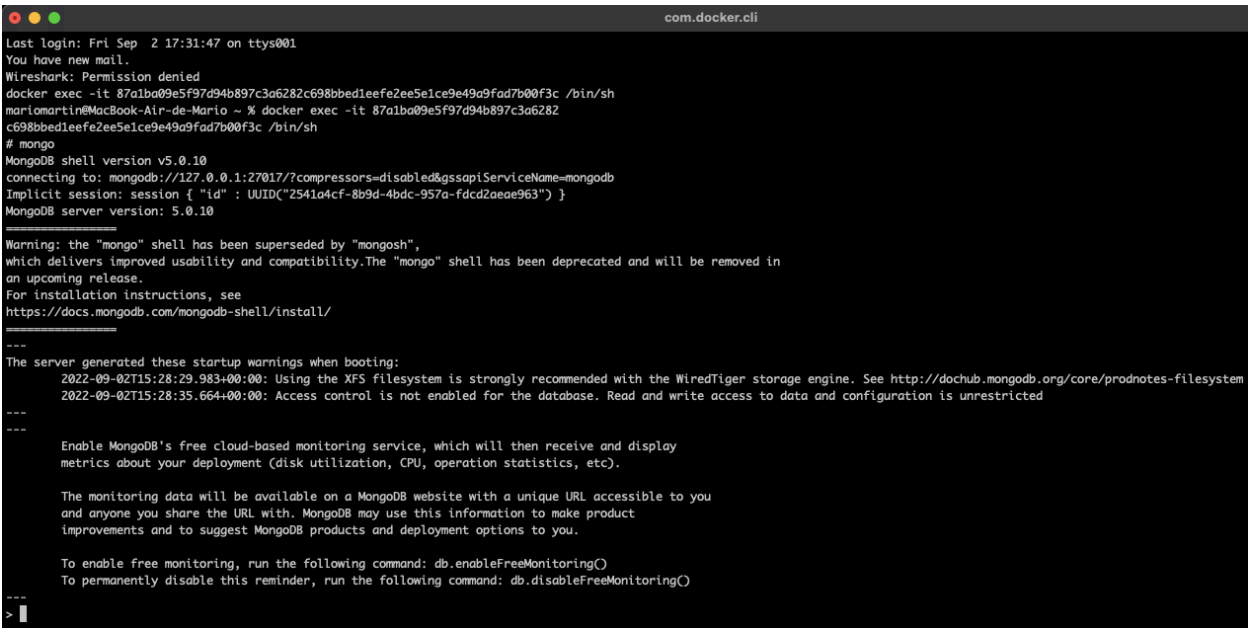


Ilustración 0.6 Comando mongo

- Vía Visual Studio Code

Para poder interactuar con la base de datos desde Visual Studio Code hay que dirigirse al apartado de extensiones y buscar “MongoDB for VS Code”. Una vez se ha instalado, aparecerá un nuevo icono en la esquina izquierda, al pulsar sobre él se mostrará la información que aparecer en la Figura.

Cuando el usuario se haya conectado a la base de datos tendrá la opción de observar de forma gráfica toda la información, tal y como se muestra en la Figura x.x

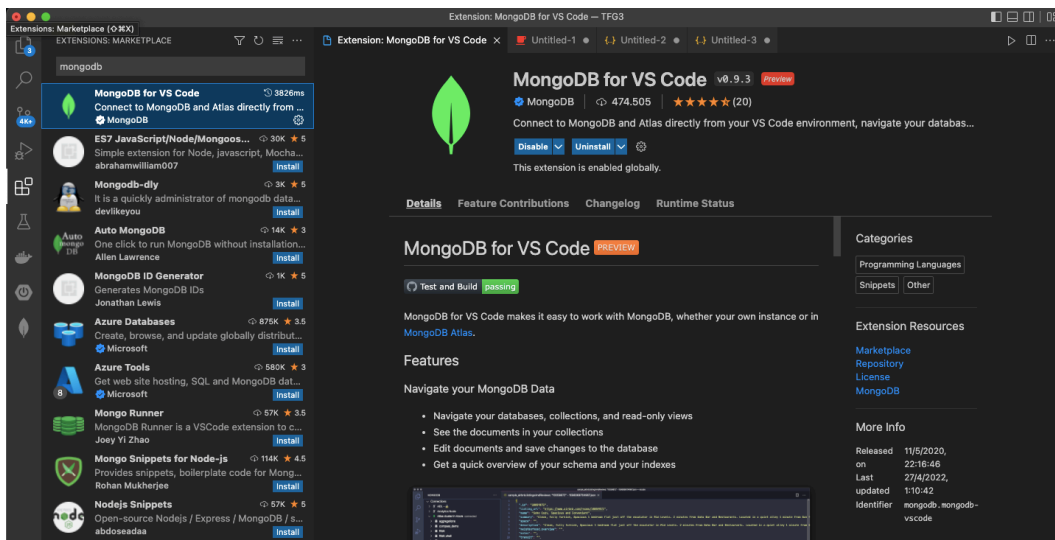


Ilustración 0.7 Extensión mongoDB VScode

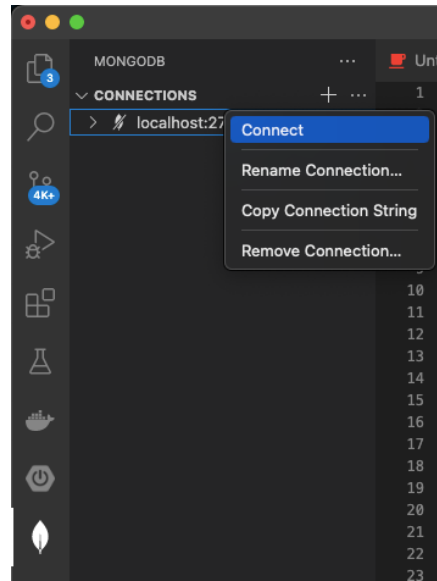


Ilustración 0.8 Conexión a mongoDB

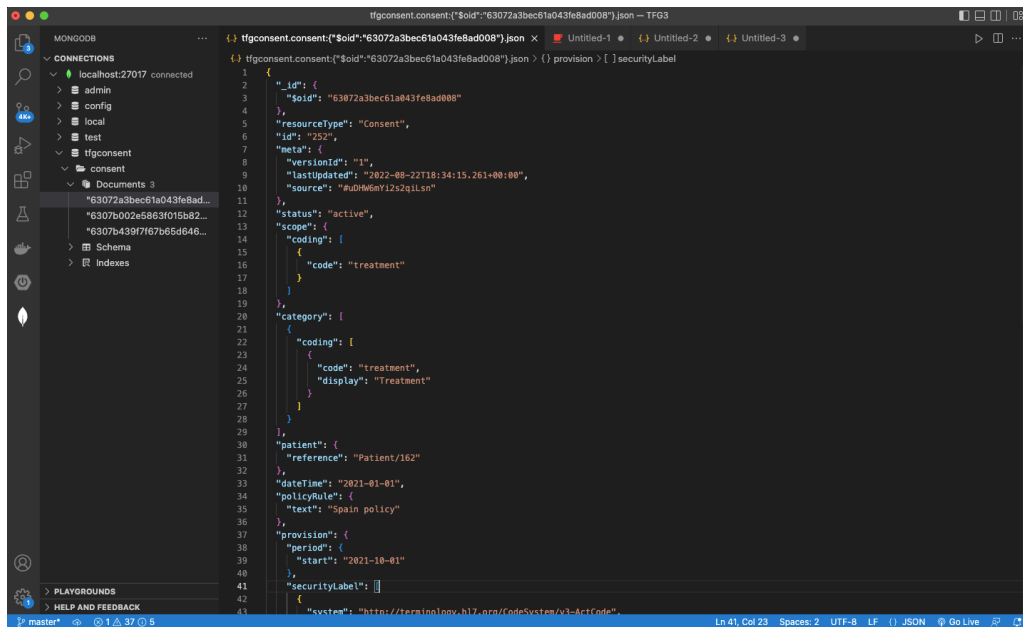


Ilustración 0.9 Menu de MongoDB en VSCode