Proyecto Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Servicio de control de acceso a información de historia clínica electrónica almacenada en sistemas HL7 FHIR

Autor: Carlos Villar Prudencio

Tutor: Jorge Calvillo Arbizu

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

Sevilla, 2020

Proyecto Fin de Máster

Máster Universitario en Ingeniería de Telecomunicación

Servicio de control de acceso a información de historia clínica electrónica almacenada en sistemas HL7 FHIR

Autor:

Carlos Villar Prudencio

Tutor:

Jorge Calvillo Arbizu

Profesor Sustituto Interino

Dpto. Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Máster: Servicio de control de acceso a información de historia clínica electrónica almacenada en sistemas HL7 FHIR

Autor: Carlos Villar Prudencio
Tutor: Jorge Calvillo Arbizu

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente/a:

Vocal/es:

Secretario/a:

acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

Resumen

Las tecnologías avanzan de forma constante, apareciendo cada día nuevos mecanismos y estándares para el manejo e intercambio de la información en cualquier área de trabajo. La evolución de los sistemas que comparten información a través de la red hace necesario que se apliquen distintos mecanismos de seguridad para garantizar el acceso restringido a información sensible.

Este trabajo se ha centrado en el análisis de los protocolos XACML, SAML y OAuth para realizar los flujos de autenticación y autorización mediante políticas, además de revisar los estándares de interoperabilidad HL7 y HL7 FHIR junto con el proyecto IPS (International Patient Summary) para el intercambio de información clínica entre países.

Posteriormente, se ha definido un sistema de control de acceso y seguridad para el intercambio de información sanitaria por medio de servicios REST, haciendo uso de los protocolos y estándares estudiados. Para la construcción del sistema se ha hecho uso de las herramientas WSO2 Identity Server y WSO2 Enterprise Integrator que permiten implementar los servicios y elementos que realizan el control de acceso.

Finalmente se han evaluado distintos escenarios aplicando las medidas de autenticación y autorización contempladas previamente, demostrando el funcionamiento de las mismas.

Palabras clave: Tecnología, servicios, seguridad, sanidad, REST, HL7, FHIR, WSO2, XACML, OAuth

Abstract

Technologies are constantly advancing, with new mechanisms and standards appearing every day for the management and exchange of information in any area of work. The evolution of the systems that share information through the network makes it necessary to apply different security mechanisms to guarantee restricted access to sensitive information.

This work focuses on the analysis of the XACML, SAML and OAUTH protocols to perform the authentication and authorization flows through policies. Furthermore, the HL7 and HL7 FHIR interoperability standars have been studied in conjunction with the IPS (International Patient Summary) project for the exchange of clinical information.

Afterwards, an access control system has been defined for the safe exchange of health information through REST services, making use of the protocols and standards studied. The WSO2 Identity Server and WSO2 Enterprise Integration tools have been used for this.

Finally, different scenarios have been evaluated applying the previously contemplated authentication and authorization measures, showing how them works.

Keywords: Tecnology, services, security, health, REST, HL7, FHIR, WSO2, XACML, OAuth

Índice

Índice de Figuras

Índice de Tablas

Índice de Códigos

Siglas

ABAC Attribute-Based Access Control.

ADT Admit/Visit Notification Message (HL7).

API Application Programming Interfaces.

CLI Command Line Interface.

EI Enterprise Integrator.

ESB Enterprise Service Bus.

EVN Event Type Segment (HL7).

FHIR Fast Healthcare Interoperability Resources.

HL7 Health Level Seven.

HTML Hyper Text Markup Languaje.

HTTP Hypertext Transfer Protocol.

IdP Identity Provider.

IPS International Patient Summary.

IS Identity Server.

JiT Just in Time.

JSON JavaScript Object Notation.

JWT JSON Web Token.

LDAP Lightweight Directory Access Protocol.

MSH Message Header Segment (HL7).

OASIS Organization for the Advancement of Structured Information Standards.

OAuth Open Authorization.

OIDC OpenID Connect.

PAP Policy Administration Point.

PDP Policy Decision Point.

PEP Policy Enforcement Point.

PID Patient Identification Segment (HL7).

PIP Policy Information Point.

QoS Quality of Service.

RBAC Role-Based Access Control.

REST Representational State Transfer.

SAML Security Assertion Markup Language.

SCIM System for Cross-domain Identity Management.

SOA Service Oriented Architecture.

SOAP Simple Object Access Protocol.

SP Service Provider.

SSO Single Sign On.

TIC Tecnologías de la Información y la Comunicación.

TLS Transport Layer Security.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

XACML eXtensible Access Control Markup Language.

XML Extensible Markup Language.

XSPA Cross-Enterprise Security and Privacy Authorization.

1 Introducción

Este apartado permitirá al lector entender y seguir la estructura del trabajo. En él se definen los motivos por los cuales se desarrolla este trabajo, así como los objetivos del mismo que se pretenden alcanzar, la justificación y relevancia del mismo y, por último, la metodología empleada para su desarrollo.

1.1 Introducción y motivación

Las tecnologías de la información y las comunicaciones (TIC) se encuentran en constante evolución y desarrollo. Un ámbito de gran valor e importancia es el ámbito sanitario en el cual se viene produciendo en los últimos años un cambio de paradigma. Día a día aparecen nuevos sistemas y aplicaciones que trabajan con información sanitaria, ya sea dentro de los propios centros sanitarios (hospitales, centros de atención, laboratorios, etc..) o aplicaciones y servicios ofrecidos directamente a las personas a través de servicios web o móvil y cuya finalidad es mejorar la atención y calidad de vida de los mismos.

La seguridad es un pilar fundamental en el mundo de las TICs y que cada día cobra mayor importancia por el valor que adquiere la información. Diseñar un sistema tan amplio que de servicio a numerosos actores de forma segura es todo un reto. Es importante establecer mecanismos de autenticación y autorización que permitan, de forma flexible y escalable, controlar el acceso y explotación de la información a los distintos actores que compondrán el sistema.

El incremento de sistemas y aplicaciones está provocando, además, un cambio en la forma de acceder y transmitir la información. En el marco de la interoperabilidad se está desarrollando una evolución en la cual donde antes se definían servicios complejos que intercambiaban una gran cantidad de información entre sistemas, ahora proliferan multitud de servicios ligeros con el objetivo de transmitir menor cantidad de información, pero de forma más rápida para ofrecer una mejor calidad de servicio (QoS) y acercar dicha información a las personas. Este cambio, como se detallará a lo largo del documento, está fomentando una migración de arquitecturas orientadas a servicios mediante protocolo SOAP, a arquitecturas centradas en el intercambio de datos a través de servicios web mediante APIs REST.

La proliferación de servicios, aplicaciones y sistemas provoca que la información manejada esté cada vez más fragmentada y a menudo los sistemas no cooperan entre ellos, impidiendo la evolución y crecimiento de los dominios. Esto está propiciando en el marco de la interoperabilidad la adopción de estándares que estructuren la información y definan esquemas que permitan la integración de sistemas heterogéneos, al convivir aplicaciones construidas con distintas arquitecturas, distintos equipos de distinta índole con software propio, etc. En el ámbito clínico se cuenta con la organización HL7 [1] que define varios estándares para la estructuración de información clínica y que está acompañando a este cambio de paradigma presentando el estándar HL7 FHIR [2] orientado a los servicios REST. FHIR no especifica un mecanismo propio de control de acceso, sino que da una serie de directrices de como podría implementarse, por lo que resultará necesario evaluar la manera de implementarlo dependiendo de la necesidad del sistema o servicio.

Por último, dentro del ámbito sanitario cabe mencionar el proyecto IPS (International Patient Summary) [3] cuyo objetivo es facilitar la atención médica entre pases, donde el escenario no puede ser más heterogéneo,

mediante la compartición de información sanitaria esencial y por medio de un documento de historia clínica resumida.

Toda esta problemática expuesta conduce a la necesidad de diseñar y construir un sistema de control de acceso a información sanitaria almacenada en FHIR donde, además, se trabaja con información personal y en muchas ocasiones de carácter sensible, cobrando aún mayor valor. La importancia y complejidad que supone diseñar este sistema sirve de motivación para llevar a cabo este trabajo.

1.2 Objetivos

El presente trabajo está compuesto por un objetivo principal y diversos objetivos de carácter secundario que permitan complementar al principal.

El objetivo principal es:

- **Diseñar un servicio de control de acceso** a información de historia clínica por medio del estándar HL7 FHIR en un sistema sanitario.

Los objetivos secundarios son:

- **Investigar protocolos y estándares** que se pueden emplear en un ámbito sanitario, ya sea para el intercambio de información, la composición y estructuración de dicha información y securización del acceso.
- **Analizar herramientas** que permitan construir y desplegar el sistema de control de acceso.
- **Mostrar el funcionamiento** del servicio de forma práctica por medio de escenarios de prueba.

1.3 Metodología

La metodología seguida para el desarrollo del trabajo y obtención de los resultados se define a continuación:

En primer lugar, junto al tutor del trabajo se establecen los objetivos y el alcance del mismo. Durante esta etapa, se contemplan y debaten diferentes opciones y alternativas a desarrollar.

En segundo lugar, una vez definidos los objetivos se deberán estudiar los diferentes elementos que componen el alcance del trabajo con el fin de poder comprenderlos y poder realizar el grueso del proyecto. Durante esta etapa, se consultarán fuentes bibliográficas de confianza, que serán empleadas durante la elaboración del documento de trabajo para respaldar la información que se muestre en el mismo y permitiendo al lector consultarlas para verificar y/o ampliar la información.

Para reforzar todo el contenido teórico asociado al trabajo, se abordará un marco práctico que muestre, en varios escenarios, el funcionamiento de los distintos elementos del mismo. Para ello, una vez concluido el estudio de todos los elementos que componen el cuerpo del trabajo, se estudiarán diversas herramientas que permitan crear y probar el escenario, mostrando los resultados obtenidos del mismo.

Por último, una vez que se logre estudiar y alcanzar los objetivos, así como demostrar el funcionamiento mediante escenarios prácticos, se plasmarán en el documento del trabajo a modo de memoria, permitiendo al lector aprender y entender de los distintos elementos y objetivos desarrollados durante el trabajo.

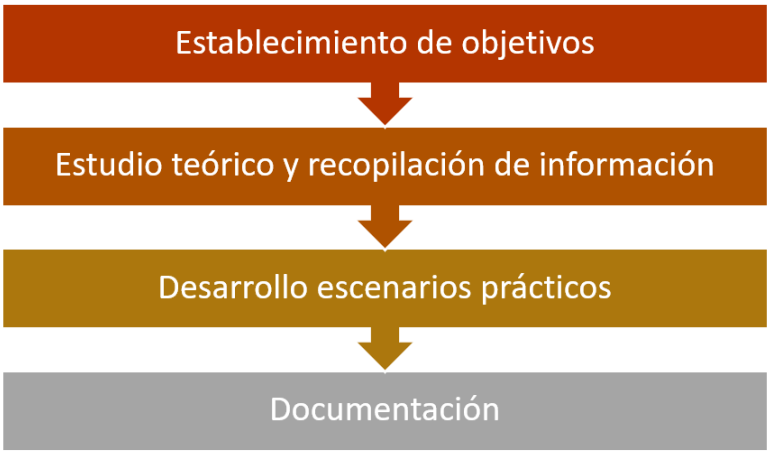

Figura 1.1 Etapas de trabajo.

2 Marco teórico

En este capítulo se pretende, en primer lugar, hacer un breve repaso a los conceptos más importantes involucrados en este trabajo y que son necesarios conocer para poder entenderlo. Entre estos conceptos se verá los protocolos XACML, SAML y OAuth 2.0 para autenticación y autorización, los estándares HL7 y HL7 FHIR para la interoperabilidad entre sistemas clínicos y el proyecto IPS para la comunicación de historia clínica esencial entre países. Posteriormente, en el siguiente capítulo, se abordarán y desarrollarán varios escenarios prácticos que permitirán entender y profundizar estos conceptos.

2.1 Introducción

El almacenamiento, acceso y explotación de la información son puntos fundamentales en el ciclo de vida de los datos. Esto no es un concepto nuevo, pero es la base de cualquier sistema informático, los cuales se encuentran en constante evolución y desarrollo, lo que provoca que surjan nuevos métodos o mecanismos capaces de llevar un paso más allá el desarrollo del ciclo de vida de los datos.

Estos desarrollos han llevado a las empresas y organismos a poder trabajar de forma remota con la información, pudiendo almacenarla y explotarla a través de la red. Esta capacidad de transmitir la información por la red permite la conexión entre distintos sistemas o aplicaciones.

Que puedan acceder distintas aplicaciones o sistemas a la misma información almacenada o mediante intercambio provoca la necesidad de tener en cuenta, entre otros, dos aspectos fundamentales: la seguridad de la información y la estructuración de la misma.

La seguridad de la información se ha convertido en un área fundamental de las tecnologías. Cada día gana más valor y se invierte más y más en conseguir mecanismos que aseguren que los datos que se almacenan sean seguros y se encuentren accesible únicamente a las personas, empresas o entidades autorizadas. Para ello es importante definir medidas de autenticación que identifiquen a las personas o empresas que acceden a la información, medidas de autorización que controlen los permisos de acceso a dicha información y medidas de auditoría que permitan conocer los accesos realizados.

La multitud de arquitecturas y lenguajes existentes en el mundo de las tecnologías provoca que, a menudo, la información sea intercambiada por sistemas o aplicaciones que entienden o manejan la información de formas diferentes, es decir, de forma heterogénea. Esto genera la necesidad de desarrollar mecanismos o estándares que permitan estructurar la información de la forma más homogénea posible.

Uno de los ámbitos en los que es fundamental todo esto es en el sanitario. Cualquier sistema sanitario se compone de multitud de aplicaciones, muchas de ellas probablemente desarrolladas y gestionadas por distintas empresas con distintas arquitecturas y softwares, que intercambian información constantemente y que podrá ser accesible por distintas personas. Esta información intercambiada entre aplicaciones en su mayoría será de carácter personal y podrá incluir información sensible, de modo que se debe asegurar que el acceso a ella sea únicamente con autorización. Todo esto debe realizarse de forma transparente para el cliente o usuario final sin afectar a la calidad del servicio.

2.2 XACML y SAML

XACML es un estándar de seguridad cuyo objetivo es el control de acceso a sistemas, aplicaciones o información, por medio de la definición de políticas y un modelo o estructura de procesado de las peticiones.

Dicho estándar define un lenguaje basado en XML para diseñar políticas de control de acceso por medio de atributos, así como un sistema compuesto por distintos elementos que en conjunto permite evaluar las peticiones recibidas y autorizar el acceso [4].

Los distintos elementos que componen el sistema y su función se definen a continuación:

- **Punto de Administración de Políticas** (*Policy Administration Point* o PAP): Punto del sistema que se encarga de la gestión y administración de las políticas de acceso.
- **Punto de Decisión de las Políticas** (*Policy Decision Point* o PDP): Entidad que evalúa las políticas aplicables a las solicitudes recibidas, consulta el PIP y toma la decisión de permitir o denegar el acceso.
- **Punto de Cumplimiento de Políticas** (*Policy Enforcement Point* o PEP): Elemento que recibe las peticiones de acceso de las distintas aplicaciones o usuarios. Desde aquí se redirige la petición al PDP y comunica la decisión tomada al origen.
- **Punto de Información de Políticas** (*Policy Information Point* o PIP): Entidad o elemento que sirve de fuente de valores de los distintos atributos.
- **Context handler**: Elemento del sistema que convierte las solicitudes de decisión al formato propio de XACML y coordina los distintos elementos del sistema. En el momento de devolver la decisión, convierte el formato XACML al formato original con el que fue recibida la solicitud.

Estos elementos permiten la orquestación del siguiente flujo para las solicitudes:

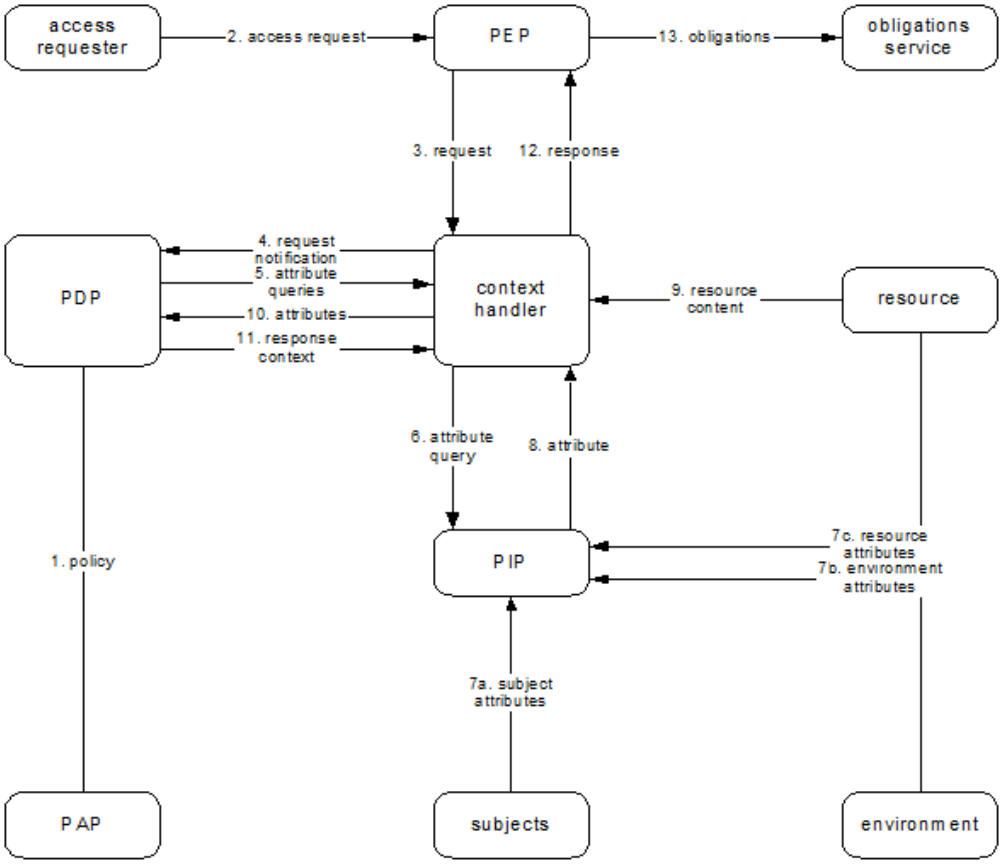

Figura 2.1 Flujo de datos XACML.

XACML define en su lenguaje un modelo o estructura con formato XML para la aplicación de políticas. Esta estructura se compone principalmente de tres elementos: reglas, políticas y conjunto de políticas.

La regla, identificada mediante la marca <*Rule*> en XML, es el elemento principal de una política. A su vez, una regla contiene varios componentes que permiten realizar su evaluación:

- Objetivo (<*Target*>): El objetivo de una regla determina sobre que recursos, usuarios, aplicaciones o acciones se debe aplicar la regla.
- Efecto (*Effect*): Especifica el resultado previsto de la aplicación de la regla. Sus posibles valores serán permitir (*Permit*) o denegar (*Deny*) la solicitud
- Condición (<*Condition*>): Determina las condiciones que se deben cumplir en una solicitud para poder aplicar la regla que la contiene.

Por su lado, una política es la unidad que se emplea en el PAP y que se aplica al flujo de datos para la evaluación de reglas. Esta se compone de los siguientes elementos:

- Objetivo (<*Target*>): Es el mismo elemento de una regla, pero aplicado al nivel de la política. Si se indica este elemento sobre la política, podría no indicarse dentro de las reglas recogidas en la misma.
- Algoritmo de combinación de reglas (*RuleCombiningAlgId*): Es un algoritmo que indica a la política como se deben combinar las distintas reglas que esta puede albergar para la toma de decisiones.
- Reglas (<*Rule*>): Son las reglas definidas anteriormente y puede haber más de una dentro de una política.
- Obligaciones (<*Obligations*>): A la hora de definir una política se pueden definir una serie de obligaciones que, en el momento de evaluación y decisión de la política, serán informadas al PEP por parte del PDP para su cumplimiento.

Finalmente, un conjunto de políticas encapsula varias políticas del mismo modo que una política puede albergar varias reglas en su interior.

Código 2.1 Estructura de una política XACML.

```
<Policy>
   <Description>...</Description>
   <Target>
      ...
   </Target>
   <Rule Effect="Permit">
      <Condition>
         ...
      </Condition>
   </Rule>
   <Rule Effect="Deny">
      <Condition>
         ...
      </Condition>
   </Rule>
</Policy>
```

Además de los principales elementos que componen las políticas, XACML define las funciones y acciones para evaluarlas y aplicarlas. En este trabajo se recogen los elementos básicos que dan pie al mismo y se recomienda al lector que si está interesado, acuda a la documentación de XACML para profundizar en la materia [4].

La evaluación de reglas y objetivos se realiza sobre distintos atributos o *claims* que permiten caracterizar o identificar a los distintos elementos del sistema (usuarios, aplicaciones, roles, etc.). Un conjunto de *claims* pueden ser recogidos en forma de dialectos o perfiles para su utilización en conjunto con fines comunes.

A su vez, SAML es un estándar de código abierto basado en XML para el intercambio de información, ya sea de autenticación como de autorización [5]. Este intercambio de información se realiza entre un proveedor de identidad y un proveedor de servicio.

El proveedor de servicio es la entidad que ofrece uno o más servicios o recursos y a la que los usuarios acceden por medio de autenticación. El proveedor de identidad es la entidad que almacena la información de los usuarios y permite corroborar que un usuario que se intenta autenticar es quien dice ser.

Este estándar define el flujo de la información que se intercambia entre el proveedor de servicio y el proveedor de identidad para verificar y autenticar al usuario.

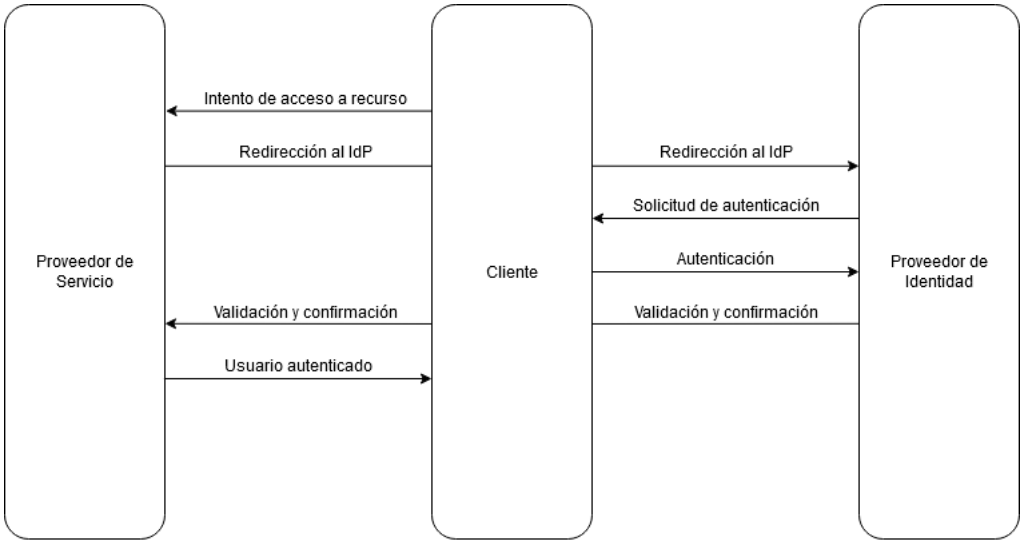

Figura 2.2 Flujo SAML.

2.3 OAuth 2.0

OAuth 2.0 es un protocolo de seguridad que permite el acceso a un servicio HTTP por parte de una aplicación externa, pudiendo ser en nombre el propietario del servicio u otorgando permiso a la aplicación para acceder por sí misma [6].

En el modelo de autenticación cliente-servidor tradicional, un cliente solicitaba el acceso a un recurso protegido por medio de autenticación, empleando credenciales proporcionadas por el propietario del recurso. Para permitir el acceso a distintas aplicaciones, el propietario tenía que compartir con todos ellos sus credenciales, generando varios problemas y limitaciones:

- Las aplicaciones debían almacenar las credenciales del propietario para poder acceder a los recursos, generalmente una contraseña en texto plano.
- Los servidores que almacenaban los recursos debían admitir la autenticación mediante contraseña.
- Las aplicaciones externas obtenían acceso general a los recursos, sin poder limitar la duración del acceso o restringirlo a recursos concretos.
- El propietario del recurso no podía revocar el acceso a una aplicación externa sin hacerlo a todas las aplicaciones a las que le había permitido el acceso.
- Los compromisos acordados con una aplicación se ligaban a una contraseña para el usuario y su acceso a todos los datos protegidos por la misma.

OAuth resuelve estos problemas por medio de una capa de autorización y la separación de las funciones de los clientes y los propietarios de los recursos. En este estándar el cliente solicita acceso a uno o varios recursos almacenados en un servidor y para su acceso se le otorgan una serie de credenciales distintas a las del propietario del recurso.

Además, en lugar de facilitar un usuario y una contraseña para el acceso a la aplicación, se genera un token de acceso, que es una cadena de caracteres encriptados y que indica un alcance o permisos concretos, una duración de validez y puede contener otros atributos para controlar el acceso. Este token es emitido por un servidor de autorización por medio de la aprobación del propietario del recurso al que se pretende acceder. Con este token, el cliente podrá acceder durante su periodo de validez a los recursos autorizados y alojados de forma protegida.

El proceso de obtención de un token y posterior acceso a un recurso se realiza por medio de una serie de pasos que se ilustran y definen a continuación:

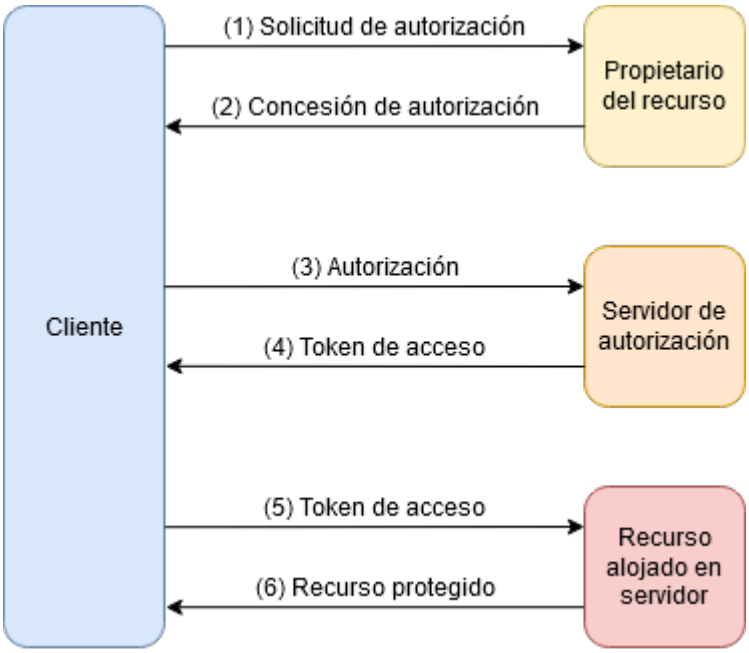

Figura 2.3 Flujo OAuth.

1. La aplicación externa solicita la autorización para acceder a un recurso concreto al propietario del recurso.
2. El propietario del recurso concede el permiso y lo registra en el servidor de autorización.
3. La aplicación accede al servidor de autorización con la autorización concedida anteriormente.
4. El servidor de autorización comprueba que la aplicación tiene permitido el acceso al recurso, genera un token y lo devuelve.
5. La aplicación solicita el acceso al servidor que aloja el recurso mediante el token.
6. El servidor comprueba la validez el token y devuelve a la aplicación el recurso solicitado

Para la limitación de acceso de una aplicación a sus usuarios, OAuth define los *scopes*. Los *scopes* son atributos que se seleccionan de forma explícita a la hora de solicitar un token y que pueden ser utilizados para controlar o limitar el acceso a los distintos usuarios [7].

OAuth no define los posibles valores o limitaciones de los *scopes* debido a que su uso dependerá de la arquitectura o necesidades del servicio o recurso. Ligado a la utilización de FHIR, se pueden utilizar para controlar el permiso de lectura o escritura de los recursos por parte de los usuarios, o el acceso a información restringida o de carácter sensible.

2.4 HL7

HL7 (Health Level Seven) es una organización internacional, iniciada en los Estados Unidos en 1987, cuyo objetivo es reducir la complejidad en el intercambio de datos sanitarios entre sistemas de información a través del desarrollo y promoción del estándar con su mismo nombre [1].

El estándar HL7, cuyo nombre hace referencia al nivel de aplicación (nivel 7) del modelo OSI de interconexión, se enfoca en la interoperabilidad para ámbitos sanitarios y define estructuras y tipos de datos para la comunicación entre distintos sistemas, permitiendo de este modo una comunicación correcta entre ellos independientemente de la arquitectura o composición de cada uno.

Este estándar cuenta con diferentes versiones: HL7 version 1 (HL7 v1), la primera versión del estándar y que asienta las bases, HL7 version 2 (HL7 v2), la cual presenta numerosas mejoras respecto a la primera y la cual es la más extendida en el mundo para la implementación de mensajería clínica, y HL7 version 3 (HL7 v3), enfocada en la semántica de la información.

Dicho estándar trabaja en torno a la definición de mensajes para el intercambio de información asociada a eventos y para ello define múltiples estructuras de mensaje con fines propios dependiendo del tipo de evento [8]. Existen numerosos mensajes definidos por HL7, los cuales se componen a su vez de segmentos que agrupan la información dependiendo de su naturaleza. A su vez, los distintos segmentos se componen de distintos campos que agrupan la información en el nivel más básico.

Un ejemplo para entender mejor la idea anterior: Cuando un paciente acude a un centro sanitario o es ingresado en un hospital, un evento que se realiza es la admisión del mismo, para lo cual se define la estructura del mensaje ADT. Dicha admisión podrá contener la información nominal del paciente, la fecha de ingreso, observaciones, etc. para lo cual se definen los distintos segmentos propios del mensaje (MSH para la información de control y cabecera, EVN para la información del evento, PID para la información nominal, etc.). Cada uno de estos segmentos a su vez contendrán distintos campos que estructuran toda la información.

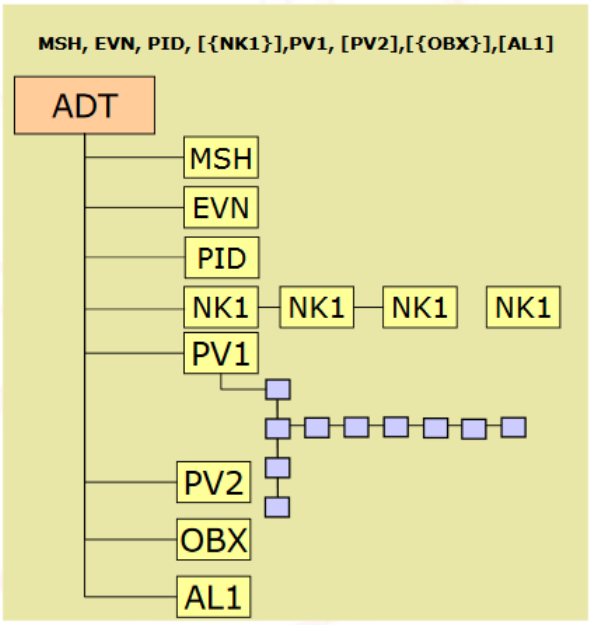

Figura 2.4 Composición mensaje ADT en HL7.

En las primeras versiones el estructurado de los mensajes se realizaba mediante el uso de delimitadores, entre los cuales se empleaba el carácter pipe "|" a partir del cual se extendió el uso de HL7 pipe para denominar al formato de estos mensajes. A partir de la versión v2.3.1 se definen pautas para definir las estructuras de los mensajes empleando codificación XML [9].

Código 2.2 Mensaje HL7 estándar.

```
MSH|^~\&|NSI||LAB||20010827120759||ADT^A01|NSI1|P|2.3||||AL
EVN|A01|18000101000000
PID|1||60719^^^^HI|26690949^^^^DNI|TORRALBA^AIDA^LIDIA||19780113|F|||POTOSI
    4032 108^^CAPITAL FEDERAL^^1899
NK1|1|CAMUS^ALBERTO|PAD|RIVADAVIA 253|42539686
PV1|1|I|301|R|||1436^PEREZ^JORGE^ALBERTO|1026^LOPEZ^NORBERTO|998^GARCIA^
    ALEJANDRO|M|||A|4|A0|N|1026^LOPEZ^NORBERTO|OB|H0100240|||||||||||||||ALV
    ||||||||20010823095130|20010823102455
IN1|1|INT^^HI|2^^^^HI~347^^^^NSI|PLAN DE SALUD
```

Código 2.3 Mensaje HL7 con codificación XML.

```
<ACK xmlns="urn:hl7-org:v2xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:schemaLocation="urn:hl7-org:v2xml ACK.xsd">
  <MSH>
    <MSH.1>|</MSH.1>
    <MSH.2>^~\&amp;</MSH.2>
    <MSH.3>
      <HD.1>LAB</HD.1>
    </MSH.3>
    <MSH.4>
      <HD.1>767543</HD.1>
    </MSH.4>
    <MSH.5>
      <HD.1>ADT</HD.1>
    </MSH.5>
    <MSH.6>
      <HD.1>767543</HD.1>
    </MSH.6>
    <MSH.7>
      <TS.1>199003141304-0500</TS.1>
    </MSH.7>
    <MSH.9>
      <MSG.1>ACK</MSG.1>
      <MSG.3>ACK</MSG.3>
    </MSH.9>
    <MSH.10>XX3657</MSH.10>
    <MSH.11>
      <PT.1>P</PT.1>
    </MSH.11>
    <MSH.12>
      <VID.1>2.4</VID.1>
    </MSH.12>
  </MSH>
  <MSA>
    <MSA.1>AR</MSA.1>
    <MSA.2>ZZ9380</MSA.2>
  </MSA>
  <ERR>
    <ERR.1>
      <ELD.1>PID</ELD.1>
      <ELD.2>1</ELD.2>
      <ELD.3>16</ELD.3>
      <ELD.4>
```

```
        <CE.1>103</CE.1>
        <CE.2>Table value not found</CE.2>
        <CE.3>HL70357</CE.3>
      </ELD.4>
    </ERR.1>
  </ERR>
</ACK>
```

2.5 REST

Transferencia de estado representacional (Representational State Transfer) o REST es un enfoque de desarrollo de la arquitectura orientada a servicios. Es una alternativa al protocolo SOAP (Simple Object Access Protocol) para el intercambio de datos, que define una interfaz de comunicación HTTP entre sistemas mediante operaciones.

REST trabaja con recursos, que son los elementos que almacenan la información y que son publicados de manera que son accesibles mediante peticiones HTTP con distintas operaciones para su intercambio y modificación de manera sencilla.

Sus principales características son:

- **Protocolo cliente/servidor sin estados:** Las peticiones HTTP que se realizan, contienen toda la información fundamental para realizar las operaciones, de modo que ni el cliente ni el servidor tengan que almacenar estados de operaciones previas.

- **Operaciones** REST hace uso de las operaciones HTTP para trabajar con los datos. Estas operaciones son: *POST* (crear), *GET* (leer), *PUT* (actualizar) y *DELETE* (eliminar)

- **Interfaz única/universal:** Los recursos REST se identifican de forma unívoca por una URI, lo que permite acceder a ellos y realizar operaciones de forma sencilla.

- **Hipermedia:** es un concepto que surge de la unión entre hipertexto y multimedia. Las aplicaciones que emplean REST suelen hacer uso de representación HTML, XML o JSON que permite moverse entre distintos recursos mediante el uso de referencias, sin necesidad de una interfaz propia.

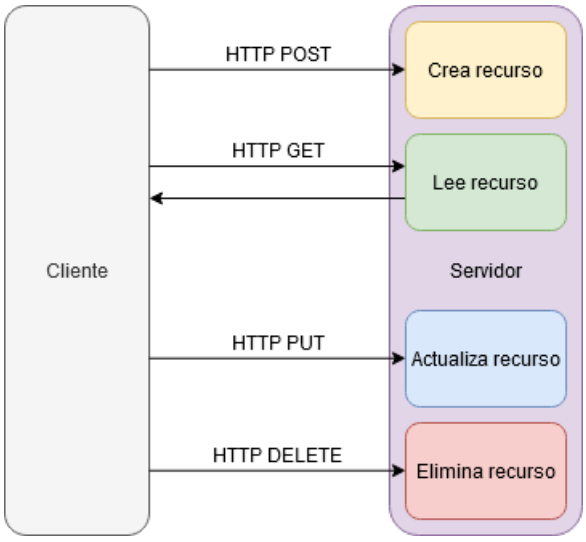

Figura 2.5 Operaciones REST sobre recursos.

2.6 HL7 FHIR

Como se mencionó en la introducción, el paradigma de las tecnologías ha propiciado la migración de servicios basados intercambio de mensajes mediante protocolo SOAP al uso de servicios REST.

Ante esto, HL7 define el estándar FHIR (Fast Healthcare Interoperability Resources), sobre el cual se desarrolla este trabajo. Dicho estándar se enfoca en el uso de recursos atomizados o granulares y diseñado para el intercambio mediante interacciones REST [2].

FHIR permite emplear los formatos de codificación XML y JSON, siendo este segundo el más extendido para este tipo de mensajería.

Código 2.4 Mensaje HL7 FHIR con codificación JSON.

```
{
  "resourceType": "Patient",
  "id": "1148350",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-05-05T08:02:48.100+00:00",
  },
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\"><div class=\"
        hapiHeaderText\">Aurelio227 <b>PFANNERSTILL264 </b></div><table class
        =\"hapiPropertyTable\"><tbody><tr><td>Identifier</td><td>999-52-3515</
        td></tr><tr><td>Address</td><td><span>Northbridge </span></div>"
  },
  "identifier": [{
    "system": "http://hl7.org/fhir/sid/us-ssn",
    "value": "999-52-3515"
  }, {
    "system": "http://standardhealthrecord.org/fhir/StructureDefinition/
        passportNumber"
  }],
  "name": [{
    "family": "Pfannerstill",
    "given": ["Aurelio"]
  }, {
    "use": "maiden"
  }],
  "gender": "male",
  "birthDate": "2002-07-23",
  "address": [{
    "city": "Northbridge",
  }],
  "maritalStatus": {
    "coding": [{
      "system": "http://terminology.hl7.org/CodeSystem/v3-MaritalStatus"
    }]
  },
  "multipleBirthBoolean": false
}
```

En el ejemplo anterior se muestra un mensaje empleando FHIR con codificación JSON de un recurso 'Patient'que recoge una serie de información asociada a un paciente y sobre la que se profundizará en los siguientes puntos.

2.6.1 Recursos FHIR

Como se ha indicado en el punto anterior, FHIR define un conjunto de recursos con características propias que permiten implementar soluciones que ayuden a agrupar y compartir la información [10].

Todos estos recursos pueden agrupar diversos conceptos, tal como se muestra en la siguiente tabla [11]:

Tabla 2.1 Tabla de recursos FHIR.

Concepto	Ejemplo	Recurso FHIR
Hallazgos clínicos		
Resultados de laboratorio	Panel de sangre, panel de hígado, etc.	*DiagnosticReport*
Estudio de imágenes	Radiografías, resonancias, etc	*DiagnosticReport*
...	...	...
Problemas del paciente, alergias y eventos adversos		
Alergia	Alergias a alimentos o medicamentos	*AllergyIntolerance*
Diagnostico clínico	Diabetes, insuficiencia cardíaca congestiva..	*Condition*
...	...	...
Historial del paciente		
Dolencias	Tos, dolor, fiebre, fatiga..	*Condition*
Historial clínico	Diabetes, insuficiencia cardíaca congestiva..	*Condition*
...	...	...

Para cada uno de estos recursos se define una estructura propia con una serie de atributos, límites, relaciones y alcance de uso propio que permiten acotar y distinguir de forma clara el uso de cada uno de ellos, facilitando de este modo su interpretación e implementación.

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal receiving health care services Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
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 LinkType (Required)

Figura 2.6 Estructura recurso Patient.

Para construir los distintos recursos se definen una serie de atributos o tipos de datos que podrán ser únicos o comunes a estos para poder recoger la información de manera estructurada [12]. A continuación, se muestran los distintos tipos de datos definidos:

- Simples/Primitivos: son únicos y recogen la información en su mínima expresión.

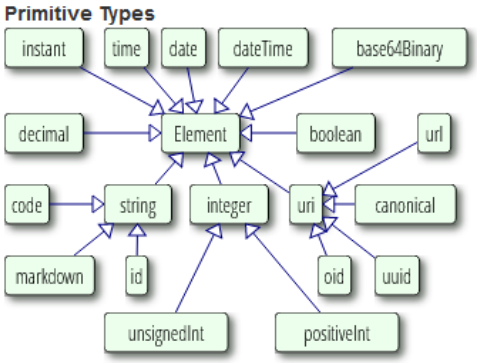

Figura 2.7 Datos simples/primitivos.

- Complejos: agrupan varios datos simples/primitivos otorgándoles significado propio.

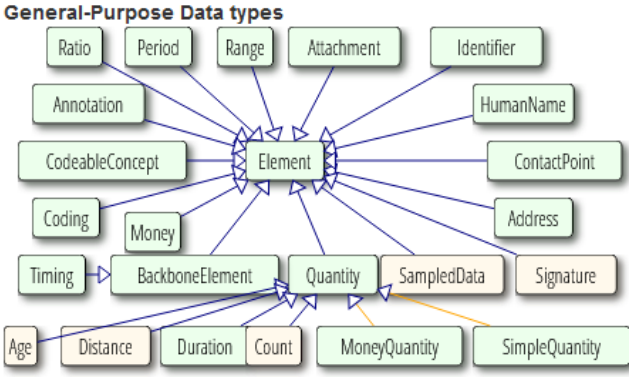

Figura 2.8 Datos complejos.

- Metadatos: tipos de datos que recogen diversa información de metadatos.

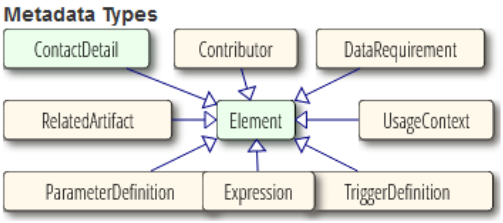

Figura 2.9 Datos de metadatos.

- Propósitos especiales: se definen con propósitos específicos.

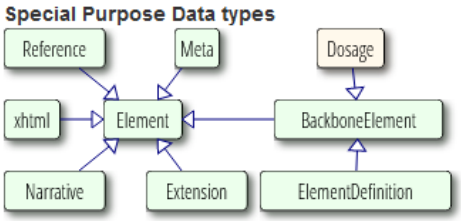

Figura 2.10 Datos de propósitos especiales.

2.6.2 Seguridad en FHIR

FHIR no es un protocolo de seguridad ni define funcionalidades en este aspecto pero si define una serie de recomendaciones y directrices a la hora de implementar este estándar de forma segura [13]. Entre estas recomendaciones se recoge el uso del protocolo TLS (Transport Layer Security) para el intercambio seguro de la información en la capa de transporte, el uso de OAuth para la autenticación y una serie de directrices para realizar la autorización y control de acceso a la información.

Con respecto a la autorización, FHIR recomienda dos modelos de control de acceso, el modelo basado en roles, también conocido como RBAC por las siglas de Role-Based Access Control y el modelo basado en atributos, conocido como ABAC, cuyas siglas provienen de Attribute-Based Access Control.

El modelo RBAC basado en roles define una serie de permisos para realizar las distintas operaciones sobre los recursos. Estos permisos son asignados a los distintos roles existentes, y estos, a su vez se asocian a los distintos usuarios. En este sentido, FHIR indica la facilidad de emplear este modelo aplicando roles a cada tipo de operación básica permitida (leer, escribir, actualizar y borrar).

En el modelo ABAC basado en atributos, los usuarios solicitan el acceso para realizar operaciones determinadas y estos permisos se concederán o no dependiendo de una serie de políticas basadas en atributos del propio usuario, el recurso al que se accede o el entorno, definidas para limitar dichas operaciones. FHIR, al igual que en el modelo anterior, expresa la facilidad de implementar este modelo al poder definirse atributos específicos para cada tipo de recurso desarrollado en el sistema, permitiendo una relación uno a uno entre atributo/permiso y recurso.

Definir todas estas reglas de decisión de control de acceso es una tarea compleja que dependerá de la información que pueda ser obtenida de los distintos elementos participantes en el sistema:

- Cliente: identidad del mismo, roles que le aplican, ubicación desde la que intenta acceder o nivel de garantía.
- Recurso: grado de confidencialidad o sensibilidad, tipo de dato o información que almacena, fechas o autor de los datos recogidos.
- Paciente: identidad del paciente, relación paciente y cliente (puede ser paciente y su correspondiente médico o enfermero asociado), política de consentimiento.
- Contexto: identidad del sistema que accede a la información, hora del día en la que se efectúa el acceso, propósito del mismo, estado del flujo de la transacción o protocolo de transporte seguro empleado.

Por último, en su documentación, FHIR contempla el uso de etiquetas para proporcionar metadatos de seguridad específicos de los recursos, permitiendo de este modo aplicar decisiones de control de acceso de manera individual a cada recurso que se consulta [14].

FHIR recoge varios tipos de etiquetas que pueden ser utilizadas dependiendo de la necesidad:

Tabla 2.2 Tabla de etiquetas FHIR.

Tipología	Etiquetas	Descripción
Contexto	Propósito de uso [15]	Estas etiquetas recogen el propósito de uso del recurso.
Sensibilidad de los datos	Código de confidencialidad [16]	Describen la sensibilidad de la información contenida en el recurso.
Control de flujo	Eliminar después de usar	Informa que el recurso debe ser eliminado después de ser utilizado.
	No reutilizar	El recurso debe ser utilizado en el momento y no está permitido su reenvío.
	Datos de prueba	Indica que los datos asociados al recurso son para uso experimental.

Estas etiquetas por si solas no aplican ningún tipo de seguridad o control de acceso a la información, pero permiten la aplicación de políticas o lógicas de negocio que permitan restringir su uso. Estas reglas aplicadas a las etiquetas deben recoger:

- Qué etiquetas se pueden utilizar.
- Cómo actuar en caso que un recurso contenga una etiqueta de seguridad no acordada.
- Obligaciones sobre el uso de estas etiquetas
- Implicaciones que conlleva el uso de etiquetas

2.7 International Patient Summary

International Patient Summary o IPS es una norma con origen en Europa [17], cuyo fin es el intercambio de datos esenciales de salud entre países de manera sencilla. HL7 ofrece una guía de implementación para recoger un resumen de la historia clínica de un paciente, de forma mínima y no exhaustiva e independiente de especialidades y condiciones, que un médico puede utilizar para dar atención médica de forma no programada [3].

Un resumen de historia clínica es un conjunto estandarizado de datos que permite recoger los datos más relevantes de la historia clínica de un paciente y que son necesarios para poder asegurar una atención médica segura.

Entre estos datos se incluye:

- Información general del paciente (nombre, fecha de nacimiento, sexo, etc.)
- Resumen médico con los datos clínicos más relevantes del paciente (alergias e intolerancias, enfermedades, implantes quirúrgicos, etc.)
- Listado de tratamientos o prescripciones en vigor del paciente.
- Otra serie de datos a modo de control o seguridad (fecha de generación del documento, persona que generó el documento, etc.)

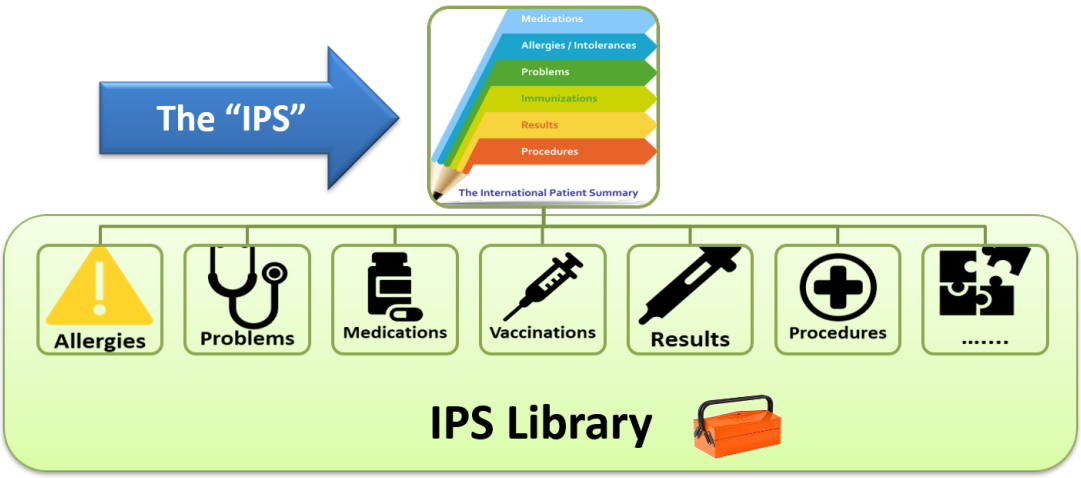

Figura 2.11 Esquema de datos IPS.

El proyecto IPS recoge y detalla los datos clínicos y el vocabulario necesario para poder generar este documento de forma estandarizada, permitiendo ser utilizado para dar atención médica tanto planificada como no planificada o de emergencia y no solo en la región o país de procedencia del paciente, si no de forma general por cualquier medico cualificado en cualquier parte del mundo.

Inicialmente el proyecto se desarrolló basado en el estándar de documentos clínicos CDA R2 pero se ha extendido y en la actualidad también se ofrece soporte para desarrollar dicho documento empleando el estándar HL7 FHIR.

2.7.1 IPS FHIR

Como se ha expuesto anteriormente, FHIR define una serie de recursos que permiten intercambiar información clínica de manera estructurada y sencilla. Si se observa la figura 2.11, los distintos elementos que componen un documento IPS se corresponden con recursos definidos en FHIR. Si bien, existen ligeras diferencias en cuanto a la información contenida en un recurso FHIR al uso y un recurso FHIR enfocado al IPS, afectan únicamente a los datos almacenados y no a la manera de procesarlos, por lo que no se han tomado en consideración en este trabajo.

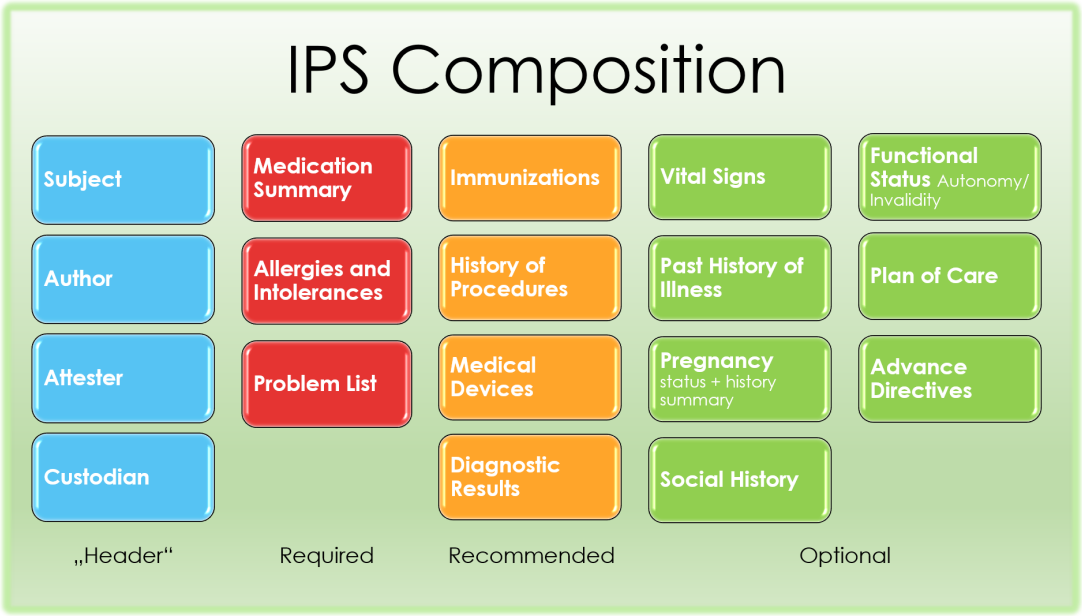

Figura 2.12 Recursos IPS FHIR.

Estos recursos, en conjunto, permiten componer un documento de resumen de historia clínica resumida. Para ello, se establece cuáles de estos recursos son imprescindibles para poder dar una atención mínima a un paciente, cuales recomendados y cuales opcionales:

- Imprescindibles: Medicaciones, alergias e intolerancias y enfermedades.
- Recomendados: Vacunaciones, historial de operaciones, implantes y dispositivos médicos y diagnósticos realizados.
- Opcionales: Histórico de enfermedades, embarazos, plan de cuidados, etc.

Junto con esta información, se deberá incluir un recurso *Composition* que contenga información general que permita identificar al paciente, el autor del documento o el dispositivo que lo genera, la organización responsable de dicho documento y un firmante o responsable del mismo.

3 Marco Práctico

Una vez que se han definido los objetivos del trabajo y habiendo desarrollado de manera teórica los distintos elementos de estudio sobre los que se apoya, se va a abordar de forma práctica. Para ello, en primer lugar, se hará una breve introducción a las distintas herramientas utilizadas y que permiten desarrollar un servicio de control de acceso, objetivo principal de este trabajo y, en segundo lugar, se desarrollarán varios casos prácticos que mostrarán el funcionamiento de los mecanismos que se pueden emplear en dicho servicio para conseguir una seguridad robusta.

3.1 Componentes

3.1.1 WSO2

WSO2 es una compañía que ofrece una plataforma *Open Source* con distintas aplicaciones bajo un modelo de arquitectura orientada a servicios (SOA) y con licencia Apache que permiten implementar aplicaciones y servicios de manera profesional.

Entre los distintos productos que WSO2 ofrece, en este trabajo se destacan el servidor de identidad (WSO2 Identity Server) y la plataforma de integración (WSO2 Enterprise Integrator) que se detallarán en los subsiguientes puntos y que se emplearán para construir los escenarios prácticos de este trabajo.

3.1.1.1 WSO2 Identity Server

WSO2 Identity Server (IS) es la herramienta de la suite de productos de WSO2 que posibilita la gestión y administración de identidades, privacidad y seguridad en un sistema. Está basado en distintos estándares abiertos como SAML, OAuth o OIDC y con la posibilidad de realizar implementación local, en la nube o híbrida [18].

Entre sus características principales se encuentran:

- **Autenticación:** El servidor de identidad permite la autenticación de usuarios incorporando su propia base de almacenamiento contra la que comprobar las credenciales de los usuarios o mediante la conexión con servidores de usuarios y bases de datos externas.
- **Single Sign On:** WSO2 Identity Server ofrece la capacidad de Single Sign On (SSO) entre sus características, permitiendo a los usuarios obtener acceso a múltiples aplicaciones proporcionando una única vez sus credenciales.
- **Autorización:** El IS ofrece un servicio de control de acceso que permite gestionar los permisos o autorizaciones de los usuarios a los distintos recursos gestionados. Para ello implementa las características del estándar XACML ya definido anteriormente.
- **Federación de identidad:** mecanismo que permite la autenticación de diferentes empresas en diferentes dominios de confianza, permitiendo a los usuarios disponer de las mismas credenciales para acceder a distintas aplicaciones. Esta característica junto al SSO permite un acceso e integración sencilla con distintas aplicaciones.

- **Gestión mediante API:** los distintos productos de WSO2 ofrecen una serie de servicios de administración en forma de API que son consumidos por la interfaz de administración de usuario pero que también se ofrecen de forma que se puedan consumir directamente. En el caso de IS permite la gestión de identidades y accesos de manera directa, característica que será empleada en los escenarios de prueba.

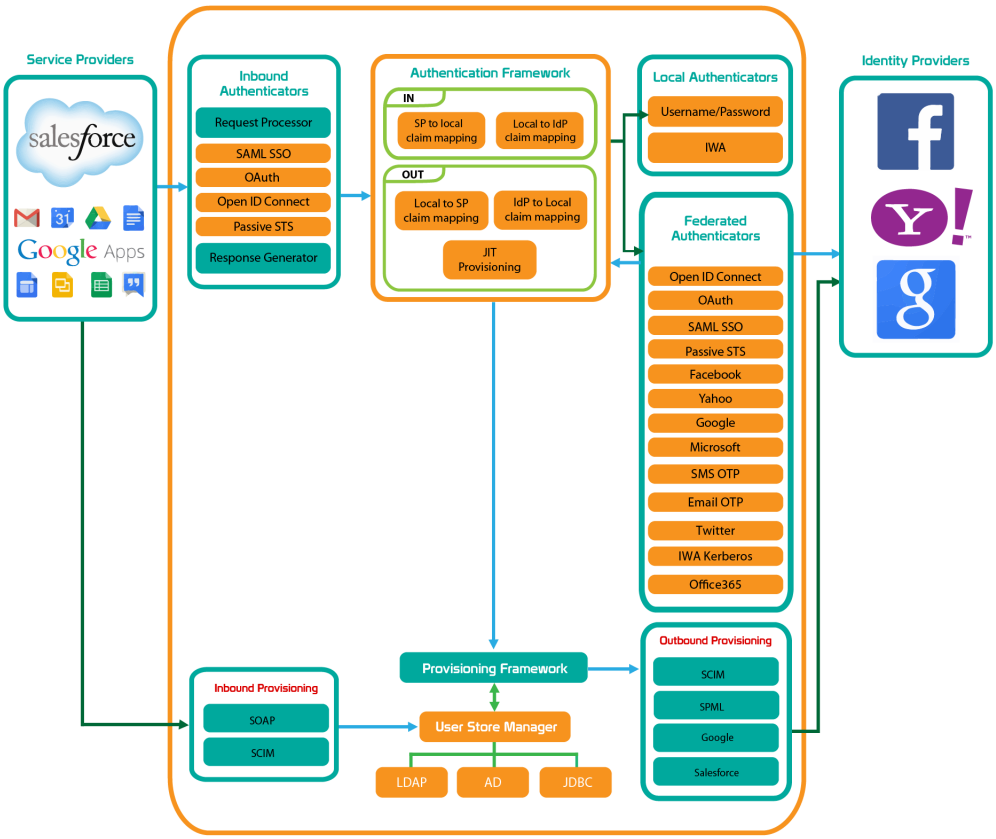

Figura 3.1 Arquitectura Identity Server.

Como se puede ver en la figura anterior, la arquitectura del IS [19] está compuesta de múltiples elementos que se detallan a continuación:

- **Service Providers:** Los proveedores de servicio (SP) son las entidades que proporcionan los servicios web. Dependen de un proveedor de identidad (IdP) para realizar la autenticación y autorización. El WSO2 Identity Server será el proveedor de identidad que se encargará de autenticar y autorizar a los usuarios del proveedor de servicio.

- **Inbound Authenticators:** Componente que realiza la autenticación recibida en el Identity Server. Trabaja con distintos tipos de solicitudes, entre las que se encuentra OAuth que es empleada en este trabajo.

- **Authentication Framework:** El marco de autenticación se encarga de la gestión de los *claims* o atributos, estableciendo la relación entre los *claims* del IS y los de los distintos SP. Además, incluye provisionamiento *"Just-in-Time"* (JiT) que permite crear usuarios en el momento que realizan el primer inicio de sesión.

- **Local Authenticators:** Los autenticadores locales son los procesos de autenticación existentes en el IS. Estos autenticadores se encargan de verificar la información de usuario y contraseña de acceso en el proceso de autenticación, contrastando con la información almacenada en el servidor.

- **Federated Authenticators:** Además de los autenticadores locales, en el IS se pueden configurar autenticadores externos, de modo que al recibir una solicitud, el Identity Server envíe la solicitud a estos autenticadores, que realizan la autenticación y devuelven la respuesta, autenticando al usuario o denegándole el acceso.
- **Identity Providers:** Los proveedores de identidad (IdP) se encargan de realizar la autenticación. Además del propio IS que actúa como proveedor de identidad pueden configurarse otros proveedores de identidad externos como Facebook o Google.
- **Provisioning Manager:** El marco de provisionamiento se encarga de todas las labores de aprovisionamiento realizadas en el Identity Server. Se integra con el gestor de usuarios y puede recibir peticiones de provisionamiento desde el *authentication framework*.
- **Authorization Manager:** El gerente de autorización se encarga de realizar la autorización y auditoría en el IS. Gestiona las autorizaciones de todas las llamadas REST o SOAP recibidas. El gestor soporta autorización basada en XACML, WS-Trust, OpenID Connect y realiza la gestión de *claims*. Además, el IS implementa una interfaz sencilla para el usuario para la implementación de políticas, así como permitir la existencia de múltiples PIPs y PDPs. También aporta un protocolo de alto rendimiento para la interacción entre PEP y PDP.
- **IdP/SP configurations:** La configuración del proveedor de servicio y del proveedor de identidad establece la base de funcionamiento de los mecanismos de autenticación y autorización del IS.
- **Indbound Provisioning:** Es el componente del IS que soporta las peticiones entrantes de provisionamiento. Permite la creación, modificación o eliminación de cuentas e identidades de usuarios solicitados. Estas solicitudes pueden realizarse en formato SOAP o SCIM.
- **User Store Manager:** El Gestor de usuarios permite al Identity Server implementar distintos sistemas de almacenamiento, ya sea por medio de su servidor LDAP interno o mediante conexión a sistemas externos por medio de conectores.
- **Claim Manager:** El gestor de *claims* permite gestionar los atributos que identifican a los usuarios almacenados en el Identity Server. Los *claims* son los atributos o características asociadas a los usuarios y pueden ser utilizados para proveer la identidad de dichos usuarios y permitir la autorización, transportándolos en tokens.
- **XACML:** XACML, como ya ha sido definido en su apartado correspondiente es un lenguaje de control de acceso basado en XML y estandarizado por OASIS que permite la definición de políticas de acceso.
- **Auditing:** Por medio del gerente de autorización, WSO2 Identity Server permite realizar auditoría de operaciones mediante un sistema de auditoría distribuida. A su vez permite monitorizar y realizar estadísticas de uso y rendimiento del servidor, así como la supervisión de sesiones y estadísticas de autenticación.
- **Identity Manager:** El gestor de identidad permite al IS adaptarse a los cambios que se pueden producir en un sistema TIC satisfaciendo los requisitos de seguridad.
- **Outbound Provisioning:** Componente que permite al IS emitir solicitudes de provisionamiento a aplicaciones externas.

3.1.1.2 WSO2 Enterprise Integrator

Es una plataforma híbrida y centrada en el uso de API pero que además admite otros tipos de arquitectura: arquitectura de microservicios, arquitectura de ESB centralizada o arquitectura basada en la nube. Esta plataforma ofrece todas las herramientas necesarias para poder realizar integraciones de estos tipos de arquitectura de una manera bastante sencilla con una avanzada interfaz gráfica [20].

Este producto es el resultado de la unificación de varias herramientas desarrolladas por WSO2 e integradas de forma conjunta ofreciendo:

- Soporte para la comunicación para distintos tipos de protocolos, servicios y sistemas.
- Microservicios en la nube y con contenedores.

- Conexión con cualquier sistema de almacén y base de datos.
- Enrutamiento, mediación y transformación de datos.
- Alto rendimiento, disponibilidad, escalabilidad y estabilidad.
- Implementación ligera y sencilla.
- Capacidad de monitorización y análisis.

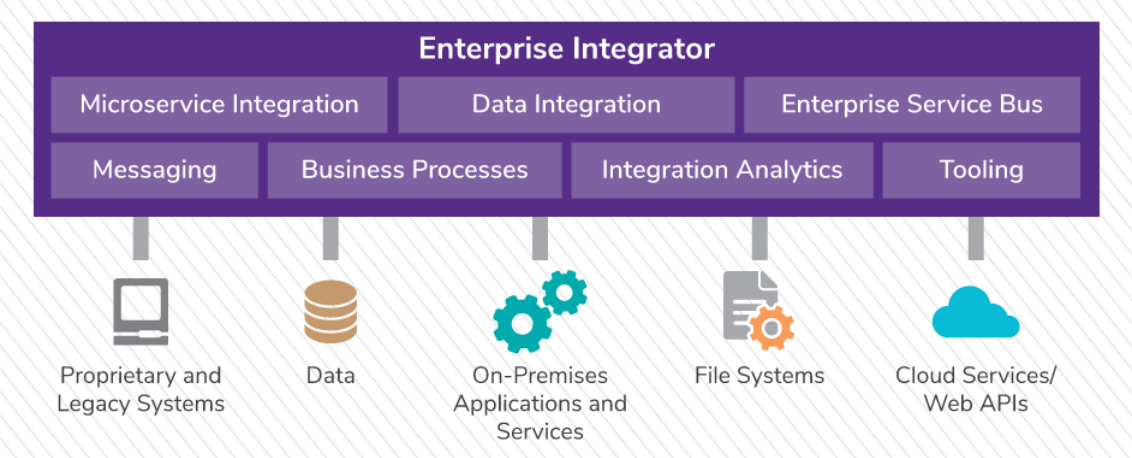

Figura 3.2 Esquema Enterprise Integrator.

En el momento de desarrollo de este trabajo, se encuentra en la versión 7.1 y se centra en las siguientes características:

- **ESB Centralizado:** El centro del EI 7.1 es el *Micro Integrator*, motor de la mensajería y basado en estándar que aporta las funciones de un ESB o bus. El ESB permite el enrutamiento, transformación y otras operaciones con los mensajes. En un ecosistema centralizado se puede emplear como capa de integración o fachada para implementar la lógica de mediación y conexión de los distintos sistemas.
- **Microservicios:** El EI está diseñado de manera ligera y pensado en el uso de contenedores y aplicaciones en la nube.
- **Integración sin código:** como se detallará en el siguiente apartado, WSO2 ofrece la herramienta WSO2 Integration Studio para el desarrollo mediante funcionalidad de arrastrar y soltar, sin necesidad de tener que desarrollar código.
- **Administración:** *Micro Integrator* ofrece un panel y una interfaz de comandos (CLI) pensada en la monitorización y análisis.
- **Streaming Integrator:** *Streaming Integrator* es un elemento ligero y basado en la nube que permite trabajar con transmisiones de datos y eventos en tiempo real.

3.1.1.3 WSO2 Integration Studio

Para facilitar la integración con el Enterprise Integrator, WSO2 también ofrece una herramienta de desarrollo enfocada en una interfaz gráfica pensada en la construcción de aplicaciones y servicios mediante un método *Drag and drop* y sencillas configuraciones pero que también permite el desarrollo mediante código XML y la integración de clases propias desarrolladas en Java u otros lenguajes.

Integration Studio permite crear recursos RESTful y definir su comportamiento interno arrastrando una serie de elementos denominados *mediadores*, unidades de procesamiento individuales con funciones específicas para el procesado de mensajes y orientados a servicios REST.

A continuación, se enumeran distintos mediadores que son de gran utilidad y han sido empleados en el desarrollo de este trabajo:

- **Call:** se utiliza para enviar mensajes desde el Integrator a un destino . Puede invocar servicios de manera síncrona o asíncrona.
- **Property:** permite extraer valores de un mensaje y almacenarlos en variables para un uso externo del mensaje.
- **Log:** permite generar mensajes de información que sean de utilidad para el desempeño de la aplicación.
- **Response:** se emplea a la hora de finalizar el procesado de un mensaje, devolviendo una respuesta al origen.
- **Header:** genera, modifica o elimina una cabecera SOAP o HTTP.
- **PayloadFactory:** permite trabajar sobre el contenido de un mensaje, reemplazando el original o modificándolo. Muy útil dentro de una orquestación que tiene que hacer múltiples llamadas.
- **OAuth:** permite comunicarse con un servidor de identidad y realizar validación de tokens OAuth para su autorización.
- **Filter:** se emplea para realizar filtrados en un flujo mediante comprobaciones sencillas. Como se mostrará más adelante, permitirá evaluar una respuesta a una consulta previa que dará lugar a varias posibles respuestas al origen.

Además de estos mediadores, otros elementos fundamentales son los denominados **Endpoints**, que son empleados para definir destinos a los que poder invocar y que, junto con el mediador **Call** se emplearán para llamar al servidor que almacenará los recursos FHIR.

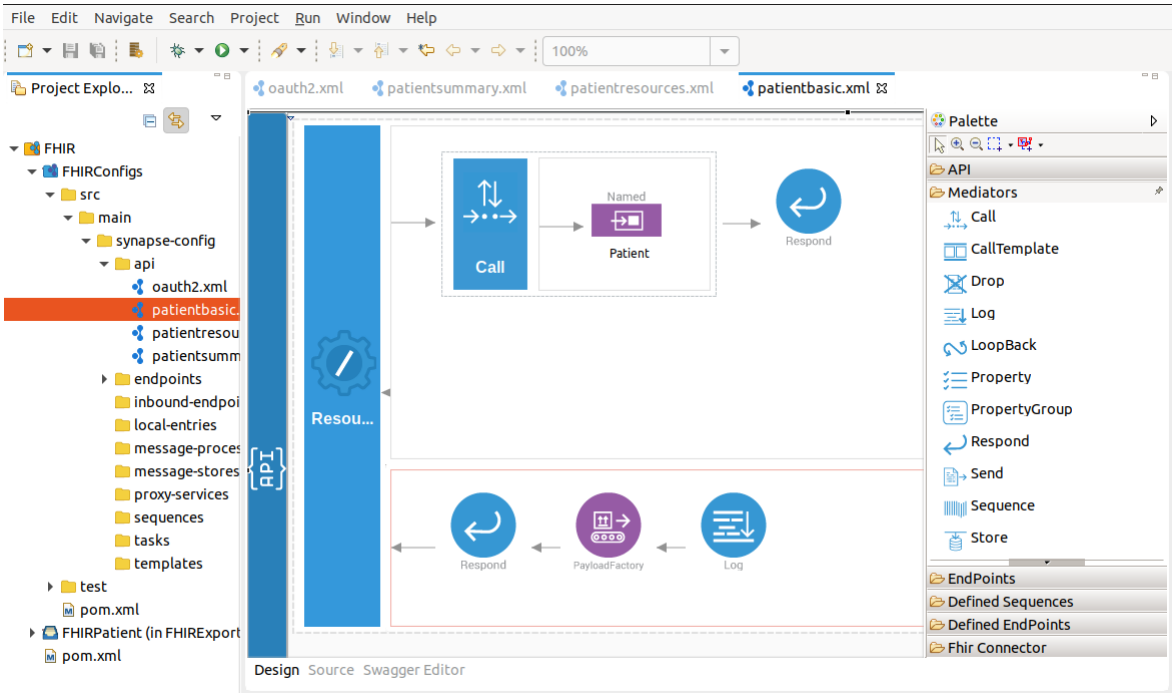

Figura 3.3 Interfaz Integration Studio con menú de mediadores.

3.1.2 Postman API Client

Postman es una plataforma colaborativa para el desarrollo API [21]. Esta plataforma ofrece una serie de herramientas entre las que se encuentra *Postman API Client*, un cliente API que permite generar y lanzar solicitudes SOAP, REST o GraphQL de manera sencilla.

Entre sus características destaca la posibilidad de realizar solicitudes complejas con distintos formatos, cabeceras e incluso transmitiendo datos de audio, vídeo, etc. Además, permite inspeccionar las respuestas

obtenidas con distintos formatos y, mostrando los códigos de respuesta y cabeceras, facilitando así la realización de pruebas y comprobaciones. Otros puntos a destacar son la integración con OAuth 2.0, usado en este trabajo, gestión de certificados SSL y uso de protocolo HTTPS para comunicación segura, o el uso de variables globales para facilitar su integración con diversos entornos [22].

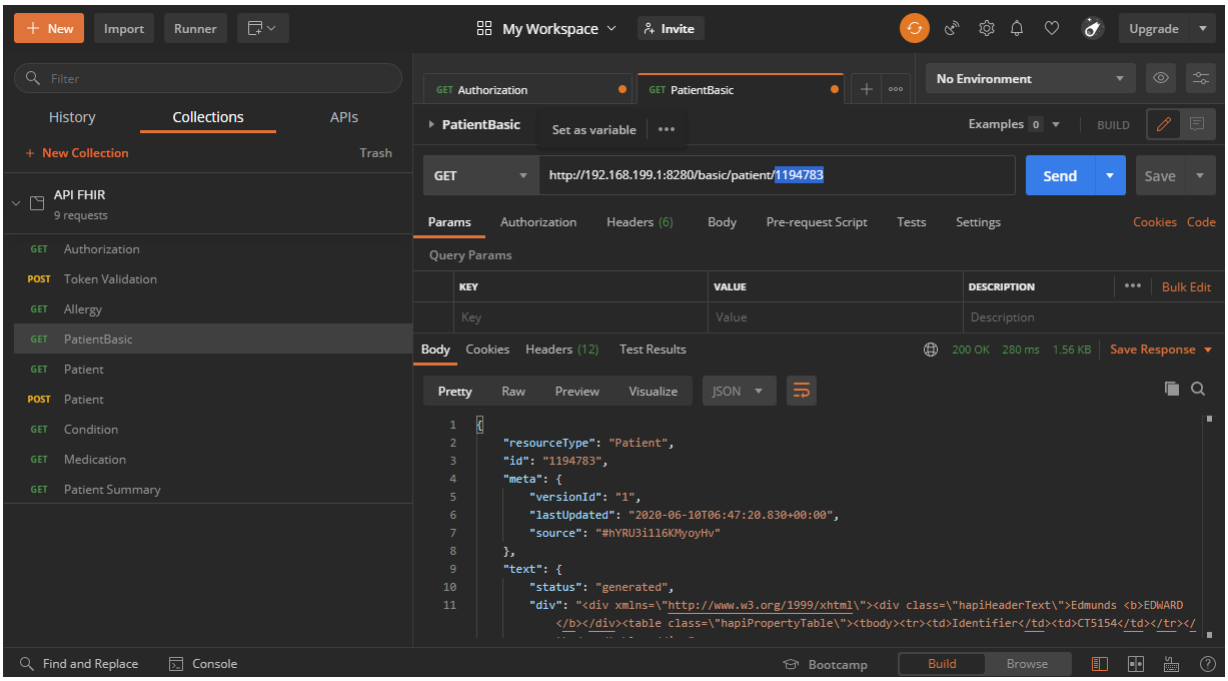

Figura 3.4 Interfaz Postman.

Esta herramienta será empleada para realizar las solicitudes en los distintos escenarios simulando a la aplicación cliente y evaluar las respuestas obtenidas, para comprobar el funcionamiento de los distintos casos de prueba.

3.1.3 HAPI FHIR Test Server

HAPI FHIR es un proyecto libre que implementa una librería HL7 FHIR para dar soporte a la interoperabilidad en el ámbito sanitario. Esta librería, desarrollada en Java y con licencia Apache, permite construir clientes y servidores basados en el estándar FHIR para su uso de manera sencilla [23].

Además de la librería, ofrece un servidor [24] de pruebas público y gratuito que permite la ejecución de pruebas y consultas y que se emplea en este trabajo para simular el servidor almacén de la información, donde se realizarán las consultas y desde donde se obtendrán los distintos recursos solicitados. Dicho servidor ofrece multitud de recursos generados por la comunidad, permitiendo consultarlos, modificarlos o generar nuevos, y permitiendo visualizarlos de manera sencilla con formato XML o JSON.

3.2 Escenarios

En esta sección se definen los distintos escenarios o casos prácticos elaborados durante el trabajo. Se partirá de un escenario básico sin seguridad al cual se irán añadiendo los distintos elementos que permitirán realizar el control de acceso desde un nivel básico a un nivel más complejo. Por último, se mostrará el escenario contemplado para poder componer el IPS FHIR definido anteriormente.

3.2.1 Escenario sin seguridad

Este primer escenario es la base de un sistema de información que permite consultar uno o más recursos FHIR.

Dicho escenario se compone de una aplicación cliente desde la cual se accederá a los distintos recursos FHIR definidos, permitiendo realizar las operaciones REST básicas (leer, escribir, actualizar y borrar).

El diagrama es el siguiente:

Figura 3.5 Escenario cliente-servidor sin seguridad.

Para la elaboración de dicho escenario se empleará Postman, que lanzará las consultas simulando a la aplicación cliente, WSO2 Enterprise Integrator que hará de fachada, publicando los distintos recursos en forma de API y a través de la cual se consultará el servidor de prueba de HAPI FHIR definido anteriormente y que almacena la información de los distintos recursos.

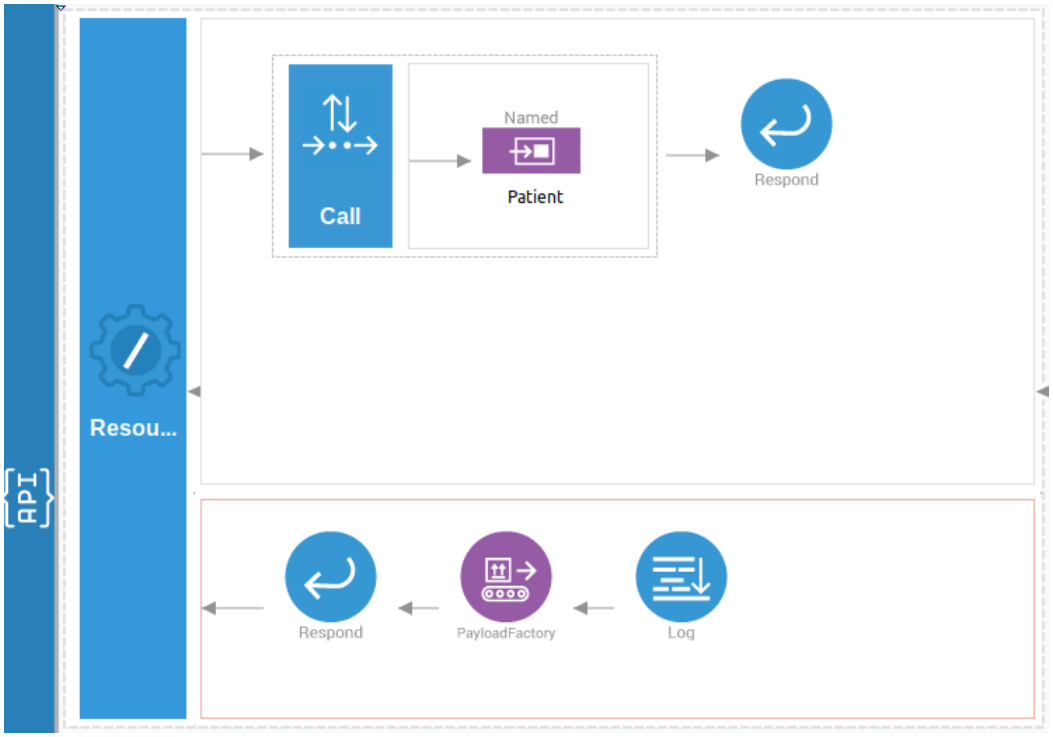

Figura 3.6 Recurso básico configurado con WSO2 Integration Studio.

En la figura 3.6 se puede observar un recurso FHIR configurado en Integration Studio. La aplicación presenta una API REST que expone múltiples recursos FHIR, como el mostrado. En este ejemplo sencillo, el recurso a consultar será del tipo *Patient* y se expone en la URL *http://192.168.199.1:8280/basic/patient/patientId*, donde el parámetro *patientId* recogerá un identificador único del paciente que se desee consultar, distinguiéndolo del resto. Una vez recibida la petición por el recurso de la API, se redirigirá la petición al servidor HAPI FHIR que almacena la información solicitada. Esto se realiza gracias al mediador *Call* que en su interior tiene configurado un componente *endpoint* apuntando a la URL *http://hapi.fhir.org/baseR4/Patient/uri.var.patientId/_history/1?_pretty=true&_format=json*, donde uri.var.patientId será el mismo identificador indicado en la solicitud original.

Además, se contempla una lógica de error de modo que en caso de producirse un error o excepción interna en la lógica del recurso, se devuelva al origen el error ocurrido.

Con este escenario se pretende mostrar cómo se realiza el acceso a los distintos recursos expuestos y visualizar un mensaje de respuesta en FHIR, permitiendo observar su formato y estructura.

3.2.1.1 Ejecución y resultados

La ejecución se realiza accediendo a la aplicación Postman, generando una nueva petición de tipo *GET* y apuntando a la URL del recurso expuesto en la API definida con Enterprise Integrator y que se ha indicado en la descripción del escenario.

Figura 3.7 Consulta recurso paciente sin seguridad en Postman.

Esto bastaría para poder realizar las consultas y ejecutar el escenario. Para generar la consulta, se pulsa sobre el botón *Send* de la aplicación, lo que lanzará la petición contra el recurso expuesto en la API y que a su vez redirigirá la petición contra el servidor, obteniendo los datos de la consulta realizada y devolviéndolos al origen, donde se mostrarán y podrán ser explotados.

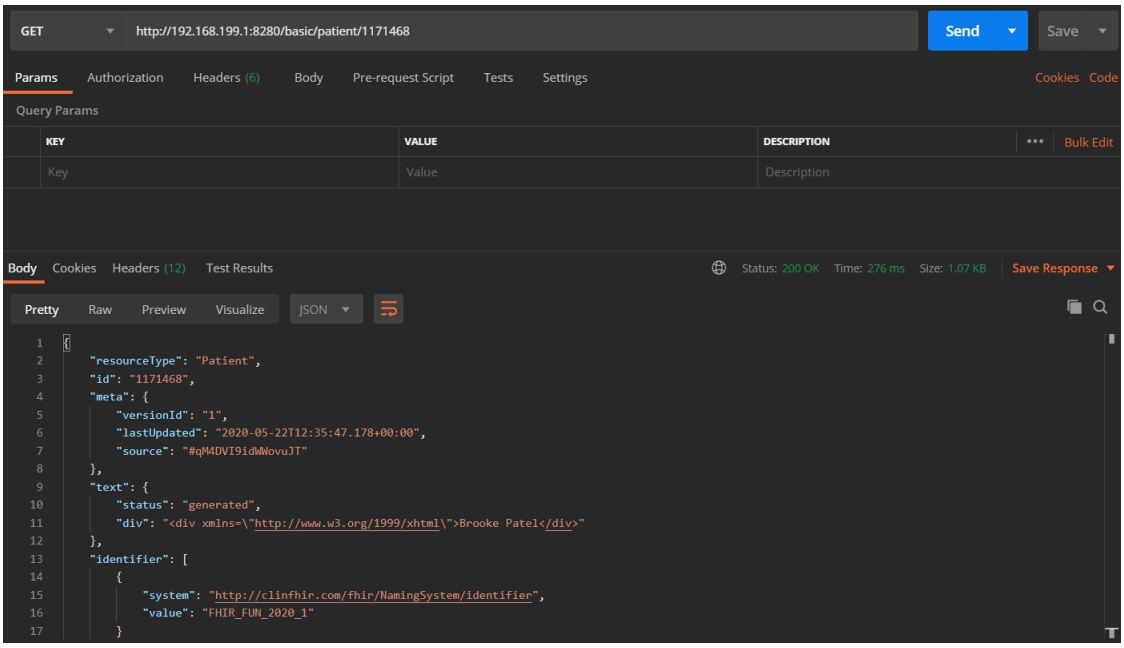

Figura 3.8 Resultado consulta paciente en escenario sin seguridad.

Como se observa en la figura 3.8, tras lanzar la consulta se recibe el recurso deseado donde se puede observar la estructura del recurso definida en FHIR y con la información almacenada sobre un paciente. La respuesta completa se muestra a continuación:

Código 3.1 Resultado esperado escenario básico.

```
{
    "resourceType": "Patient",
    "id": "1171468",
    "meta": {
        "versionId": "1",
        "lastUpdated": "2020-05-22T12:35:47.178+00:00",
```

```
        "source": "#qM4DVI9idWWovuJT"
    },
    "text": {
        "status": "generated",
        "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Brooke Patel</div>"
    },
    "identifier": [
        {
            "system": "http://clinfhir.com/fhir/NamingSystem/identifier",
            "value": "FHIR_FUN_2020_1"
        }
    ],
    "name": [
        {
            "use": "official",
            "text": "Brooke Patel",
            "family": "Patel",
            "given": [
                "Brooke"
            ]
        }
    ],
    "gender": "female",
    "birthDate": "2000-05-22"
}
```


En caso de error debido a consultar un paciente indicando un identificador inexistente o no almacenado en el servidor, se obtendría un mensaje de error indicando lo ocurrido, como se muestra a continuación:

Código 3.2 Resultado erroneo escenario básico.


```
{
    "resourceType": "OperationOutcome",
    "text": {
        "status": "generated",
        "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\"><h1>Operation
            Outcome</h1><table border=\"0\"><tr><td style=\"font-weight: bold
            ;\">ERROR</td><td>[]</td><td><pre>Resource Patient/11714682/_history
            /1 is not known</pre></td>\n\t\t\t\t\t\n\t\t\t\t\n\t\t\t</tr>\n\t\t
            </table>\n\t</div>"
    },
    "issue": [
        {
            "severity": "error",
            "code": "processing",
            "diagnostics": "Resource Patient/11714682/_history/1 is not known"
        }
    ]
}
```


3.2.2 Escenario con autenticación

A partir del escenario anterior, que contempla la llamada desde Postman a recursos expuestos por el servidor HAPI FHIR, se añade un primer elemento de seguridad. En ese caso se añadirá la autenticación, haciendo uso del componente WSO2 Identity Server como proveedor de identidad que alojará los distintos usuarios del sistema, un proveedor de servicio que empleará OAuth 2.0 y que controlará las solicitudes de acceso y

emisión de tokens JWT, y empleando también el componente WSO2 Enterprise Integrator que actuará como fachada de los recursos FHIR para realizar la validación de tokens y el resto de lógica que será utilizada en los subsiguientes escenarios.

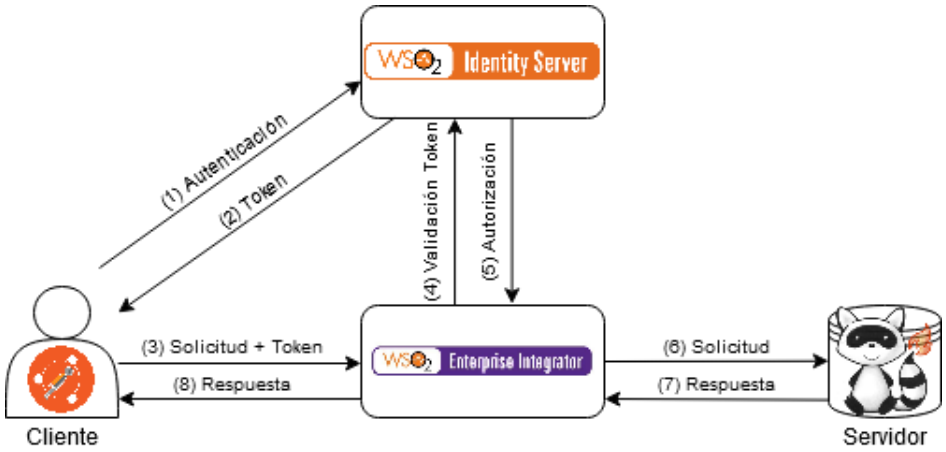

Figura 3.9 Escenario cliente-servidor con token.

En la figura 3.9 se puede observar el diagrama del escenario, en el que el primer paso para su ejecución será una invocación al Identity Server para realizar la autenticación y obtención de un token de acceso.

Este proceso de autenticación desde el punto de vista de la arquitectura del IS, consistirá en una solicitud enviada por un usuario y recibida en el *Inbound Authenticator*, el cual procesará la petición y la trasladará al *Authentication Framework* que comprobará dónde se debe realizar la autenticación y la remitirá al *Local Authenticator*, puesto que la autenticación se realiza en el propio IS y no por un componente externo. Una vez realizada la autenticación, comprobando las credenciales emitidas por el usuario, se genera una respuesta al origen, devolviendo el token correspondiente.

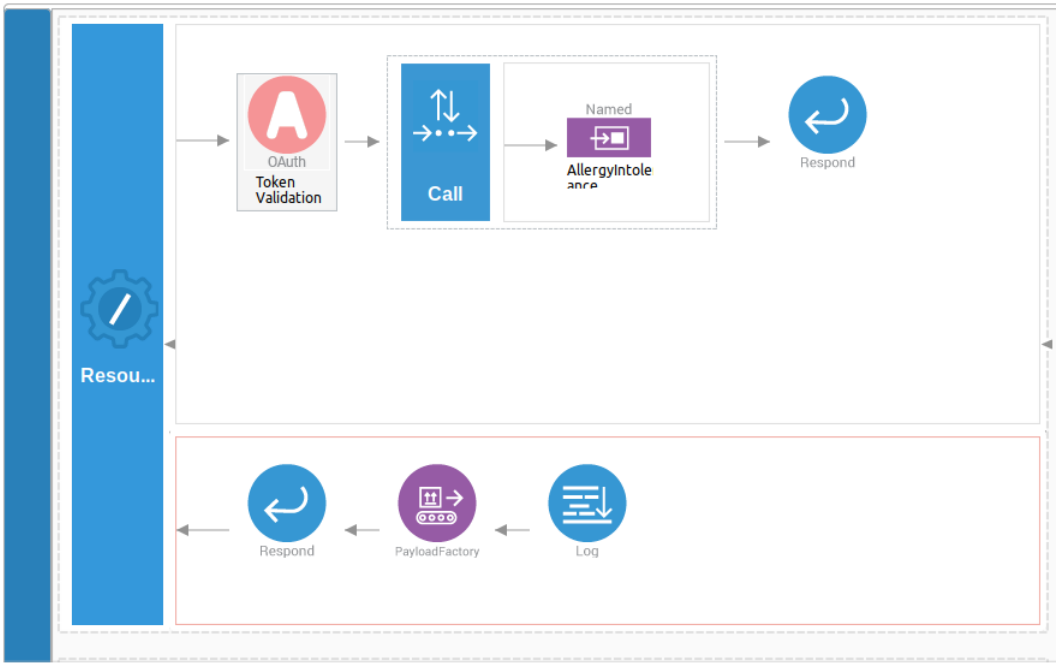

Figura 3.10 Escenario con validación de token configurado en Enterprise Integrator.

Una vez obtenido el token de acceso, se lanzará una solicitud de tipo *GET*, similar al escenario anterior pero añadiendo una cabecera *Authorization* que incluirá el token devuelto por el IS. Una vez recibida la solicitud en el Enterprise Integrator, en primer lugar, se tomará el token de acceso por el mediador *OAuth* y se invocará al IS para validar que el token recibido por parte del usuario es válido. Una vez validado por parte del IS, devolverá una respuesta al EI indicando el resultado, lo que provocará que se lance la consulta hacia el servidor HAPI FHIR para consultar el recurso solicitado y devolver la respuesta al usuario originario de la petición. En caso de que la validación no fuese satisfactoria (por ser un token incorrecto o caducado) no se lanzará la consulta hacia el servidor y se le comunicará un mensaje de error al usuario.

El proceso de validación de tokens por parte del IS se conoce como introspección, para lo cual el Identity Server publica un servicio que realiza esta comprobación y que puede ser consultado en la URL *https://localhost:9444/services/OAuth2TokenValidationService.OAuth2TokenValidationServiceHttpsSoap11Endpoint/*. Este proceso recibirá el token por parte del EI, comprueba que el token es válido y está activo, además de poder aplicar políticas de control de acceso como se verá en el siguiente escenario y devolverá una respuesta.

Este proceso dentro del Enterprise Integrator se ilustra en la figura 3.10.

3.2.2.1 Ejecución y resultados

Para la ejecución de este escenario, en primer lugar, será necesario solicitar un token para un usuario. Mediante Postman, se accede a la petición *Authorization* previamente configurada (ver anexo D) y se pulsará sobre el botón *Get New Access Token*. Si se encuentra bien configurada la petición y el IS, se deberá abrir una ventana en la que se deberá indicar las credenciales de un usuario, en este caso será *Fulanito*. Una vez indicadas, se deberá pulsar el botón *Continue*, que enviará la solicitud al IS, realizando el proceso descrito anteriormente.

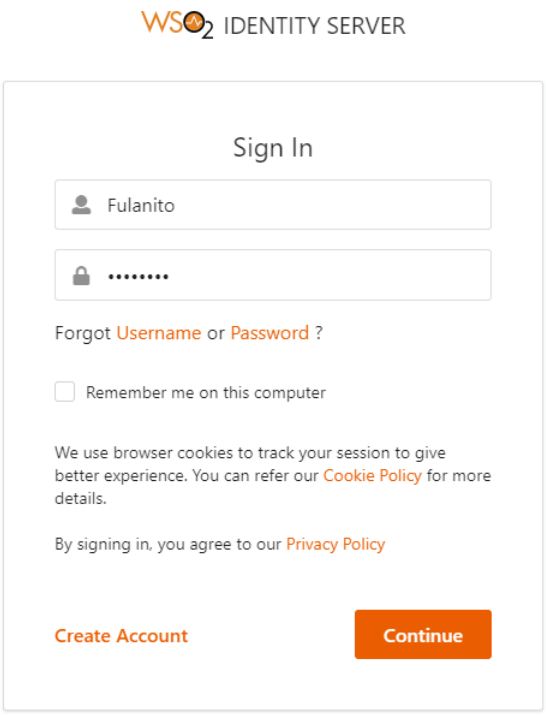

Figura 3.11 Proceso de autenticación y obtención de token I.

Una vez realizado el proceso de autenticación, se devolverá el token de acceso con la duración que se haya establecido previamente en el proveedor de servicio.

Este token deberá ser utilizado a continuación para solicitar el recurso deseado y publicado por el EI. En esta ocasión se consultará un recurso *AllergyIntolerance* que recoge las alergias e intolerancias de un paciente y que se encuentra publicado en la URL *http://192.168.199.1:8280/basic/patient/patientId/allergyintolerance*.

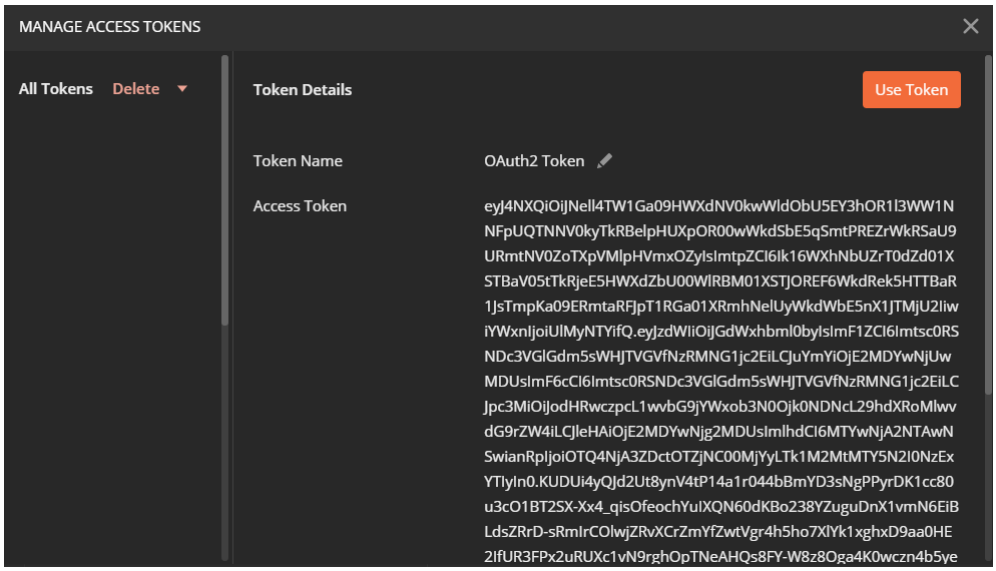

Figura 3.12 Token devuelto por el IS.

Se deberá configurar la petición apuntando a esta url y además, incluir una cabecera *Authorization* con el token en su interior.

Figura 3.13 Consulta recurso de alergias con token en Postman.

Realizada la configuración de la petición se deberá lanzar, enviando la solicitud junto con el token al EI, que en primer lugar, consultará al IS para comprobar la validez de dicho token y una vez comprobado, hará la llamada correspondiente al servidor HAPI FHIR a la URL *http://hapi.fhir.org/baseR4/AllergyIntolerance?_pretty=true&patient=uri.var.patientId&_format=json*, obteniendo el recurso deseado con la información sobre las alergias del paciente consultado.

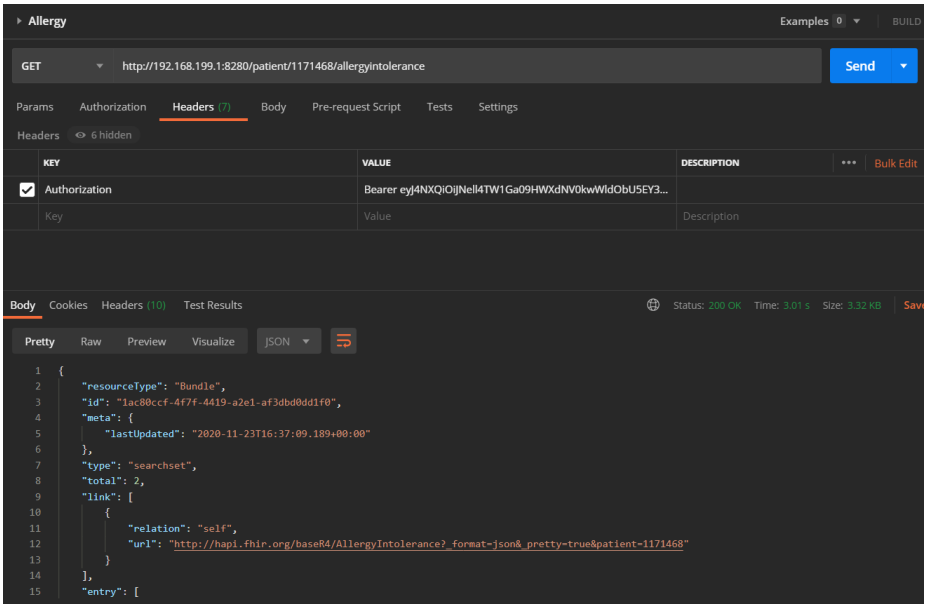

Figura 3.14 Resultado consulta alergia en escenario con autenticación.

Un fragmento del recurso obtenido se puede ver a continuación:

Código 3.3 Fragmento de recurso AllergyIntolerance obtenido en la consulta con token.

```
{
    "resourceType": "Bundle",
    "id": "1ac80ccf-4f7f-4419-a2e1-af3dbd0dd1f0",
    "meta": {
        "lastUpdated": "2020-11-23T16:37:09.189+00:00"
    },
    "type": "searchset",
    "total": 2,
    "link": [
        {
            "relation": "self",
            "url": "http://hapi.fhir.org/baseR4/AllergyIntolerance?_format=json&
                _pretty=true&patient=1171468"
        }
    ],
    "entry": [
        {
            "fullUrl": "http://hapi.fhir.org/baseR4/AllergyIntolerance/1171545",
            "resource": {
                "resourceType": "AllergyIntolerance",
                "id": "1171545",
                ...
            }
            ]
        }
    ]
}
```

Por contra, si se introduce un token incorrecto o caducado, la respuesta obtenida sería la siguiente:

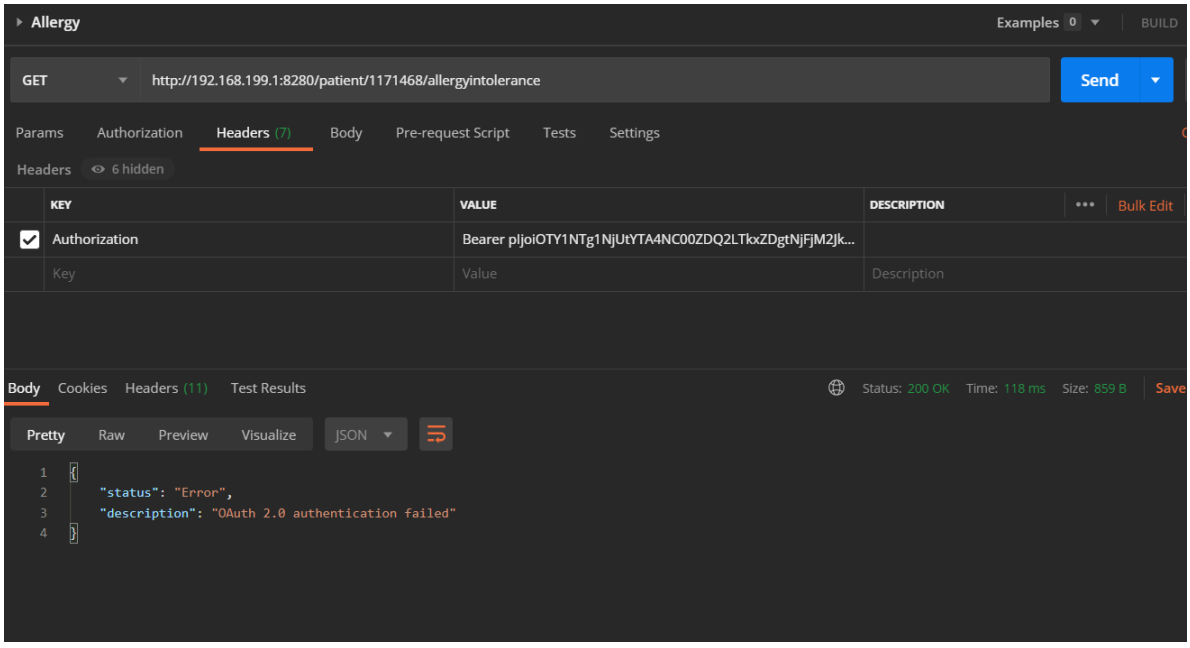

Figura 3.15 Resultado de una validación de token fallida.

3.2.3 Escenario con autenticación y autorización

Este escenario es una extensión del anterior en el que además de realizar la validación de token por parte del IS se aplicarán políticas XACML para realizar un control de acceso más exhaustivo a los recursos.

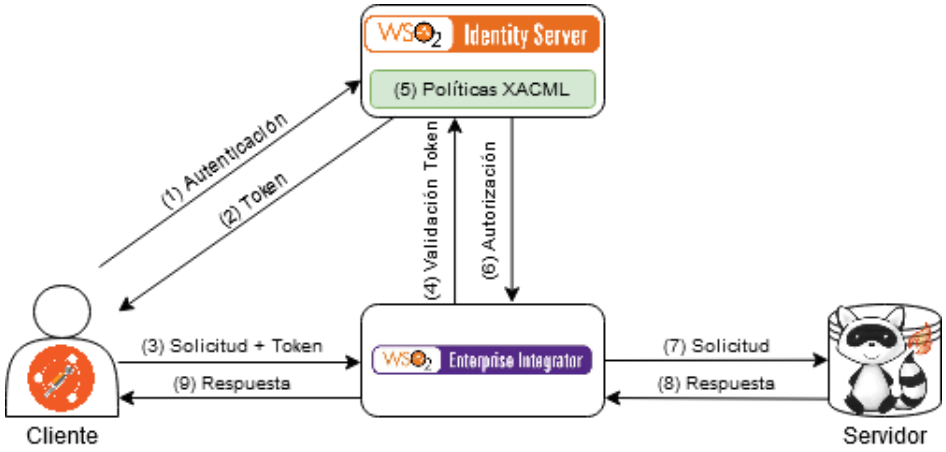

Figura 3.16 Escenario cliente-servidor con token y políticas XACML.

Como se observa en la figura 3.16, este escenario complementa al anterior añadiendo el uso de políticas XACML para el control de acceso y autorización de los usuarios a los distintos recursos. La gestión y aplicación de estas políticas se realiza en el IS, como se ha mostrado en su apartado correspondiente.

Dichas políticas son gestionadas en el PAP del IS, pudiendo visualizarlas, editarlas y publicarlas en el PDP para que se apliquen en el momento que se realizan las validaciones de los tokens. El proceso de generación y publicación de las políticas en el IS se describe en el anexo de configuración del IS (Anexo C).

En el escenario se evalúa la aplicación y decisión de las llamadas aplicando dos posibles políticas basadas en roles y atributos, mezclando los modelos RBAC y ABAC: una para el rol de enfermero y otra para el rol de médico y con una serie de *scopes* permitidos a cada rol.

La primera de estas políticas aplicará cuando el usuario *Menganito*, al cual se le ha otorgado el rol *enfermero* solicite un recurso. El proceso de invocación será el mismo que el realizado en el escenario anterior, obteniendo un token a partir de una llamada previa al IS y solicitando el recurso al EI. El EI, a su vez, invocará el IS para realizar la validación del token, donde además de comprobar que el token sea válido, se aplicará la política.

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  PolicyId="scope_based_token_validation_role_enfermero" RuleCombiningAlgId="urn:oas
   <Description>This policy template provides ability to validate OAuth2 access token to a given service provider(defined by SP_NAME) in the valid
   <Target>
      <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Oauth2</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/sp/sp-name" Category="http://wso2.org/identity/sp" DataType="http://www.
            </Match>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">token_validation</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/identity-action/action-name" Category="http://wso2.org/identity/identity
            </Match>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">enfermero</AttributeValue>
               <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role" Category="urn:oasis:names:tc:xacml:2.0:subject:role" I
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Rule Effect="Deny" RuleId="permit_by_scope">
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-name" Category="http://wso2.org/identity/oauth-scope"
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">confidential</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-name" Category="http://wso2.org/identity/oauth-scope"
            </Apply>
         </Apply>
      </Condition>
   </Rule>
</Policy>
```

Figura 3.17 Política XACML aplicable a un usuario con rol enfermero.

En la figura 3.17 se puede observar la política aplicable al rol *enfermero*. Para que se aplique se deberán cumplir las tres condiciones que se observan en la mitad superior: el proveedor de servicio debe ser *Oauth2* (el creado para estos escenarios), la acción deberá ser *token_validation* y el rol deberá ser *enfermero*. Si se cumplen estas condiciones se aplicará la política y se concederá o negará el acceso dependiendo de los *scopes* contenidos en el token.

Se ha contemplado que el rol *enfermero* únicamente tenga permitido el uso del *scope read*, de modo que un enfermero (en este caso *Menganito*) solo tenga acceso de lectura a los distintos recursos. El resto de *scopes* implementados (*write* y *confidential*) están restringidos al rol *médico*, como se observa en la figura 3.18.

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  PolicyId="scope_based_token_validation_role_medico" RuleCombiningAlgId="urn:oasis:
   <Description>This policy template provides ability to validate OAuth2 access token to a given service provider(defined by SP_NAME) in the valic
   <Target>
      <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Oauth2</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/sp/sp-name" Category="http://wso2.org/identity/sp" DataType="http://www.
            </Match>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">token_validation</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/identity-action/action-name" Category="http://wso2.org/identity/identity
            </Match>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">medico</AttributeValue>
               <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role" Category="urn:oasis:names:tc:xacml:2.0:subject:role" I
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Rule Effect="Permit" RuleId="permit_by_role_and_scope">
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-name" Category="http://wso2.org/identity/oauth-scope"
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-name" Category="http://wso2.org/identity/oauth-scope"
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">confidential</AttributeValue>
               <AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-name" Category="http://wso2.org/identity/oauth-scope"
            </Apply>
         </Apply>
      </Condition>
   </Rule>
   <Rule Effect="Deny" RuleId="deny_others"></Rule>
</Policy>
```

Figura 3.18 Política XACML aplicable a un usuario con rol médico.

Esta segunda política, se aplica de forma similar a la anterior con la única diferencia de que aplicará al rol *médico*. Esta política permitirá el acceso a los distintos usuarios que puedan poseer el rol de médico y les permitirá acceder a los distintos recursos para leer (mediante el *scope read*), como para escribir (mediante el *scope write*), así como acceder a información restringida como se contempla en el siguiente escenario.

3.2.3.1 Ejecución y resultados

Para la ejecución de este escenario se han contemplado distintos casos: acceso a lectura de un recurso por parte de un enfermero, acceso a escritura por un enfermero y acceso a escritura de un recurso por un médico.

Para la ejecución del primer caso se toma al usuario *Menganito*, el cual posee el rol *enfermero* y se solicita un token con el *scope read* de forma similar a como se ha mostrado en el escenario anterior. Una vez obtenido el token y antes de realizar la solicitud, se ha querido realizar una demostración del contenido del token obtenido. Para ello, se ha construido en Postman una petición denominada *Token Validation* que permite invocar al servicio de introspección del IS y el cual devuelve el contenido del token en formato JSON.


```
{
    "nbf": 1606323444,
    "scope": "read",
    "active": true,
    "token_type": "Bearer",
    "exp": 1606327044,
    "iat": 1606323444,
    "client_id": "klsDR477TiFvnlXrSTe_74L4mcsa",
    "username": "Menganito"
}
```


Figura 3.19 Contenido token usuario enfermero.

La respuesta muestra distintos parámetros del token, entre los que se encuentran el *scope* solicitado, un indicador de si el token se encuentra activo o está caducado, el tipo, la caducidad del mismo o el usuario asociado.

Con este token se podrá realizar invocaciones a los distintos recursos con la operación *GET* para leer su contenido y obteniendo la respuesta correspondiente, como se ha mostrado hasta ahora.

El siguiente caso consiste en intentar acceder mediante un usuario con el rol *enfermero* a un recurso mediante la operación *POST* para registrarlo. Para ello, se ha creado un nuevo recurso en la API que contempla el uso de dicha operación y que permite registrar nuevos recursos de tipo *Patient* en el servidor. Adicionalmente se ha contemplado una comprobación dentro del recurso que permite la creación de pacientes cuando el token facilitado en la invocación contenga el scope *write* para impedir que los usuarios sin este atributo puedan acceder. Dicho recurso se encuentra publicado en la URL *http://192.168.199.1:8280/patient* y para invocarlo será necesario incluir, además del token, un recurso *Patient* dentro del cuerpo del mensaje.

Para ejecutar este caso, partiendo del token obtenido en el caso anterior, se genera la petición, indicando la URL del servicio e incluyendo la cabecera *Authorization* que contiene el token y cuerpo del mensaje con el recurso *Patient* FHIR.

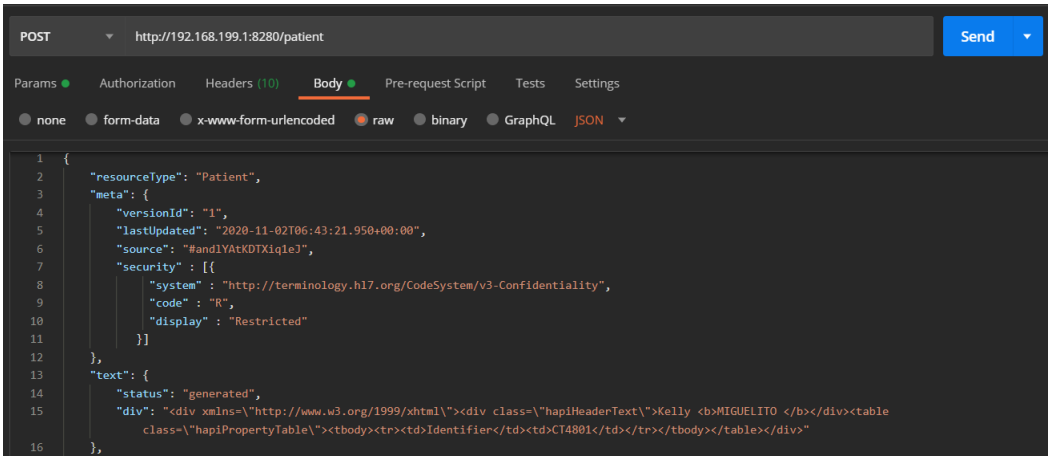

Figura 3.20 Solicitud de creación de recurso *Patient*.

Una vez generada la petición, se envía al recurso publicado en el EI, el cual invocará al IS, donde se realizará la validación del token y se aplicará la política XACML asociada al rol *enfermero* de manera correcta debido a que dispone únicamente del *scope read*, como se muestra en la figura 3.19. Una vez el IS da la respuesta de la validación al EI, se comprueba si el token contiene el *scope write* asociado a la operación *POST*, y al no contenerlo, se denegará el acceso informando que no se tiene el permiso adecuado, tal y como se puede ver en el siguiente mensaje extraído:

Código 3.4 Mensaje de acceso no autorizado por no disponer de permiso de escritura.

```
{
    "status": "Error",
    "description": "Acceso no autorizado. No se dispone del permiso
        correspondiente."
}
```

Por otro lado, y para cerrar este caso, se comprueba el resultado al solicitar el permiso de escritura por parte del usuario con rol *enfermero* para poder realizar la operación *POST*. Para ello, a la hora de realizar la solicitud del token por parte del usuario *Menganito*, se indicará el *scope write*. En una primera instancia se obtendrá el token con el *scope* asociado, pero a la hora de invocar al recurso y realizar la validación y aplicación de la política XACML, al tener restringido su uso como, como se pudo ver en la figura 3.17, se denegará el acceso y se le devolverá nuevamente el mensaje de error.

El último caso consiste en el uso de un usuario con rol médico para obtener acceso tanto de lectura como de escritura a los distintos recursos. Para ello, se toma al usuario *Fulanito*, el cual posee el rol *medico* y se solicita un token con los *scopes read, write y confidential*. Una vez obtenido el token, se puede observar su contenido haciendo uso del servicio de introspección del IS obteniendo un resultado como el que se muestra a continuación:


```
{
    "nbf": 1606328902,
    "scope": "confidential read write",
    "active": true,
    "token_type": "Bearer",
    "exp": 1606332502,
    "iat": 1606328902,
    "client_id": "klsDR477TiFvnlXrSTe_74L4mcsa",
    "username": "Fulanito"
}
```


Figura 3.21 Contenido token usuario médico.

Visualizado el interior del token, se puede proceder a invocar un recurso para leerlo o crear uno mediante la operación *POST*. Si se realiza lo primero, se obtendrá la misma respuesta observada por el usuario enfermero, ya que ambos tienen el permiso para leer, otorgado por el *scope read*. Si se invoca la operación *POST*, esta vez al realizarse la validación y aplicar la política XACML correspondiente, se permitirá el acceso y se podrá generar el recurso, obteniendo como respuesta el recurso creado.

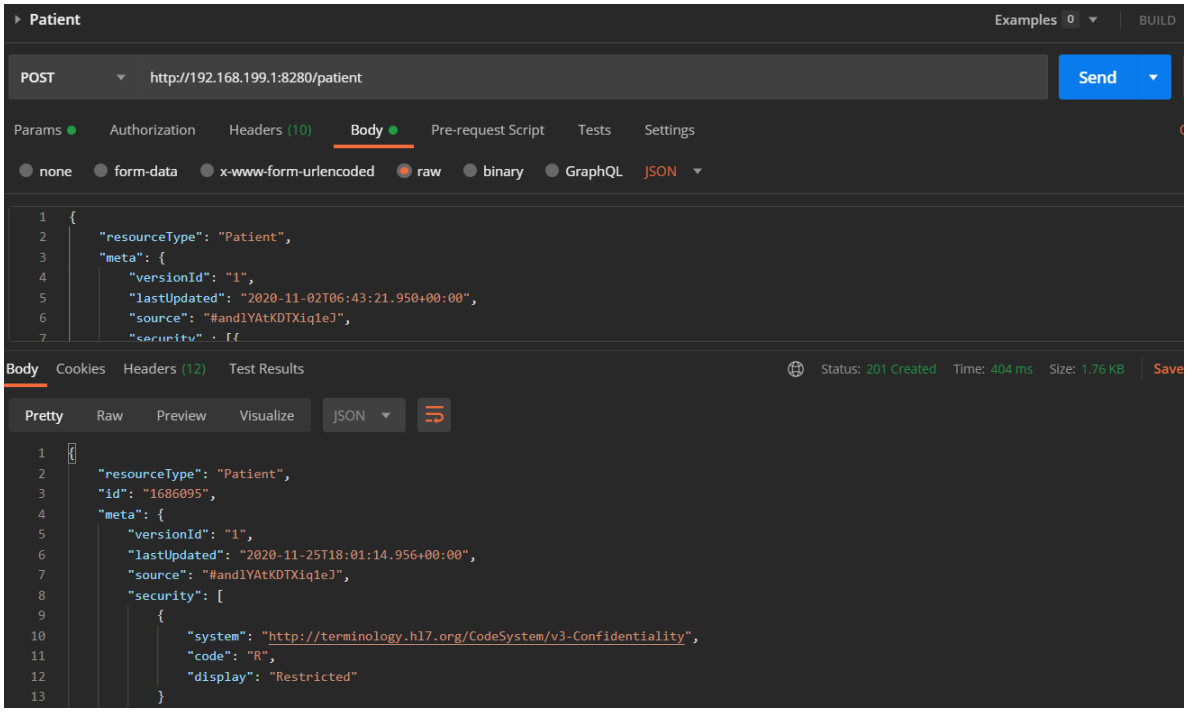

Figura 3.22 Recurso creado mediante la operación POST.

En esta ocasión se ha podido realizar la operación tras superar el proceso de autenticación, autorización mediante política XACML y comprobación del permiso correspondiente para realizar la operación.

Para finalizar este escenario y de forma adicional, se puede comprobar que se ha generado un nuevo recurso en el servidor, observando que la respuesta a la petición contiene un código HTTP *201 Created* y el recurso obtenido en la respuesta es el mismo que se ha enviado, añadiendo un campo *id* con su identificador para poder acceder a él, y otra serie de metadatos, como la fecha de actualización o la versión del mismo.

3.2.4 Escenario con contenido restringido

El último de los escenarios para la aplicación de control de acceso a información clínica empleando el estándar FHIR contempla el acceso a información de carácter sensible o restringida mediante el uso de etiquetas.

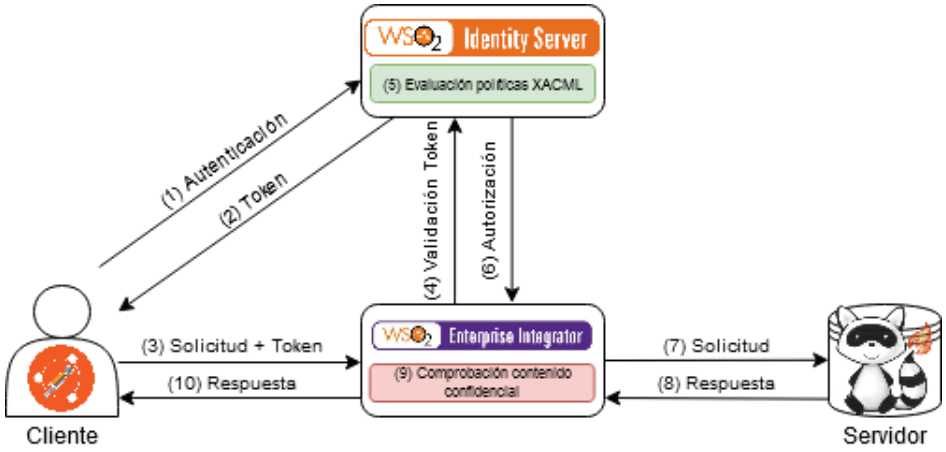

Figura 3.23 Escenario cliente-servidor con token, políticas y contenido restringido.

En este caso, se aplicará la lógica de control de acceso antes y después de consultar un recurso, de modo que en una primera etapa se realizará la validación de token y aplicación de políticas XACML para poder consultar el recurso, y en la segunda, tras obtener el recurso, se verificará la existencia de una etiqueta avisando de que la información consultada tiene carácter sensible o restringido y comprobando que el usuario que ha realizado la solicitud tiene permiso para acceder a dicha información.

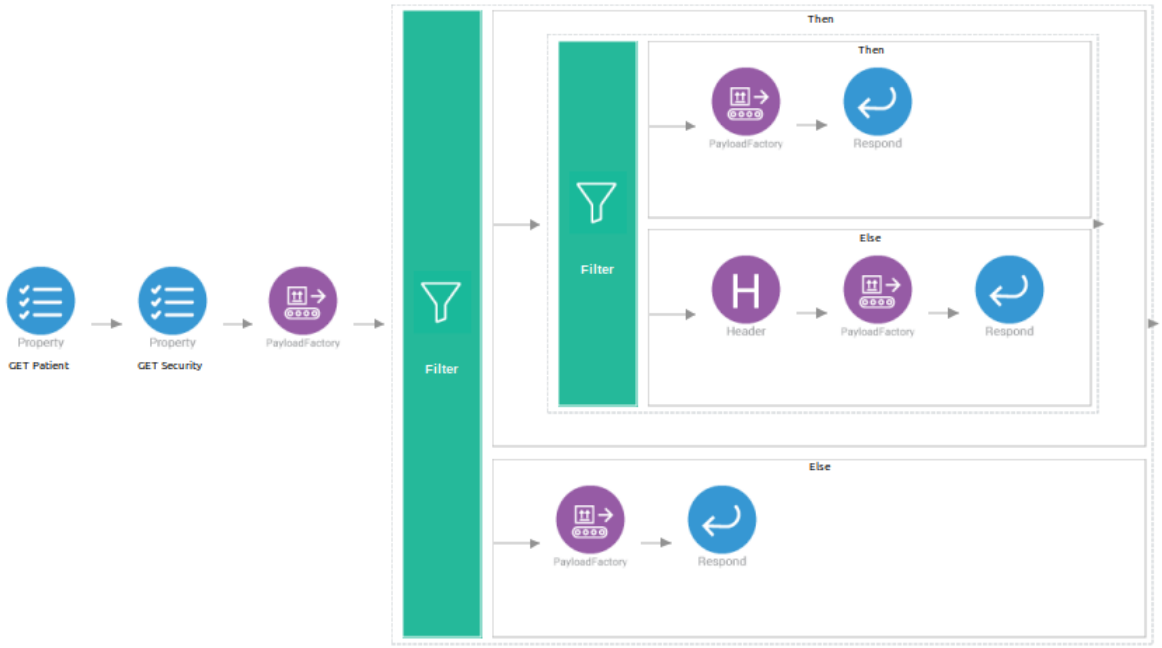

Figura 3.24 Escenario con validación de etiquetas configurado en Enterprise Integrator.

Como se observa en la figura 3.24, la lógica que implementa el EI contempla dos filtros. El primero de ellos, comprueba la existencia de la etiqueta de seguridad en el recurso obtenido tras consultar el servidor.

En caso de no existir, se devolverá la respuesta directamente al origen de la petición. En caso de existir, se aplicará el segundo filtro, que comprobará la existencia del *scope confidential* en el token facilitado. En caso de existir dicho *scope*, se devolverá el recurso consultado al solicitante y, en caso de no existir, se notificará un mensaje de error indicando el problema.

3.2.4.1 Ejecución y resultados

Para la ejecución se contemplarán nuevamente dos casos: acceso con permiso de visualización de información sensible y acceso sin ella.

El primer caso será el acceso a la información sin el permiso correspondiente. Para ejecutar la solicitud, se empleará el usuario con rol *enfermero*, que como vimos en las políticas XACML, no tiene permitido el uso del *scope confidential*, de modo que no dispondrá de acceso a la información.

Se ejecutará la consulta, tras obtener un token, invocando la URL *http://192.168.199.1:8280/patient/patientId*, donde el valor de *patientId* será el identificador de un paciente que disponga de la etiqueta de seguridad, informando que se trata de un paciente con información sensible.

Tras realizar la invocación, se validará el token, autorizando el acceso en caso de indicar únicamente el *scope read* y denegándolo en caso de haber solicitado el *scope confidential* o *write*. Tras esto, y suponiendo que únicamente se ha solicitado acceso con el primer *scope*, se consultará el recurso, se obtendrá respuesta en el EI, que comprobará la existencia de la etiqueta, en este caso con valor *'R'* y comprobará si el token contiene el *scope confidential*. Al no disponer de dicho *scope* se denegará el recurso, emitiendo en su lugar un mensaje informando del error.

Código 3.5 Mensaje de acceso restringido a información sensible.

```
{
    "status": "Error",
    "description": "Acceso restringido"
}
```

El segundo caso muestra el acceso a la información con el permiso, asociado al *scope confidential*, que está disponible por los usuarios con el rol de médico. Nuevamente, será necesario obtener un token y realizar la invocación al servicio, donde se realizará la misma lógica y donde esta vez, al comprobar la existencia del *scope* correspondiente, se devolverá al origen el recurso solicitado, pudiéndose observar en su interior, la existencia de la etiqueta que informa del carácter sensible de la información.

Código 3.6 Mensaje de acceso restringido a información sensible.

```
{
    "resourceType": "Patient",
    "id": "1686317",
    "meta": {
        "versionId": "1",
        "lastUpdated": "2020-11-25T19:09:10.057+00:00",
        "source": "#andlYAtKDTXiq1eJ",
        "security": [
            {
                "system": "http://terminology.hl7.org/CodeSystem/v3-
                    Confidentiality",
                "code": "R",
                "display": "Restricted"
            }
        ]
    },
    ...
    "name": [
        {
```

```
            "family": "Detal",
            "given": [
                "Miguelito"
            ]
        }
    ],
    "gender": "male"
}
```

3.2.5 Escenario con IPS

Por último y para terminar, se ha querido incluir un escenario basado en la construcción de una consulta a los distintos recursos que componen un documento de resumen de historia clínica de un paciente con IPS FHIR. Como ya se indicó en sus correspondientes apartados, FHIR recoge la información en distintos recursos con significado propio y el documento IPS recoge varios de ellos, algunos de ellos con carácter requerido u obligatorio y otros únicamente opcionales. En un escenario real, algunos de estos recursos o incluso todos, pueden estar almacenados y custodiados por sistemas diferentes, por lo que se hace complicado que un único sistema posea la información necesaria para exponer un recurso único con toda ella. En un escenario desacoplado para poder componer el documento, un solicitante debería solicitar, uno a uno a cada sistema, el recurso que almacena para finalmente poder construir el documento. Es aquí donde entra en juego y gana valor la inclusión del Enterprise Integrator como un elemento central que gestione y exponga los distintos recursos existentes, permitiendo exponer un recurso único al que un usuario realice una solicitud y se genere un proceso de orquestación y agregación que consulte posteriormente cada uno de los distintos recursos de forma independiente, procese toda la información para componer el documento y lo devuelva al solicitante.

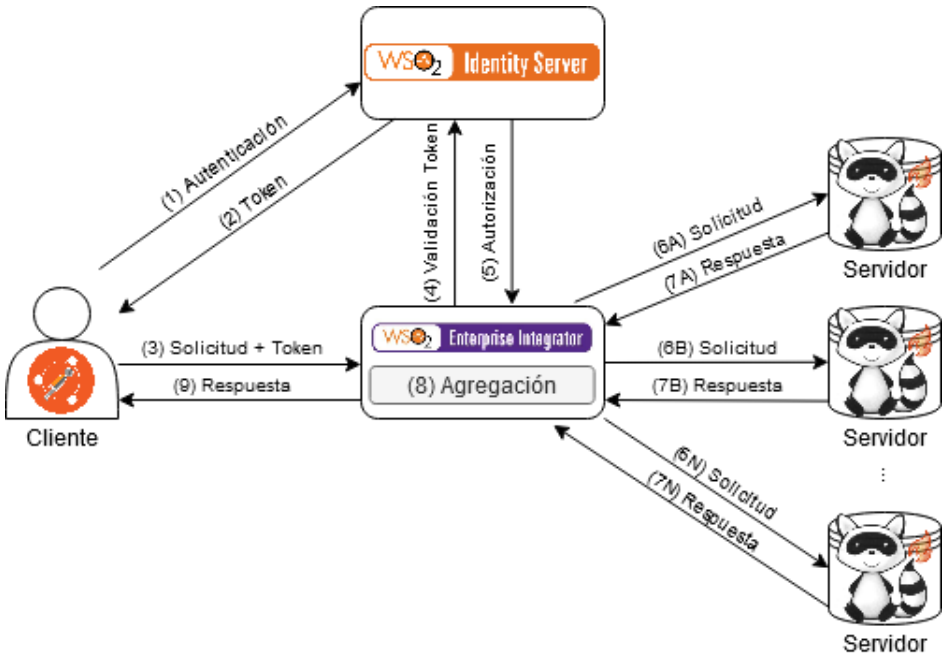

Figura 3.25 Escenario composición IPS.

Este proceso se ha realizado, como se podrá ver en la figura 3.25, definiendo un recurso que en su interior invoca al resto de recursos expuestos en la API y que han sido empleados anteriormente, recogiendo cada respuesta, evaluando la información devuelta y generando con ello un recurso único que es devuelto al usuario que realiza la solicitud.

En este punto se toman en cuenta dos consideraciones importantes. Por un lado, la gestión de las autorizaciones, y por el otro, la generación de respuestas en caso de que algunos de los recursos solicitados no tengan registrada ninguna información o no se encuentren disponibles en el momento de la consulta.

Para la primera de ellas, FHIR menciona en su documentación que se debe contemplar, cuando se trabaje con recursos que puedan contener otros en su interior, si se permite el acceso al recurso completo de manera individual, se debe permitir el acceso al resto de recursos contenidos, pero no da directrices de cómo llevarlo a cabo. En este punto se contemplan varias opciones:

- Controlar el acceso al recurso completo y admitir el resto de accesos sin realizar ningún control adicional. Este es el mecanismo más sencillo pero que a su vez permite un menor control sobre los accesos a la información.
- Controlar el acceso al recurso completo y a cada uno de los recursos individuales con un único permiso. Este mecanismo es más complejo puesto que se tiene que controlar el permiso tanto en el recurso expuesto como en cada uno de los recursos individuales.
- Controlar el acceso al recurso completo con un permiso propio y a cada uno de los recursos individuales con sus propios permisos. Este es el más complejo de todos y a su vez el más seguro.

Dependiendo del escenario y sus necesidades será conveniente aplicar una u otra de estas opciones.

Para la segunda de las consideraciones, FHIR define una serie de valores estáticos para cada recurso que permite informar de forma estandarizada cuando no existe información sobre alguno de los recursos o no se tiene conocimiento, ya que se considera imprescindible discernir e informar adecuadamente cuando no se tiene información de, por ejemplo, alergias debido a que no se hayan realizado pruebas, y cuando se han realizado pruebas y no se ha detectado ninguna.

A continuación, se muestra un ejemplo de un recurso de alergia cuando no se tiene información registrada y cuando no existen alergias detectadas de un paciente:

Código 3.7 Recurso AllergyIntolerance cuando no se tiene información sobre alergias de un paciente.

```
[{
  "resource": {
    "resourceType": "AllergyIntolerance",
    "id": "0",
    "meta": {
      "versionId": "1"
    },
    "code": {
      "coding": [{
        "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv-ips",

        "code": "no-allergy-info",
        "display": "No information about allergies"
      }]
    },
    "patient": {
      "reference": "Patient/1171468"
    }
  }
}]
```

Código 3.8 Recurso AllergyIntolerance cuando no se han detectado alergias sobre un paciente.

```
[{
  "resource": {
    "resourceType": "AllergyIntolerance",
    "id": "0",
    "meta": {
      "versionId": "1"
```

```
    },
    "code": {
      "coding": [{
        "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv-ips",

        "code": "no-known-allergies",
        "display": "No known allergies"
      }]
    },
    "patient": {
      "reference": "Patient/1171468"
    }
  }
}]
```

3.2.5.1 Ejecución y resultados

Para la ejecución de este escenario, se deberá invocar el recurso, adjuntando un token previamente solicitado en la cabecera *Authorization* y expuesto en la siguiente URL: *http://192.168.199.1:8280/patient/patientId/summary*

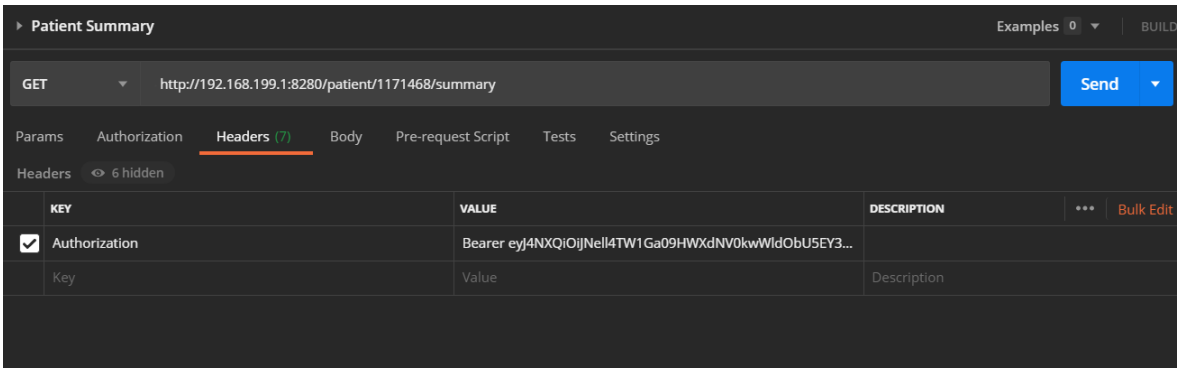

Figura 3.26 Consulta recurso Patient Summary en Postman.

Una vez lanzada la petición, en primer lugar, se realizará la validación del token facilitado. Una vez validado, se lanzará una consulta interna al primero de los recursos expuestos, en este caso *Composition*, con la información de cabecera del documento. Después, se consultará el siguiente recurso que compone el documento, que realizará su lógica interna y devolverá una respuesta. Esta respuesta se evalúa, comprobando si se ha devuelto información o no. En caso de no obtenerse respuesta, se generará un mensaje indicando que no se conoce información del recurso asociado, tal y como se ha indicado anteriormente.

Esta lógica se repetirá por cada uno de los distintos recursos contemplados en el escenario y finalmente se generará el documento IPS completo uniendo las distintas respuestas obtenidas de cada recurso.

Código 3.9 Fragmento de recurso Patient Summary.

```
{
    "resourceType": "Bundle",
    "id": "29ed51b9-1a38-4081-a9e6-8128f08a2d51",
    "meta": {
        "lastUpdated": "2020-09-24T08:07:33.075+00:00"
    },
    "type": "searchset",
    "link": [
        {
            "relation": "self",
            "url": "http://hapi.fhir.org/baseR4/Bundle?_pretty=true"
```

```
        },
        {
            "relation": "next",
            "url": "http://hapi.fhir.org/baseR4?_getpages=29ed51b9-1a38-4081-
                a9e6-8128f08a2d51&_getpagesoffset=20&_count=20&_pretty=true&
                _bundletype=searchset"
        }
    ],
    "entry": [
        {
            "resource": {
                "resourceType": "Composition",
                "subject": {
                    "reference": "Patient/1681277"
                },
                "date": "2018-06-28T12:30:02+01:00",
        },
        {
            "resource": {
                "resourceType": "AllergyIntolerance",
                ...
        }
        },
        {
            "resource": {
                "resourceType": "Condition",
                ...
        }
        },
    {
            "resource": {
                "resourceType": "MedicationStatement",
                ...
        }
    },
        {
            "resource": {
                "resourceType": "MedicationStatement",
                ...
        }
    },
    ...
        {
            "resource": {
                "resourceType": "Immunization",
                ...
                }
            }
        }
    ]
}
```

En el mensaje anterior se puede observar un fragmento de la respuesta a un recurso Patient Summary en el cual se contempla la cabecera (*Composition*) y la existencia de al menos un recurso alergia (*AllergyIntolerance*), un problema (*Condition*), dos medicaciones (*MedicationStatement*) y una vacunación (*Immunization*).

4 Conclusiones

La elaboración de este trabajo ha supuesto todo un reto debido a la cantidad de elementos que se han de tomar en consideración a la hora de diseñar y construir un sistema de control de acceso. No existe un método único que permita diseñar el sistema y se deben tener en cuenta muchos actores en el mismo.

Los distintos protocolos estudiados, en especial XACML para la definición y uso de políticas, junto con OAuth para realizar los flujos de autenticación y autorización, presentan grandes bondades en cuanto a posibilidades de uso y funcionamiento, ofreciendo en conjunto los mecanismos necesarios para poder llevar a cabo un control exhaustivo del acceso a la información.

En lo referente al marco práctico, tras el estudio y uso del software ofrecido por WSO2, resulta complicado su uso en primera instancia debido a la gran cantidad de herramientas y opciones que permiten. El uso de su documentación se hace fundamental para poder configurar por primera vez los distintos elementos que se han contemplado en el trabajo. Se considera interesante mencionar que la herramienta WSO2 Integration Studio, pese a las bondades que ofrece en cuanto a sencillez para diseñar los recursos contemplados, resulta complicada de utilizar, al ser poco intuitiva en muchas ocasiones a la hora de configurar parámetros y elementos. Además, esta sencillez algunas veces podrá penalizar el desarrollo, obligando a usar alternativas a los mediadores que ofrece como se ha visto a la hora de invocar al servicio de introspección de tokens y visualizar su contenido.

El uso de la herramienta Postman y el servidor público ofrecido por HAPI FHIR han facilitado enormemente la consecución de los objetivos, simplificando la ejecución de las distintas operaciones realizadas y obteniendo un sistema de almacenamiento basado en FHIR sin necesidad de construir uno específico.

Cabe mencionar que, si bien es cierto que los escenarios ejecutados son limitados en cuanto a los usuarios, roles, permisos y número de componentes por sencillez, podrían ser utilizados como base para el diseño de un sistema de control real de manera sencilla.

Como resumen final, se concluye que diseñar un sistema de control de acceso no es algo sencillo y que dependerá de cada escenario en el que se pretenda implementar, pero existen gran cantidad de mecanismos y herramientas que definen por completo soluciones para poder llevarlo a cabo de manera eficaz y robusta.

4.1 Resultados y objetivos

En relación con los objetivos marcados al inicio del desarrollo de este trabajo, se considera que se han alcanzado.

Objetivo principal:

- **Diseñar un servicio de control de acceso:** Se han estudiado los distintos componentes necesarios para diseñar un servicio de control de acceso en un sistema sanitario que hace uso del estándar HL7 FHIR, mostrando el uso de los distintos mecanismos, protocolos y estándares que lo componen.

Objetivos secundarios:

- **Investigar protocolos y estándares:** Como se ha mencionado en el objetivo principal, se han estudiado los distintos protocolos y estándares que se pueden emplear, tanto para un sistema sanitario, con los estándares HL7, HL7 FHIR y el proyecto IPS para la interoperabilidad de la información clínica, como para un sistema de seguridad, mediante los estándares de autenticación y autorización OAuth, XACML y SAML.
- **Analizar herramientas:** Se han analizado las distintas herramientas de la suite ofrecida por WSO2 para poder llevar a cabo los distintos escenarios prácticos que se han contemplado durante el trabajo.
- **Mostrar el funcionamiento:** Con los distintos escenarios contemplados se ha mostrado el funcionamiento de cada elemento del sistema de control de acceso, añadiendo en cada uno de ellos un nivel mayor de complejidad.

4.2 Próximos pasos

Finalizado el desarrollo de este trabajo y habiendo alcanzado los objetivos planteados en el inicio, se podrían llevar a cabo varias modificaciones o mejora para extender el funcionamiento del mismo.

Un primer paso podría ser la separación de los proveedores de identidad de los proveedores de servicio. En un sistema grande, como puede ser un sistema sanitario, podrán convivir numerosos proveedores de servicio que ofrezcan diversos servicios y tal vez más de un proveedor de identidad. En este trabajo se ha mostrado el funcionamiento de los distintos escenarios con el IS como único elemento que integra al proveedor de servicio y al proveedor de identidad, pero como se mostró en la arquitectura de WSO2 Identity Server, existen mecanismos para la integración con proveedores de servicio y proveedores de identidad externos al mismo. Esto supondría facilitar la integración entre distintas aplicaciones y servicios, a la vez que complicaría el sistema de control de acceso, añadiendo más actores que controlar. Del mismo modo, se podría hacer uso de distintos sistemas de almacenamiento para la gestión de usuarios.

A su vez, sería interesante hacer uso de un dialecto de *claims* distinto del que ofrece WSO2 por defecto. Para un entorno sanitario, OASIS define el perfil XSPA para XACML 2.0 enfocado en entornos sanitarios. Este perfil define una serie de atributos y normas que se pueden emplear para llevar a cabo la autenticación de usuarios y su autorización mediante políticas [25].

Otro tema interesante podría ser el estudio e implementación de un servicio FHIR para almacenar la información. Con la propia librería que ofrece HAPI FHIR parece sencillo de crear este servidor y donde además de poder almacenarse los distintos recursos para uso, se podrían implementar más medidas de control de acceso. Aunque en un sistema centralizado como el que se ha desarrollado en los distintos trabajos, la gestión del control de acceso se ha aplicado a un elemento central del sistema, en un sistema descentralizado, cada aplicación que almacenase información debería tener un sistema de control de acceso con medidas propias o consensuadas para asegurar la información. También se podría considerar un sistema mixto, en el cual exista un sistema centralizado que controle la autenticación y autorización pero que delegue la última decisión a cada sistema, de modo que estos puedan contemplar distintas medidas, como el control de información sensible, de modo que esta información no llegue a salir de su dominio.

Por último, se podría profundizar en el desarrollo del IPS ya que el objetivo de su estudio en este trabajo ha sido enfocado para una implementación con control de acceso sobre el mismo y no para desarrollarlo con todas sus características.

Bibliografía

[1] HL7. *http://www.hl7spain.org/wp-content/uploads/2012/08/semHL7_introSeminario.pdf*.

[2] FHIR. *https://www.hl7.org/fhir/http.html*.

[3] IPS. *http://international-patient-summary.net/mediawiki/index.php?author=Main_Page*.

[4] XACML. *http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html*.

[5] SAML. *http://saml.xml.org/saml-specifications*.

[6] OAuth 2.0. *https://tools.ietf.org/html/rfc6749*.

[7] OAuth scopes. *https://oauth.net/2/scope/*.

[8] HL7 v2. *http://www.hl7spain.org/wp-content/uploads/2012/08/SemHL7_Detalles_V2.pdf*.

[9] HL7 v2.3. *https://www.hl7.org/implement/standards/product_brief.cfm?product_id=225*.

[10] Recursos FHIR. *https://www.hl7.org/fhir/resourcelist.html*.

[11] Guía de recursos FHIR. *http://hl7.org/fhir/resourceguide.html*.

[12] Datos FHIR. *https://www.hl7.org/fhir/datatypes.html*.

[13] Seguridad FHIR. *https://www.hl7.org/fhir/security.html*.

[14] Etiquetas FHIR. *https://hl7.org/FHIR/security-labels.html*.

[15] Propósito de uso de etiquetas FHIR. *https://www.hl7.org/fhir/v3/PurposeOfUse/vs.html*.

[16] Tabla de confidencialidad FHIR. *https://www.hl7.org/fhir/v3/ConfidentialityClassification/vs.html*.

[17] Norma IPS. *https://standards.cen.eu/dyn/www/f?p=204:110:0::::FSP_PROJECT,FSP_ORG_ID:65797,6232&cs=161C7CF4C93D84B07327805D4194C55BA*.

[18] WSO2 Identity Server. *https://is.docs.wso2.com/en/latest/*.

[19] Arquitectura IS. *https://is.docs.wso2.com/en/latest/get-started/architecture/*.

[20] WSO2 Enterprise Integrator. *https://wso2.com/integration/*.

[21] Postman. *https://www.postman.com/*.

[22] Postman Client. *https://www.postman.com/product/api-client/*.

[23] HAPI FHIR. *https://hapifhir.io/*.

[24] HAPI Server. *http://hapi.fhir.org/#*.

[25] XSPA. *http://docs.oasis-open.org/xacml/xspa/v1.0/xacml-xspa-1.0-cs02.html*.

[26] Descarga Enterprise Integrator. *https://wso2.com/enterprise-integrator/6.6.0*.

[27] Descarga Integration Studio. *https://wso2.com/integration/integration-studio/*.

[28] Documentación WSO2 Integration Studio. *https://ei.docs.wso2.com/en/latest/micro-integrator/develop/WSO2-Integration-Studio/*.

[29] Descarga Identity Server. *https://wso2.com/identity-and-access-management/#*.

[30] Configuración Identity Server. *https://docs.wso2.com/display/IS530/Adding+and+Configuring+a+Service+Provider*.

[31] Descarga Postman Client. *https://www.postman.com/downloads/*.

5 Anexos

A Instalación y configuración de WSO2 Enterprise Integrator

Instalación

La instalación del componente Enterprise Integrator de WSO2 comienza con la descarga del producto desde su página oficial [26]. En este caso se ha optado por la versión 6.6.0. Accediendo a la página de WSO2 se podrá descargar el fichero *wso2ei-windows-installer-x64-6.6.0.msi* con el cual se podrá realizar la instalación del componente. Una vez descargado, se deberá ejecutar y seguir las indicaciones del instalador para proceder con su instalación. Una vez seguidos todos los pasos, se realizará la instalación y configuración de los componentes básicos para poder ejecutar el Enterprise Integrator.

Configuración

Para iniciar el componente, se deberá ejecutar el archivo *launcher_integrator.bat* que se encontrará alojado en el directorio *<WSO2_HOME>\Enterprise Integrator\6.6.0\bin* siendo *<WSO2_HOME>* el directorio raíz del equipo que albergará todos los componentes de WSO2 (en el caso de este trabajo será *C:\Archivos de Programa\WSO2*). Una vez ejecutado el fichero, se abrirá un terminal en el cual se podrán observar los distintos *logs* lanzados por el componente durante su inicio y durante su ejecución.

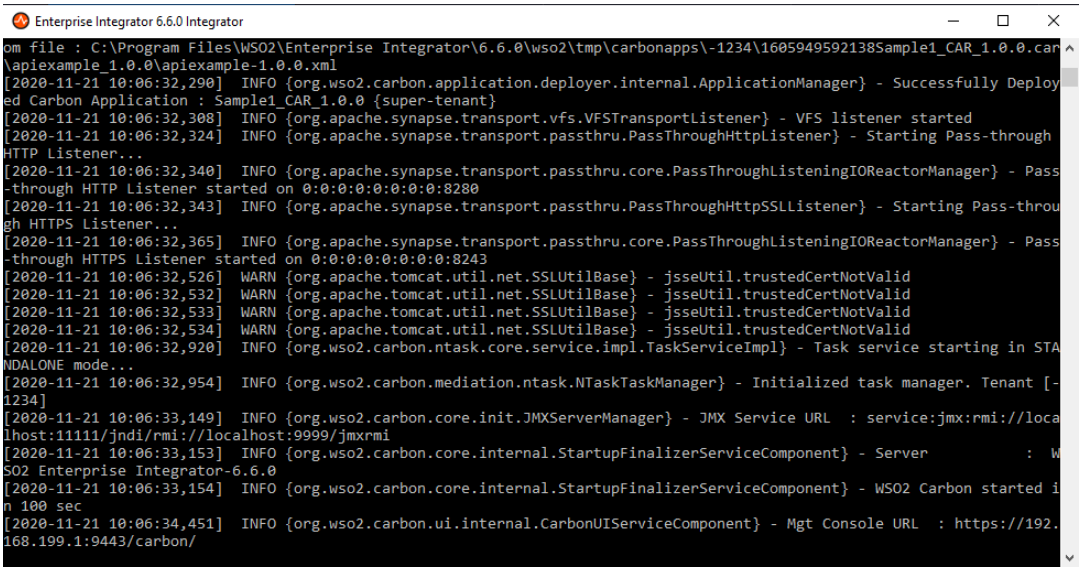

Figura A.1 Terminal de WSO2 Enterprise Integrator.

Una vez finalizada la iniciación del componente, se podrá observar en el terminal la URL para acceder a la interfaz de administración, en este caso, dicha URL es *https://192.168.199.1:9443/carbon/* y podremos acceder a ella a través de cualquier navegador web. Por defecto, en todos los componentes de WSO2 el protocolo empleado para el acceso será HTTPS y el puerto configurado será el 9443 y, a su vez, el usuario de acceso por defecto será *admin* y la contraseña *admin*.

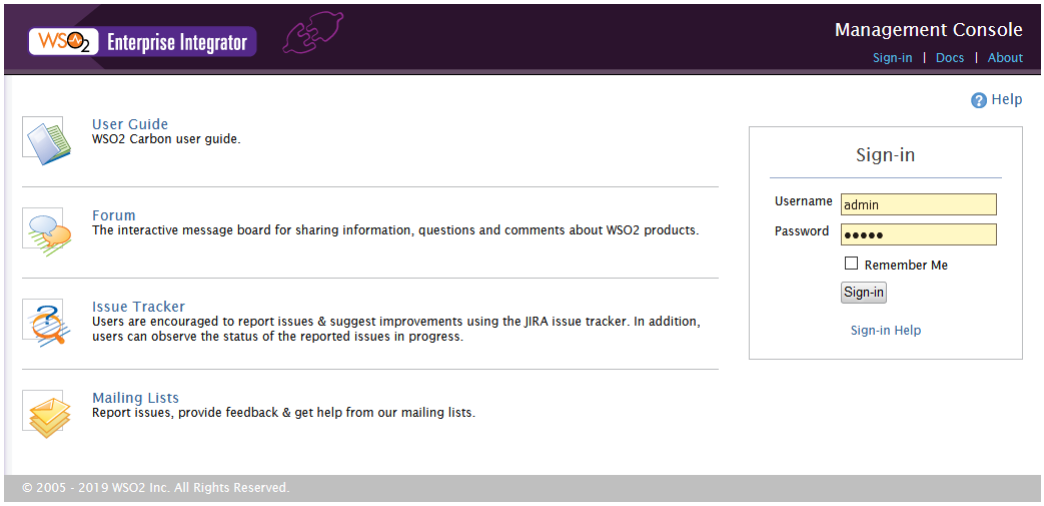

Figura A.2 Pantalla de inicio de sesión del Enterprise Integrator.

Una vez se ha accedido a la interfaz de administración y tras indicar el usuario y contraseña adecuados, se accederá a la interfaz principal del componente, donde se observará información básica del servidor como su dirección, versión, sistema operativo, etc. En el menú lateral de la izquierda se encuentran todas las herramientas que ofrece la interfaz para realizar configuraciones y administrar el servidor.

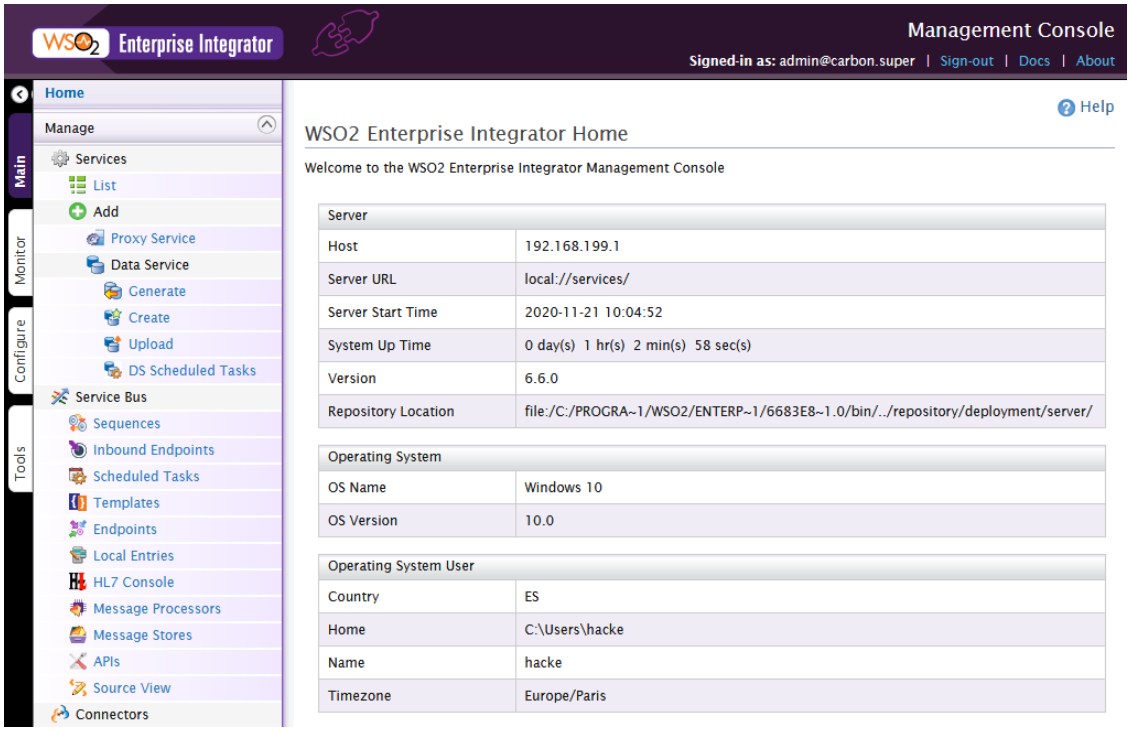

Figura A.3 Interfaz principal Enterprise Integrator.

Para este trabajo, las herramientas que se emplearán serán *APIs* dentro de las opciones del menú *Service Bus* y *List* y *Add* en el menú *Carbon Applications*. Las herramientas *List* y *Add* nos permitirán añadir y gestionar las aplicaciones desplegadas en el Enterprise Integrator y la herramienta *APIs* permite observar y gestionar las distintas APIs generadas por las aplicaciones desplegadas.

En este trabajo se ha empleado el WSO2 Integration Studio para realizar el desarrollo de varias APIs que exponen los distintos recursos que componen los escenarios de estudio del mismo, y que se exportan como una aplicación desplegable en el EI.

Una vez desarrollada la aplicación (ver anexo siguiente) y exportada como el fichero *FHIR-API_1.0.0.car*, se deberá acceder a la interfaz de administración del Enterprise Integrator para poder añadirla al servidor mediante la opción *Add* dentro del menú *Carbon Applications* antes mencionado. Una vez se ha accedido, se mostrará la opción para añadir el fichero de la aplicación almacenado en el equipo, donde se deberá seleccionar el fichero antes mencionado y pulsar sobre el botón *Upload*. Hecho esto, se comenzará a desplegar la aplicación en el servidor, pudiéndose observar el resultado en el terminal y donde si todo funcionado correctamente, se debería poder observar el mensaje *Successfully Deployed Carbon Application*.

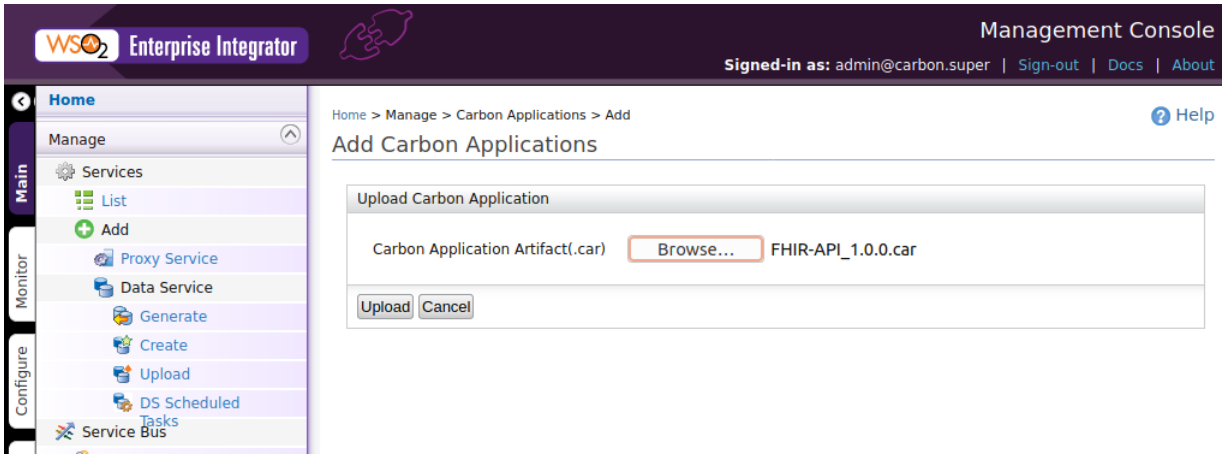

Figura A.4 Interfaz para el despliegue de aplicaciones en Enterprise Integrator.

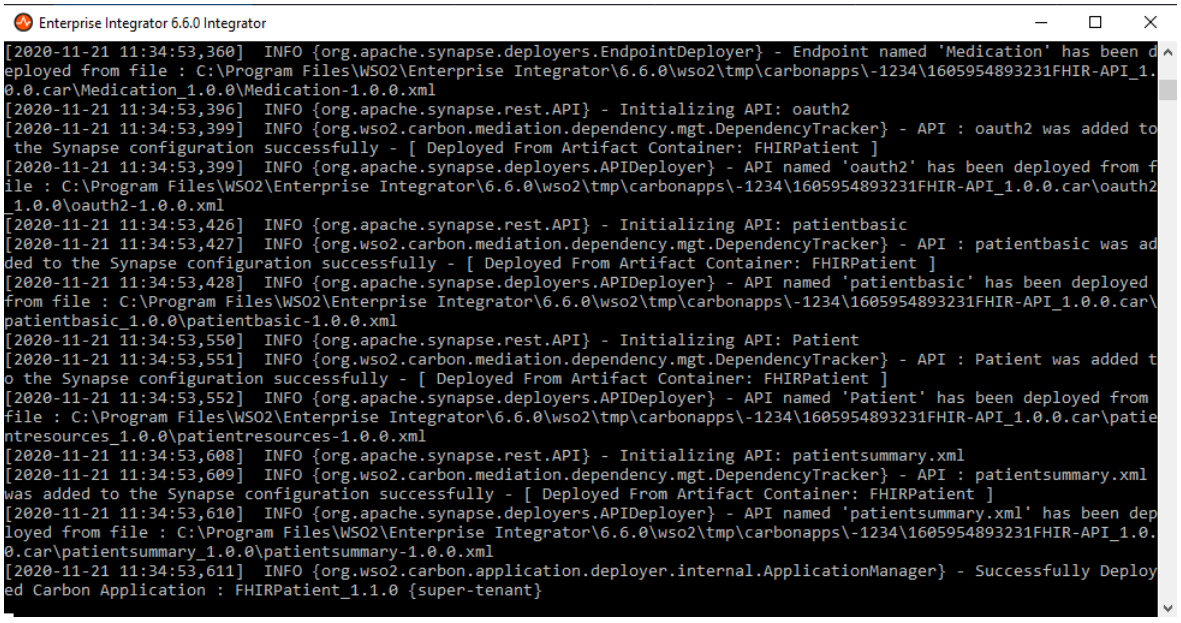

Figura A.5 Resultado del despliegue de la aplicación en el terminal del Enterprise Integrator.

Con esto, la aplicación quedaría desplegada, pudiéndose observar en la opción *List* en el mismo menú que la opción anterior. En esta nueva ventana se podrá observar la aplicación desplegada y se permitirá realizar varias acciones: *Delete* para quitar la aplicación del servidor, *Redeploy* para volver a desplegar la aplicación si fuese necesario y *Download* para descargar el fichero que contiene la aplicación.

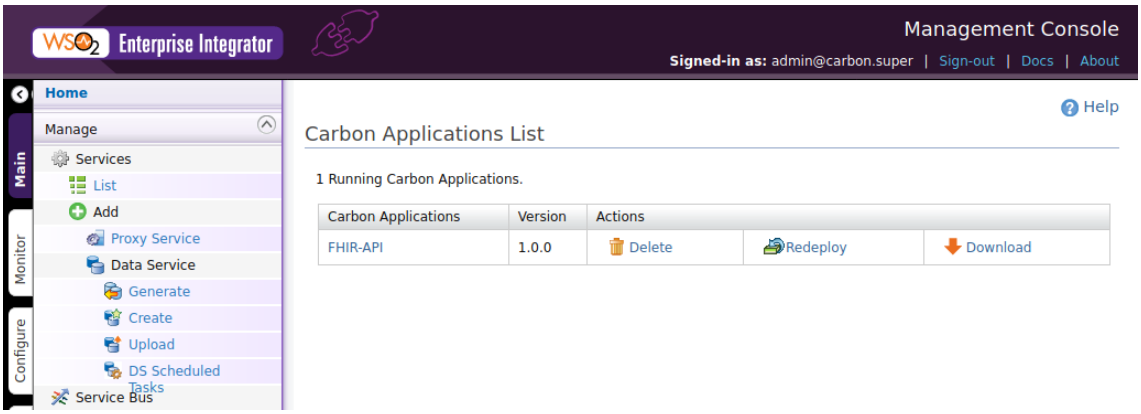

Figura A.6 Listado con la aplicación desplegada en el servidor Enterprise Integrator.

Por último, accediendo a la herramienta *APIs* y habiendo desplegado la aplicación desarrollada, se podrán observar las distintas APIs desplegadas en el servidor, además de permitir generar nuevas manualmente. Aquí se mostrarán las distintas URLs de acceso a las APIs expuestas por el servidor y se podrá acceder a cada una de ellas por medio de las diferentes acciones que ofrece el menú, donde se podrán habilitar estadísticas y trazas de las mismas por medio de las acciones *Enable Statistics* y *Enable Tracing*, editarlas o eliminarlas por medio de las acciones *Edit* y *Delete* y en el caso de estar desarrolladas con swagger se podría mostar su formato y lanzar pruebas mediante las acciones *swagger* y *Try This API*.

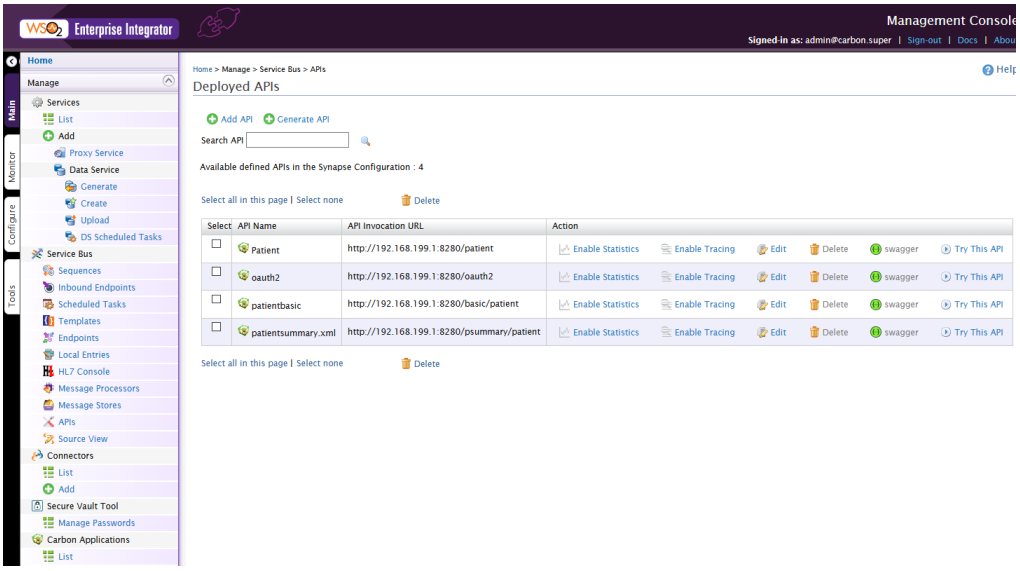

Figura A.7 Listado de APIs desplegadas en el servidor Enterprise Integrator.

B Instalación y uso de WSO2 Integration Studio

Instalación

Para implementar la lógica necesaria en los distintos escenarios de prueba, se empleará la herramienta WSO2 Integration Studio, que permitirá definir el comportamiento interno de los distintos recursos que serán empleados.

Para su instalación, en un inicio se optó por la instalación de la herramienta bajo el sistema operativo Windows 10, donde se encuentran albergados el resto de componentes, pero tras encontrarse numerosos errores y bloqueos en la herramienta, se ha optado por una instalación en una máquina virtual en el mismo equipo haciendo uso del software VMware Worksation y una imagen de Ubuntu en su versión 20.04.01 LTS.

El instalable se encuentra disponible en el sitio web de WSO2 [27], donde se podrá descargar el fichero *WSO2-Integration-Studio-7.1.0-linux-gtk-x86_64.zip*. Este fichero comprimido albergará en su interior todo lo necesario para la ejecución sin necesidad de realizar ninguna instalación. Para ello, se deberá extraer en el directorio deseado por la persona que vaya a hacer uso del mismo y ejecutar el fichero *IntegrationStudio*, lo que arrancará la aplicación.

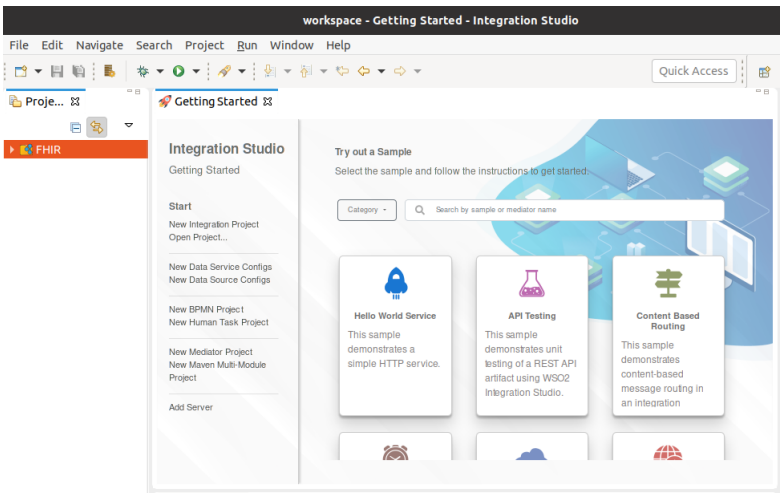

Figura B.1 Interfaz de inicio de Integration Studio.

Configuración y desarrollo de la aplicación

Una vez se ha arrancado Integration Studio, se procede a diseñar y desarrollar los distintos escenarios de prueba contemplados en el trabajo.

Para ello, en primer lugar, se genera un nuevo proyecto con la opción *New Integration Project* incluida en el menú inicial de la aplicación. Esto lanzará una nueva ventana para la creación y configuración del proyecto. En esta ventana se deberá otorgar un nombre a dicho proyecto y se podrán seleccionar los elementos iniciales incluidos en el proyecto. En este caso, los elementos que contendrán el proyecto serán un módulo ESB para lo que se deberá seleccionar la opción *Create ESB Configs* y un módulo para exportar la aplicación y poder desplegarla en el Enterprise Integrator, mediante la opción *Create Composite Exporter*. Por defecto, los distintos elementos que se incluyen en el proyecto serán nombrados partiendo del nombre indicado al proyecto, que en este caso será *FHIR* por el contexto del trabajo. También se deberá indicar un directorio en el que se almacenará dicho proyecto, indicándose uno por defecto. Una vez indicada esta información, se podrá pulsar el botón *Finish* para que el proyecto sea generado.

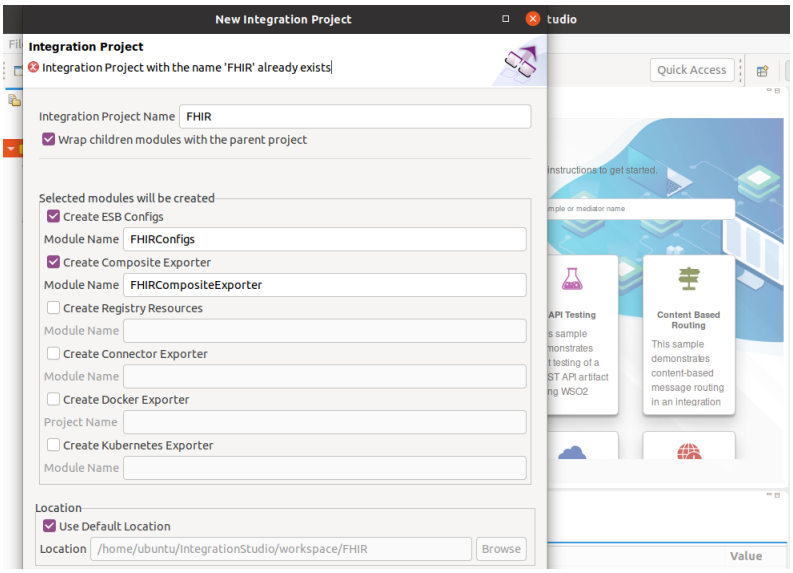

Figura B.2 Creación del proyecto en Integration Studio.

Con el proyecto creado, se deberán crear los elementos que contendrá la aplicación. En este trabajo, se quiere desarrollar una API que actúa como fachada o puerta de acceso a distintos recursos FHIR y que además gestiona la lógica para realizar comprobación de tokens, control de errores y distinta lógica y orquestación de los mensajes dependiendo del escenario.

Para esto, pulsando con el botón derecho del ratón sobre el elemento *FHIRConfigs* que se ha debido generar en el explorador de proyectos, se deberá seleccionar la opción *New* en el desplegable obtenido y posteriormente en la opción *REST API*, con lo que se podrá generar una API en el proyecto. Al seleccionar la opción, aparecerá un nuevo menú para generar y configurar la API, donde se deberá indicar *Create A New API Artifact* en la primera ventana del menú y pulsar sobre el botón *Next* para avanzar por el asistente. En la nueva ventana se deberá indicar el nombre en el apartado *Name* y un *Context*, que será la ruta raíz para acceder a la API. Una vez indicado esto, se deberá pulsar en el botón *Finish* para terminar de generar el fichero que contendrá la API.

Una vez finalizado el asistente de creación de la API, se generará el fichero correspondiente en la ruta *FHIRConfigs/src/main/synapse-config/api*. A partir de aquí se podrán desarrollar los distintos recursos expuestos por la API y su lógica interna. Esto puede hacerse de forma gráfica mediante el arrastre de los distintos *mediadores* y elementos que ofrece Studio en forma de paleta y que ya han sido presentados en su apartado correspondiente, o bien, por medio de código XML, para lo cual se deberá pulsar la pestaña *Source*, disponible en la parte inferior de la ventana de edición.

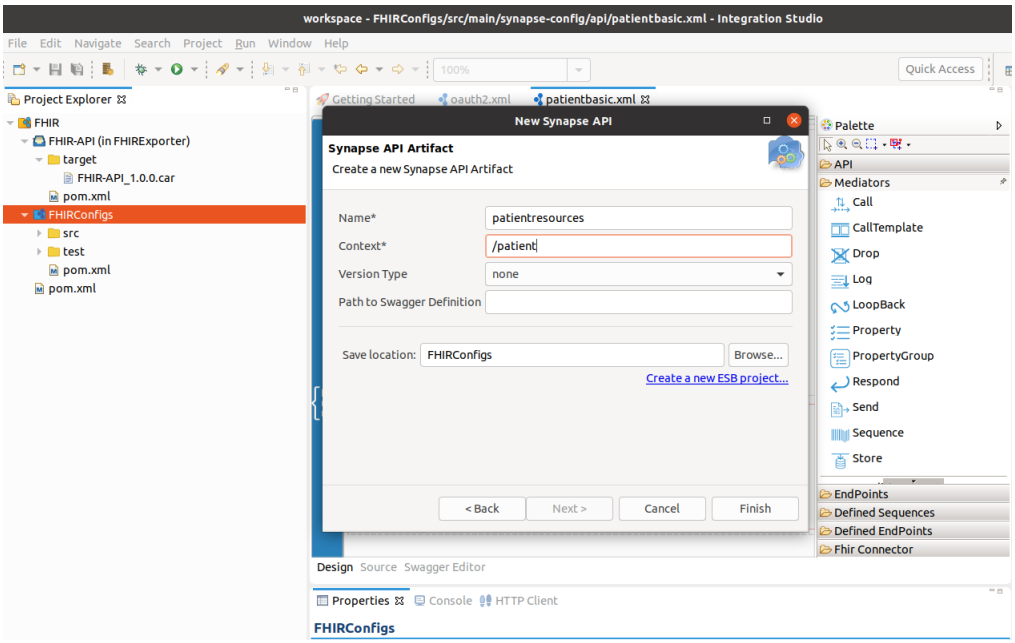

Figura B.3 Creación de una API con Integration Studio.

En este punto se podrán desarrollar los distintos escenarios contemplados en el trabajo incorporando y configurando los distintos elementos necesarios dependiendo de cada caso.

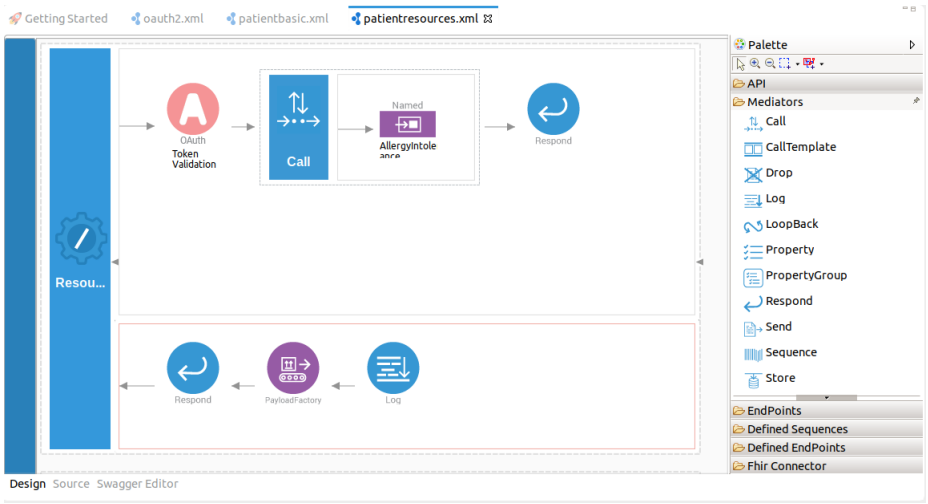

Figura B.4 Vista gráfica de un escenario en Integration Studio.

Exportación de la aplicación

Cuando han sido desarrollados los distintos escenarios que componen el trabajo (códigos adjuntos al final del trabajo) se podrá exportar y generar un fichero en formato *.car* que podrá ser importado y desplegado en el Enterprise Integrator.

Para la exportación de la aplicación se hace uso de uno de los componente del proyecto generados durante su creación, el *Composite Exporter*. Para ello, se deberá pulsar con el botón derecho del ratón en el elemento *FHIRExporter*, disponible en el menú de navegación del proyecto, y posteriormente en la opción *Export...*. Esto lanzará un nuevo asistente en el que se deberá elegir el tipo de exportación que se desea realizar, que para este trabajo será *Composite Application Archive [CAR]* dentro de la carpeta *WSO2*.

Figura B.5 Vista de código de un escenario en Integration Studio.

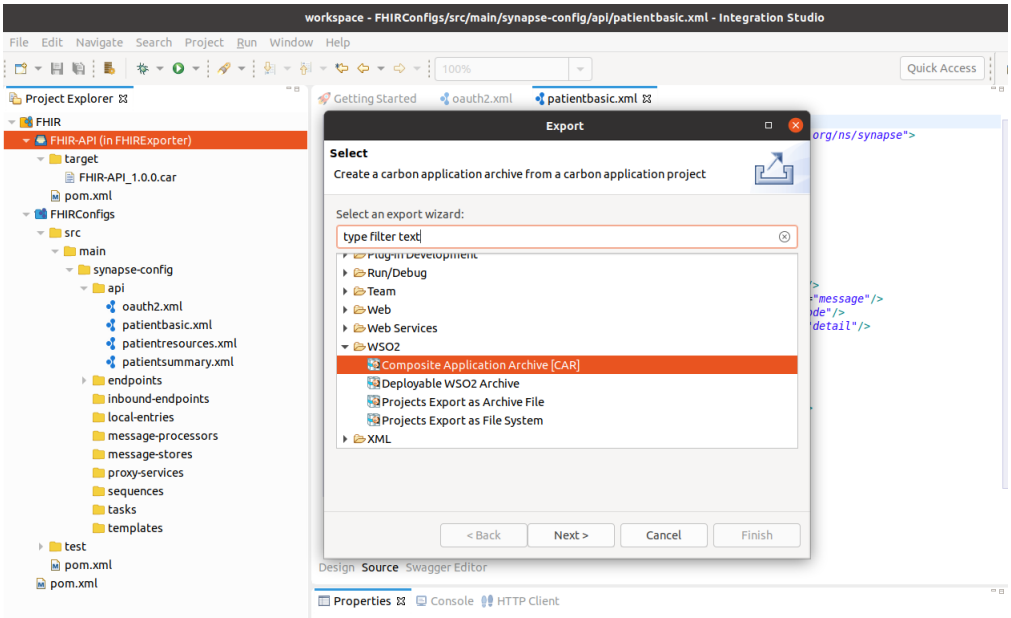

Figura B.6 Exportación de la aplicación desarrollada con Integration Studio.

Con el tipo de exportación seleccionado, se desplegará una nueva ventana donde se deberán indicar distintos parámetros del fichero de exportación: su nombre, su versión y la carpeta destino donde se desea realizar la exportación. Una vez indicados estos parámetros, se deberá pulsar el botón *Next*, avanzando en el menú a una nueva página en la cual se deberán especificar los elementos del proyecto que se desean exportar. Pulsando en el *check* de la carpeta raíz se podrán seleccionar todos los elementos del proyecto desarrollado para realizar la exportación. Una vez hecho esto, pulsando en el botón *Finish* se completará la exportación y se generará el fichero con la aplicación.

Una vez se ha obtenido el fichero con la aplicación, se podrá acudir al Enterprise Integrator e importarla, como se indicó en el anexo anterior.

WSO2 Integration Studio es una herramienta que permite multitud de opciones que no son empleadas en este trabajo, por lo que se recomienda acceder a la documentación oficial para obtener más información [28].

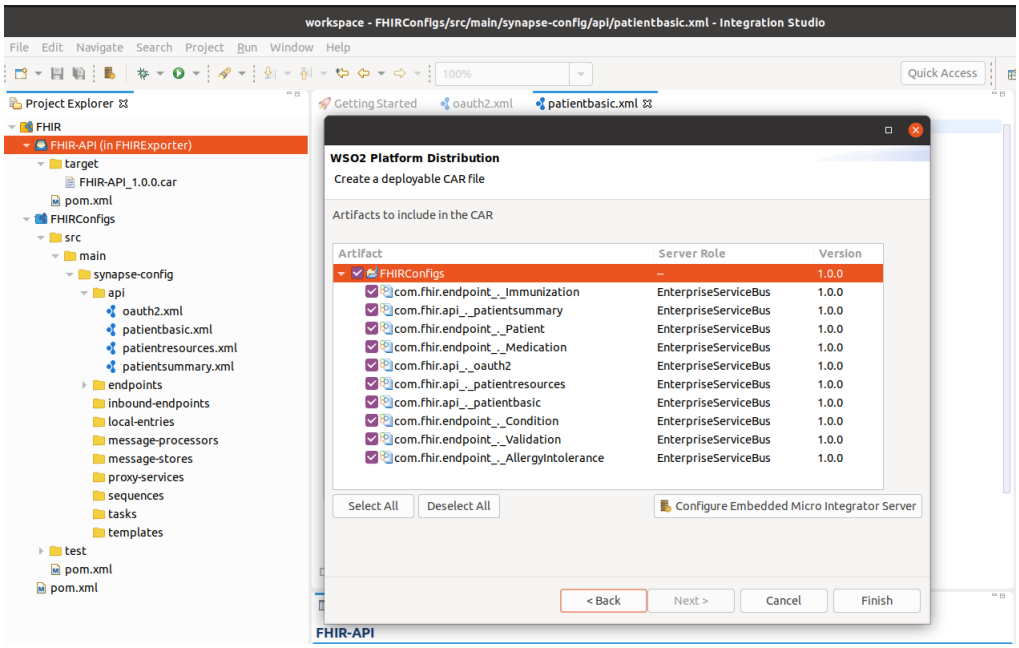

Figura B.7 Listado de elementos a exportar en la aplicación.

C Instalación y configuración de WSO2 Identity Server

Instalación

El proceso de instalación y configuración de WSO2 Identity Server empieza, una vez más, por acceder a su página oficial y descargando la herramienta [29]. En esta ocasión, se ha descargado la última versión que se corresponde con la 5.10.0 en el momento de realización de este trabajo. Una vez en el sitio web de WSO2, se descarga el fichero *wso2is-windows-installer-x64-5.10.0.msi* y se instala mediante su asistente, de manera similar al Enterprise Integrator.

Configuración básica

Una vez instalado, puesto que todos los componentes de WSO2 emplean el puerto 9443 por defecto para su funcionamiento y ya que se ha instalado en el mismo equipo que aloja al Enterprise Integrator, se deberá modificar el puerto para no provocar un conflicto. Para ello, se deberá acceder al documento *deployment.toml*, que se puede encontrar en el directorio *<WSO2_HOME>\Identity Server\5.10.0\repository\conf* y modificar el *offset* que por defecto se encuentra con valor a 0. WSO2 emplea este parámetro para establecer el puerto de funcionamiento de sus servicios, sumando a su puerto por defecto (9443) el valor de dicho parámetro, por lo que con darle un valor diferente al 0 cambiará el puerto del servicio. En este caso, para seguir el orden, se ha modificado el valor de 0 a 1, lo que repercute en que el puerto de funcionamiento del servicio será el 9444.

deployment.toml

```
1  [server]
2  hostname = "localhost"
3  node_ip = "127.0.0.1"
4  base_path = "https://$ref{server.hostname}:${carbon.management.port}"
5  offset = "1"
6
7  [admin_service.wsdl]
8  enable= true
9
```

Figura C.1 Fichero "deployment.toml" modificado de WSO2 Identity Server.

Una vez realizado y guardado el cambio, se podrá ejecutar el servicio Identity Server en el equipo. Para ello, se deberá ejecutar el fichero *launcher_wso2server.bat*, abriéndose un terminal que mostrará información sobre la ejecución del servicio, de forma análoga al EI. Una vez terminado de iniciarse el servicio, se mostrará la URL de acceso a la interfaz de administración del servicio, observándose el valor del puerto modificado que se ha indicado anteriormente.

Similar al procedimiento realizado con el servicio Enterprise Integrator, accediendo mediante un navegador a la URL indicada en la terminal, se accederá a la pantalla de inicio de sesión del servicio, donde se deberán

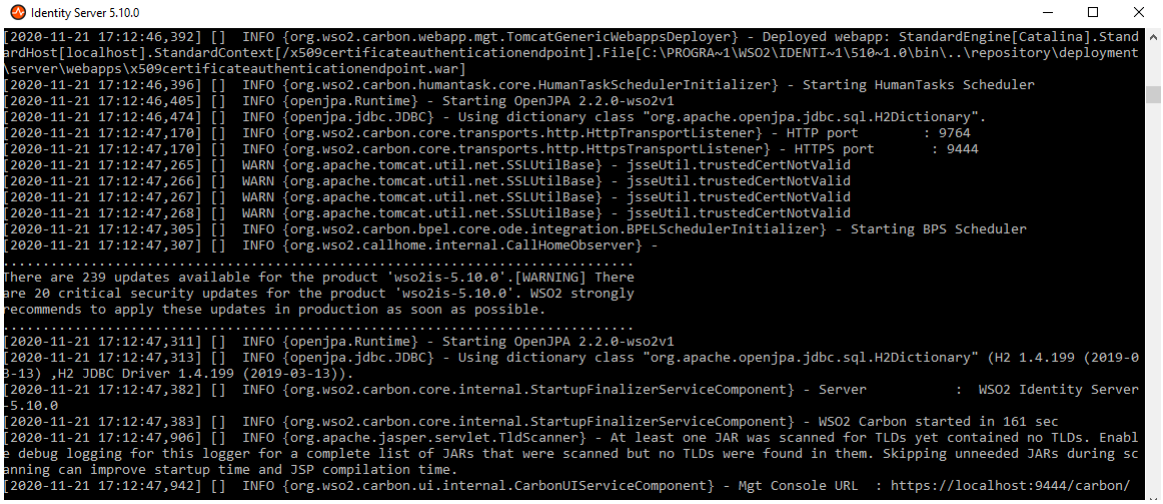

Figura C.2 Terminal de WSO2 Identity Server.

indicar los mismos valores de usuario y contraseña por defecto que los empleados anteriormente, es decir, usuario *admin* y contraseña *admin*.

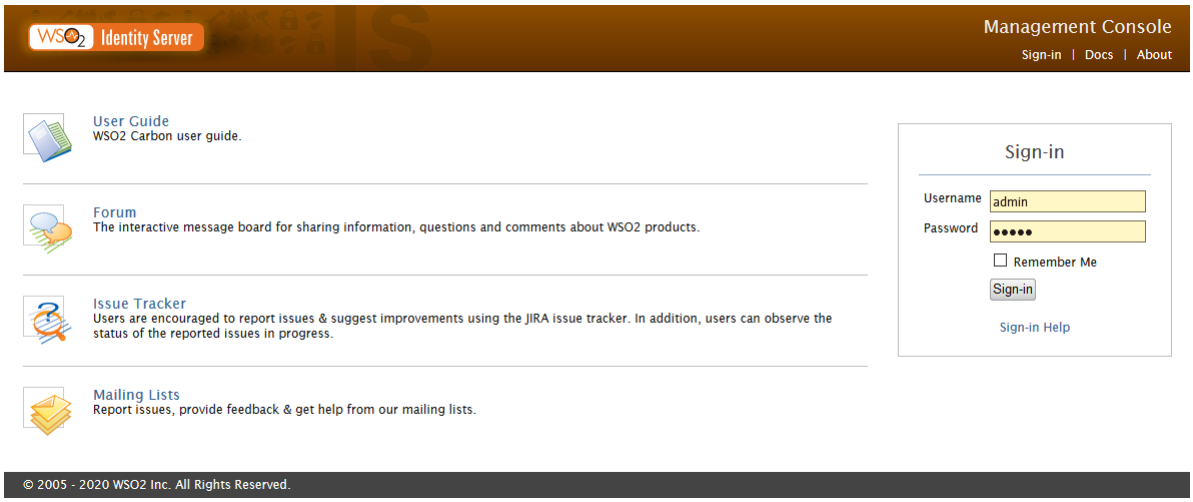

Figura C.3 Pantalla de inicio de sesión del Identity Server.

Una vez introducidas las credenciales de acceso, se mostrará la interfaz principal del servicio, donde se puede encontrar información del servicio en su parte central y las distintas herramientas en el menú lateral, alojado en la parte izquierda. En esta ocasión, las opciones que resultan de interés para el desarrollo de este trabajo son las correspondientes con *Users and Roles* y *Service Providers* en el apartado *Identity* y *PAP* junto con *PDP* en el apartado *Entitlement*, donde las primeras se centran en la identidad y autenticación de usuarios y la segunda en la autorización y control de acceso basado en políticas.

Configuración de usuarios y roles

En primer lugar, en el apartado *Users and Roles* se gestionan los roles y usuarios del sistema, pudiendo crearlos, modificarlos o eliminarlos según la necesidad. Antes de comenzar a crear usuarios, es conveniente definir los distintos roles que se desean emplear en el sistema, en este caso, se generarán un rol *médico* y otro *enfermero*. Para ello, seleccionando la opción *Add* en el menú lateral izquierdo de la interfaz se accede a la interfaz de creación de usuarios y roles.

Una vez aquí, se deberá seleccionar por el momento la opción *Add New Role* para generar un nuevo rol.

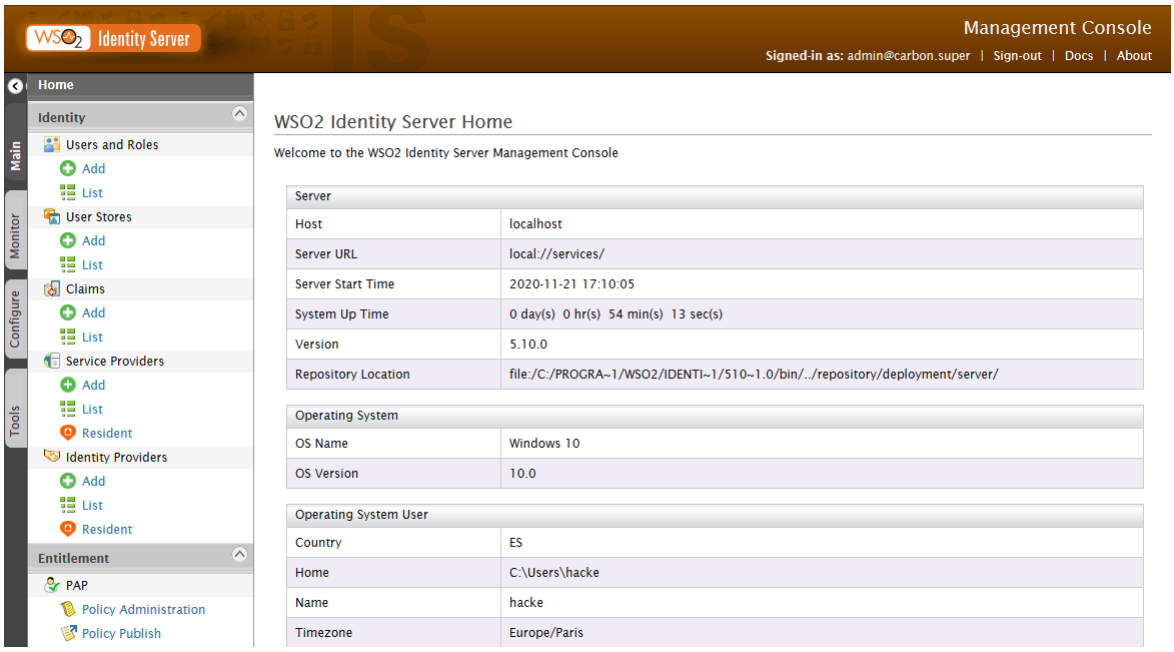

Figura C.4 Interfaz principal Identity Server.

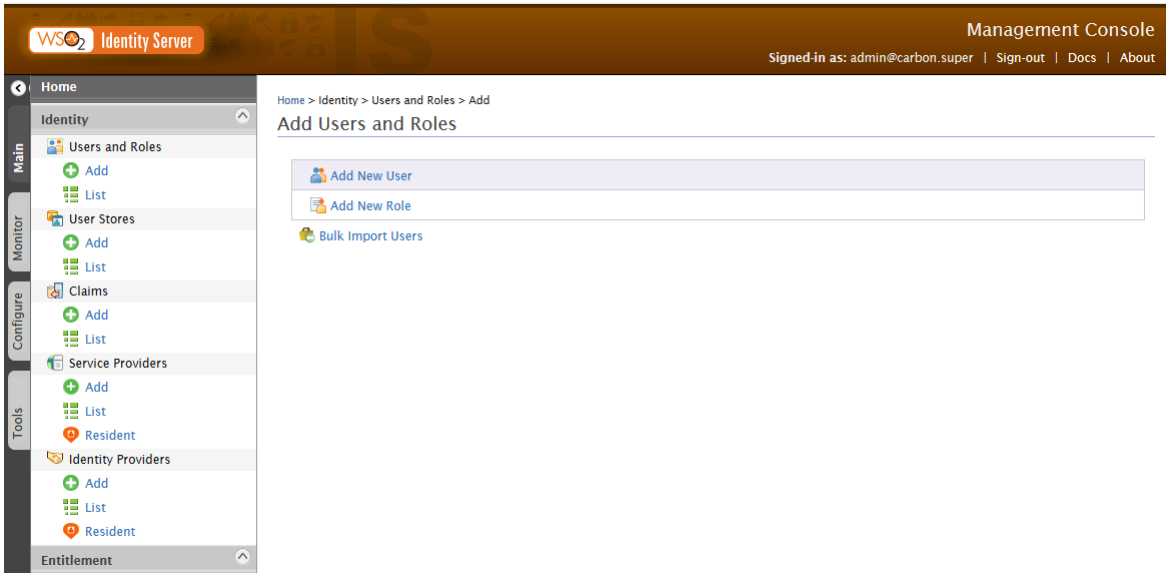

Figura C.5 Interfaz para la creación de usuarios y roles.

En los siguientes párrafos se empleará la opción *Add New User* para generar varios usuarios en el sistema. Cuando se ha seleccionado la opción para crear un nuevo rol aparecerá una nueva página en la que se deberá seleccionar, por un lado el dominio, que para los roles de los usuarios se dejará con el valor *PRIMARY*, y por el otro lado el nombre, que serán los indicados anteriormente. Pulsando en la opción *Next*, se abrirá un menú en el que se podrán indicar los distintos permisos asociados a cada rol, pero para este trabajo se dejará el valor indicado por defecto. Pulsando el botón *Finish* quedará el rol creado y disponible para su uso.

Para la creación de usuarios, una vez creados los roles, accediendo nuevamente al menú de creación de usuarios y roles, se deberá seleccionar la opción *Add New User*, desplegando de este modo el asistente para la creación de los distintos usuarios que se emplearán en el trabajo. En primer lugar, se deberá indicar el dominio, donde en este caso y por defecto solo dejará elegir la opción *PRIMARY*, el usuario, la contraseña y la confirmación de la contraseña. Pulsando el botón *Next*, avanzará el asistente y permitirá seleccionar un rol para asociarlo al usuario que se está creando, donde se podrán seleccionar los roles recientemente

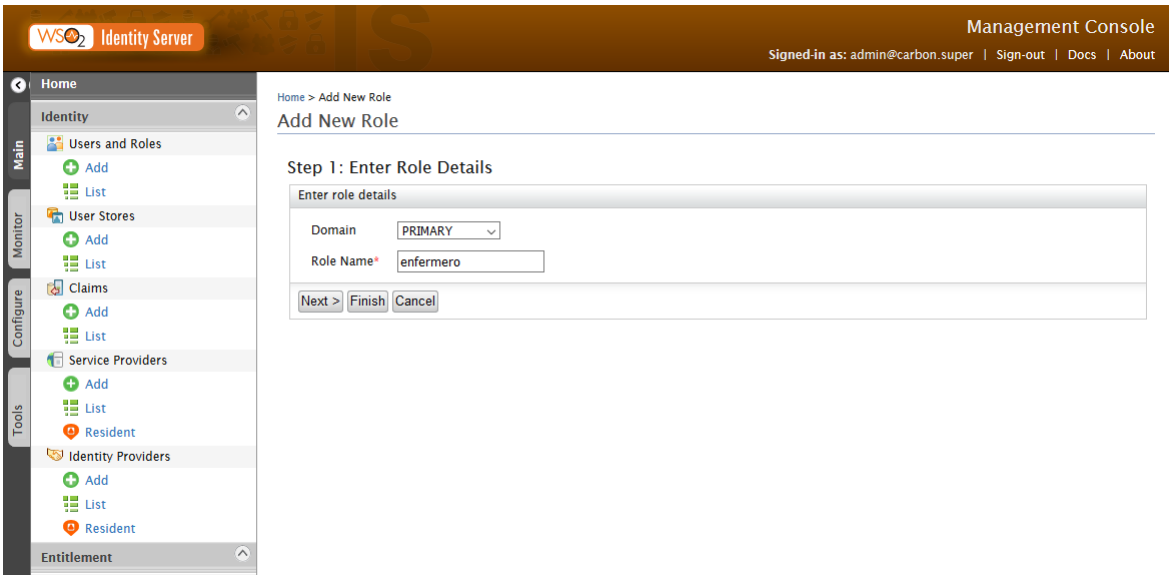

Figura C.6 Creación del rol *enfermero*.

creados. Una vez seleccionado el rol, se deberá pulsar el botón *Finish* para concluir la creación del usuario, que quedará asociado al rol indicado. Para este trabajo se definirán los usuarios *Fulanito*, con rol *médico* y *Menganito* con rol *enfermero*.

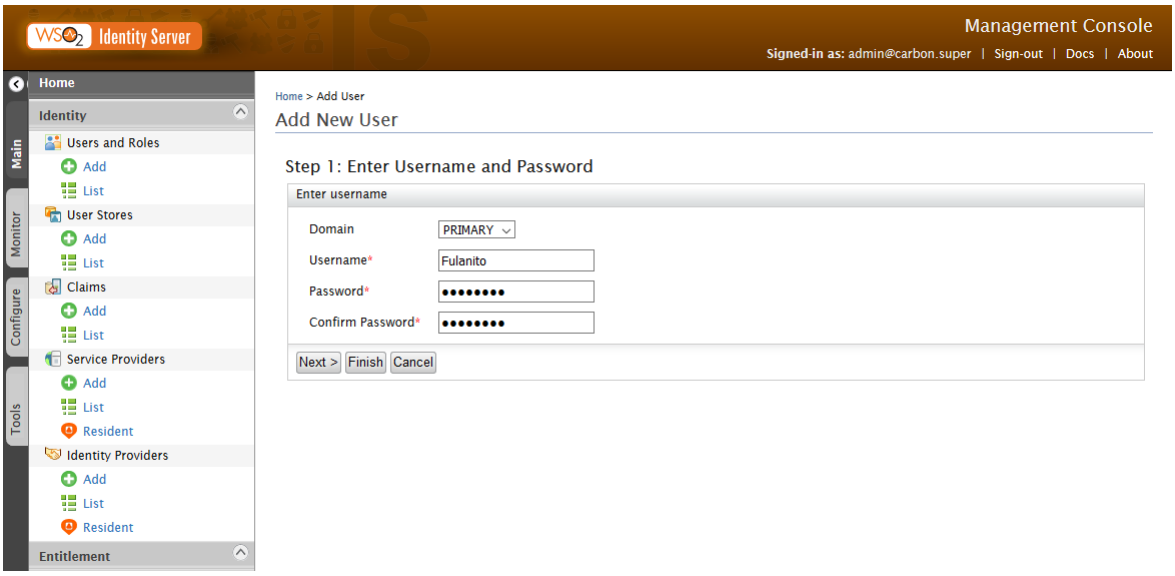

Figura C.7 Creación del usuario *Fulanito*.

Con esto, quedarán creados los distintos usuarios y roles contemplados en los escenarios.

Configuración del proveedor de servicio

El siguiente paso será configurar el proveedor de servicio (*Service Provider*) que permitirá a los usuarios autenticarse y solicitar un token mediante OAuth 2.0 para obtener acceso a los distintos recursos. Para ello, se deberá acceder desde el menú lateral de la interfaz del IS a la opción *Add* de la sección *Service Providers*. Esto desplegará un asistente de configuración para la creación del proveedor de servicio donde se deberá indicar el tipo de configuración (en este trabajo se hará de forma manual) e indicar el nombre y una descripción para el proveedor.

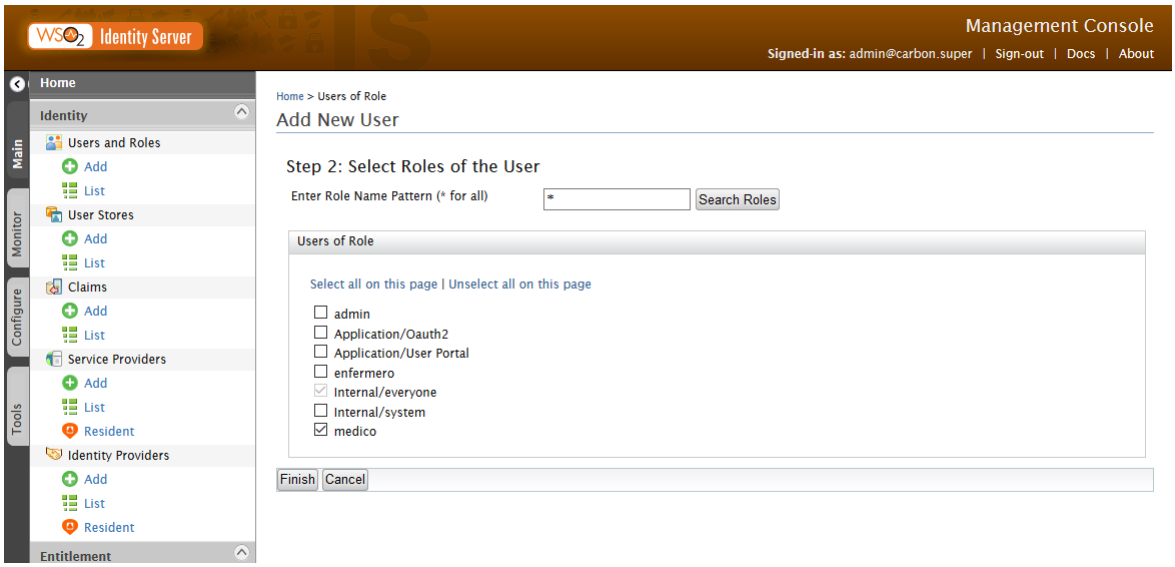

Figura C.8 Asociación del usuario *Fulanito* al rol *médico*.

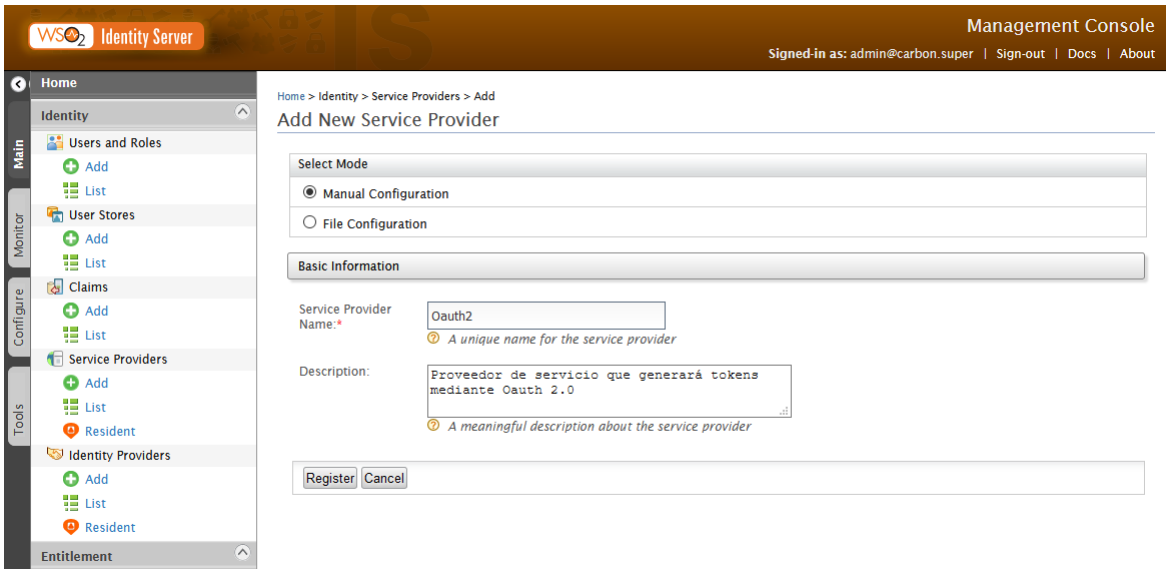

Figura C.9 Creación de un Service Provider en Identity Server.

Una vez introducida esta información, pulsando el botón *Register* se creará el proveedor de servicio y se mostrará una nueva ventana con la información básica indicada en el punto anterior y un menú de configuración para los distintos elementos del proveedor de servicio. Este trabajo se ha centrado en el uso del proveedor de servicio para realizar una autenticación básica y la expedición de tokens con OAuth pero este menú cuenta con muchas otras opciones de configuración posibles para lo cual se recomienda al lector que acceda a la documentación de WSO2 si está interesado en profundizar en alguna de ellas [30].

Para la configuración del proveedor de servicio, se deberá indicar la URL de acceso al proveedor en la opción *Access URL*. En este trabajo se ha indicado la URL *https://localhost:9444/oauth2*, correspondiente con la dirección local y puerto del IS y el nombre del proveedor por sencillez.

Como se indicaba en su apartado correspondiente, HL7 FHIR recomienda el uso de OAuth 2.0 para la autenticación, por lo que se ha optado por esta opción para el desarrollo de los escenaros prácticos. Para ello, se deberá acceder al apartado *OAuth/OpenID Connect Configuration* dentro del menú *Inbound Authentication Configuration* y pulsando el botón *Configure* la primera vez que se accede a dicho apartado.

En esta nueva sección se configurará el uso de OAuth, pudiendo seleccionar la versión del protocolo a

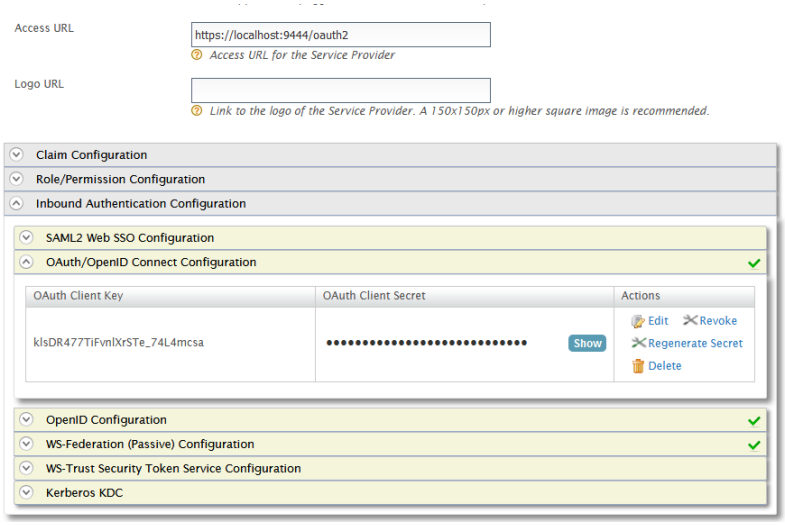

Figura C.10 Interfaz de configuración del Service Provider.

implementar, el tipo de flujos permitidos para la obtención de token o *Grant Type* y otra serie de parámetros que pueden ser útiles dependiendo de la necesidad del servicio, como la duración de los tokens o el algoritmo de encriptación del mismo. Por simplicidad, únicamente se ha marcado la casilla *XACML Scope Validator* que se empleará para contemplar la validación de *scopes* al realizar el proceso de autorización con políticas XACML que se realizará posteriormente. Además, se ha indicado la URL *https://localhost:9444/oauth2/token* para la redirección de las peticiones una vez autorizadas, de modo que obtengan el token solicitado.

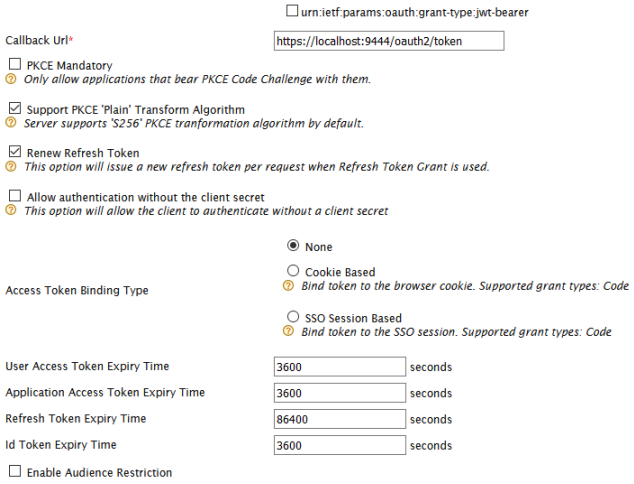

Figura C.11 Configuración de OAuth 2.0 en el Service Provider.

Una vez realizada la configuración se deberá pulsar el botón *Add* o *Update* dependiendo de si es la primera vez que se realiza la configuración o es una modificación. Con esto quedará configurado el proveedor de servicio para trabajar con OAuth 2.0 para la autenticación y generación de tokens. Como se observa en la figura C.10, una vez configurado el apartado anterior, se mostrarán los atributos *OAuth Client Key* y *OAuth Client Secret* que serán el identificador y la clave secreta propias de la aplicación y que se deberán configurar en el cliente para poder solicitar los tokens de acceso.

Configuración de políticas

Por último, para el control de acceso se realizará autorización basada en la aplicación de políticas XACML a las peticiones recibidas.

La gestión y configuración de las políticas en el IS se realiza en el apartado *Entitlement*.Para su configuración, en primer lugar, se debe acceder al PAP mediante la opción *Policy Administration* en el menú lateral izquierdo de la interfaz del IS. Dentro del PAP se gestionan las políticas, pudiendo añadir nuevas o visualizar y editar las que ya estén almacenadas.

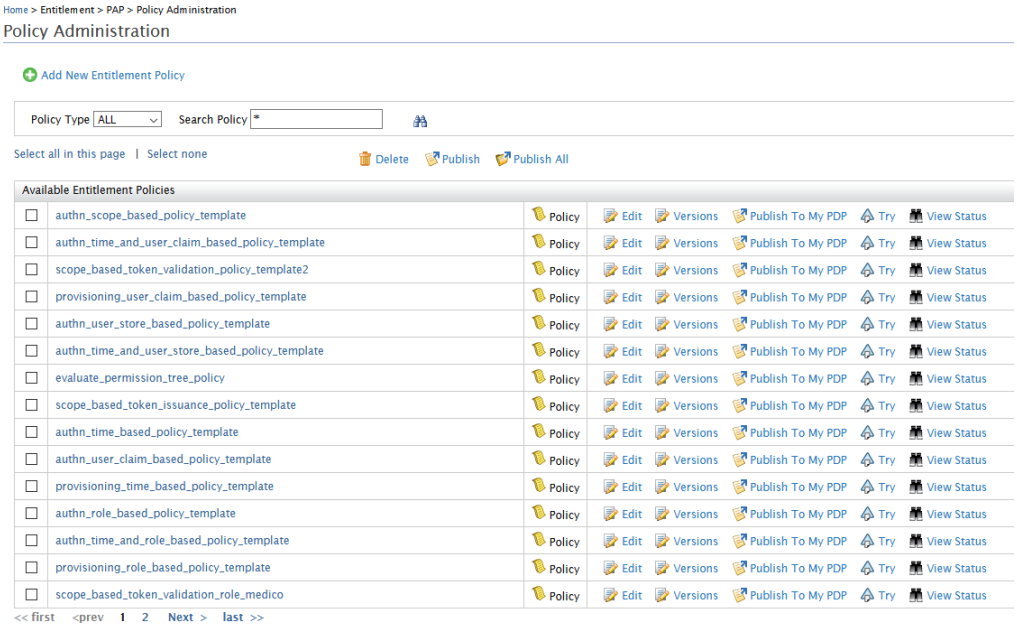

Figura C.12 PAP en Identity Server.

Para la generación de políticas en este trabajo, se ha tomado como referencia la política *scope_based_token_-validation_policy_template*, incluida por defecto en el IS en la versión 5.10 utilizada para la construcción de los escenarios de prueba. Dichas políticas se han adaptado mediante el editor XML incluido en el PAP para evaluar los distintos *claims* asociados al proveedor de servicio, acción, rol y *scopes* que incluye WSO2 en el IS por defecto.

Una vez creadas las políticas que se emplean en los distintos escenarios, se procede a publicarlas en el PDP para su aplicación durante el proceso de validación de token y realizar el control de acceso. Para ello, se debe pulsar en el botón *Publish To My PDP* disponible a la derecha de cada política, lo que lanzará una nueva ventana con un formulario con opciones para la publicación de la política deseada. Estas opciones contemplan:

- Acción de publicación: permite indicar la acción que se realiza sobre la política, pudiendo ser añadir, actualizar, ordenar, habilitar, deshabilitar o eliminar en el PDP.
- Estado de publicación: indica el estado en el que se publicará la política, pudiendo ser habilitado o deshabilitado.
- Establecer versión: permite indicar la versión de la política, pudiendo ser una nueva versión al publicar o una anterior.
- Establecer orden: durante la publicación se indicar el orden en el que se debe aplicar la nueva política, pudiendo indicar el orden por defecto o eligiéndolo manualmente.

Las distintas políticas publicadas se pueden encontrar en el PDP, donde se podrá visualizar, activar o desactivar, modificar su orden de aplicación eliminar. Para acceder a él, se debe pulsar en la opción *Policy View* del menú del IS. En la figura C.14 se puede observar la interfaz del PDP con las distintas políticas publicadas y activas.

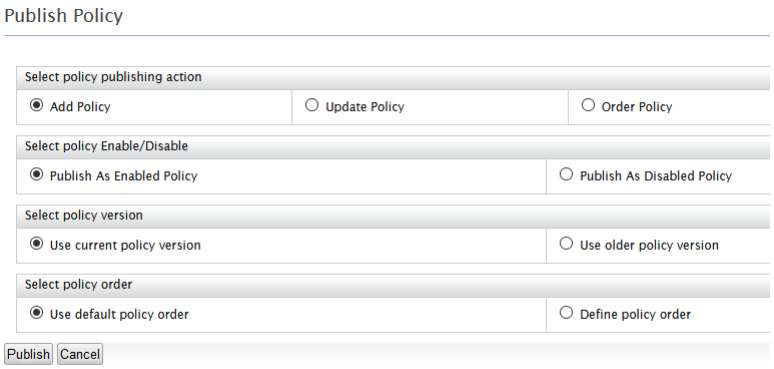

Figura C.13 Menú de publicación de políticas en el IS.

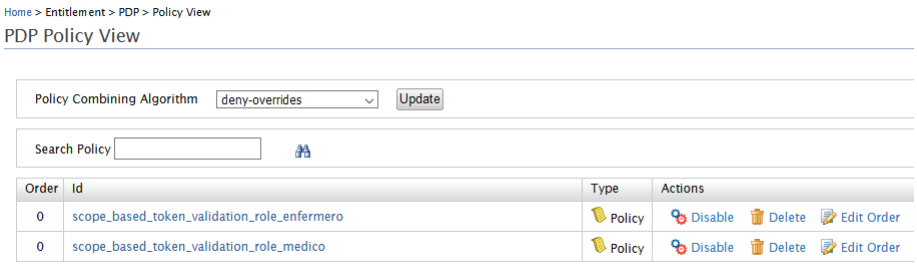

Figura C.14 PDP en Identity Server.

D Instalación y configuración de Postman

Instalación

Para la instalación y configuración de Postman API Client, en primer lugar, será necesario descargar desde la web oficial el fichero instalable [31], que en el momento de desarrollo de este trabajo es *Postman-win32-7.36.0-Setup.exe*.

Una vez descargado se deberá ejecutar el fichero y seguir el asistente de instalación para realizar una instalación correcta en el equipo.

Después de instalarse, se podrá lanzar la aplicación, observando su pantalla inicial en la figura D.1.

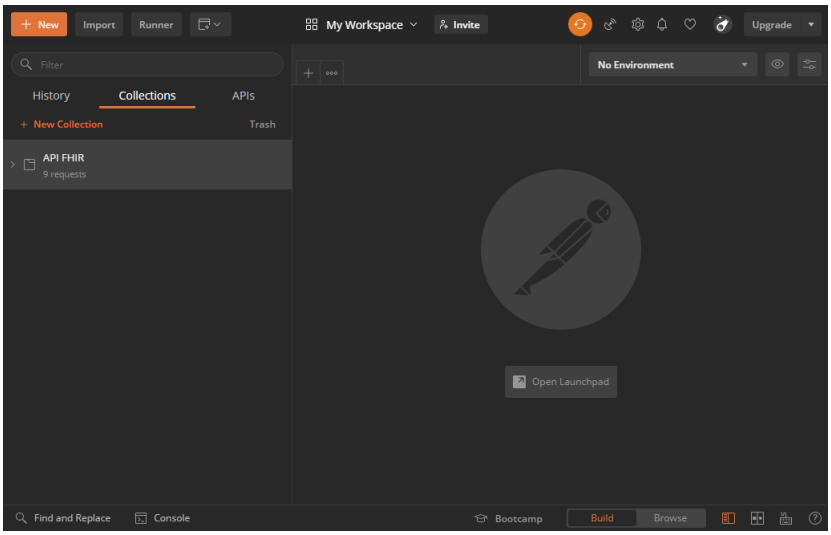

Figura D.1 Interfaz inicial Postman.

Configuración de las peticiones

Para configurar los distintos escenarios de prueba, en primer lugar, se creará una colección, que albergará las distintas operaciones REST que se realizarán en el trabajo. Para ello, se deberá seleccionar el botón *New* presente en la esquina superior izquierda de la aplicación. Una vez pulsado, se mostrará una ventana con distintas opciones, en la que se deberá seleccionar *Collection* para generar una nueva colección. Una vez hecho esto, se deberá otorgar un nombre y opcionalmente una descripción para crear dicha colección y comenzar a trabajar.

Con la colección creada, se deberán generar las distintas peticiones (*Request*) para lanzar las solicitudes a los distintos recursos. Para ello, se deberá pulsar nuevamente el botón *New* antes mencionado, seleccionando

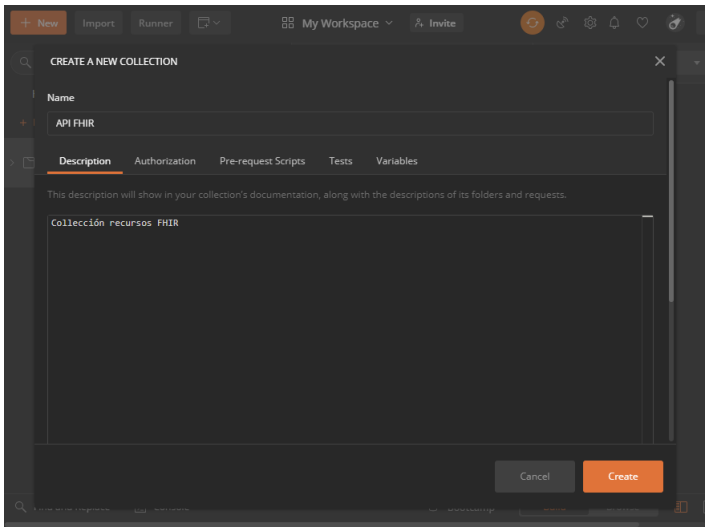

Figura D.2 Generación de una nueva colección en Postman.

la opción Request en esta ocasión. Se desplegará una nueva página donde se deberá indicar el nombre que se le quiere otorgar a la petición y una descripción opcional. Además, se deberá seleccionar la colección anteriormente generada para crear la petición dentro de esta. Con esto, se generará la petición dentro de la colección.

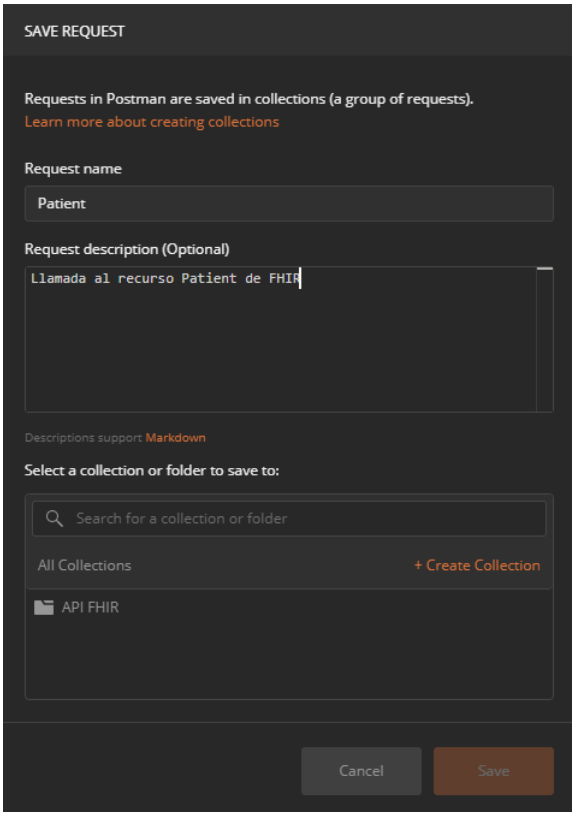

Figura D.3 Generación de una nueva petición en Postman.

El siguiente paso será configurar la petición para lanzar las consultas de forma correcta:

- En primer lugar, se deberá indicar la operación a realizar, ya sea *GET*, *POST*, *PUT*, *DELETE*, etc...

- En segundo lugar, se debe indicar la URL a la que se quiere lanzar la petición. La URL deberá coincidir con la indicada en Enterprise Integrator una vez que se ha desplegado la aplicación desarrollada y con la ruta del recurso que se desee consultar, incluyendo en el caso de este trabajo, el identificador del recurso concreto a consultar.

- Por último, se deberán indicar, en caso de que corresponda, distintas cabeceras para transmitir información deseada. En distintos escenarios de este trabajo será necesario indicar la cabecera *Authorization* para transmitir el token de autorización.

- Adicionalmente, en el caso de que la operación de la petición fuese de tipo *POST* o *PUT*, se debería añadir un elemento *Body* que debería contener el recurso a enviar con la petición.

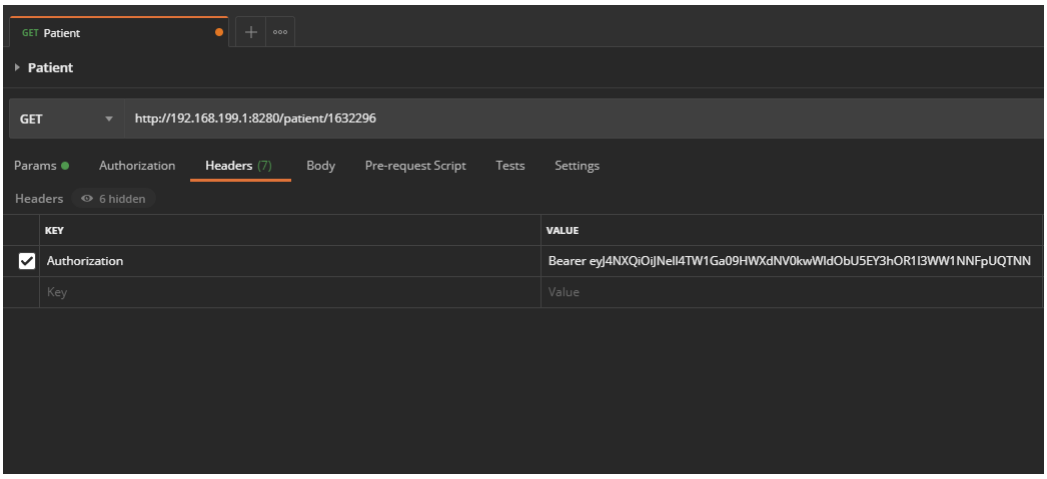

Figura D.4 Configuración de los parámetros de una petición en Postman.

Para realizar la autenticación y obtener un token de autorización mediante OAuth 2.0 para consumir los recursos, se deberá generar, además de las peticiones de cada recurso deseado, una petición concreta para realizar la autenticación y obtención de autorización. Para ello, se deberá generar una petición concreta en la que habrá que configurar la pestaña *Authorization* como se muestra en la figura 5.33.

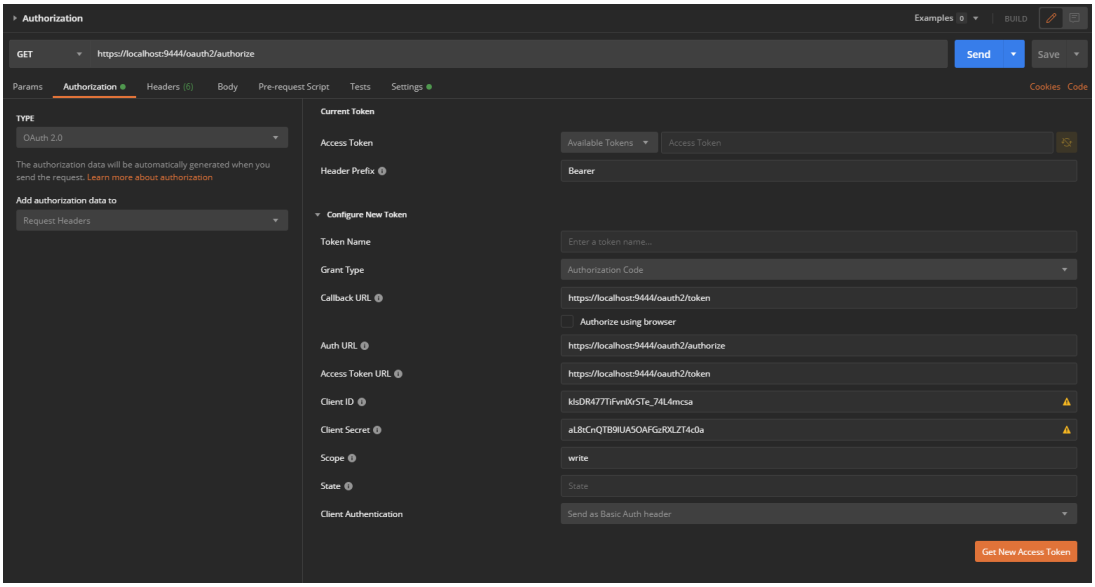

Figura D.5 Configuración de los parámetros para la obtención de token en Postman.

La configuración realizada es la siguiente:

- **Type:** Se deberá seleccionar *OAuth 2.0*, que será el protocolo empleado en este trabajo.
- **Grant Type:** Se seleccionará *Authorization Code* para indicar el tipo de token a devolver.
- **Callback URL:** Se indicará la URL a la que debe ser redirigida la solicitud una vez authorizada.
- **Auth URL:** URL del servicio de autorización.
- **Access Token URL:** Será la URL del servicio de autenticación para la obtención de token.
- **Client ID:** Se debe indicar el identificador del proveedor de servicio generado durante su configuración para identificar a la aplicación.
- **Client Secret:** Se debe indicar la clave secreta de la aplicación, generada junto con el *Client ID*.
- **Scope:** Atributos que deberá incorporar el token que se podrán emplear posteriormente para comprobar permisos.

E Ficheros

Integration Studio

Código E.1 API Oauth2 para una validación más detallada del token (oauth2.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<api context="/oauth2" name="oauth2" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="GET" uri-template="/token">
        <inSequence>
            <payloadFactory media-type="xml">
                <format>
                    <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
                        envelope" xmlns:xsd="http://org.apache.axis2/xsd" xmlns:
                        xsd1="http://dto.oauth2.identity.carbon.wso2.org/xsd">
                        <soap:Header/>
                        <soap:Body>
                            <xsd:validate>
                                <!--Optional:-->
                                <xsd:validationReqDTO>
                                    <!--Optional:-->
                                    <xsd1:accessToken>
                                        <!--Optional:-->
                                        <xsd1:identifier>$1</xsd1:identifier>
                                        <!--Optional:-->
                                        <xsd1:issuer>?</xsd1:issuer>
                                        <!--Optional:-->
                                        <xsd1:tokenType>Bearer</xsd1:tokenType>
                                    </xsd1:accessToken>
                                    <!--Zero or more repetitions:-->
                                    <xsd1:context>
                                        <!--Optional:-->
                                        <xsd1:key>?</xsd1:key>
                                        <!--Optional:-->
                                        <xsd1:value>?</xsd1:value>
                                    </xsd1:context>
                                    <!--Zero or more repetitions:-->
                                    <xsd1:requiredClaimURIs>?</xsd1:
                                        requiredClaimURIs>
                                </xsd:validationReqDTO>
                            </xsd:validate>
                        </soap:Body>
```

```
                    </soap:Envelope>
                </format>
                <args>
                    <arg evaluator="xml" expression="$trp:Authorization"/>
                </args>
            </payloadFactory>
            <call>
                <endpoint>
                    <address format="soap11" uri="https://localhost:9444/services
                        /OAuth2TokenValidationService.
                        OAuth2TokenValidationServiceHttpsSoap11Endpoint/">
                        <suspendOnFailure>
                            <initialDuration>-1</initialDuration>
                            <progressionFactor>-1</progressionFactor>
                            <maximumDuration>0</maximumDuration>
                        </suspendOnFailure>
                        <markForSuspension>
                            <retriesBeforeSuspension>0</retriesBeforeSuspension>
                        </markForSuspension>
                    </address>
                    <property name="SOAPAction" scope="transport" value="
                        validation"/>
                    <property name="Authorization" scope="transport" value="Basic
                         YWRtaW46YWRtaW4="/>
                </endpoint>
            </call>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
        </faultSequence>
    </resource>
    <resource methods="POST" uri-template="/prueba">
        <inSequence>
            <payloadFactory media-type="text">
                <format>&lt;inline&gt;
&lt;soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:
    xsd="http://org.apache.axis2/xsd" xmlns:xsd1="http://dto.oauth2.identity.
    carbon.wso2.org/xsd"&gt;&lt;soap:Header/&gt;&lt;soap:Body&gt;&lt;xsd:
    validate&gt;&lt;!--Optional:--&gt;&lt;xsd:validationReqDTO&gt;&lt;!--
    Optional:--&gt;&lt;xsd1:accessToken&gt;&lt;!--Optional:--&gt;&lt;xsd1:
```

```
    identifier&gt;eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3WW1NNFpUQT.
    NNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0Zo.
    TXpVMlpHVmxOZyIsImtpZk16W.
    XhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsT.
    mpKa09ERmtaRFJpT1RGa01XRmhNelUyWkdWbE5nX1JT.MjU2IiwiYWxnIjoiUlMyNTYifQ.
    eyJzdWIiOiJ1c3VhcmlvM.
    SIsImF1ZmtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MDkzMzMs.
    ImF6cmtscORSNDc3VGlGdm5sWHJTVGVf.
    NzRMNG1jc2EiLCJzY29wZSI6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxo.
    b3N0OjkONDNcL29hdXRoMlwvdG9rZW4iLCJleHAiOjE2MDM5MTI5MzMsImlhdCI6MTYwM.
    zkwOTMzMywianRpIjoiZjI1ZTY0ODMtNWUxZi00OTJ.hLWI3M2UtZWU1OGM4NDY5NTljIn0.
    p9SJWSbBA0R9oGR_qXy0iTWhJE1kPF__-
    S31ZhMKAPLbWEmmQjGGTZVdgHnAtiM4jHbqmtWp5J6sFg.
    Ab7CQDqVqygKAZaWt4jjYyAsdQgp6t0gR4PAxO2aRzTs.7fQDaf-1
    LYTiAbNewn2ofOO7kyevizxpvr9Ik76CvlUwkm4NfOAQ3.
    Raz47xSiTLI7SOuFFpQmljD7hKtBfC_sWAXBBnpE1cZPuIqvcoq9D71Q4fibUjlxdKiGIG-
    GhcwVfhQMa28-mTXN-
    cwQzOujTEZP3zof71XMbGtD8TjPBwrjDbAPmEz6x8igEpFsSc8ZNjFpjdBlFwdY_Uaa3ToQox-
    QLy3XA&lt;/xsd1:identifier&gt;&lt;!--Optional:--&gt;&lt;xsd1:issuer&gt;?&lt
    ;/xsd1:issuer&gt;&lt;!--Optional:--&gt;&lt;xsd1:tokenType&gt;Bearer&lt;/
    xsd1:tokenType&gt;&lt;/xsd1:accessToken&gt;&lt;!--Zero or more repetitions
    :--&gt;&lt;xsd1:context&gt;&lt;!--Optional:--&gt;&lt;xsd1:key&gt;?&lt;/xsd1
    :key&gt;&lt;!--Optional:--&gt;&lt;xsd1:value&gt;?&lt;/xsd1:value&gt;&lt;/
    xsd1:context&gt;&lt;!--Zero or more repetitions:--&gt;&lt;xsd1:
    requiredClaimURIs&gt;?&lt;/xsd1:requiredClaimURIs&gt;&lt;/xsd:
    validationReqDTO&gt;&lt;/xsd:validate&gt;&lt;/soap:Body&gt;&lt;/soap:
    Envelope&gt;
&lt;/inline&gt;</format>
                <args/>
            </payloadFactory>
            <call>
                <endpoint key="Validation"/>
            </call>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
        </faultSequence>
    </resource>
</api>
```

Código E.2 API básica sin lógica de autenticación/autorización (patientbasic.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<api context="/basic/patient" name="patientbasic" xmlns="http://ws.apache.org/
    ns/synapse">
    <resource methods="GET" uri-template="/{patientId}">
        <inSequence>
            <call>
                <endpoint key="Patient"/>
            </call>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
        </faultSequence>
    </resource>
</api>
```

Código E.3 API principal con autenticación y autorización (patientresources.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<api context="/patient" name="Patient" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="GET" uri-template="/{patientId}/allergyintolerance">
        <inSequence>
            <oauthService description="Token Validation" password="admin"
                remoteServiceUrl="https://localhost:9444/services/
                OAuth2TokenValidationService.
                OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
                admin"/>
            <call>
                <endpoint key="AllergyIntolerance"/>
            </call>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
```

```
            <property expression="get-property('ERROR_CODE')" name="code"/>
            <property expression="get-property('ERROR_DETAIL')" name="detail
                "/>
        </log>
        <payloadFactory media-type="json">
            <format>{ "status": "Error",
"description": "$1" }</format>
            <args>
                <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
            </args>
        </payloadFactory>
        <respond/>
    </faultSequence>
 </resource>
 <resource methods="GET" uri-template="/{patientId}/composition">
    <inSequence>
        <oauthService description="Token Validation" password="admin"
            remoteServiceUrl="https://localhost:9444/services/
            OAuth2TokenValidationService.
            OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
            admin"/>
        <call>
            <endpoint key="Composition"/>
        </call>
        <respond/>
    </inSequence>
    <outSequence/>
    <faultSequence>
        <log level="custom">
            <property name="text" value="An unexpected error occured"/>
            <property expression="get-property('ERROR_MESSAGE')" name="
                message"/>
            <property expression="get-property('ERROR_CODE')" name="code"/>
            <property expression="get-property('ERROR_DETAIL')" name="detail
                "/>
        </log>
        <payloadFactory media-type="json">
            <format>{ "status": "Error",
"description": "$1" }</format>
            <args>
                <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
            </args>
        </payloadFactory>
        <respond/>
    </faultSequence>
 </resource>
 <resource methods="GET" uri-template="/{patientId}/condition">
    <inSequence>
        <oauthService description="Token Validation" password="admin"
            remoteServiceUrl="https://localhost:9444/services/
            OAuth2TokenValidationService.
            OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
            admin"/>
        <call>
            <endpoint key="Condition"/>
        </call>
        <respond/>
```

```
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
        </faultSequence>
    </resource>
    <resource methods="GET" uri-template="/{patientId}/medication">
        <inSequence>
            <oauthService description="Token Validation" password="admin"
                remoteServiceUrl="https://localhost:9444/services/
                OAuth2TokenValidationService.
                OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
                admin"/>
            <call>
                <endpoint key="Medication"/>
            </call>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
        </faultSequence>
    </resource>
    <resource methods="GET" uri-template="/{patientId}/immunization">
        <inSequence>
            <oauthService description="Token Validation" password="admin"
                remoteServiceUrl="https://localhost:9444/services/
```

```
            OAuth2TokenValidationService.
            OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
            admin"/>
        <call>
            <endpoint key="Immunization"/>
        </call>
        <respond/>
    </inSequence>
    <outSequence/>
    <faultSequence>
        <log level="custom">
            <property name="text" value="An unexpected error occured"/>
            <property expression="get-property('ERROR_MESSAGE')" name="
                message"/>
            <property expression="get-property('ERROR_CODE')" name="code"/>
            <property expression="get-property('ERROR_DETAIL')" name="detail
                "/>
        </log>
        <payloadFactory media-type="json">
            <format>{ "status": "Error",
"description": "$1" }</format>
            <args>
                <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
            </args>
        </payloadFactory>
        <respond/>
    </faultSequence>
</resource>
<resource methods="GET" uri-template="/{patientId}">
    <inSequence>
        <call>
            <endpoint>
                <http method="get" statistics="enable" uri-template="http
                    ://192.168.199.1:8280/oauth2/token">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
                </http>
            </endpoint>
        </call>
        <property description="GET Response" expression="$body//" name="
            scope" scope="default" type="OM" xmlns:soapenv="http://schemas.
            xmlsoap.org/soap/envelope/"/>
        <call>
            <endpoint key="Patient"/>
        </call>
        <property description="GET Patient" expression="json-eval($)" name="
            patient" scope="default" type="STRING"/>
        <property description="GET Security" expression="json-eval($.meta.
            security[0].code)" name="security" scope="default" type="STRING
            "/>
        <payloadFactory media-type="json">
```

```
                <format>$1</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:scope"/>
                </args>
            </payloadFactory>
            <filter regex="R" source="$ctx:security">
                <then>
                    <filter regex="confidential" source="json-eval($.Body.
                        Envelope.Body.validateResponse.return.scope[0])">
                        <then>
                            <payloadFactory media-type="json">
                                <format>$1</format>
                                <args>
                                    <arg evaluator="xml" expression="$ctx:patient
                                        "/>
                                </args>
                            </payloadFactory>
                            <respond/>
                        </then>
                        <else>
                            <header name="HTTP_SC" scope="transport" value="401"/>

                            <payloadFactory media-type="json">
                                <format>{ "status": "Error",
    "description": "Acceso restringido"}</format>
                                <args/>
                            </payloadFactory>
                            <respond/>
                        </else>
                    </filter>
                </then>
                <else>
                    <payloadFactory media-type="json">
                        <format>$1</format>
                        <args>
                            <arg evaluator="xml" expression="$ctx:patient"/>
                        </args>
                    </payloadFactory>
                    <respond/>
                </else>
            </filter>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
    "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
```

```
            </args>
        </payloadFactory>
        <respond/>
    </faultSequence>
</resource>
<resource methods="POST">
    <inSequence>
        <property description="GET Request" expression="json-eval($)" name="
            request" scope="default" type="STRING"/>
        <call>
            <endpoint>
                <http method="get" statistics="enable" uri-template="http
                    ://192.168.199.1:8280/oauth2/token">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
                </http>
            </endpoint>
        </call>
        <property description="GET Response" expression="$body//" name="
            scope" scope="default" type="OM"/>
        <payloadFactory media-type="json">
            <format>$1</format>
            <args>
                <arg evaluator="xml" expression="$ctx:scope"/>
            </args>
        </payloadFactory>
        <filter regex="write" source="json-eval($.Body.Envelope.Body.
            validateResponse.return.scope[2])">
            <then>
                <payloadFactory media-type="json">
                    <format>$1</format>
                    <args>
                        <arg evaluator="xml" expression="$ctx:request"/>
                    </args>
                </payloadFactory>
                <call>
                    <endpoint>
                        <http method="post" uri-template="http://hapi.fhir.
                            org/baseR4/Patient?_format=json&amp;_pretty=true">
                            <suspendOnFailure>
                                <initialDuration>-1</initialDuration>
                                <progressionFactor>-1</progressionFactor>
                                <maximumDuration>0</maximumDuration>
                            </suspendOnFailure>
                            <markForSuspension>
                                <retriesBeforeSuspension>0</
                                    retriesBeforeSuspension>
                            </markForSuspension>
                        </http>
                        <property name="Content-Type" scope="transport" value
                            ="application/json"/>
```

```
                    </endpoint>
                </call>
                <respond/>
            </then>
            <else>
                <payloadFactory media-type="json">
                    <format>{ "status": "Error",
"description": "Acceso no autorizado. No se dispone del permiso
    correspondiente."}</format>
                    <args/>
                </payloadFactory>
                <respond/>
            </else>
        </filter>
    </inSequence>
    <outSequence/>
    <faultSequence/>
</resource>
<resource methods="GET" uri-template="/{patientId}/summary">
    <inSequence>
        <oauthService description="Token Validation" password="admin"
            remoteServiceUrl="https://localhost:9444/services/
            OAuth2TokenValidationService.
            OAuth2TokenValidationServiceHttpsSoap11Endpoint/" username="
            admin"/>
        <property expression="get-property('uri.var.patientId')" name="
            patientId" scope="default" type="STRING"/>
        <property description="GET Authorization" expression="$trp:
            Authorization" name="token" scope="default" type="STRING"/>
        <call>
            <endpoint>
                <http method="get" uri-template="http://192.168.199.1:8280/
                    patient/{uri.var.patientId}/composition">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
                </http>
            </endpoint>
        </call>
        <property description="GET Composition" expression="json-eval($.
            entry)" name="composition" scope="default" type="STRING"/>
        <header description="SET Authorization" expression="$ctx:token" name
            ="Authorization" scope="transport"/>
        <call>
            <endpoint>
                <http method="get" uri-template="http://192.168.199.1:8280/
                    patient/{uri.var.patientId}/allergyintolerance">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
```

```
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
                </http>
            </endpoint>
        </call>
        <filter regex="0" source="json-eval($.total)">
            <then>
                <payloadFactory media-type="json">
                    <format>[{
  "resource": {
    "resourceType": "AllergyIntolerance",
    "id": "0",
    "meta": {
      "versionId": "1"
    },
    "code": {
      "coding": [{
        "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv-ips",

        "code": "no-allergy-info",
        "display": "No allergy info"
      }]
    },
    "patient": {
      "reference": "Patient/$1"
    }
  }
}]</format>
                    <args>
                        <arg evaluator="xml" expression="$ctx:patientId"/>
                    </args>
                </payloadFactory>
                <property description="Non AllergyIntolerance" expression="
                    json-eval($)" name="allergyintolerance" scope="default"
                    type="STRING"/>
            </then>
            <else>
                <property description="GET AllertgyIntolerance" expression="
                    json-eval($.entry)" name="allergyintolerance" scope="
                    default" type="STRING"/>
            </else>
        </filter>
        <header description="SET Authorization" expression="$ctx:token" name
            ="Authorization" scope="transport"/>
        <call>
            <endpoint>
                <http method="get" uri-template="http://192.168.199.1:8280/
                    patient/{uri.var.patientId}/condition">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
```

```
                    </http>
                </endpoint>
            </call>
            <filter regex="0" source="json-eval($.total)">
                <then>
                    <payloadFactory media-type="json">
                        <format>[{
        "resource": {
          "resourceType": "Condition",
          "id": "0",
          "meta": {
            "versionId": "1"
          },
          "clinicalStatus": [{
            "coding": [{
              "system": "http://terminology.hl7.org/CodeSystem/condition-
                  clinical",
              "code": "active",
              "display": "Active"
            }]
          }],
          "verificationStatus": [{
            "coding": [{
              "system": "http://terminology.hl7.org/CodeSystem/condition-ver-
                  status",
              "code": "confirmed",
              "display": "Confirmed"
            }]
          }],
          "category": [{
            "coding": [{
              "system": "http://loinc.org",
              "code": "75326-9",
              "display": "Problem"
            }],
            "text": "Diagnosis"
          }],
          "code": {
            "coding": [{
              "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv
                  -ips",
              "code": "no-problem-info",
              "display": "No information about current problems"
            }]
          },
          "subject": {
            "reference": "Patient/$1"
          }
        }
}]</format>
                        <args>
                            <arg evaluator="xml" expression="$ctx:patientId"/>
                        </args>
                    </payloadFactory>
                    <property description="No Condition" expression="json-eval($)
                        " name="condition" scope="default" type="STRING"/>
                </then>
```

```
                <else>
                    <property description="GET Condition" expression="json-eval($
                        .entry)" name="condition" scope="default" type="STRING"/>
                </else>
            </filter>
            <header description="SET Authorization" expression="$ctx:token" name
                ="Authorization" scope="transport"/>
            <call>
                <endpoint>
                    <http method="get" uri-template="http://192.168.199.1:8280/
                        patient/{uri.var.patientId}/medication">
                        <suspendOnFailure>
                            <initialDuration>-1</initialDuration>
                            <progressionFactor>-1</progressionFactor>
                            <maximumDuration>0</maximumDuration>
                        </suspendOnFailure>
                        <markForSuspension>
                            <retriesBeforeSuspension>0</retriesBeforeSuspension>
                        </markForSuspension>
                    </http>
                </endpoint>
            </call>
            <filter regex="0" source="json-eval($.total)">
                <then>
                    <payloadFactory media-type="json">
                        <format>[{
  "resource": {
    "resourceType": "MedicationStatement",
    "id": "0",
    "meta": {
      "versionId": "1"
    },
    "status": "active",
    "medicationCodeableConcept": {
      "coding": [{
        "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv-ips",

        "code": "no-medication-info",
        "display": "No information about current medications"
      }]
    },
    "subject": {
      "reference": "Patient/$1"
    }

  }
}]</format>
                        <args>
                            <arg evaluator="xml" expression="$ctx:patientId"/>
                        </args>
                    </payloadFactory>
                    <property description="No Medication" expression="json-eval($
                        )" name="medication" scope="default" type="STRING"/>
                </then>
                <else>
```

```
                <property description="GET Medication" expression="json-eval(
                    $.entry)" name="medication" scope="default" type="STRING
                    "/>
            </else>
        </filter>
        <header description="SET Authorization" expression="$ctx:token" name
            ="Authorization" scope="transport"/>
        <call>
            <endpoint>
                <http method="get" uri-template="http://192.168.199.1:8280/
                    patient/{uri.var.patientId}/immunization">
                    <suspendOnFailure>
                        <initialDuration>-1</initialDuration>
                        <progressionFactor>-1</progressionFactor>
                        <maximumDuration>0</maximumDuration>
                    </suspendOnFailure>
                    <markForSuspension>
                        <retriesBeforeSuspension>0</retriesBeforeSuspension>
                    </markForSuspension>
                </http>
            </endpoint>
        </call>
        <filter regex="0" source="json-eval($.total)">
            <then>
                <payloadFactory media-type="json">
                    <format>[{
      "resource": {
        "resourceType": "Immunization",
        "id": "0",
        "meta": {
          "versionId": "1"
        },
        "status": "completed",
        "vaccineCode": {
          "coding": [{
            "system": "http://hl7.org/fhir/uv/ips/CodeSystem/absent-unknown-uv
                -ips",
            "code": "no-immunization-info",
            "display": "No information about current immunizations"
          }]
        },
        "patient": {
          "reference": "Patient/$1"
        },
        "occurrenceDateTime": "unknown"
      }
      }]</format>
                    <args>
                        <arg evaluator="xml" expression="$ctx:patientId"/>
                    </args>
                </payloadFactory>
                <property description="No Immunization" expression="json-eval
                    ($)" name="immunization" scope="default" type="STRING"/>
            </then>
            <else>
```

```
                    <property description="GET Immunization" expression="json-
                        eval($.entry)" name="immunization" scope="default" type="
                        STRING"/>
                </else>
            </filter>
            <payloadFactory media-type="json">
                <format>{
  "resourceType": "Bundle",
  "id": "29ed51b9-1a38-4081-a9e6-8128f08a2d51",
  "meta": {
    "lastUpdated": "2020-09-24T08:07:33.075+00:00"
  },
  "type": "searchset",
  "link": [ {
    "relation": "self",
    "url": "http://hapi.fhir.org/baseR4/Bundle?_pretty=true"
  }, {
    "relation": "next",
    "url": "http://hapi.fhir.org/baseR4?_getpages=29ed51b9-1a38-4081-a9e6-8128
        f08a2d51&amp;_getpagesoffset=20&amp;_count=20&amp;_pretty=true&amp;
        _bundletype=searchset"
  } ],
  "entry":
  $1,
  $2,
  $3,
  $4,
  $5
}</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:composition"/>
                    <arg evaluator="xml" expression="$ctx:allergyintolerance"/>
                    <arg evaluator="xml" expression="$ctx:condition"/>
                    <arg evaluator="xml" expression="$ctx:medication"/>
                    <arg evaluator="xml" expression="$ctx:immunization"/>
                </args>
            </payloadFactory>
            <respond/>
        </inSequence>
        <outSequence/>
        <faultSequence>
            <log level="custom">
                <property name="text" value="An unexpected error occured"/>
                <property expression="get-property('ERROR_MESSAGE')" name="
                    message"/>
                <property expression="get-property('ERROR_CODE')" name="code"/>
                <property expression="get-property('ERROR_DETAIL')" name="detail
                    "/>
            </log>
            <payloadFactory media-type="json">
                <format>{ "status": "Error",
   "description": "$1" }</format>
                <args>
                    <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
                </args>
            </payloadFactory>
            <respond/>
```

```
        </faultSequence>
    </resource>
</api>
```

Código E.4 Endpoint al recurso AllergyIntolerance (AllergyIntolerance.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="AllergyIntolerance" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/
        AllergyIntolerance?_pretty=true&amp;patient={uri.var.patientId}&amp;
        _format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.5 Endpoint al recurso Composition (Composition.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Composition" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/Composition?
        _pretty=true&amp;patient={uri.var.patientId}&amp;_format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.6 Endpoint al recurso Condition (Condition.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Condition" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/Condition?
        _pretty=true&amp;patient={uri.var.patientId}&amp;_format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.7 Endpoint al recurso Immunization (Immunization.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Immunization" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/Immunization?
        _pretty=true&amp;patient={uri.var.patientId}&amp;_format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.8 Endpoint al recurso Medication (Medication.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Medication" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/
        MedicationStatement?_pretty=true&amp;patient={uri.var.patientId}&amp;
        _format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.9 Endpoint al recurso Patient (Patient.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Patient" xmlns="http://ws.apache.org/ns/synapse">
    <http method="get" uri-template="http://hapi.fhir.org/baseR4/Patient/{uri.
        var.patientId}/_history/1?_pretty=true&amp;_format=json">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
</endpoint>
```

Código E.10 Endpoint al servicio de validación (Validation.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="Validation" xmlns="http://ws.apache.org/ns/synapse">
```

```
    <http method="post" uri-template="https://localhost:9444/services/
        OAuth2TokenValidationService.
        OAuth2TokenValidationServiceHttpsSoap11Endpoint/">
        <suspendOnFailure>
            <initialDuration>-1</initialDuration>
            <progressionFactor>1.0</progressionFactor>
        </suspendOnFailure>
        <markForSuspension>
            <retriesBeforeSuspension>0</retriesBeforeSuspension>
        </markForSuspension>
    </http>
    <property name="Authorization" scope="axis2" value="Basic YWRtaW46YWRtaW4=
        "/>
</endpoint>
```

Identity Server

Código E.11 Service Provider exportado de Identity Server (Oauth2.xml).

```
<?xml version="1.0" encoding="UTF-8"?><ServiceProvider>
  <ApplicationName>Oauth2</ApplicationName>
  <Description>Oauth2 service</Description>
  <JwksUri/>
  <InboundAuthenticationConfig>
    <InboundAuthenticationRequestConfigs>
      <InboundAuthenticationRequestConfig>
        <InboundAuthKey>Oauth</InboundAuthKey>
        <InboundAuthType>openid</InboundAuthType>
        <InboundConfigType>standardAPP</InboundConfigType>
        <Properties/>
      </InboundAuthenticationRequestConfig>
      <InboundAuthenticationRequestConfig>
        <InboundAuthKey>klsDR477TiFvnlXrSTe_74L4mcsa</InboundAuthKey>
        <InboundAuthType>oauth2</InboundAuthType>
        <InboundConfigType>standardAPP</InboundConfigType>
        <inboundConfiguration><![CDATA[<?xml version="1.0" encoding="UTF-8"
            standalone="yes"?>
<oAuthAppDO>
    <oauthConsumerKey>klsDR477TiFvnlXrSTe_74L4mcsa</oauthConsumerKey>
    <oauthConsumerSecret>aL8tCnQTB9lUA5OAFGzRXLZT4c0a</oauthConsumerSecret>
    <applicationName>Oauth2</applicationName>
    <callbackUrl>https://localhost:9444/oauth2/token</callbackUrl>
    <oauthVersion>OAuth-2.0</oauthVersion>
    <grantTypes>refresh_token urn:ietf:params:oauth:grant-type:saml2-bearer
        implicit password client_credentials iwa:ntlm urn:ietf:params:oauth:
        grant-type:device_code authorization_code urn:ietf:params:oauth:grant-
        type:uma-ticket account_switch </grantTypes>
    <scopeValidators>
        <scopeValidator>XACML Scope Validator</scopeValidator>
    </scopeValidators>
    <pkceSupportPlain>true</pkceSupportPlain>
    <pkceMandatory>false</pkceMandatory>
    <state>ACTIVE</state>
    <userAccessTokenExpiryTime>3600</userAccessTokenExpiryTime>
    <applicationAccessTokenExpiryTime>3600</applicationAccessTokenExpiryTime>
```

```
    <refreshTokenExpiryTime>86400</refreshTokenExpiryTime>
    <idTokenExpiryTime>3600</idTokenExpiryTime>
    <audiences/>
    <bypassClientCredentials>false</bypassClientCredentials>
    <renewRefreshTokenEnabled>true</renewRefreshTokenEnabled>
    <requestObjectSignatureValidationEnabled>false</
        requestObjectSignatureValidationEnabled>
    <idTokenEncryptionEnabled>false</idTokenEncryptionEnabled>
    <idTokenEncryptionAlgorithm>null</idTokenEncryptionAlgorithm>
    <idTokenEncryptionMethod>null</idTokenEncryptionMethod>
    <tokenType>JWT</tokenType>
</oAuthAppDO>
]]></inboundConfiguration>
        <Properties/>
      </InboundAuthenticationRequestConfig>
      <InboundAuthenticationRequestConfig>
        <InboundAuthKey>Oauth</InboundAuthKey>
        <InboundAuthType>passivests</InboundAuthType>
        <InboundConfigType>standardAPP</InboundConfigType>
        <Properties/>
      </InboundAuthenticationRequestConfig>
    </InboundAuthenticationRequestConfigs>
  </InboundAuthenticationConfig>
  <LocalAndOutBoundAuthenticationConfig>
    <AuthenticationSteps>
      <AuthenticationStep>
        <StepOrder>1</StepOrder>
        <LocalAuthenticatorConfigs>
          <LocalAuthenticatorConfig>
            <Name>BasicAuthenticator</Name>
            <DisplayName>basic</DisplayName>
            <IsEnabled>false</IsEnabled>
            <Properties/>
          </LocalAuthenticatorConfig>
        </LocalAuthenticatorConfigs>
        <FederatedIdentityProviders/>
        <SubjectStep>false</SubjectStep>
        <AttributeStep>false</AttributeStep>
      </AuthenticationStep>
    </AuthenticationSteps>
    <AuthenticationType>local</AuthenticationType>
    <alwaysSendBackAuthenticatedListOfIdPs>false</
        alwaysSendBackAuthenticatedListOfIdPs>
    <UseTenantDomainInUsername>false</UseTenantDomainInUsername>
    <UseUserstoreDomainInRoles>true</UseUserstoreDomainInRoles>
    <UseUserstoreDomainInUsername>false</UseUserstoreDomainInUsername>
    <SkipConsent>false</SkipConsent>
    <skipLogoutConsent>false</skipLogoutConsent>
    <EnableAuthorization>true</EnableAuthorization>
  </LocalAndOutBoundAuthenticationConfig>
  <RequestPathAuthenticatorConfigs/>
  <InboundProvisioningConfig>
    <ProvisioningUserStore/>
    <IsProvisioningEnabled>false</IsProvisioningEnabled>
    <IsDumbModeEnabled>false</IsDumbModeEnabled>
  </InboundProvisioningConfig>
  <OutboundProvisioningConfig>
```

```
    <ProvisioningIdentityProviders/>
  </OutboundProvisioningConfig>
  <ClaimConfig>
    <RoleClaimURI/>
    <LocalClaimDialect>true</LocalClaimDialect>
    <IdpClaim/>
    <ClaimMappings/>
    <AlwaysSendMappedLocalSubjectId>false</AlwaysSendMappedLocalSubjectId>
    <SPClaimDialects/>
  </ClaimConfig>
  <PermissionAndRoleConfig>
    <Permissions/>
    <RoleMappings/>
    <IdpRoles/>
  </PermissionAndRoleConfig>
  <IsSaaSApp>false</IsSaaSApp>
  <ImageUrl/>
  <AccessUrl>https://localhost:9444/oauth2</AccessUrl>
  <IsDiscoverable>false</IsDiscoverable>
</ServiceProvider>
```

Código E.12 Política XCAML para enfermero (política_enefermero.xml).

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="
    scope_based_token_validation_role_enfermero" RuleCombiningAlgId="urn:oasis:
    names:tc:xacml:1.0:rule-combining-algorithm:first-applicable" Version
    ="1.0">
<Description>This policy template provides ability to validate OAuth2 access
    token to a given service provider in the validation flow based on the scope
    .</Description>
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Oauth2</
    AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/sp/sp-name" Category
    ="http://wso2.org/identity/sp" DataType="http://www.w3.org/2001/XMLSchema#
    string" MustBePresent="true">
</AttributeDesignator>
</Match>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    token_validation</AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/identity-action/
    action-name" Category="http://wso2.org/identity/identity-action" DataType="
    http://www.w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Match>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">enfermero</
    AttributeValue>
<AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
    Category="urn:oasis:names:tc:xacml:2.0:subject:role" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
```

```
</Match>
</AllOf>
</AnyOf>
</Target>
<Rule Effect="Deny" RuleId="permit_by_scope">
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</
    AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-
    name" Category="http://wso2.org/identity/oauth-scope" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">confidential
    </AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-
    name" Category="http://wso2.org/identity/oauth-scope" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Apply>
</Apply>
</Condition>
</Rule>
</Policy>
```

Código E.13 Política XCAML para médico (politica_medico.xml).

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="
    scope_based_token_validation_role_medico" RuleCombiningAlgId="urn:oasis:
    names:tc:xacml:1.0:rule-combining-algorithm:first-applicable" Version
    ="1.0">
<Description>This policy template provides ability to validate OAuth2 access
    token to a given service provider in the validation flow based on the scope
    ..</Description>
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Oauth2</
    AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/sp/sp-name" Category
    ="http://wso2.org/identity/sp" DataType="http://www.w3.org/2001/XMLSchema#
    string" MustBePresent="true">
</AttributeDesignator>
</Match>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    token_validation</AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/identity-action/
    action-name" Category="http://wso2.org/identity/identity-action" DataType="
    http://www.w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Match>
```

```
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">medico</
    AttributeValue>
<AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
    Category="urn:oasis:names:tc:xacml:2.0:subject:role" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Match>
</AllOf>
</AnyOf>
</Target>
<Rule Effect="Permit" RuleId="permit_by_role_and_scope">
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</
    AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-
    name" Category="http://wso2.org/identity/oauth-scope" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</
    AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-
    name" Category="http://wso2.org/identity/oauth-scope" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">confidential
    </AttributeValue>
<AttributeDesignator AttributeId="http://wso2.org/identity/oauth-scope/scope-
    name" Category="http://wso2.org/identity/oauth-scope" DataType="http://www.
    w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Apply>
</Apply>
</Condition>
</Rule>
<Rule Effect="Deny" RuleId="deny_others">
</Rule>
</Policy>
```

Postman

Código E.14 Colección y recursos configurados en Postman.

```
{
  "info": {
    "_postman_id": "a1ca9893-e2ac-42fa-907b-88c50a3ac368",
    "name": "API FHIR",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.
        json"
```

```
},
"item": [
  {
    "name": "Authorization",
    "protocolProfileBehavior": {
      "strictSSL": false
    },
    "request": {
      "auth": {
        "type": "oauth2",
        "oauth2": [
          {
            "key": "accessTokenUrl",
            "value": "https://localhost:9444/oauth2/token",
            "type": "string"
          },
          {
            "key": "authUrl",
            "value": "https://localhost:9444/oauth2/authorize",
            "type": "string"
          },
          {
            "key": "redirect_uri",
            "value": "https://localhost:9444/oauth2/token",
            "type": "string"
          },
          {
            "key": "scope",
            "value": "write",
            "type": "string"
          },
          {
            "key": "clientSecret",
            "value": "aL8tCnQTB9lUA5OAFGzRXLZT4c0a",
            "type": "string"
          },
          {
            "key": "clientId",
            "value": "klsDR477TiFvnlXrSTe_74L4mcsa",
            "type": "string"
          },
          {
            "key": "addTokenTo",
            "value": "header",
            "type": "string"
          },
          {
            "key": "tokenType",
            "value": "",
            "type": "string"
          },
          {
            "key": "accessToken",
            "value": "",
            "type": "string"
          }
        ]
```

```
        },
        "method": "GET",
        "header": [],
        "url": {
          "raw": "https://localhost:9444/oauth2/authorize",
          "protocol": "https",
          "host": [
            "localhost"
          ],
          "port": "9444",
          "path": [
            "oauth2",
            "authorize"
          ]
        }
      },
      "response": []
    },
    {
      "name": "Token Validation",
      "protocolProfileBehavior": {
        "strictSSL": false,
        "disabledSystemHeaders": {}
      },
      "request": {
        "auth": {
          "type": "noauth"
        },
        "method": "POST",
        "header": [
          {
            "key": "Authorization",
            "value": "Basic YWRtaW46YWRtaW4=",
            "type": "text"
          }
        ],
        "body": {
          "mode": "urlencoded",
          "urlencoded": [
            {
              "key": "token",
              "value": "eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3WW1
              NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0Zo
              TXpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HW
              XdZbU00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa0
              1XRmhNelUyWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQaeyJzdWIiOiJ1
              c3VhcmlvMSIsImF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiL
              CJuYmYiOjE2MDMxMjM3MTMsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNz
              RMNG1jc2EiLCJzY29wZSI6InBob25lIiwiaXNzIjoiaHR0cHM6XC9cL2xvY2F
              saG9zdDo5NDQzXC9vYXV0aDJcL3Rva2VuIiwiZXhwIjoxNjAzMTI3MzEzLCJp
              YXQiOjE2MDMxMjM3MTMsImp0aSI6IjczOWMzYTFiLTU2ZTEtNGQ1Yi05Mzg3L
              TE1ZGQ4ZWI2YzE0MCJ9dWsIwS2nypx1yGlQIWhxVw1lbZiqE2ptHXBTaSCBcO
              aVXAhCEJEaz7Bv4rWnKCfoTBj3MAafMxqObympn5V_24WLxnJ8I31qeBFJ_TP
              UxaxCysCjGZNKtlUKV3U8qtiJc_SvWa44XM3r8dBLx9XEERC88QDR7GLSwm87
              dJKLx0BQWAYEbA5rVFvqPlXLDJW5Lb2Q4q36o0F96BTzLB98XU_vDkNSz9dsW
              DElqlRlhlRJXEM29xPtTC4i_ikBOG5yuu1dkzipiRBU8y6b91wLYHNg5xsDj2
              1pTmw0oIHZC2FZqtj71oHFBkHADzVx_yjLsuA01h4Vqns1htzjZbCHB6g",
```

```
            "type": "text"
          }
        ],
        "options": {
          "urlencoded": {}
        }
      },
      "url": {
        "raw": "https://localhost:9444/oauth2/introspect",
        "protocol": "https",
        "host": [
          "localhost"
        ],
        "port": "9444",
        "path": [
          "oauth2",
          "introspect"
        ],
        "query": [
          {
            "key": "username",
            "value": "usuario1",
            "disabled": true
          },
          {
            "key": "password",
            "value": "usuario1",
            "disabled": true
          },
          {
            "key": "grant_type",
            "value": "password",
            "disabled": true
          }
        ]
      }
    },
    "response": []
  },
  {
    "name": "Allergy",
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
          WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
          XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
          00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6Imtsc0RSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6Imtsc0RSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0Ojk0NDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
          R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
```

```
          XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
          NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
          -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
          wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
          Nbg",
          "type": "text"
        }
      ],
      "url": {
        "raw": "http://192.168.199.1:8280/patient/1171468/allergyintolerance",
        "protocol": "http",
        "host": [
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "patient",
          "1171468",
          "allergyintolerance"
        ]
      }
    },
    "response": []
  },
  {
    "name": "Composition",
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
          WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
          XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
          00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkONDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
          R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
          XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
          NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
          -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
          wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
          Nbg",
          "type": "text"
        }
      ],
      "url": {
        "raw": "http://192.168.199.1:8280/patient/1171468/allergyintolerance",
        "protocol": "http",
        "host": [
```

```
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "patient",
          "1171468",
          "allergyintolerance"
        ]
      }
    },
    "response": []
  },
  {
    "name": "PatientBasic",
    "request": {
      "method": "GET",
      "header": [],
      "url": {
        "raw": "http://192.168.199.1:8280/basic/patient/1194783",
        "protocol": "http",
        "host": [
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "basic",
          "patient",
          "1194783"
        ]
      }
    },
    "response": []
  },
  {
    "name": "Patient",
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
          WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
          XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
          00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6Imtsc0RSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6Imtsc0RSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0Ojk0NDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
          R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
```

```
          XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
          NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
          -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
          wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
          Nbg",
          "type": "text"
        }
      ],
      "url": {
        "raw": "http://192.168.199.1:8280/patient/1632296",
        "protocol": "http",
        "host": [
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "patient",
          "1632296"
        ],
        "query": [
          {
            "key": "",
            "value": null,
            "disabled": true
          }
        ]
      }
    },
    "response": []
  },
  {
    "name": "Patient",
    "request": {
      "method": "POST",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
          WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
          XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
          00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkONDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
          R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
          XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
          NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
          -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
          wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
          Nbg",
          "type": "text"
```

```
          "disabled": true
        },
        {
          "key": "Content-Type",
          "value": "application/json",
          "type": "text"
        }
      ],
      "body": {
        "mode": "raw",
        "raw": "{\r\n\t\"resourceType\": \"Patient\",\r\n\t\"meta\": {\r\n\t\t
            \"versionId\": \"1\",\r\n\t\t\"lastUpdated\": \"2020-11-02T06
            :43:21.950+00:00\",\r\n\t\t\"source\": \"#andlYAtKDTXiq1eJ\",\r\n\
            t\t\"security\" : [{\r\n \"system\" : \"http://terminology.hl7.org
            /CodeSystem/v3-Confidentiality\",\r\n \"code\" : \"R\",\r\n \"
            display\" : \"Restricted\"\r\n       }]\r\n\t},\r\n\t\"text\": {\r
            \n\t\t\"status\": \"generated\",\r\n\t\t\"div\": \"<div xmlns=\\\"
            http://www.w3.org/1999/xhtml\\\"><div class=\\\"hapiHeaderText
            \\\">Kelly <b>MIGUELITO </b></div><table class=\\\"
            hapiPropertyTable\\\"><tbody><tr><td>Identifier</td><td>CT4801</td
            ></tr></tbody></table></div>\"\r\n\t},\r\n\t\"identifier\": [{\r\n
            \t\t\"type\": {\r\n\t\t\t\"coding\": [{\r\n\t\t\t\t\"system\": \"
            http://terminology.hl7.org/CodeSystem/v2-0203\",\r\n\t\t\t\t\"code
            \": \"MR\",\r\n\t\t\t\t\"display\": \"Medical Record Number\"\r\n\
            t\t}],\r\n\t\t\t\"text\": \"Medical Record Number\"\r\n\t\t},\r\
            n\t\t\"value\": \"CT4801\"\r\n\t}, {\r\n\t\t\"type\": {\r\n\t\t\t
            \"coding\": [{\r\n\t\t\t\t\"system\": \"http://terminology.hl7.org
            /CodeSystem/v2-0203\",\r\n\t\t\t\t\"code\": \"NIIP\",\r\n\t\t\t\t
            \"display\": \"National Insurance Payor Identifier\"\r\n\t\t\t}],\
            r\n\t\t\t\"text\": \"National Insurance Payor Identifier\"\r\n\t\t
            },\r\n\t\t\"value\": \"9800000118\"\r\n\t}],\r\n\t\"name\": [{\r\n
            \t\t\"family\": \"Detal\",\r\n\t\t\"given\": [\"Miguelito\"]\r\n\t
            }],\r\n\t\"gender\": \"male\"\r\n}"
      },
      "url": {
        "raw": "http://192.168.199.1:8280/patient",
        "protocol": "http",
        "host": [
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "patient"
        ],
        "query": [
          {
            "key": "",
            "value": null,
            "disabled": true
          }
        ]
      }
    },
    "response": []
```

```
    },
    {
      "name": "Condition",
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
            WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
            XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
            00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
            yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
            mF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
            QyNDcsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
            6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkONDNcL29hdXRoMlwv
            dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
            mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
            R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
            XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
            NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
            -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
            wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
            Nbg",
            "type": "text"
          }
        ],
        "url": {
          "raw": "http://192.168.199.1:8280/patient/1171468/condition",
          "protocol": "http",
          "host": [
            "192",
            "168",
            "199",
            "1"
          ],
          "port": "8280",
          "path": [
            "patient",
            "1171468",
            "condition"
          ]
        }
      },
      "response": []
    },
    {
      "name": "Medication",
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
            WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
            XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
            00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
```

```
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkONDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
          R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
          XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
          NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
          -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1OOXn0YyiBuBM_ePX7mAdM
          wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
          Nbg",
          "type": "text"
        }
      ],
      "url": {
        "raw": "http://192.168.199.1:8280/patient/1171468/medication?=",
        "protocol": "http",
        "host": [
          "192",
          "168",
          "199",
          "1"
        ],
        "port": "8280",
        "path": [
          "patient",
          "1171468",
          "medication"
        ],
        "query": [
          {
            "key": "",
            "value": ""
          }
        ]
      }
    },
    "response": []
  },
  {
    "name": "Patient Summary",
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3
          WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoT
          XpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU
          00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelU
          yWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQseyJzdWIiOiJGdWxhbml0byIsI
          mF1ZCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJuYmYiOjE2MDM5MT
          QyNDcsImF6cCI6ImtscORSNDc3VGlGdm5sWHJTVGVfNzRMNG1jc2EiLCJzY29wZSI
          6InJlYWQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkONDNcL29hdXRoMlwv
          dG9rZW4iLCJleHAiOjE2MDM5MTc4NDcsImlhdCI6MTYwMzkxNDI0NywianRpIjoiZ
          mRhOGQ0MzQtNTc3Yi00ZTViLThhYjctZWJkYWIwMzFkNdjg4In0b2YjSWq1OP4L3J
```

```
            R2iMSV3mbtOQ7Vg0vegFgHYmWMBbopQ0o7tve3Awfgs_90zEdpfYWm5IRxuWLunZC
            XxCl9mZMiqW-soIxoP6Nynewj46Et4zNvfbh6PE7LTzBBtAA4j3axrYRzYxh9kWMB
            NNoZrAgbE4g6pmEeVCF8oWZM8GYMc9Kin_-WgDfdbFmOle4BDWhJvYn8VMHa7NUHE
            -8FjWEutz3EWDF-GVScxsq_PLA3ZsWzEhltpuKjLojB1O0Xn0YyiBuBM_ePX7mAdM
            wHabXLbEDfG3-So6QR1gVPk34x5aNH-mtK_TETI0fTqWId3_z3CANwGk36Ixprvrx
            Nbg",
            "type": "text"
          }
        ],
        "url": {
          "raw": "http://192.168.199.1:8280/patient/1171468/summary",
          "protocol": "http",
          "host": [
            "192",
            "168",
            "199",
            "1"
          ],
          "port": "8280",
          "path": [
            "patient",
            "1171468",
            "summary"
          ]
        }
      },
      "response": []
    }
  ],
  "protocolProfileBehavior": {}
}
```