Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Mecanismo de control de acceso para soluciones IoT de FIWARE en escenarios sanitarios

Autor: Fernando Ramos Rojas
Tutor: Jorge Calvillo Arbizu

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

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Mecanismo de control de acceso para soluciones IoT de FIWARE en escenarios sanitarios

Autor:
Fernando Ramos Rojas

Tutor:
Jorge Calvillo Arbizu
Departamento de Ingeniería Telemática

Dpto. Ingeniería Telématica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Mecanismo de control de acceso para soluciones IoT de FIWARE en escenarios sanitarios

Autor: Fernando Ramos Rojas

Tutor: Jorge Calvillo Arbizu

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Después de una intensa etapa de 4 años donde he tenido dificultades de todo tipo, siempre he tenido el apoyo de mi familia, amigos y de mi pareja. Siempre se los agradeceré por su apoyo y cariño.

Durante estos 4 años he conocido nuevos amigos, a quienes recordaré toda la vida. Gracias Pablo, Jesús, Ana, Sergio, Daniel, Natalia, Raúl, … por haber sido mi gran apoyo en este tiempo y también por haber compartido tantos momentos juntos.

Dar las gracias a mi pareja, por estar siempre presente tanto para animarme como para apoyarme en cualquier decisión que tomase.

Agradecer a mis padres, por su apoyo en los momentos más difíciles durante la época de exámenes, por aguantar cuando estaba de mal humor y por haberme animado cuando más lo necesitaba.

Recordar también a mi abuela, aunque no esté presente, sé que esté donde esté me estará apoyando.

Gracias a mi tutor, Jorge Calvillo, por saber guiarme en todo momento y enseñarme nuevas referencias que me han sido de gran ayuda para el proyecto que hemos trabajado.

Fernando Ramos Rojas

Sevilla, 2020

Resumen

Los beneficios que proporciona IoT (Internet of Things) como el despliegue de elementos sensores y actuadores autónomos o el análisis de grandes cantidades de datos, motivan que se esté aplicando en diversos entornos, incluyendo el sanitario. Sin embargo, la información de contexto manejada en escenarios de salud como, por ejemplo, los datos de pacientes tienen un carácter especialmente sensible, y requiere el despliegue de mecanismos de seguridad robustos y fiables.

En este proyecto se propone el diseño y desarrollo de un mecanismo de autenticación y autorización en un entorno IoT que controle la publicación, suscripción y recepción de los datos, evitando la participación de entidades no autorizadas. Este mecanismo se llevará a cabo mediante varios módulos proporcionados por FIWARE para la seguridad en IoT que serán adaptados teniendo en cuenta los requisitos específicos del dominio sanitario y desplegados simultáneamente para construir una solución completa. Los módulos FIWARE que sirven de base para el mecanismo de autenticación y autorización son Authzforce, PEP Proxy Wilma y Keyrock.

El objetivo de estas tres herramientas es el de evitar que ninguna entidad que no haya aportado credenciales válidas y tampoco haya cumplido una serie de reglas definidas en un conjunto de políticas dentro de una aplicación registrada pueda acceder a Orion.

El desarrollo y la validación del mecanismo de seguridad se ha llevado a cabo mediante el planteamiento de varios escenarios, abordando primero los más simples, para concluir posteriormente con escenarios más complejos.

Abstract

The benefits provided by IoT (Internet of Things), such as the deployment of autonomous sensor elements and actuators or the analysis of large amounts of data, motivate its application in various environments, including healthcare. However, the context information handled in health scenarios. For example, patient data is particularly sensitive, and requires the deployment of robust and reliable security mechanisms.

This project proposes the design and development of an authentication and authorization mechanism in an IoT environment that controls the publication, subscription, and reception of data, avoiding the participation of unauthorized entities. This mechanism will be carried out through various modules provided by FIWARE for security in IoT that will be adapted considering the specific requirements of the healthcare domain and deployed simultaneously to build a complete solution. The FIWARE modules that serve as the basis for the authentication and authorization mechanism are Authzforce, PEP Proxy Wilma and Keyrock.

The objective of these three components is to prevent any entity that has not provided valid credentials and has not complied with a series of rules defined in a set of policies within a registered application that can access Orion.

The development and validation of the security mechanism has been carried out by proposing several scenarios, first addressing the simplest ones, to later conclude with more complex scenarios.

Índice

Agradecimientos **9**

Resumen **11**

Abstract **13**

Índice **14**

Índice de Tablas **16**

Índice de Figuras **17**

1 Introducción **21**
1.1 Motivación y objetivos *21*
1.1.1 Motivación 21
1.1.2 Objetivos 21
1.2 Solución Planteada *22*
1.3 Plan de trabajo *22*

2 Estado del arte **25**
2.1 Internet of Thing (IoT). *25*
2.2 Modelo de control de acceso basado en atributos (ABAC). *25*
2.2.1 XACML. 26
2.3. FIWARE. *31*
2.3.1 Orion Context Broker 31
2.3.2 Identity Manager Keyrock 32
2.3.3 PEP Proxy Wilma 33
2.3.4 Authzforce 33
2.4 Niveles de seguridad. *34*
2.5 Tecnologías y recursos software. *35*
2.5.1 Máquina virtual Ubuntu 18.04.2 LTS 35
2.5.2 Postman 35
2.5.3 Docker 36
2.5.4 MongoDB y MySQL 37

3 Resultados **38**
3.1 Funcionalidad de cada componente en el sistema planteado. *38*
3.1.1 Keyrock 39
3.1.2 PEP Proxy Wilma 39
3.1.3 Authzforce 39
3.2 Diseño de los escenarios de prueba *41*
3.3 Proceso de autenticación. *43*
3.4 Proceso de autorización *45*
3.5 Escenarios planteados. *45*
3.5.1 Escenario 1: Autorización a los agentes IoT para la publicación. 45
3.5.2 Escenario 2: Autorización a usuarios (médicos) para la consulta de la actividad física de los pacientes. 59
3.5.3 Escenario 3: Autorización a los usuarios (médicos) para la suscripción a entidades. 68
3.5.4 Escenario 4. Autorización para el administrador del sistema. 75

4 Conclusiones y líneas futuras 79

Anexo A: Instalación y configuración de docker 81

Anexo B: Instalación y configuración de Orion, Authzforce, Wilma y Keyrock. 83

Anexo C: Instalación de Accumulator-Server. 92

Anexo D: Instalación de Postman. 94

Referencias 96

ÍNDICE DE TABLAS

Tabla 1. Consulta de información. 23
Tabla 2. Trabajo inicial de los componentes. 23
Tabla 3. Conocimiento de Wilma y Authzforce. 23
Tabla 4. Implementación de los escenarios. 24
Tabla 5. Pruebas y validación. 24
Tabla 6. Documentación. 24
Tabla 7. Tiempo total. 24
Tabla 8. Usuarios registrados del Hospital Central. 41
Tabla 9. Agentes IoT registrados del Hospital Central. 41
Tabla 10. Usuarios registrados de la Residencia de Sevilla. 42
Tabla 11. Agentes IoT registrados de la Residencia de Sevilla 43

ÍNDICE DE FIGURAS

Figura 1. Esquema de la solución planteada. 22
Figura 2. Arquitectura ABAC. 26
Figura 3. Ejemplo simple de la sentencia <Target>. 28
Figura 4. Ejemplo simple de la sentencia <Target>. 28
Figura 5. Ejemplo de la directiva <Apply>. 29
Figura 6. Ejemplo de la sentencia <ObligationExpressions>. 29
Figura 7. Logo de Fiware. 31
Figura 8. Logo OCB. 32
Figura 9. Esquema Context Element. 32
Figura 10. Logo de Keyrock. 33
Figura 11. Logo Authzforce. 33
Figura 12. Nivel autenticación. 34
Figura 13. Nivel básico de seguridad. 34
Figura 14. Nivel avanzado de seguridad. 35
Figura 15. Máquina virtual Ubuntu. 35
Figura 16. Logo de Postman. 36
Figura 17. Logo Docker. 36
Figura 18. Logo Docker-Compose. 36
Figura 19. Logo MongoDB. 37
Figura 20. Logo MySQL. 37
Figura 21. Esquema de la solución final planteada. 38
Figura 22. Ejemplo de entidad de tipo "ActividadFisica". 39
Figura 23. Creación de un dominio de políticas. 40
Figura 24. Ejemplo de creación de una política. 40
Figura 25. Ejemplo de activación de una política. 41
Figura 26. Esquema de los elementos pertenecientes al Hospital Central. 42
Figura 27. Esquema de los elementos pertenecientes a la Residencia. 43
Figura 28. Solicitud de un token de acceso. 44
Figura 29. Comando para obtener en base 64 los valores oauth_client id y secret juntos. 44
Figura 30. Token de acceso. 44
Figura 31. Cabecera X-Auth-Token en Postman. 44
Figura 32. Respuesta al no indicar la cabecera Auth-token. 44

Figura 33. Respuesta al indicar un token inválido. 45
Figura 34. Escenario de publicación. Función getRESTPolicy. 46
Figura 35. Valor de la variable req.body. 46
Figura 36. Valor de req.body.toString('utf8'). 46
Figura 37. Mensaje de solicitud de autorización. 47
Figura 38. Mensaje de solicitud de autorización. 47
Figura 39. Mensaje de solicitud de autorización. 48
Figura 40. Escenario publicación. Política. 48
Figura 41. Escenario publicación. Política. 49
Figura 42. Escenario publicación. Política. 50
Figura 43. Escenario publicación. Política. 51
Figura 44. Respuesta de un usuario no autorizado. 51
Figura 45. Función de redirección hacia Orion. 51
Figura 46. Esquema de funcionamiento del escenario 1. 52
Figura 47. Escenario Publicación. Prueba 1. 52
Figura 48. Escenario Publicación. Prueba 2. 53
Figura 49. Escenario de publicación. Prueba 3. 53
Figura 50. Escenario de publicación. Prueba 4. 53
Figura 51. Escenario Publicación. Prueba 5. 54
Figura 52. Escenario Publicación. Prueba 6. 54
Figura 53. Escenario actualización de entidades. Función authorizeAzf y llamada a la función check Permissions. 55
Figura 54. Implementación de la función checkPermissions. 56
Figura 55. Implementación de la función getRESTPolicy. 57
Figura 56. Parte de la politíca para la actualización de una entidad. 57
Figura 57. Escenario publicación 1.1. Prueba 1. 58
Figura 58. Escenario publicación 1.1. Prueba 2. 58
Figura 59. Escenario publicación 1.1. Prueba 3. 58
Figura 60. Escenario publicación 1.1. Prueba 4. 59
Figura 61. Escenario consulta. Implementación authorizeAzf. 59
Figura 62. Escenario de consulta. Implementación de getRESTPolicy. 60
Figura 63. Escenario de consulta. Implementación de getRESTPolicy. 61
Figura 64. Escenario de consulta. Implementación de getRESTPolicy. 61
Figura 65. Escenario de consulta. Política. 62
Figura 66. Escenario de consulta. Política. 63
Figura 67. Implementación de la función redirRequest. 63
Figura 68. Esquema de funcionamiento del escenario 2. 64
Figura 69. Error debido a no eliminar la cabecera Content-Type en una consulta. 64
Figura 70. Escenario de consulta. Prueba 1. Petición. 65

Figura 71. Escenario de consulta. Prueba 1. Respuesta. 65
Figura 72. Escenario de consulta. Prueba 2. 66
Figura 73. Escenario de consulta. Prueba 3. Petición. 66
Figura 74. Escenario de consulta. Prueba 3. Respuesta. 68
Figura 75. Escenario de consulta. Prueba 4. 68
Figura 76. Escenario de suscripción. Implementación authorizeAzf. 69
Figura 77. Escenario de suscripción. Implementación authorizeAzf. 69
Figura 78. Escenario de suscripción. Política. 70
Figura 79. Escenario de suscripción. Política. 71
Figura 80. Esquema de funcionamiento del escenario 3. 72
Figura 81. Escenario suscripción. Prueba 1. 73
Figura 82. Escenario suscripción. Prueba 1. 73
Figura 83. Escenario suscripción. Prueba 2. 74
Figura 84. Escenario suscripción. Prueba 3. 74
Figura 85. Escenario suscripción. Prueba 4. 75
Figura 86. Escenario para el administrador. Política. 76
Figura 87. Escenario para el administrador. Prueba 1. Petición. 76
Figura 88. Escenario para el administrador. Prueba 1. Respuesta. 77
Figura 89. Escenario para el administrador. Prueba 2. Petición. 77
Figura 90. Escenario para el administrador. Prueba 2. Respuesta. 78
Figura 91. Comprobación de la instalación de Docker y Docker-Compose. 82
Figura 92. Estructura de la carpeta escenario_sanitario 83
Figura 93. Services.sh. 84
Figura 94. Funciones startContainers y stoppingContainers. 84
Figura 95. Función displayServices. 84
Figura 96. Configuración de Orion en el fichero Docker-compose.yml. 85
Figura 97. Configuración de Keyrock en el fichero Docker-compose.yml. 86
Figura 98. Configuración de Authzforce en el fichero Docker-compose.yml. 87
Figura 99. Configuración de MongoDB y MySQL en el fichero Docker-compose.yml. 87
Figura 100. Comprobación de las versiones de node y npm. 88
Figura 101. Configuración de PEP Proxy Wilma. 88
Figura 102. Configuración de PEP Proxy Wilma. 89
Figura 103. Tabla authzforce en MySQL. 89
Figura 104. Tabla pep_proxy en MySQL. 90
Figura 105. Tabla role en MySQL. 90
Figura 106. Tabla user en MySQL. 91
Figura 107. Tabla oauth_client en MySQL. 91
Figura 108. Ejemplo de una notificación recibida. 93
Figura 109. Interfaz gráfica de Postman. 94

1 INTRODUCCIÓN

1.1 Motivación y objetivos

1.1.1 Motivación

La Agencia Europea de Seguridad de las Redes y de la Información ha publicado un estudio sobre las principales amenazas que afrontan los llamados *Smart hospitals* (hospitales inteligentes) [1]. El estudio destaca el malware como el principal factor de riesgo para este tipo de centro médico y advierte de que cada nuevo dispositivo conectado es un objetivo potencial para un ataque.

Las consecuencias de un ataque informático a un hospital son especialmente graves dada la enorme importancia de lo que hay en juego, como por ejemplo la integridad de los datos personales sanitarios.

Un hospital inteligente se define como aquel que se apoya en el Internet de las Cosas (IoT) para mejorar los procedimientos de atención a los pacientes. Lo que hace que un hospital sea inteligente es el uso de sistemas y aparatos interconectados que llevan a mejorar la atención y tratamiento de los pacientes.

Más allá de los hospitales inteligentes, centrándonos en el ámbito sanitario en general, podemos comprobar que las tecnologías de la información y la comunicación (TIC) basadas en el Internet de las Cosas están posibilitando el desarrollo de nuevas capacidades y servicios soportados por la tecnología en este dominio. Pueden desde hacer un seguimiento de la salud de los pacientes hasta mejorar la accesibilidad del paciente o facilitar aspectos puramente administrativos, como gestionar mejor los datos o accesibilidad al historial.

Aunque IoT está en auge en diversos dominios, en el sanitario su aplicación todavía está limitada. Uno de los motivos sería la falta de seguridad de los datos sanitarios registrados, debido a que el almacenamiento de los datos de una serie de paciente tiene como riesgo la filtración o robo de datos de salud de los pacientes.

Por ello en este trabajo se va a desarrollar un mecanismo de control de acceso para IoT en cualquier escenario de salud aplicando una serie de componentes de FIWARE. Como caso de uso se utilizará un dispositivo sensor integrado en una prenda textil (camiseta inteligente) que obtiene datos de la actividad física de un paciente. Este dispositivo ha sido desarrollado por el Grupo de Ingeniería Biomédica de la ETSI. La información obtenida por las camisetas es publicada por agentes IoT y, mediante los componentes FIWARE, se pretende controlar qué usuarios pueden consultar o suscribirse a estos datos, y qué agentes IoT los pueden publicar.

También otra de las motivaciones del proyecto es poner en práctica los conocimientos obtenidos en el Grado y además indagar sobre nuevas tecnologías no vistas durante la carrera.

1.1.2 Objetivos

El principal objetivo del proyecto es el diseño y desarrollo de un mecanismo de control de acceso en el dominio sanitario a partir de un conjunto de componentes de FIWARE, que son: Orion Context Broker, PEP Proxy Wilma, Authzforce y Keyrock.

El desarrollo y validación de este mecanismo se lleva a cabo mediante el planteamiento de varios escenarios. El primer escenario tiene como objetivo evitar que agentes IoT no autorizados puedan publicar información captada por la camiseta inteligente, ya que podría haber un agente IoT malicioso que intente publicar información falsa sobre una serie de pacientes. El segundo escenario se centra en la consulta de información buscando evitar que usuarios no autorizados puedan consultar dicha información publicada por los agentes IoT; si no evitamos esto el escenario no sería fiable, ya que no todas las personas deben tener acceso a la información personal de un paciente. El tercer escenario limitará la suscripción de alguna entidad publicada previamente en función de qué usuario esté realizando esta operación y de la dirección a la que va a recibir las

notificaciones.

Para conseguir el correcto funcionamiento de estos escenarios, los cuales podrían funcionar simultáneamente, es necesaria la interacción de los diferentes componentes FIWARE. Siendo imprescindible el desarrollo de una serie de políticas mediante Authzforce para satisfacer estos escenarios planteados.

En el siguiente apartado se va a describir brevemente la función que va a tener cada componente en nuestro sistema.

1.2 Solución Planteada

La solución desarrollada pretende proporcionar capacidades de seguridad a aquellos escenarios sanitarios soportados por el paradigma IoT (particularizado en los componentes FIWARE).

En cualquier escenario IoT tenemos agentes que publican y consumen información de contexto.

Ante una petición de publicación, suscripción o consulta, Wilma, que será el componente intermediario de la aplicación, recoge dicha solicitud, y, en primer lugar, consulta a Keyrock la autenticidad de la entidad que está intentando acceder el sistema. En la solicitud de Wilma ha de aparecer una cabecera donde se indique una contraseña que identifica al usuario en cuestión. Dicha contraseña se le denomina *token,* y es proporcionada por Keyrock en caso de que el usuario esté registrado en él.

Por consiguiente, una vez autenticado el usuario (incluso el Agente IoT debe estar autenticado), Wilma acudirá a Authzforce para obtener una decisión proveniente de este componente, es decir, Authzforce comprobará si dicho usuario está autorizado para obtener el recurso que pretende acceder.

Wilma será el componente que aplique dicha decisión. Si ha sido autorizado, Wilma redireccionará la petición a Orion Context Broker donde la respuesta dependerá de qué acción y recurso pretendía acceder el usuario. Por ejemplo, la publicación de información por parte de un agente IoT autorizado o bien la consulta de datos por parte de un médico autorizado. En caso contrario, se mostrará un mensaje indicando que no está autorizado.

Por otro lado, en nuestro proyecto los agentes IoT van a ser simulados mediante Postman, componente que va a ser explicado más adelante.

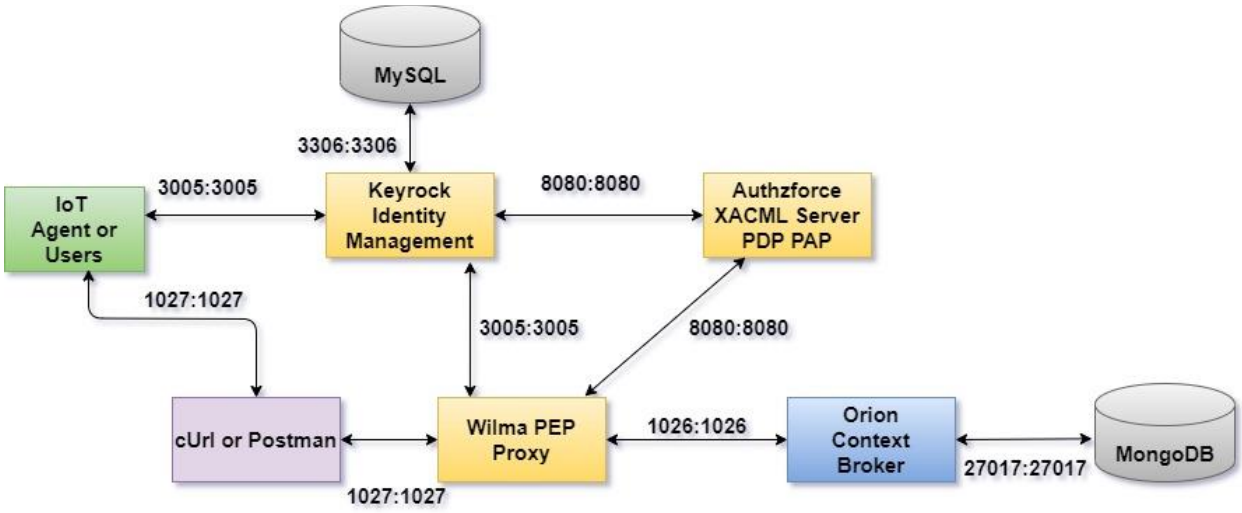

Figura 1. Esquema de la solución planteada.

1.3 Plan de trabajo

En este apartado se van a detallar las tareas que se han realizado para llevar a cabo el proyecto, y una comparación del tiempo estimado y el tiempo real dedicado a cada una de ellas.

Estas tareas son las siguientes:

- Tarea 1: Búsqueda y obtención de la información. En esta actividad inicial se estudiará la documentación de FIWARE [2], con el objetivo de conocer la plataforma, el uso de sus componentes y su funcionamiento.

Consulta de información	**Tiempo estimado**	**Tiempo real**
Investigación sobre FIWARE y sus componentes	35 horas	48 horas
Total	35 horas	48 horas

Tabla 1. Consulta de información.

- Tarea 2: Instalación de los diferentes componentes. En esta tarea también se ha tenido en cuenta la comprobación de la conexión entre todos los componentes y el tiempo invertido en la solución de los los errores que se han obtenido en la instalación de estos.

Trabajo inicial de los componentes	**Tiempo estimado**	**Tiempo real**
Instalación de los componentes y comprobación de la conexión entre ellos	25 horas	45 horas
Total	25 horas	45 horas

Tabla 2. Trabajo inicial de los componentes.

- Tarea 3: Esta tarea va a consistir en tratar de entender los diferentes archivos de configuración de Wilma y de conocer un lenguaje nuevo, XACML, que es usado en Authzforce para establecer diferentes políticas con el fin de autorizar a diferentes usuarios.

Conocimiento de Wilma y Authzforce	**Tiempo estimado**	**Tiempo real**
Conocimiento de los archivos de configuración de Wilma	10 horas	10 horas
Búsqueda de información sobre el lenguaje XACML	12 horas	15 horas
Total	22 horas	25 horas

Tabla 3. Conocimiento de Wilma y Authzforce.

- Tarea 4: Esta tarea va a tratar sobre el diseño y desarrollo de los posibles escenarios que se pueden implementar y el tiempo dedicado sobre los mismos.

Implementación de los escenarios	**Tiempo estimado**	**Tiempo real**
Planteamiento de los posibles escenarios	10 horas	15 horas
Desarrollo de los escenarios	70 horas	90 horas
Total	80 horas	105 horas

Tabla 4. Implementación de los escenarios.

- Tarea 5: Pruebas y validación de los diferentes escenarios planteados.

Pruebas y validación	**Tiempo estimado**	**Tiempo real**
Pruebas de cada escenario	5 horas	8 horas
Corrección de errores en la puesta en marcha de los escenarios	5 horas	10 horas
Total	10 horas	18 horas

Tabla 5. Pruebas y validación.

- Tarea 6: Realización de la memoria de fin de Grado

Documentación	**Tiempo estimado**	**Tiempo real**
Memoria del trabajo	80 horas	85 horas
Total	80 horas	85 horas

Tabla 6. Documentación.

Tiempo total estimado y real concluido.

Total	**252 horas**	**326 horas**

Tabla 7. Tiempo total.

2 ESTADO DEL ARTE

EN este apartado vamos a explicar brevemente cada una de las tecnologías utilizadas en este proyecto, y se detallarán los recursos utilizados para la implementación de este trabajo.

2.1 Internet of Thing (IoT).

La definición de IoT podría ser la agrupación e interconexión de dispositivos y objetos a través de una red, donde todos ellos podrían ser visibles e interaccionar. Respecto al tipo de objetos o dispositivos podría ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos como pueden ser el frigorífico, el calzado o la ropa. Cualquier cosa que se pueda imaginar podría ser conectada a internet e interaccionar sin necesidad de la intervención humana, el objetivo por tanto es una interacción de máquina a máquina.

Internet ha evolucionado rápidamente y esto ha permitido que IoT sea ya una realidad y no sólo una visión de futuro. La fama de esta tecnología radica principalmente en todas las aplicaciones y posibilidades que nos proporciona tanto para mejorar tanto la vida cotidiana de las personas como los entornos empresariales, donde se está implantando desde hace algún tiempo.

El sector sanitario no está al margen de esta creciente necesidad. En este ámbito en concreto, se analiza esta tecnología para presentar numerosas oportunidades a los profesionales sanitarios para el seguimiento de los pacientes. Cada día más dispositivos aparecen para ayudar a médicos y pacientes en el tratamiento y seguimiento de sus enfermedades.

No obstante, IoT puede tener limitaciones. En el caso del dominio sanitario, mantener la integridad de los datos, garantizar que no puedan ser manipulados, de forma intencionada o por error, y garantizar la privacidad de los pacientes, cuyos dispositivos envían su información de salud de forma continua, resulta una obligación no solo legal, sino también ética para los profesionales.

2.2 Modelo de control de acceso basado en atributos (ABAC).

El modelo de control de acceso basado en atributos (ABAC) define un paradigma por el que se conceden los derechos de acceso a usuarios mediante el uso de políticas combinando una serie de atributos.

Conceptualmente, los atributos son un conjunto de pares clave-valor. Además, pueden utilizarse sin límite de uso en las políticas de autorización. Los atributos se pueden clasificar en cuatro categorías:

- Atributos del sujeto: atributos que describen al usuario que intenta el acceso, por ejemplo, edad, autorización, departamento, rol, título del trabajo ...
- Atributos de acción: atributos que describen la acción que se está intentando realizar, por ejemplo, publicar, consultar, suscribir …
- Atributos de recurso: atributos que describen el objeto al que se accede, por ejemplo, el departamento al que pertenece el recurso, la clasificación o sensibilidad, la ubicación, tipo de información que contiene …
- Atributos contextuales (entorno): atributos que tratan el tiempo, la ubicación o aspectos dinámicos del escenario de control de acceso.

Las políticas de autorización son declaraciones que reúnen atributos para expresar lo que puede suceder y lo que no. Algunos ejemplos:

- Un usuario puede ver un documento si el documento está en el mismo departamento que el usuario.
- Denegar cualquier acceso que se produzca antes de las nueve de la mañana.

Con ABAC se pueden tener tantas políticas como se desee, con el fin de adaptar muchos escenarios diferentes.

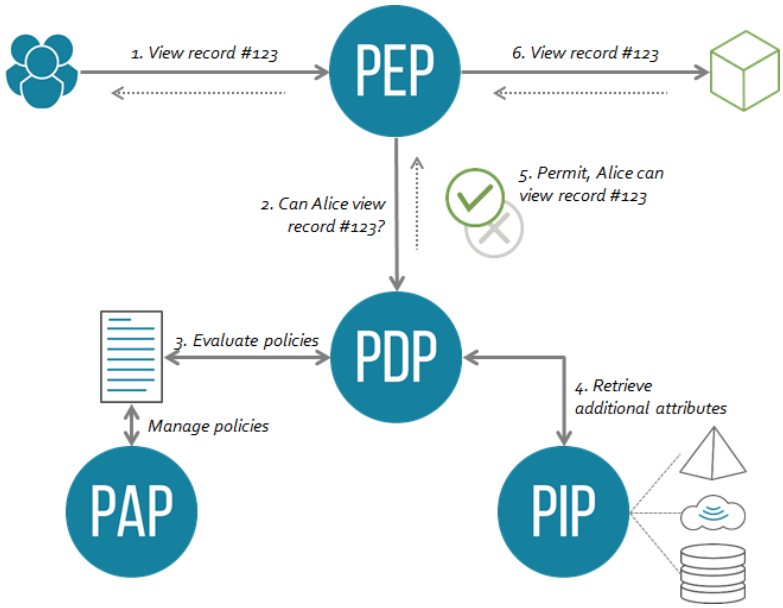

Figura 2. Arquitectura ABAC.

La arquitectura recomendada por este modelo de control de acceso sería la siguiente:

El PEP o Punto de aplicación de políticas es responsable de proteger los recursos (aplicaciones o datos) sobre los que se aplica el modelo de control de acceso. El PEP inspecciona la solicitud y genera una solicitud de autorización que envía al PDP.

El PAP o Punto de administración de políticas es el encargado de definir inequívocamente las políticas. Antes de la aplicación de las políticas, deben ser administradas. Esta es la fuente de la que dependerá el PDP.

El PDP o Punto de decisión de políticas es el cerebro de la arquitectura. Esta es la pieza que evalúa las solicitudes entrantes en función de las políticas con las que se ha configurado. El PDP devuelve una decisión de Permitir / Denegar. El PDP también puede usar PIP para recuperar los metadatos faltantes.

El PIP o Punto de información de políticas permite conectar al PDP con fuentes externas de atributos, por ejemplo, LDAP o bases de datos.

El principal estándar que implementa ABAC en la actualidad es XACML, lenguaje de marcado de control de acceso eXtensible.

2.2.1 XACML.

XACML (eXtensible Access Control Markup Language)[3] es un estándar de OASIS que define un modelo de datos XML para elaborar políticas de autorización, así como la lógica a seguir para evaluarlas en un contexto de solicitud de acceso.

XACML sigue el modelo ABAC y la evaluación se puede realizar con los datos adicionales recuperados de Policy Information Point (PIP), que está definido por la arquitectura de referencia XACML.

En XACML, los atributos se definen como:

- Un identificador.
- Un tipo de datos, por ejemplo, cadena. Hay casi 20 tipos de datos para elegir en XACML, desde entero y doble hasta fecha, hora y booleano.
- Una categoría, por ejemplo, acceso-sujeto. Hay 4 categorías predeterminadas en XACML que se utilizan regularmente en las políticas. La categoría es similar a la función gramatical del atributo en la oración.

De forma predeterminada, los atributos XACML son siempre bolsas de valores. Esto significa que, de forma predeterminada, el Punto de decisión de políticas (PDP) de XACML siempre considerará un atributo como

una bolsa de valores, incluso si esta bolsa contiene un valor único o ninguno.

Los elementos principales del estándar XACML son:

- Rule: Una regla es la unidad política más elemental. Consiste en aplicar condiciones sobre los atributos de la solicitud de acceso para poder llegar a una decisión (Permitir o Denegar). Los componentes principales de una regla son:
  - Target: Define un conjunto de condiciones simplificadas para el sujeto, el recurso y la acción que se deben cumplir para que un conjunto de políticas o reglas se apliquen a una solicitud determinada. Solo se puede comparar un atributo con un valor estático y no con un conjunto de valores. Un <Target> vacío coincide con cualquier solicitud.

    Este componente se puede estructurar en diferentes elementos:

    - <Target> debe contener una secuencia conjuntiva de elementos <AnyOf>, es decir, tiene que haber al menos una coincidencia positiva entre cada elemento <AnyOf>.
    - El elemento <AnyOf> debe contener una secuencia disyuntiva de elementos <AllOf>, es decir, tiene que haber solo una coincidencia positiva en un elemento <AllOf>, no entre todos como ocurría con <AnyOf>.
    - El elemento<AllOf> ha de contener una secuencia conjuntiva de directivas <Match>, debe identificar un conjunto de entidades haciendo coincidir los valores de los atributos en un elemento <Attributes> del contexto de la solicitud con el valor del atributo incorporado.
    - <Match> representa una expresión booleana sobre los atributos del contexto de la solicitud. MatchId especifica la función que se utilizará para realizar la evaluación de la coincidencia, el elemento <AttributeValue> indica el valor específico que se pretende comparar y el elemento <AttributeDesignator> especifica la categoría del atributo que va a ser comparado.

En las Figuras 3 y 4 se muestran ejemplos de cómo se combinan estas clausulas para verificar el cumplimiento de una política.

Como XACML utiliza el control de acceso basado en atributos, todos los atributos se clasifican en cuatro categorías principales:

- Usuario que solicita el acceso. (urn:oasis:names:tc:xacml:3.0:attribute-category:subject). Entre sus atributos pueden estar su identificador, su rol, su organización a la que pertenece, relación con el recurso al que está accediendo, etc.
- Recurso que pretende acceder el usuario. (urn:oasis:names:tc:xacml:3.0:attribute-category:resource). Por ejemplo, /v2/entities o/v2/subscriptions.
- Tipo de acción que solicita el usuario. (urn:oasis:names:tc:xacml:3.0:attribute-category:action). Por ejemplo, POST, GET, DELETE, PUT o PATCH.
- Contexto, esta categoría hace referencia a aquellos atributos que no están asociados con la acción, recurso y usuario. Por ejemplo, franja horaria a la que se pretende acceder, o bien un atributo especifico enviado por Wilma del que se pretende evaluar con un cierto valor. Esta categoría será vital para la realización del proyecto. (urn:oasis:names:tc:xacml:3.0:attribute-category:environment)

Caso donde la condición A y B deben cumplirse.

```
<Target>
  <AnyOf>
    <AllOf>
      <Match> condicionA </Match>
      <Match> condicionB </Match>
    </AllOf>
  </AnyOf>
</Target>
```

Figura 3. Ejemplo simple de la sentencia <Target>.

Caso donde se puede cumplir la condición A o la B

```
<Target>
  <AnyOf>
    <AllOf>
      <Match> condicionA </
    AllOf>
    <AllOf>
      <Match> condicionB </Match>
    </AllOf>
  </AnyOf>
</Target>
```

Figura 4. Ejemplo simple de la sentencia <Target>.

- Effect: El efecto de la regla indica la consecuencia prevista después de una evaluación exitosa para la regla en cuestión. Se permiten dos valores, “Permitir” y “Denegar”.
- Condition: Las condiciones solo existen en las reglas. Las condiciones son esencialmente una forma avanzada de un <Target> que puede usar una gama más amplia de funciones y, lo que es más importante, puede usarse para comparar dos o más atributos juntos. <Condition> contiene un elemento <Expression>, con la restricción de que el tipo de datos de retorno <Expression> debe ser un booleano. Los siguientes elementos están en el grupo de sustitución de elementos <Expression>:
    - <Apply>: Aplica una función a los argumentos que derivan de este elemento. El elemento <Apply> debe contener el siguiente atributo: FunctionId, que indica el identificador de la función que se aplicará a los argumentos, dichos argumentos pueden ser <AttributeValue>, <AttributeDesignator> o bien otras funciones. Se muestra un ejemplo del proyecto en la Figura 5.

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
  AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization" DataType="http://www.w3.org/2001/XMLSchema#string"
  MustBePresent="false" />
 </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
  </Apply>
```

Figura 5. Ejemplo de la directiva <Apply>.

El primer <Apply> va a aplicar la función de igualdad de Strings. El segundo <Apply> indica que la bolsa de valores de la categoría environment:organization debe contener un único valor, si esta vacía o contiene más de un valor devuelve indeterminado. El atributo de esta categoría será comparado con el valor especificado en <AttributeValue> gracias al primer <Apply> comentado. Y si es exactamente igual, devolverá "True", es decir un valor booleano tal y como se comentó anteriormente.

- <VariableReference>: Se utiliza para hacer referencia a un valor definido dentro de la misma política, es decir, <Policy>.

- Obligation expressions y Advice expressions: Una regla, política o conjunto de políticas puede contener una o más expresiones de obligación o consejo. Cuando se evalúa tal regla, política o conjunto de políticas, la expresión de obligación o consejo se evaluará como una obligación o consejo respectivamente, que se pasa al siguiente nivel de evaluación solo si el resultado de la regla, política o política conjunto que se está evaluando coincide con el valor del atributo FulfillOn de la obligación o el atributo AppliesTo del aviso.

El consejo es similar a la obligación y comparte gran parte de su sintaxis. La diferencia es contractual: el PEP puede ignorar cualquier consejo que reciba. Las PEP no tienen que cumplir con las declaraciones de asesoramiento; Las PEP pueden considerar o descartar la declaración.

```
<ObligationExpressions>
  <ObligationExpression FulfillOn = "Permit" ObligationId =
"email_obligation">
    <AttributeAssignmentExpression AttributeId = " e mail">
       <AttributeValue DataType = "
http://www.w3.org/2001/XMLSchema#string "> pamodaaw @ gmail .com
</AttributeValue>
     </AttributeAssignmentExpression>
  </ObligationExpression>
</ObligationExpressions>
```

Figura 6. Ejemplo de la sentencia <ObligationExpressions>.

En la expresión de obligación anterior, la obligación se agregará a la respuesta cuando solo la decisión sea "Permit". El valor de obligación, "pamodaaw@gmail.com" se agregará a la respuesta hacia el PEP.

- Policy: Contiene un conjunto de elementos <Rule> y un procedimiento específico para combinar los resultados de su evaluación. Es la unidad básica de la política utilizada por el PDP, por lo que está destinada a formar la base de una decisión de autorización. El elemento <Policy> contiene los siguientes atributos y elementos:

  - PolicyId: Identificador de la política. Es responsabilidad del PAP asegurarse de que no haya dos políticas visibles para el PDP que tengan el mismo identificador.

  - Version: El número de versión de la Política.

  - RuleCombiningAlgId: El algoritmo de combinación de reglas especifica el procedimiento mediante el cual se combinan los resultados de la evaluación de las reglas de los componentes al evaluar la política. Algunos ejemplos de algoritmos de combinación estándar son:

  - Deny-overrides (urn: oasis: names: tc: xacml: 3.0: rule-combine-algorítm: deny-overrides). Si alguna decisión es "Deny", el resultado es " Deny".
  - Permit-overrides(urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides). Si alguna decisión es " Permit", el resultado es " Permit".
  - Deny-unless-permit (urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit). El resultado será “Deny”, excepto si la decisión es “Permit” donde el resultado será “Permit”. Este algoritmo va a ser el utilizado en el proyecto.

- Target: El elemento <Target> define la aplicabilidad de una <Policy> a un conjunto de solicitudes de decisión. Concretamente, en nuestro proyecto se comparará el tipo de aplicación permitido con el tipo de aplicación que pertenece el usuario que pretende acceder al sistema.
- Rule: Podrá haber cualquier número de reglas dentro de una política.

- PolicySet: Es un elemento de nivel superior en el esquema de política XACML. <PolicySet> es una agregación de otros conjuntos de políticas. Los conjuntos de políticas pueden incluirse en un elemento <PolicySet> adjunto, ya sea directamente utilizando el elemento <PolicySet> o indirectamente utilizando el elemento <PolicySetIdReference>.

  El elemento <PolicySet> contiene los siguientes atributos y elementos:

  - PolicySetId: Identificador de conjunto de políticas. Es responsabilidad del PAP asegurarse de que no haya dos políticas visibles para el PDP que tengan el mismo identificador.
  - Version: El número de versión de PolicySet, para la modificación de alguna política debemos incrementar en uno el número de la versión. Más adelante, se detallará la creación de las políticas.
  - PolicyCombiningAlgId: El algoritmo de combinación de políticas especifica el procedimiento mediante el cual se combinan los resultados de la evaluación de las políticas. Se utilizan los mismos algoritmos que las reglas. Se usará "urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit",el cual dará como resultado “Deny”, excepto si la decisión es “Permit” donde el resultado será “Permit”.
  - Target: El elemento <Target> define la aplicabilidad de un conjunto de políticas a un conjunto de solicitudes de decisión. En nuestro proyecto estará vacío, ya que la aplicabilidad del sistema se dará en el <Target> perteneciente al elemento <Policy>, al <Target> perteneciente a <Rule> y las condiciones que se establezcan en <Condition>.
  - Policy: Podrá haber cualquier número de políticas dentro de un conjunto de políticas.

XACML define una larga lista de funciones para manipular y comparar atributos con otros atributos y valores, se indicarán concretamente aquellas funciones que van a ser utilizadas en el proyecto:

- Igualdad, desigualdad y otras funciones de emparejamiento. Por ejemplo:
  - urn:oasis:names:tc:xacml:1.0:function:string-equal. Esta función debe tomar dos argumentos de tipo de string y deberá devolver un booleano, si el valor de los dos argumentos es exactamente el mismo devolverá True, en caso contrario False.
- Funciones aritméticas.
- Funciones de comparación no numéricas.
  - urn:oasis:names:tc:xacml:2.0:function:time-in-range. Esta función debe tomar tres argumentos de tipo Time y deberá devolver un booleano. Devolverá True si el primer argumento cae en la franja horaria establecida por el segundo y tercer argumento. De lo contrario, devolverá False.
- Funciones de cadena.
  - urn:oasis:names:tc:xacml:1.0:function:string-one-and-only. Esta función garantiza que su argumento se evalúa como una bolsa que contiene exactamente un valor, ya que AttributeDesignator que devuelve una bolsa de valores. Dicha bolsa, puede estar vacía, que contenga un único valor o una lista de valores tal y como se comentó anteriormente. La misma acción tendrá la siguiente función, urn:oasis:names:tc:xacml:1.0:function:time-one-and-only.

  - urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of. Esta función permite comparar una lista de valores con el valor de un atributo especifico.
  - urn:oasis:names:tc:xacml:1.0:function:string-bag. Esta función indica que la bolsa de valores que se va a indicar va a ser de tipo String.
  - urn:oasis:names:tc:xacml:3.0:function:string-starts-with. Esta función debe tomar dos argumentos de tipo String y devolverá un booleano. El resultado será "True" si la segunda cadena comienza con la primera cadena y "False" en caso contrario.
- Funciones lógicas.
  - urn:oasis:names:tc:xacml:1.0:function:and
  - urn:oasis:names:tc:xacml:1.0:function:or
- Funciones de set y bag.
- Funciones de orden superior.
- Funciones de expresión regular.
- Funciones XPath.

2.3. FIWARE.

FIWARE [2] es una plataforma de código abierto que define un conjunto de estándares para la gestión de información de contexto siguiendo el paradigma IoT. FIWARE proporciona una serie de componentes que facilitan el despliegue de soluciones inteligentes con el fin de obtener y gestionar información del entorno para la toma de decisiones automatizadas.

Figura 7. Logo de Fiware.

El elemento principal que permite a FIWARE recopilar y administrar la información de contexto es el Orion Context Broker, un servidor web que permite peticiones de actualización o suscripción a los datos, utilizando una API de tipo REST sobre el protocolo de transporte HTTP.

También se describirán los elementos FIWARE Generic Enabler. Estos componentes permiten llevar a cabo la tarea de securización del sistema y el procesamiento de la información de contexto producida por los agentes IoT.

2.3.1 Orion Context Broker

Orion Context Broker [4] es el componente principal para cualquier solución que se desarrolle bajo el paraguas de FIWARE, ya que mediante él se puede administrar la información de contexto, consultarla y actualizarla. El Context Broker está rodeado a su vez de elementos adicionales que permiten suministrar datos de diversas fuentes y entornos que, trabajando en este elemento se convierten en herramientas para el procesamiento de datos, análisis, control de acceso o la implementación y propia configuración de la plataforma FIWARE.

Figura 8. Logo OCB.

Una función principal soportada por el OCB es el de lograr una disociación total entre productores y consumidores de contexto. Los productores de contexto son aquellos que publican datos sin saber qué, dónde y cuándo los consumidores de contexto consumirán los datos publicados; por lo tanto, no necesitan estar conectados a ellos. Por otro lado, los consumidores de contexto consumen información de contexto de su interés, sin que conozcan al productor de contexto que publica un evento en particular. Están interesados en el evento en sí, y no en quien lo generó.

Un elemento de contexto se refiere a la información que es producida o observada y que puede ser relevante para su procesamiento, análisis y extracción de nuevo conocimiento. Tiene asociado un valor definido, que consiste en una secuencia de uno o más triplas que se refieren a atributos de un elemento de contexto. Además, proporciona información relevante a una entidad en particular, la cual puede ser un componente físico o parte de una aplicación. Por ejemplo, un elemento de contexto puede contener valores de atributos asociados a una habitación como la última temperatura medida, los metros cuadrados que mide y el color de la pared. Por lo general, elemento de contexto contiene un id (EntityId) y un tipo (EntityType) que identifica exclusivamente a una entidad. Adicionalmente, pueden existir metadatos, vinculados a los atributos a un elemento de contexto. Sin embargo, la existencia de metadatos vinculados a un atributo de elemento de contexto es opcional.

En resumen, un elemento de contexto puede contener los siguientes parámetros:

- Un EntityId y un EntityType único que identifica la entidad a la cual los datos de contexto hacen referencia.
- Una secuencia de uno o más atributos de elementos de datos.
- Metadatos opcionales vinculados a atributos.

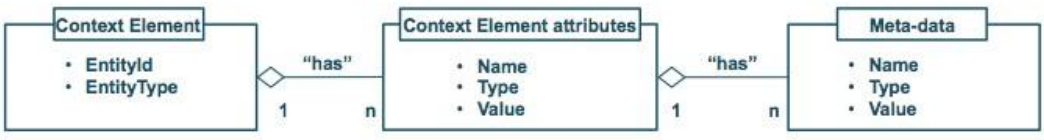

Figura 9. Esquema Context Element.

El OCB utiliza la base de datos MongoDB para almacenar el estado actual de las entidades, sin almacenar información histórica de sus cambios. Para este propósito se debe utilizar una base de datos externa al OCB.

Orion no proporciona autenticación "nativa" ni ningún mecanismo de autorización para hacer cumplir el control de acceso, por tanto, para llevar a cabo los diferentes escenarios debemos utilizar una serie de componentes FIWARE para garantizar diferentes niveles de seguridad.

2.3.2 Identity Manager Keyrock

Keyrock [5] es el componente encargado de proporcionar mecanismos para gestionar la identidad de aquellos usuarios que intenten acceder al sistema, con la finalidad de conocer en todo momento quién o qué está accediendo a la información de contexto del sistema.

Keyrock implementa una interfaz gráfica donde el administrador puede registrar aplicaciones, organizaciones y usuarios y asignar diferentes roles a los usuarios creados. Keyrock se interconecta con la base de datos MySQL para almacenar toda la información relativa a aplicaciones, organizaciones, usuarios, token de acceso y roles.

Para la autenticación, lo habitual, será primero solicitar a Keyrock el token de acceso, indicando el usuario y contraseña en el mensaje de solicitud. En caso de estar registrado dicho par usuario-contraseña en la base de

datos, Keyrock dará como respuesta el token de acceso, el cual identificará al usuario.

Por consiguiente, el usuario para solicitar algún recurso en el sistema ha de indicar en la propia solicitud el token de acceso. La finalidad de esto sería que Keyrock decida si se ha autenticado correctamente en función de si el token indicado por el usuario está almacenado en la base de datos.

Figura 10. Logo de Keyrock.

2.3.3 PEP Proxy Wilma

El FIWARE Generic Enabler PEP Proxy Wilma [6], es el componente que va a permitir la activación del mecanismo de autenticación y autorización de nuestro sistema, siendo el intermediario entre Keyrock, la aplicación y Authzforce.

Las solicitudes de proxy deben hacerse con una cabecera especial HTTP Header: X-Auth-Token. Este encabezado contiene el token de acceso obtenido mediante Keyrock.

PEP-Proxy Wilma actuará como Punto de Aplicación de la Política (PEP en sus siglas del inglés), teniendo el siguiente funcionamiento:

1. Intercepta las peticiones de los usuarios a un recurso.
2. Realiza una petición de autenticación a Keyrock, el cual consultará si dicho usuario está registrado mediante su token y, por consiguiente, emitirá una decisión de acceso.
3. Wilma escuchará la decisión de autenticación, denegando el acceso del usuario o realizando una petición a Authzforce para obtener una decisión de autorización en función del usuario, recurso y acción que pretenda realizar.
4. Wilma denegará el acceso al usuario o dará como respuesta la petición que ha solicitado consultado de esta manera a Orion.

2.3.4 Authzforce

AuthzForce [7] es un componente FIWARE que proporciona un marco de control de acceso basado en atributos (ABAC), facilitando una API para obtener decisiones de autorización basadas en políticas de autorización y solicitudes de autorización desde un PEP. Authzforce evalúa decisiones de autorización basadas en políticas XACML y atributos relacionados con una solicitud de acceso determinada (por ejemplo, identidad del solicitante, recurso solicitado, acción solicitada), siguiendo la lógica de evaluación de políticas definida en XACML.

Figura 11. Logo Authzforce.

Authzforce además de actuar como PDP, puede actuar como punto de administración de políticas (PAP), esto significa que los PolicySets se pueden crear y modificar utilizando llamadas API directamente a Authzforce.

Sin embargo, no existe una GUI para crear o modificar una <PolicySet>. Por tanto, creamos dichas políticas mediante el terminal. La explicación de la creación y puesta en funcionamiento de las políticas se llevará a

cabo en el Apartado 3.1.4.

2.4 Niveles de seguridad.

Dentro del marco de la seguridad en sistemas informáticos se definen tres niveles de control de acceso.

- Nivel Autenticación de Seguridad: Se obtiene cuando el sistema dispone de un mecanismo de autenticación capaz de tener una gran cantidad de usuarios asociados a varias aplicaciones. Cada usuario tendrá acceso a los recursos que la aplicación en la que esté registrado permita. Con este nivel podremos permitir o denegar desde Keyrock el acceso a todos o a ninguno de los recursos en función de si el usuario es válido o no.

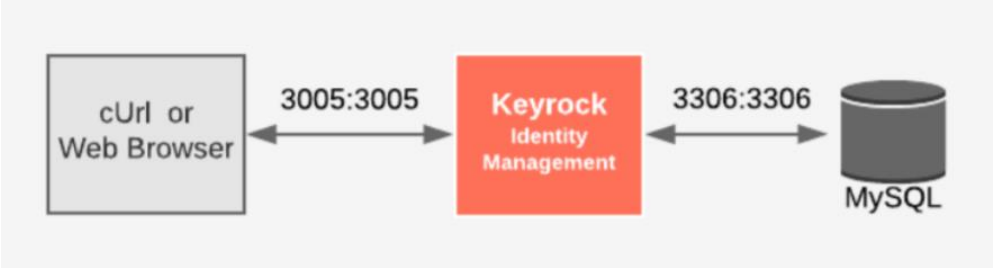

Figura 12. Nivel autenticación.

En este nivel de seguridad, Keyrock ejerce como PDP, por lo que Keyrock es el encargado de tomar la decisión de aceptar o denegar el acceso de un usuario en función de si el usuario es válido o no.

- Nivel Básico de Seguridad: Se logra a partir de la asignación de roles a los diferentes usuarios que están registrados en el sistema. Primero se crean una serie de organizaciones a las que van a pertenecer un conjunto de usuarios. Después de establecer las diferentes organizaciones, se crean una serie de roles a las que se le asignan permisos a cada uno de estos roles. En este nivel, Keyrock sigue siendo el encargado de la toma de decisiones, y también Wilma sigue siendo el componente que ejecute dicha decisión.

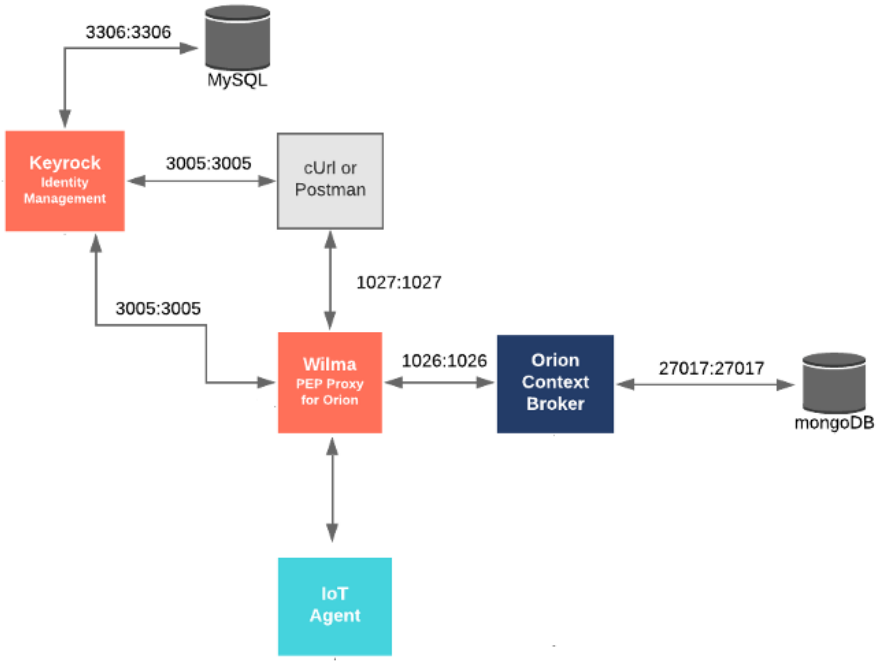

Figura 13. Nivel básico de seguridad.

- Nivel Avanzado de Seguridad: Se logra al incluir un nuevo componente denominado Authzforce, el cual va a proporcionar una API para obtener decisiones de autorización basadas en unas políticas determinadas y en las peticiones procedentes de los PEPs. Dichas políticas son generadas mediante el estándar XACML que describe el formato de las políticas de autorización y lógica de evaluación.

De esta manera, este nuevo componente va a ejercer como Punto de Decisión de la Política, en vez de Keyrock como se ha descrito anteriormente.

Cabe destacar que Keyrock ha de estar presente en este nivel avanzado de seguridad para proporcionar el nivel de autenticación, además PEP Proxy Wilma para funcionar tiene que estar conectado obligatoriamente a un Idm.

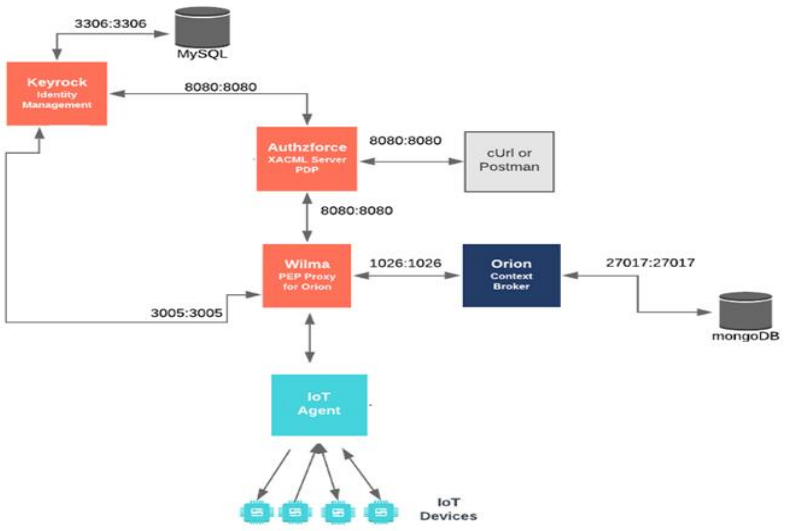

Figura 14. Nivel avanzado de seguridad.

2.5 Tecnologías y recursos software.

2.5.1 Máquina virtual Ubuntu 18.04.2 LTS

Para el desarrollo del proyecto se ha decidido usar el Sistema Operativo Ubuntu 18.04.2 LTS. La máquina virtual se ha creado utilizando VirtualBox 6.1.12 ya que por medio de esta aplicación es posible instalar sistemas operativos adicionales, dentro de otro sistema operativo anfitrión, en este caso Windows, cada uno con su propio ambiente virtual.

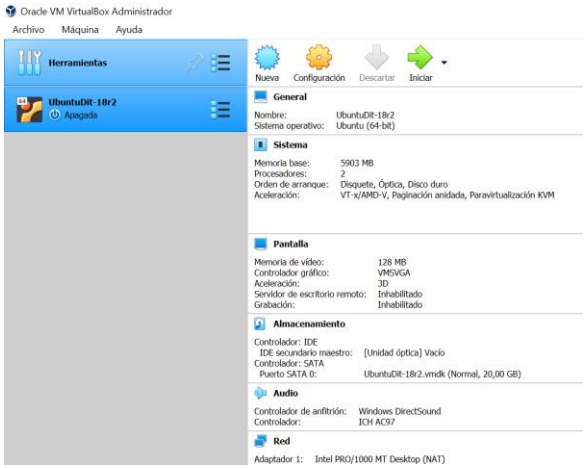

Figura 15. Máquina virtual Ubuntu.

2.5.2 Postman

Postman [8] es un servicio que permite probar un cliente REST realizando simples peticiones de la API, todo ello a través de una interfaz web en la que podemos interactuar fácilmente. Proporciona diferentes tipos de peticiones (GET, POST, PUT, …). Este programa permite simular diversas peticiones a Orion Context Broker:

- Un médico puede realizar consultas hacia entidades creadas previamente, las cuales identifican la actividad física de un paciente en concreto.
- Un médico puede suscribirse a entidades con el fin de obtener notificaciones cuando algún atributo de dicha entidad suscrita se actualice.
- Un agente IoT puede crear una entidad correspondiente a la actividad física de un paciente.

Figura 16. Logo de Postman.

2.5.3 Docker

Docker [9] es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Figura 17. Logo Docker.

Docker es un "emulador" de entornos aislado para poder ejecutar programas sin que afecte al sistema operativo (SO) y pudiéndose llevar y replicar en otros SS.OO. o entornos.

Docker consta de imágenes y contenedores:

- Una imagen es la especificación inerte, inmutable, una foto del estado y de unas piezas de software que incluyen desde la aplicación que queremos ejecutar hasta las librerias y todo lo necesario para que se ejecute encima del sistema operativo que lo contiene.
- Un contenedor es un entorno aislado con la instanciación de una imagen, el cual se puede configurar.

No obstante, hace falta incluir la herramienta Docker-compose [10] para poder hacer pulling de aplicaciones que estén compuestas por varios contenedores relacionados entre sí. En este proyecto, los contenedores Orion Context Broker y MongoDB están relacionados, al igual que Keyrock con MySQL y Authzforce. Se detallará la instalación de estos componentes en el ANEXO A.

Figura 18. Logo Docker-Compose.

2.5.4 MongoDB y MySQL

MongoDB [11] es una base de datos orientada a documentos. Esto quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Orion Context Broker va a utilizar MongoDB para almacenar toda la información relativa a las entidades.

Figura 19. Logo MongoDB.

MySQL [12] es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo.

En este proyecto MySQL se encargará de almacenar toda la información referente a Keyrock idm, con el fin de poder llevar a cabo la correcta autenticación del usuario en el sistema.

Figura 20. Logo MySQL.

3 RESULTADOS

Antes de describir cada escenario, se va a explicar la solución planteada final, desarrollando más en detalle la funcionalidad de los diferentes componentes usados.

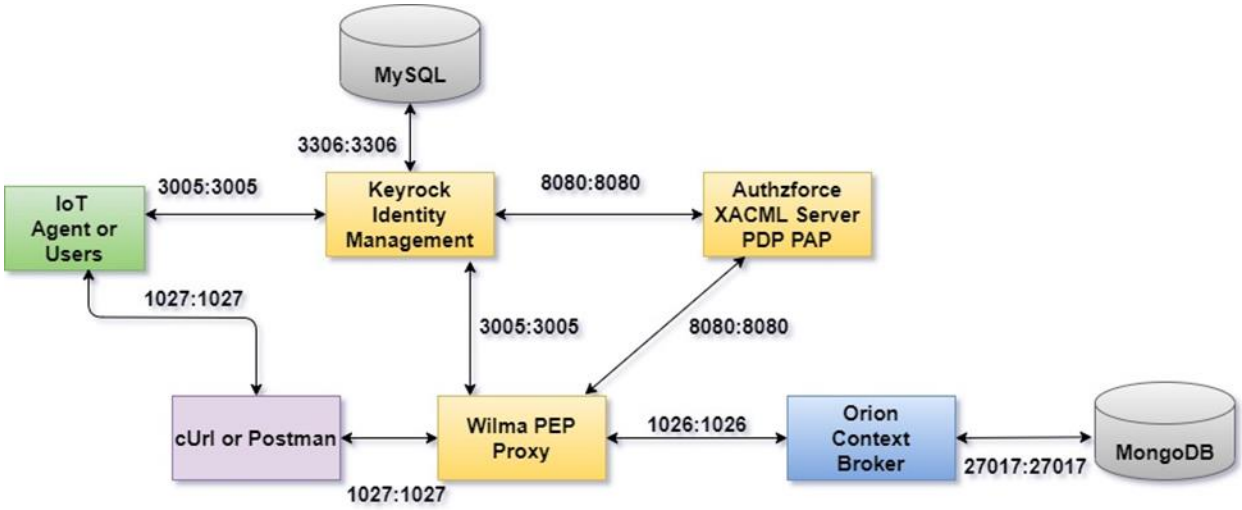

Figura 21. Esquema de la solución final planteada.

3.1 Funcionalidad de cada componente en el sistema planteado.

3.1.1 Orion Context Broker

Orion Context Broker va a ser el componente que se pretende proteger de entidades no autorizadas. Orion permite la creación de entidades y la consulta y suscripción de estas.

Para este proyecto se va a trabajar con entidades que contienen datos de la actividad física de un paciente, a partir de la información recogida y enviada a un agente IoT por la camiseta inteligente. El tipo de estas entidades es "ActividadFisica" y su id comenzará de esta manera, "urn:ngsi-ld:sensor:". Además, contendrá una serie de atributos:

- "publisher": Es el agente IoT que publica la entidad.
- "id_patient": Es el usuario que utiliza la camiseta.
- "timestamp": Indica la hora a la que se realizó la prueba.
- "id_device": Identifica la camiseta usada.
- "organization": Organización perteneciente al paciente y camiseta.
- "accumulates_metabolic_expenditure". Se trata del gasto calórico total del paciente.
- "instant_metabolic_expenditure". Es el gasto calórico instantáneo en el momento en el que se envía el mensaje.

```
{
  "id":"urn:ngsi-ld:sensor:001","type":"ActividadFisica",
 "publisher": "Agente2001",
 "id_patient": "222222222A",
 "timestamp": {"type": "DateTime", "value":"2020-07-24T09:30:23.238Z"},
  "id_device" : "11:1E:C0:25:E6:99",
  "organization": "ResidenciaSevilla",
  "accumulated_metabolic_expenditure" :{"type": "float", "value": "5.59"},
  "instant_metabolic_expenditure":{"type": "float", "value": "0.05"}
}
```


Figura 22. Ejemplo de entidad de tipo "ActividadFisica".

La configuración de Orion llevada a cabo en el proyecto se comenta en el Anexo B.

3.1.1 Keyrock

Keyrock se ha configurado para escuchar peticiones en el puerto 3005, y su función consiste en garantizar la identidad de todo aquel usuario o agente IoT que quiera acceder al sistema para lo cual tendrá que estar registrado previamente.

El registro de los usuarios, organizaciones, PEP Proxy, roles, etc., lo puede realizar el administrador del sistema mediante la interfaz gráfica de Keyrock. Pero se ha obtado a crear un estado inicial de la base de datos MySQL donde ya se configura todo el registro. El estado inicial se explica en el Anexo B.

Wilma acudirá a Keyrock para delegar una decisión sobre la autenticación del usuario o agente IoT. El proceso de autenticación en este proyecto se explica en el apartado 3.2.

3.1.2 PEP Proxy Wilma

Wilma está configurado para escuchar peticiones en el puerto 1027. Cuando intercepte una petición, Wilma comprobará la autenticación del usuario consultando a Keyrock, proporcionándole a este componente el token de acceso indicado por el usuario. Si el usuario está registrado, Wilma enviará a Authzforce una solicitud de decisión de autorización para comprobar si el usuario puede realizar una consulta, publicación o suscripción de una entidad. En dicha solicitud, Wilma indicará una serie de parámetros de gran importancia para Authzforce. Los cuales son:

- El rol que tiene el usuario que está intentando acceder, Wilma obtiene este valor a partir de la respuesta que le da Keyrock al ser autenticado correctamente.
- El id de la aplicación donde está registrado el usuario.
- La acción que pretende realizar, por ejemplo, POST, GET, DELETE O PATCH.
- El recurso que pretende acceder el usuario, en nuestro proyecto será /v2/entities o /v2/subscriptions.
- Se enviarán atributos que tratan el tiempo y aspectos dinámicos del escenario de control de acceso, como por ejemplo los atributos expuestos en la publicación de una entidad.

Si Authzforce permite la acción solicitada por el usuario sobre el recurso que haya indicado Wilma redirigirá la petición hacia Orion, el cual dará una respuesta dependiendo de la petición que haya realizado el usuario en cuestión.

Wilma se ha instalado mediante código fuente con el fin de poder modificar los ficheros de configuración para cumplir los requisitos de los diferentes escenarios que se plantearán a continuación. En cada escenario se especifican los ficheros que se han modificado.

3.1.3 Authzforce

Authzforce va a permitir obtener decisiones de autorización basadas en políticas de autorización, y peticiones de autorizacion de PEPs, en este caso Wilma. En nuestro sistema, Authzforce además de actuar como PDP, va a actuar como punto de administración de políticas (PAP), por tanto, podremos crear y modificar PolicySets utilizando directamente Authzforce.

La creación y puesta en marcha de las políticas se podría realizar en tres pasos:

1. Hay que crear un dominio donde se van a almacenar las políticas. Mediante el siguiente comando:

```
salas@linux:~$ curl --location --request POST 'http://localhost:8080/authzforce-ce/domains' \
> --header 'Content-Type: application/xml' \
> <domainProperties xmlns="http://authzforce.github.io/rest-api-model/xmlns/authz/5" externalId="escenario_sanitario"/>'
```

Figura 23. Creación de un dominio de políticas.

La respuesta indicará el id del dominio que será necesario para el siguiente paso.

A la misma vez que se crea el dominio, se establecen por defecto una política y dos ficheros de configuración de dominio.

- 0.1.0.xml: Dicha política por defecto permite el acceso a cualquier usuario.
- pdp.xml: Dicho fichero indica la cantidad de PolicySets y de VariableDefinition que puede haber en una política.
- properties.xml: Indica el número máximo de políticas en un dominio y número máximo de versiones por política.

2. Para crear un PolicySet para un dominio determinado en Authzforce, hay que realizar una solicitud POST al /authzforce-ce/domains/domain-id/pap/policies, incluyendo el conjunto completo de reglas XACML para cargar. La respuesta contiene la identificación interna de la política contenida en Authzforce e información sobre las PolicySet versiones disponibles. Las reglas del nuevo PolicySet no se aplicarán hasta que PolicySet se active. En la Figura 24 se muestra un ejemplo.

```
curl --location --request POST 'http://localhost:8080/authzforce-ce/domains/ffsR-txyEeqvSwJCrBIBDA/pap/policies' \
--header 'Content-Type: application/xml' \
--data-raw '<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicySetId="f8194af5-8a07-486a-9581-c1f05d05483c" Version="48"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
   <Description>Politicas para escenarios sanitarios</Description>
   <Target />
<Policy PolicyId="escenario_sanitario_suscripcion" Version="1.0" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
      <Description>Politicas para la suscripcion por parte de los medicos</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">escenario_sanitario</AttributeValue>
                  <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                  DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
               </Match>
            </AllOf>
         </AnyOf>
      </Target>
      <Rule RuleId="Reglas_Medicos_Hospital_Central_suscripcion" Effect="Permit">
         <Description>Reglas medicos del Hospital Central suscripcion</Description>
         <Target>
           <AnyOf>
               <AllOf>
                  <Match MatchId="urn:oasis:names:tc:xacml:3.0:function:string-starts-with">
                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/v2/suscriptions</AttributeValue>
                     <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                     AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
                     DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                  </Match>
               </AllOf>
            </AnyOf>
```

```
            <AnyOf>
               <AllOf>
                  <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">POST</AttributeValue>
                     <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                     DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                  </Match>
               </AllOf>
            </AnyOf>
         </Target>
</Rule>
   </Policy>
</PolicySet>'
```

Figura 24. Ejemplo de creación de una política.

Tras realizar esta solicitud POST, se crea un fichero donde se almacena la política definida. El nombre que tendrá dicho fichero será: VERSION.xml; en este ejemplo sería 48.xml.

3. Para activar un PolicySet, hay que realizar una solicitud PUT al /authzforce-ce/domains/domain-id/pap/pdp.properties, incluyendo el valor de PolicySetId dentro de la directiva <rootPolicyRefExpresion>. Dicho valor se indicó en el código XACML enviado en la solicitud anterior. La respuesta devuelve información sobre la PolicySet aplicada. La solicitud sería la que se muestra en la Figura 25.

```
salas@linux:~$ curl --location --request PUT 'http://localhost:8080/authzforce-ce/domains/ffsR-txyEeqvSwJCrBIBDA/pap/pdp.properties' \
> --header 'Content-Type: application/xml' \
> --data-raw '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><pdpPropertiesUpdate xmlns="http://authzforce.github.io/rest-api-model/xmlns/authz/5">
> <rootPolicyRefExpression>f8194af5-8a07-486a-9581-c1f05d05483c</rootPolicyRefExpression></pdpPropertiesUpdate>'
```

Figura 25. Ejemplo de activación de una política.

A continuación, se va a describir el diseño de los escenarios de prueba.

3.2 Diseño de los escenarios de prueba

Se ha pensado en un escenario sanitario general con dos organizaciones, Hospital Central y Residencia, independientes entre sí. Es decir, aunque harán uso de la misma plataforma IoT (y compartirán el mismo Orion) buscaremos una separación lógica que vendrá soporta por el mecanismo de control de acceso.

Mediante Keyrock se ha configurado la organización Hospital Central donde se han registrado: 4 médicos, 2 de ellos trabajan en el turno de mañana y los otros dos en el turno de tarde. El turno de mañana va de 9 de la mañana a 3 de la tarde, y el turno de tarde de 3 de la tarde a 9 de la noche.

Médico	Correo	Contraseña	Horario
José	jose_hospital_central@tfg.com	test	Mañana
Pablo	pablo_hospital_central@tfg.com	test	Mañana
Fernando	fernando_hospital_central@tfg.com	test	Tarde
María	maria_hospital_central@tfg.com	test	Tarde

Tabla 8. Usuarios registrados del Hospital Central.

También se han registrado dos Agentes IoT mediante Keyrock para esta organización, donde uno de ellos va a estar en pruebas, de modo que va a estar asociado a un solo paciente y a una única camiseta. Mientras que el otro va a estar asociado a tres pacientes diferentes pudiendo publicar información de la actividad física de dos camisetas que sean utilizadas por estos pacientes. Además, se ha establecido un horario de publicación, de 9 de la mañana a 5 de la tarde, el cual se debe cumplir. Fuerta de este horario, no estará autorizada la publicación.

Los Agentes IoT se han registrado como usuarios de la organización para que puedan ser autenticados correctamente y por consiguiente puedan ser autorizados en función de lo que pretendan publicar.

Agente_IoT	Correo	Contraseña	IdPacientes	IdCamisetas
Agente IoT 1000	agente_iot_hospital_central_1000@tfg.com	test	123456789F	00:1E:C0:25:E6:99
Agente IoT 1001	agente_iot_hospital_central_1001@tfg.com	test	123456789F, 987654321A, 123456789A	00:1E:C0:25:E6:99, 00:1E:C0:25:F6:90

Tabla 9. Agentes IoT registrados del Hospital Central.

Se ha registrado la dirección IP de 3 PCs de esta organización para asegurar que las notificaciones sean enviadas a dichas direcciones ante la actualización de alguna entidad a la que algún médico del Hospital Central se haya suscrito.

- 172.18.1.1
- 172.18.1.2
- 172.18.1.3

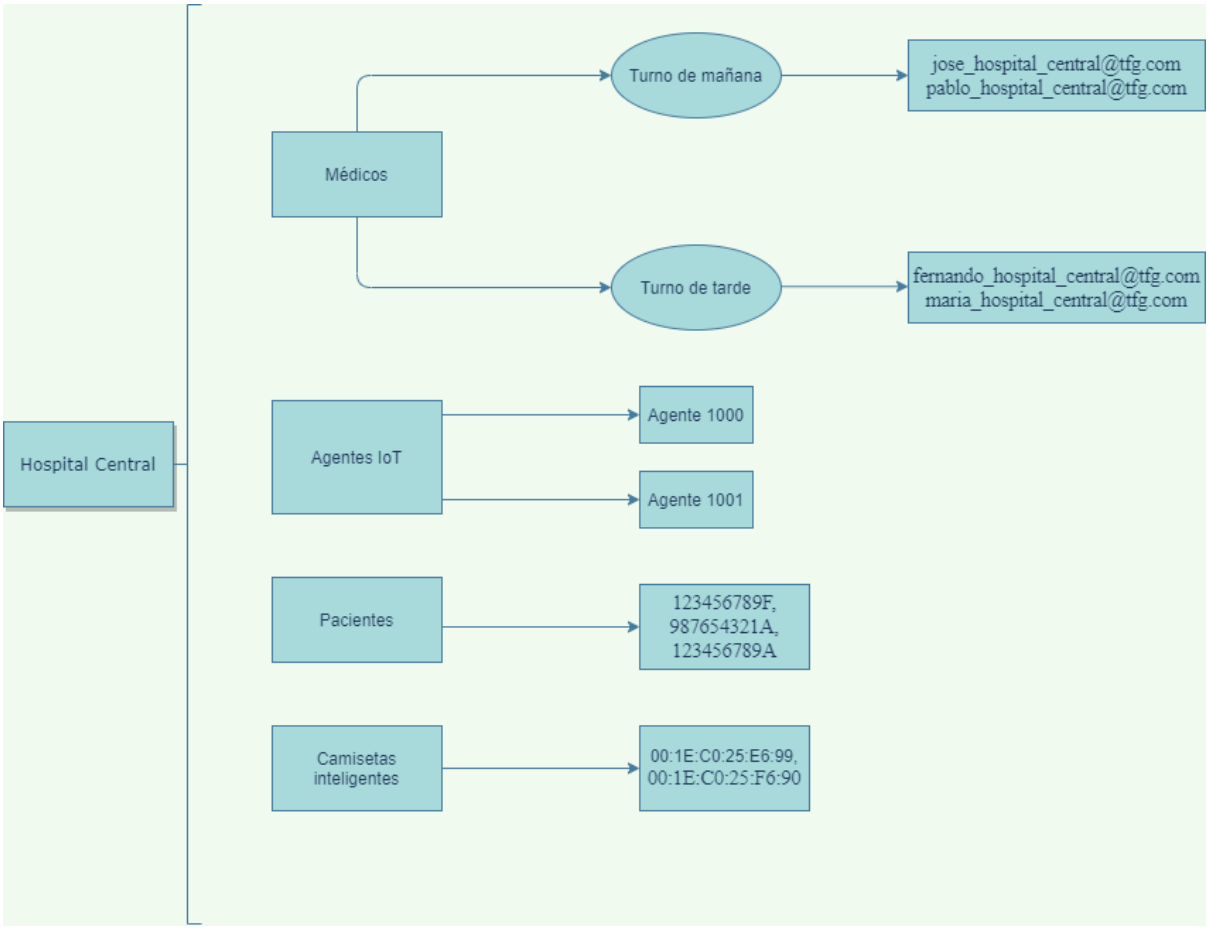

Figura 26. Esquema de los elementos pertenecientes al Hospital Central.

Con respecto a la Residencia, se va a trabajar de la misma manera, 4 médicos, dos de ellos de turno de mañana y los otros dos con turno de tarde. Lo mismo con los Agentes IoT, uno de ellos de prueba y el otro más complejo. El horario de publicación establecido en esta organización es de 11 de la mañana a 7 de la tarde.

Médico	Correo	Contraseña	Horario
Ana	ana_residencia_sevilla@tfg.com	test	Mañana
Sergio	sergio_residencia_sevilla@tfg.com	test	Mañana
Rafael	rafael_residencia_sevilla@tfg.com	test	Tarde
Marta	marta_residencia_sevilla@tfg.com	test	Tarde

Tabla 10. Usuarios registrados de la Residencia de Sevilla.

Agente_IoT	Correo	Contraseña	IdPacientes	IdCamisetas
Agente IoT 2000	agente_iot_residencia_sevilla_2000@tfg.com	test	111111111A	11:1E:C0:25:E6:99
Agente IoT 2001	agente_iot_residencia_sevilla_2001@tfg.com	test	111111111A, 222222222A, 333333333A	11:1E:C0:25:E6:99, 11:1E:C0:25:F6:90

Tabla 11. Agentes IoT registrados de la Residencia de Sevilla

Se han registrado también otras direcciones IP correspondientes a 3 PCs distintos de esta organización. Al trabajar con la misma máquina virtual se ha optado trabajar con la misma subred, pero no tiene por qué estar en la misma. La única restricción que se debe asegurar es que Orion sea capaz de tener una conexión con la dirección que va a recibir las notificaciones. Por lo contrario, no se recibirá notificaciones a dicha dirección.

- 172.18.1.4
- 172.18.1.7
- 172.18.1.10

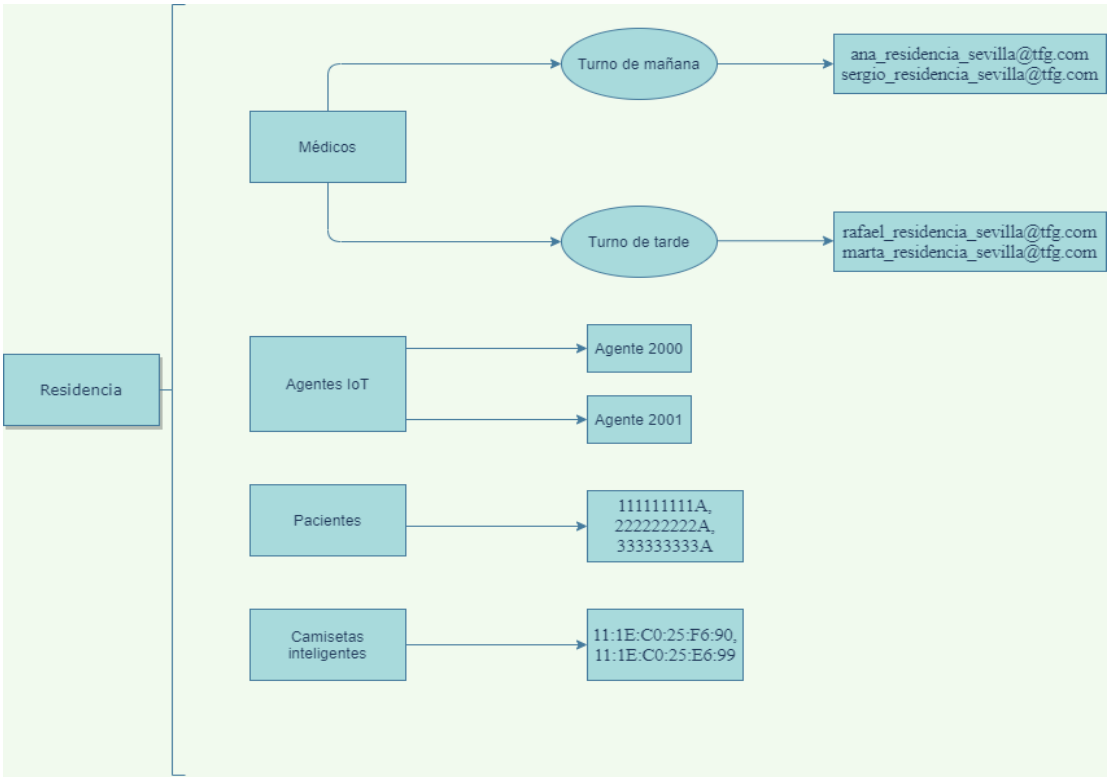

Figura 27. Esquema de los elementos pertenecientes a la Residencia.

Por último, se ha registrado un administrador que va a poder consultar la actividad física de cualquier paciente del sistema, y, además, podrá observar qué suscripciones hay activas.

3.3 Proceso de autenticación.

Antes de realizar las pruebas de validación del mecanismo de seguridad para la publicación de información, consulta y suscripción a entidades, se va a explicar cómo se lleva a cabo la autenticación en el sistema. La autenticación se realizará siempre de la misma manera y además es obligatoria en todos los escenarios, ya que la comunicación con Wilma requiere obligatoriamente de un token de acceso.

En primer lugar, para obtener el token de acceso proporcionado por Keyrock, el usuario o agente IoT debe

mandar un mensaje POST a este componente, un ejemplo de solicitud de un token:

```
salas@linux:~/escenario_sanitario$ curl -X POST \
>   http://localhost:3005/oauth2/token \
>   -H 'Accept: application/json' \
>   -H 'Authorization: Basic ZXNjZW5hcmlvX3Nhbml0YXJpbzplc2NlbmFyaW9fc2FuaXRhcmlvX3NlY3JldA==' \
>   -H 'Content-Type: application/x-www-form-urlencoded' \
>   -d 'username=agente_iot_hospital_central_1000@tfg.com&password=test&grant_type=password'
```

Figura 28. Solicitud de un token de acceso.

En esta petición se pueden observar las siguientes cabeceras:

- Authorization: Basic ZXN…: Esta cabecera indica el nivel de seguridad del sistema junto al valor de oauth_client id y oauth_client secret codificado en base 64.

  Oautch client id y secret se han preconfigurado en la puesta en marcha de Keyrock gracias al estado inicial de la base de datos MySQL, explicado en el Anexo B.

  Este valor codificado en base64 se obtiene al ejecutar el siguiente comando:

```
salas@linux:~/escenario_sanitario$ echo escenario_sanitario:escenario_sanitario_secret| base64
ZXNjZW5hcmlvX3Nhbml0YXJpbzplc2NlbmFyaW9fc2FuaXRhcmlvX3NlY3JldAo=
```

Figura 29. Comando para obtener en base 64 los valores oauth_client id y secret juntos.

- El cuerpo del mensaje se corresponde a username y password, donde se debe indicar el correo y contraseña del usuario que pretenda acceder al sistema. En este caso, el usuario había sido previamente registrado en Keyrock, explicado en el Anexo B.

  La respuesta nos mostrará el token asociado al usuario registrado. En caso de que no esté registrado, no proporcionará el token.


```
{"access_token":"c98a004623736d95afb5095c9febc16f8ea70503",
"token_type":"Bearer",
"expires_in":3599,
"refresh_token":"40423ac630c9e7e9cd8cbc08a62fbe0d64f7b843",
"scope":["bearer"]}
```


Figura 30. Token de acceso.

Una vez obtenido el token de acceso que permite ser autenticado en el sistema, el agente IoT o usuario podrá proceder al envío de la petición de publicación, consulta o suscripción a Wilma, indicando obligatoriamente en el mensaje la siguiente cabecera:

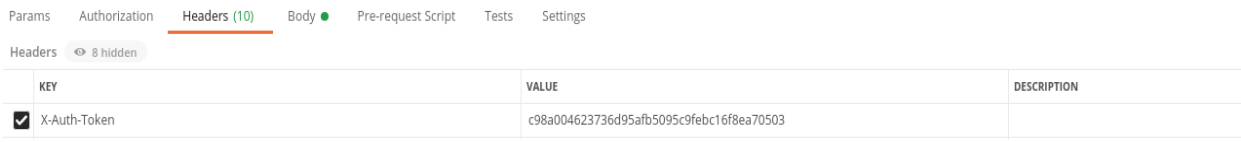

Figura 31. Cabecera X-Auth-Token en Postman.

Donde el valor de la cabecera X-Auth-token debe ser el token proporcionado por Keyrock. En caso de que no se indique dicha cabecera, cualquier petición dará la siguiente respuesta:

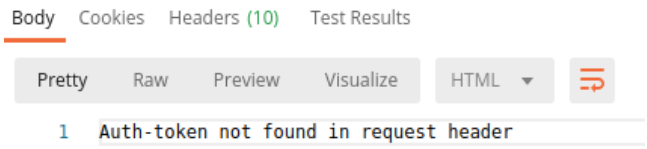

Figura 32. Respuesta al no indicar la cabecera Auth-token.

Cuando se envía una petición a Wilma, este elemento consultará a Keyrock si dicho usuario está registrado en el sistema mediante el token indicado en la cabecera X-Auth-Token. Si está registrado, Keyrock le mandará como respuesta a Wilma el id del usuario, el rol al que pertenece y aplicación en la que está trabajando. Por lo contrario, se mostrará el siguiente mensaje:

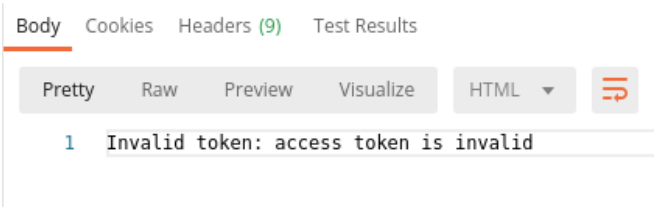

Figura 33. Respuesta al indicar un token inválido.

3.4 Proceso de autorización

El proceso de autorización comienza una vez que el usuario o agente IoT se haya autenticado correctamente en el sistema. Wilma obtiene tras la exitosa autenticación el id del usuario, el rol al que pertenece y aplicación en la que está trabajando gracias a Keyrock. Además, en caso de ser una publicación de una entidad por parte de un agente IoT o de la suscripción a una entidad en concreto por parte de un médico, Wilma obtiene los atributos indicados en el cuerpo de dichos mensajes.
Por consiguiente, Wilma manda a Authzforce una solicitud de autorización indicando sobre el propio mensaje los parámetros recibidos por Keyrock, y en ciertos casos, también los atributos obtenidos de la petición. Authzforce evalúa dichos datos recibidos mediante una serie de políticas activas para obtener una decisión, siendo esta enviada a Wilma con el fin de ser efectuada.

A continuación, se va an describir cada uno de los escenarios planteados.

3.5 Escenarios planteados.

3.5.1 Escenario 1: Autorización a los agentes IoT para la publicación.

En este primer escenario se desa asegurar qué agentes IoT no autorizados no puedan publicar información en Orion, y, además, que la información publicada sea la correcta. Algunos ejemplos de este escenario son:

- Un agente IoT perteneciente al Hospital Central no puede publicar información sobre la Residencia.
- Un agente IoT no puede publicar información correspondiente a un paciente o una camiseta con los que no esté asociado.
- Un agente IoT no puede publicar fuera de la franja horaria permitidad en la organización.

Centrándonos más en la autorización, Wilma envía la solicitud de autorización a Authzforce mediante la llamada de una función perteneciente al archivo, azf.js. Para la publicación de alguna entidad no es necesario modificar el fichero root.js con respecto al proporcionado por el repositorio. Pero el fichero azf.js sí será necesario modificarlo para poder obtener los diferentes atributos expuestos por los agentes IoT en el cuerpo del mensaje, concretamente en la función getRESTPolicy, mostrada en la Figura 34.
En primer lugar, debemos diferenciar el tipo de solicitud y recurso al que se está accediendo. Un agente IoT puede publicar una entidad nueva en Orion con una petición POST o actualizar algún campo de una entidad ya existente con una petición tipo PATCH. En primer lugar, analizaremos el caso de petición POST al recurso /v2/entities. En el apartado 3.5.1.1 se analizará el control sobre publicaciones tipo PATCH.

```
const getRESTPolicy = function (roles, action, resource, appId,req,peticionget,tabla) {

    log.info("Checking authorization to roles", roles, "to do ", action, " on ", resource, "and app ", appId);
    if((action == 'POST' && resource == '/v2/entities')||(action == 'PATCH')){
        var publisher = '';
        var organization = '';
        var id_patient = '';
        var id_device = '';
        if(action == 'PATCH'){
            organization = tabla[0];
            log.info(organization);
            publisher = tabla[1];
            log.info(publisher);
            id_patient = tabla[2];
            log.info(id_patient);
            id_device = tabla[3];
            log.info(id_device);

        }
        else{
            publisher = JSON.parse(req.body.toString('utf8'))['publisher'];
            organization = JSON.parse(req.body.toString('utf8'))['organization'];
            id_patient = JSON.parse(req.body.toString('utf8'))['id_patient'];
            id_device = JSON.parse(req.body.toString('utf8'))['id_device'];
```

Figura 34. Escenario de publicación. Función getRESTPolicy.

```
2020-08-30 12:53:35.036  - INFO: AZF-Client - <Buffer 20 20 20 20 20 7b 0a 20 20
 20 20 20 20 20 22 69 64 22 3a 22 75 72 6e 3a 6e 67 73 69 2d 6c 64 3a 73 65 6e 7
3 6f 72 3a 30 30 33 22 2c 22 74 79 70 65 22 ... 398 more bytes>
```

Figura 35. Valor de la variable req.body.

```
2020-08-30 12:53:35.038  - INFO: AZF-Client -      {
       "id":"urn:ngsi-ld:sensor:003","type":"ActividadFisica",
      "publisher": "Agente1001",
      "id_patient": "987654321A",
      "timestamp": {"type": "DateTime", "value":"2020-07-24T09:30:23.238Z"},
       "id_device" :"00:1E:C0:25:E6:99",
           "organization": "HospitalCentral",
           "accumulated_metabolic_expenditure" :{"type": "float", "value": "4.59
"},
           "instant_metabolic_expenditure":{"type": "float", "value": "0.07"}
    }
```

Figura 36. Valor de req.body.toString('utf8').

En el bloque else de la función getRESTPolicy obtenemos los diferentes atributos expuestos por el agente IoT, en caso de que no se encuentre alguno, el valor de la variable será *undefined*. Req.body devuelve todo el cuerpo del mensaje del Agente IoT en formato Buffer, un ejemplo sería el que se observa en la Figura 35. Por consiguiente, lo convertimos primero en String para que sea leíble. Por último, lo convertimos a JSON para que podamos encontrar específicamente el valor de un atributo en concreto, un ejemplo se puede observar en la Figura 36.

Los atributos que deben ser obtenidos son:

- Publisher: Indica qué Agente IoT pretende realizar la publicación.
- Id_patient: Identificador de un paciente.
- Id_device: Identificador de una camiseta inteligente.
- Organization: Organización a la que pertenece el Agente IoT.

Wilma procederá a desarrollar el mensaje de solicitud de autorización que será enviado a Authzforce.

```
const  XACMLPolicy = {
     "Request":{
        "xmlns":"urn:oasis:names:tc:xacml:3.0:core:schema:wd-17",
        "CombinedDecision": "false",
        "ReturnPolicyIdList":"false",
        "Attributes":[
            {
                "Category":"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject",
                "Attribute":[
                ]
            },
            {
                "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
                "Attribute":[
                    {
                        "AttributeId":"urn:oasis:names:tc:xacml:1.0:resource:resource-id",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": appId
                        }
                    },
                    {
                        "AttributeId":"urn:thales:xacml:2.0:resource:sub-resource-id",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": escapeXML(resource)
                        }
                    }
                ]
            },
            {
                "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:action",
                "Attribute":{
                    "AttributeId":"urn:oasis:names:tc:xacml:1.0:action:action-id",
                    "IncludeInResult": "false",
                    "AttributeValue":{
                        "DataType":"http://www.w3.org/2001/XMLSchema#string",
                        "$t": action
                    }
```

Figura 37. Mensaje de solicitud de autorización.

En la Figura 37 se indican algunos de los parámetros que van a ser enviados a Authzforce para que lleve a cabo la evaluación de la autorización. Concretamente los parámetros indicados serían el recurso al que se pretende acceder, la aplicación en la que se está trabajando y el tipo de acción que ha solicitado el usuario.

```
        }
    },
    {
        "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
         "Attribute":{
            "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:id_device",
            "IncludeInResult": "false",
            "AttributeValue":{
                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                "$t": id_device
            }
        }
    },
    {
        "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
         "Attribute":{
            "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:organization",
            "IncludeInResult": "false",
            "AttributeValue":{
                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                "$t": organization
            }
        }
    },
    {
        "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
         "Attribute":{
            "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:id_patient",
            "IncludeInResult": "false",
            "AttributeValue":{
                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                "$t": id_patient
            }
        }
    },
```

Figura 38. Mensaje de solicitud de autorización.

```
                {
                    "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
                     "Attribute":{
                        "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:publisher",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": publisher
                        }
                    }
                },
            ]
        }
    };

    // create Attribute only roles is not empty because XACML schema requires that an Attribute has at least one AttributeValue
if(roles.length > 0) {
       XACMLPolicy.Request.Attributes[0].Attribute[0] =
                        {
                            "AttributeId":"urn:oasis:names:tc:xacml:2.0:subject:role",
                            "IncludeInResult": "false",
                            "AttributeValue": [
                                // One per role
                                // {
                                // "DataType":"http://www.w3.org/2001/XMLSchema#string",
                                // "$t":"Manager"
                                // }
                            ]
                        };

      for (const i in roles) {
        XACMLPolicy.Request.Attributes[0].Attribute[0].AttributeValue[i] = {
            //"AttributeId":"urn:oasis:names:tc:xacml:2.0:subject:role",
            //"IncludeInResult": "false",
            //"AttributeValue":{
                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                "$t": roles[i]
            //}
        };
      }
}
```

Figura 39. Mensaje de solicitud de autorización.

En la Figura 38 se muestran los atributos imprescindibles enviados para asegurar el mecanismo de seguridad para la publicación del escenario sanitario que hemos planteado. El motivo de esto es porque se va a enviar los valores de los atributos id_device, organization, id_patient y Publisher para que sean evaluados en Authzforce.

Por último y no menos importante, en la Figura 39 se puede observar que también se enviará el rol que tiene el usuario en el sistema. Mediante este atributo podremos diferenciar en Authzforce los agentes IoT del Hospital Central y los pertenecientes a la Residencia.

Authzforce evaluará una a una las políticas almacenadas en el sistema. En caso de no cumplirse ninguna, denegará la autorización al usuario.

La política creada para el escenario de publicación se muestra en la Figura 40.

```
<Policy PolicyId="escenario_sanitario_publicacion" Version="1.0" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
   <Description>Politicas para el envio de informacion por parte de los agentes iot registrados</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">escenario_sanitario</AttributeValue>
               <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
               AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
               DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Rule RuleId="Reglas_Agente_IoT_Hospital_Central" Effect="Permit">
      <Description>Regla para agentes IoT Hospital Central</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">/v2/entities</AttributeValue>
                  <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                  AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
                  DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
               </Match>
            </AllOf>
         </AnyOf>
```

```
    <AnyOf>
       <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">POST</AttributeValue>
             <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
             DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
          </Match>
       </AllOf>
    </AnyOf>
 </Target>
```

Figura 40. Escenario publicación. Política.

La primera parte de la política va a ser igual para todos los escenarios, se comprueba si la aplicación que está trabajando el usuario es *escenario_sanitario*. Además, en esta política va a haber dos reglas, una para los agentes IoT del Hospital Central y otra regla para los agentes IoT de la Residencia. Vamos a detallar la regla de los agentes IoT del Hospital Central ya que para la Residencia será similar.

El tipo de acción y el recurso al que se pretende acceder van a ser igual para las dos reglas, POST y /v2/entities.

Ahora se van a comentar las condiciones que se han establecido en esta regla.

```
<Condition>
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
       </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:id_patient"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">123456789F</AttributeValue>
       </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:publisher"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Agente1000</AttributeValue>
       </Apply>
```

```
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:id_device"
    DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
 </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">00:1E:C0:25:E6:99</AttributeValue>
  </Apply>
  </Apply>
```

Figura 41. Escenario publicación. Política.

En la Figura 41 se describe una de las dos restricciones contenidas en una sentencia OR. La primera restricción trata sobre el Agente IoT 1000, donde se compara los diferentes atributos con ciertos valores concretos.

- Organization ha de ser obligatoriamente HospitalCentral.
- Id_patient debe ser 123456789F.
- Publisher debe ser Agente 1000.
- Id_device tendría que ser 00:1E:C0:25:E6:99.

Para que sea válido deben cumplirse las diferentes comparaciones descritas.

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
    DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
 </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">123456789F</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">987654321A</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">123456789A</AttributeValue>
  </Apply>
  <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
  AttributeId="urn:oasis:names:tc:xacml:1.0:environment:id_patient"
  DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
</Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:publisher"
    DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
 </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Agente1001</AttributeValue>
  </Apply>
```

```
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">00:1E:C0:25:E6:99</AttributeValue>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">00:1E:C0:25:F6:90</AttributeValue>
       </Apply>
       <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:id_device"
       DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
       </Apply>
       </Apply>
       </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Agente_IoT_Hospital_Central</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
      AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
      DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
   </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
         <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
      </Apply>
      </Apply>
</Condition>
```

Figura 42. Escenario publicación. Política.

La Figura 42 muestra la segunda restricción de la sentencia OR que ha de cumplirse. Trata sobre el Agente IoT 1001 que, como ya se indicó, es más complejo que el anterior, por este motivo va a haber más valores posibles por cada atributo.

- Organization ha de ser obligatoriamente HospitalCentral.
- Id_patient puede ser 123456789F, 987654321A o 123456789A.
- Publisher debe ser Agente 1001.
- Id_device puede ser 00:1E:C0:25:E6:99 o 00:1E:C0:25:F6:90.

Al igual que antes, debe cumplirse cada una de las comparaciones para que sea válida la sentencia.

```
        <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
             <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
             DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
          </Apply>
          </Apply>
    </Condition>
  </Rule>
```

Figura 43. Escenario publicación. Política.

Por último, en la Figura 43, se puede observa que al mismo nivel que la sentencia OR, hay otra sentencia AND. Dicha sentencia comprueba si el rol del usuario es *Agente_IoT_Hospital_Central*, y también, comprueba si la hora a la que se está intentando publicar está dentro de la franja horaria permitida, en este caso sería de 9:00:00 a 17:00:00.

Para la regla de la Residencia, el planteamiento va a ser igual, pero los valores a comparar serán distintos. Estos valores se corresponderán a los indicados al principio de este Apartado.

En resumen, se permitirá la publicación si la sentencia OR ha sido válida, es decir, si cumple los requisitos del Agente IoT 1000 o del Agente IoT 1001. Y, además, debe tener el rol correcto y que la hora de publicación sea la permitida.

En caso de no cumplir ninguna política, el usuario obtendrá la siguiente respuesta:

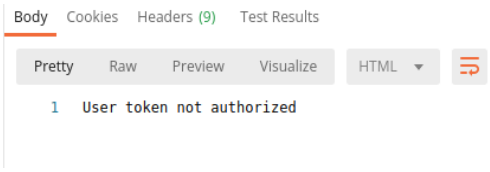

Figura 44. Respuesta de un usuario no autorizado.

Si Authzforce devuelve *Permit* a Wilma, es que el usuario está autorizado. Por tanto, Wilma redireccionaré el recurso a Orion mediante la llamada de la siguiente función.

```
proxy.sendData(protocol, options, req.body, res);
```

Figura 45. Función de redirección hacia Orion.

A continuación, se va a mostrar un esquema del funcionamiento de este escenario:

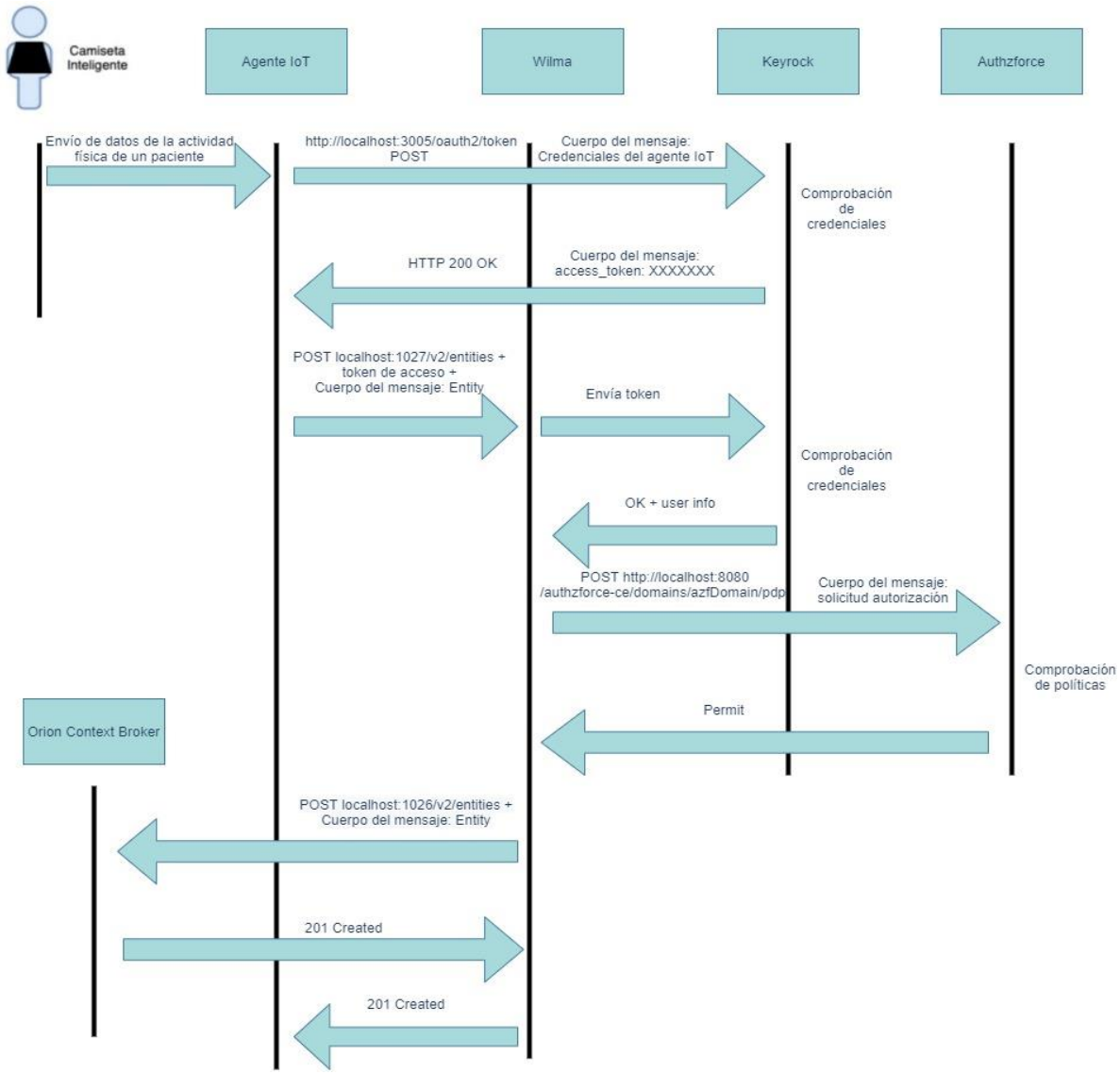

Figura 46. Esquema de funcionamiento del escenario 1.

Pruebas y resultados:

1. Agente IoT 1000 publica la actividad física del paciente 123456789F utilizando la camiseta 00:1E:C0:25:E6:99 a las 14:50. Se observará que se ha autorizado la publicación al Agente IoT 1000

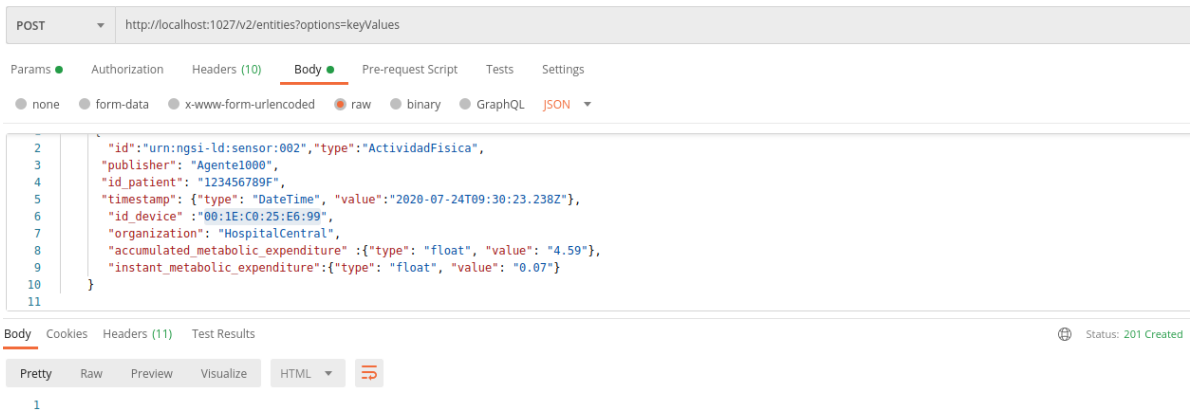

Figura 47. Escenario Publicación. Prueba 1.

2. Agente IoT 1000 publica la actividad física del paciente 987654321A utilizando la camiseta 00:1E:C0:25:E6:99 a las 14:54. Obtendremos que dicho Agente IoT no está autorizado.

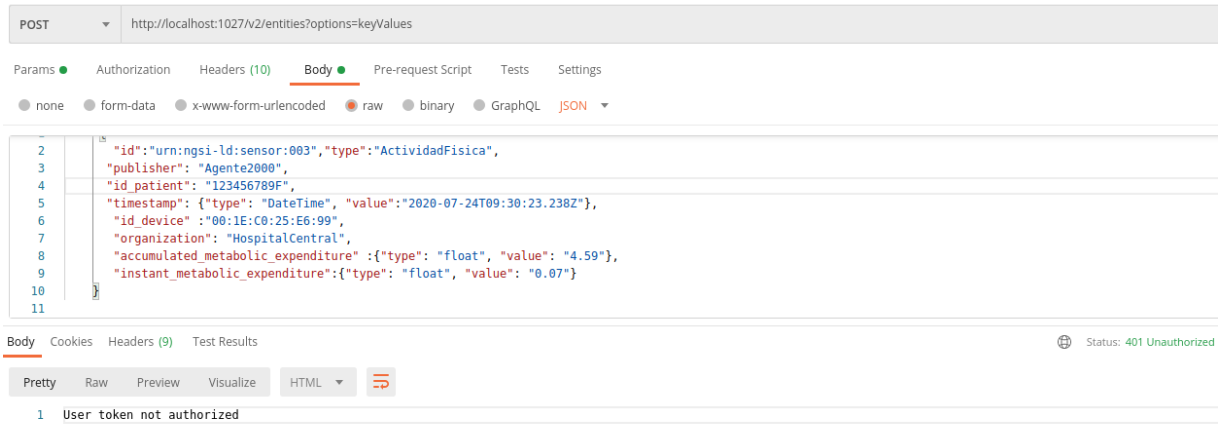

Figura 48. Escenario Publicación. Prueba 2.

3. Agente IoT 1000 publica la actividad física del paciente 123456789F utilizando la camiseta 00:1E:C0:25:E6:99 a las 15:05 indicando que el publisher es el Agente2000 ya sea por equivocación o por suplantación de identidad.

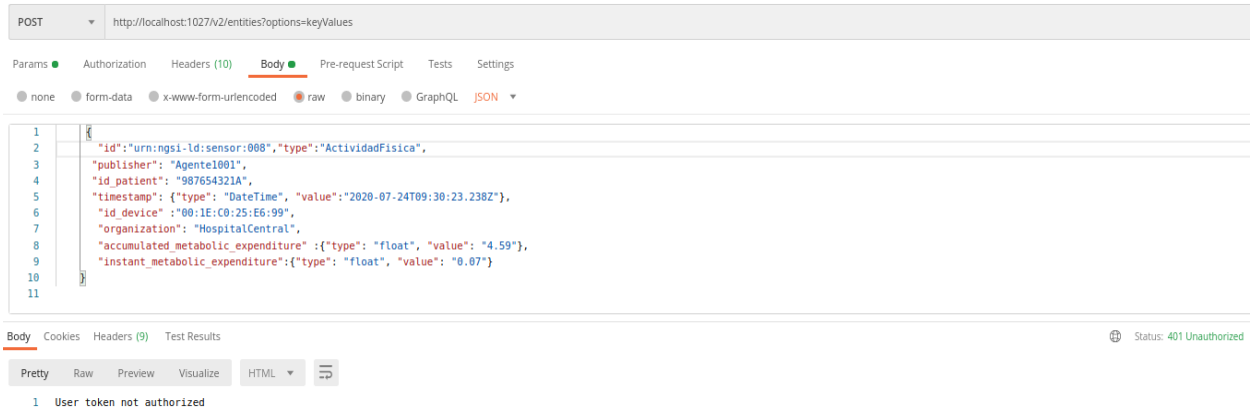

Figura 49. Escenario de publicación. Prueba 3.

4. Agente IoT 1001 publica la actividad física del paciente 987654321A utilizando la camiseta 00:1E:C0:25:F6:90 a las 18:00:00. No estará autorizado porque no se le está permitido la publicación a dicha hora.

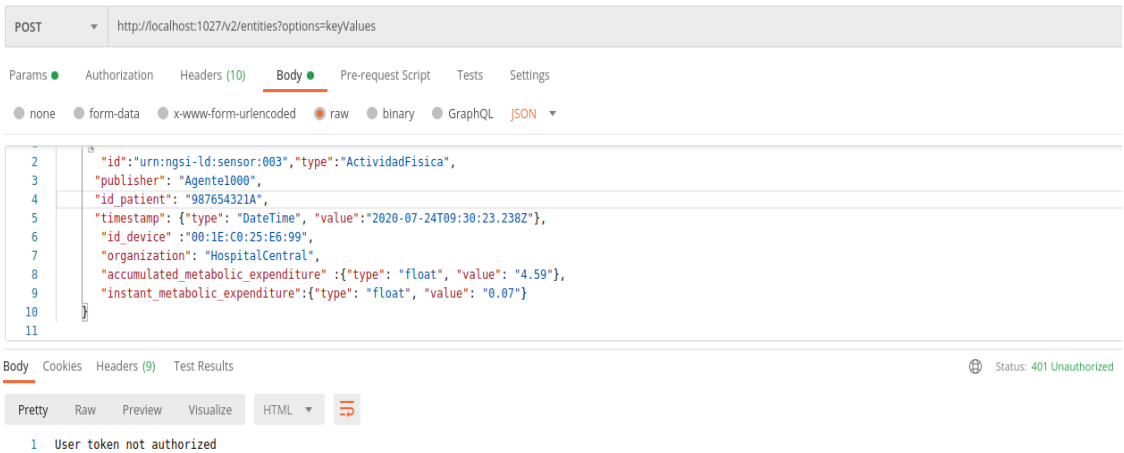

Figura 50. Escenario de publicación. Prueba 4.

5. Agente IoT 2001 publica la actividad física del paciente 111111111A utilizando la camiseta 11:1E:C0:25:E6:99 a las 18:00:00. Se observará que el agente tiene autorización para publicar ya que se le está permitido en dicha franja horaria y además está asociado a dicho paciente y camiseta.

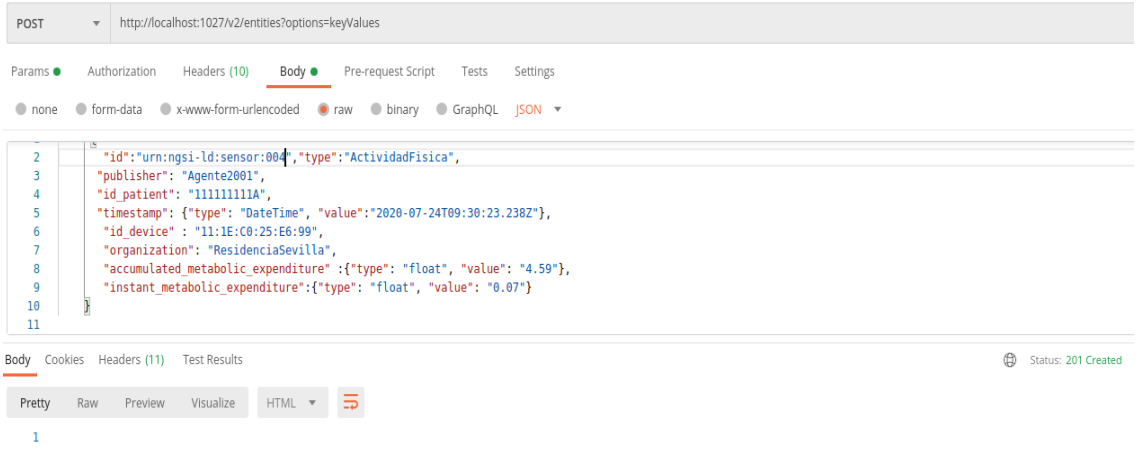

Figura 51. Escenario Publicación. Prueba 5.

6. Usuario correctamente autenticado pero que no pertenece ni al rol de *Agente_IoT_Hospital_Central* ni *Agente_IoT_Residencia_Sevilla* publica información.

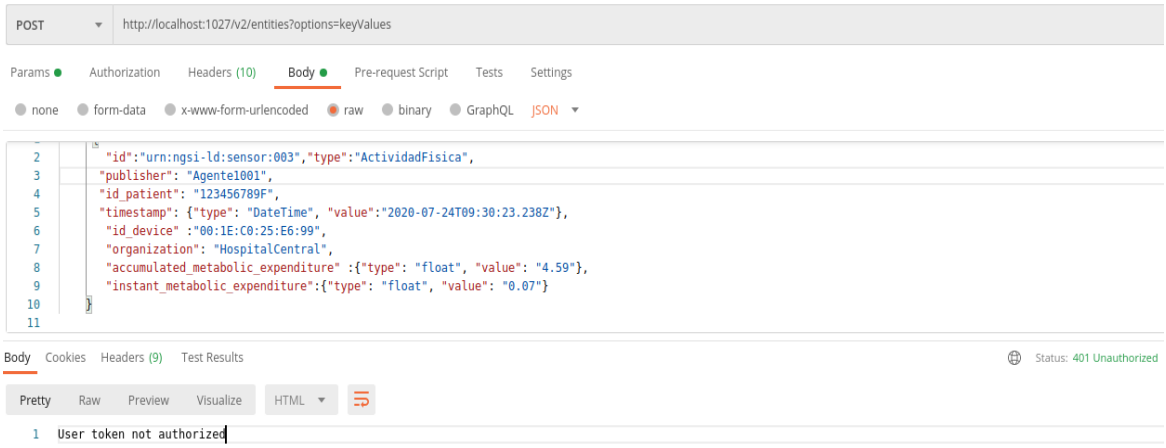

Figura 52. Escenario Publicación. Prueba 6.

3.5.1.1 Escenario 1.1: Autorización a los agentes IoT para la actualización de alguna entidad.

Ahora vamos a explicar un caso particular de publicación, concretamente, la actualización de algunos atributos de una entidad previamente creada.

A diferencia de la publicación de una nueva entidad, ahora sí vamos a modificar el archivo root.js ya que para saber si el agente IoT está autorizado a actualizar una entidad en concreto es necesario saber qué atributos tiene dicha entidad para luego evaluarlos en Authzforce. Para poder conseguir estos atributos a partir del identificador de la entidad debemos realizar una petición GET previa a Orion Context Broker.

```
const AZF = require('./../lib/azf.js').AZF;

  function sleep(ms) {
return new Promise(resolve => setTimeout(resolve, ms));
  }

  async function authorizeAzf(req, res, authToken, userInfo) {
      let array = "";
      var publisher = "";
      var id_patient = "";
      var id_device = "";
      const exec = require('child_process').exec;
      var organization= '';
      var entidades = '';
      var link = '';

      if((req.url != '/v2/entities' && (req.method == 'GET' || req.method == 'PATCH')) || (req.url == '/v2/subscriptions' && req.method == 'POST')){
          if(req.url != '/v2/entities' &&(req.method == 'GET' || req.method == 'PATCH')){

              link = 'curl -X GET  http://localhost:1026'+ req.url;
          }

          exec(link, (err, stdout, stderr) => {
              if (err) {
                  log.info(`exec error: ${err}`);
              }
              entidades = stdout;
              if( req.method == 'PATCH'){
                  organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
                  log.info(organization);
                  publisher =  JSON.parse(entidades.toString('utf8'))['publisher']['value'];
                  log.info(publisher);
                  id_patient =  JSON.parse(entidades.toString('utf8'))['id_patient']['value'];
                  log.info(id_patient);
                  id_device =  JSON.parse(entidades.toString('utf8'))['id_device']['value'];
                  log.info(id_device);

                  array = [organization,publisher,id_patient,id_device];
              }
              else{
                  organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
              }
          });}
      await sleep(1000);
      ,
AZF.checkPermissions(
  authToken,
  userInfo,
    req,organization,array,
  function() {
    redirRequest(req, res, userInfo);
  },
  function(status, e) {
    if (status === 401) {
        log.error('User access-token not authorized: ', e);
      res.status(401).send('User token not authorized');
    } else if (status === 404) {
      log.error('Domain not found: ', e);
      res.status(404).send(e);
    } else {
      log.error('Error in AZF communication ', e);
      res.status(503).send('Error in AZF communication');
    }
  },
  tokensCache
);
```

Figura 53. Escenario actualización de entidades. Función authorizeAzf y llamada a la función check Permissions.

Primero comprobamos si estamos ante una petición PATCH al recurso /v2/entities. En dicho caso, asignamos a una variable como cadena el comando a ejecutar, donde req.url será /v2/entities/entity-id/attrs. Mediante el comando exec y la variable asignada anteriormente, obtendremos la salida de Orion gracias a la variable stdout. Dicha salida se corresponderá a los diferentes atributos de la entidad especificada, en caso de que no existe dicha entidad, Orion devolverá [].

Para obtener cada atributo se realizará lo mismo que en el escenario de publicación, es decir, mediante JSON.parse(….toString('utf8'))[atributo_específico] incluyendo una última columna indicando ['value'].

Por último, almacenaremos cada atributo en una variable Array nombrada *tabla,* para luego pasar como argumento dicha variable a la función AZF.checkPermissions definida en el archivo azf.js.

Hay que destacar que la función exec tarda unos segundos en obtener respuesta, haciendo que la ejecución de

Wilma continúe aun sin obtener los atributos de la consulta a Orion. De este modo no se conseguía el objetivo que se planteaba ya que Authzforce no tenía en cuenta los atributos de dicha entidad para la evaluar la autorización. Para solventar este inconveniente se implementa una función sleep, para que la ejecución de Wilma se detenga hasta obtener los atributos de la consulta.

```
const checkPermissions = function(authToken, userInfo, req,peticionget,tabla, callback, callbackError, cache) {

    const roles = getRoles(userInfo);
    const appId = userInfo.app_id;
    const azfDomain = userInfo.app_azf_domain;


    let xml;
    const action = req.method;
    const resource = req.path;

    if (config.authorization.azf.custom_policy) {
        log.info('Checking auth with AZF...');
        xml = require('./../policies/' + config.azf.custom_policy).getPolicy(roles, req, appId);
    } else {
        if (cache[authToken] &&
            cache[authToken][action] &&
            cache[authToken][action].indexOf(resource) !== -1) {

            log.info('Permission in cache...');

            callback();
            return;
        }
        log.info('Checking auth with AZF...');
        xml = getRESTPolicy(roles, action, resource, appId,req,peticionget,tabla);
    }
```

Figura 54. Implementación de la función checkPermissions.

En la función checkPermissions pasaremos diferentes parámetros como argumentos a la función getRESTPolicy para que sean enviados en la solicitud de autorización hacia a Authzforce. Estos parámetros son:

- Roles
- Action
- Resource
- appId
- req
- peticionget
- tabla

Algunos parámetros solo tendrán utilidad dependiendo del escenario, por tanto, se detallarán en la explicación correspondiente.

```
const getRESTPolicy = function (roles, action, resource, appId,req,peticionget,tabla) {

    log.info("Checking authorization to roles", roles, "to do ", action, " on ", resource, "and app ", appId);
    if((action == 'POST' && resource == '/v2/entities')||(action == 'PATCH')){
        var publisher = '';
        var organization = '';
        var id_patient = '';
        var id_device = '';
        if(action == 'PATCH'){
            organization = tabla[0];
            log.info(organization);
            publisher = tabla[1];
            log.info(publisher);
            id_patient = tabla[2];
            log.info(id_patient);
            id_device = tabla[3];
            log.info(id_device);

        }
        else{
            publisher = JSON.parse(req.body.toString('utf8'))['publisher'];
            organization = JSON.parse(req.body.toString('utf8'))['organization'];
            id_patient = JSON.parse(req.body.toString('utf8'))['id_patient'];
            id_device = JSON.parse(req.body.toString('utf8'))['id_device'];
            log.info(req.body);
            log.info(req.body.toString('utf8'));
            log.info(JSON.parse(req.body.toString('utf8')));

            }
```

Figura 55. Implementación de la función getRESTPolicy.

En primer lugar, debemos diferenciar el tipo de solicitud y recurso al que se está accediendo, en este caso sería un PATCH al recurso /v2/entities.

Entraríamos en el if, donde asignamos los diferentes atributos almacenados en la variable Array *tabla* a diferentes variables.

Ahora se procede a desarrollar el mensaje de solicitud de autorización, que será exactamente igual que el indicado en la publicación de una entidad nueva.

La política creada para el escenario de actualización de entidad ya creada será muy similar al de publicación. Las condiciones serán las mismas, lo único que va a variar va a ser el tipo de acción para esta política, que ahora sería PATCH y no POST como antes.

```
<AnyOf>
   <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">PATCH</AttributeValue>
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
      </Match>
   </AllOf>
</AnyOf>
```

Figura 56. Parte de la politíca para la actualización de una entidad.

Pruebas y resultados:

1. Agente IoT 1000 actualiza los atributos *accumulated_metabolic_expenditure* e *instant_metabolic_expenditure* de la entidad *urn:ngsi-ld:sensor:002*. Dicha entidad ha sido creada por dicho agente, y, además, está intentando actualizarla dentro de la franja horaria permitida.

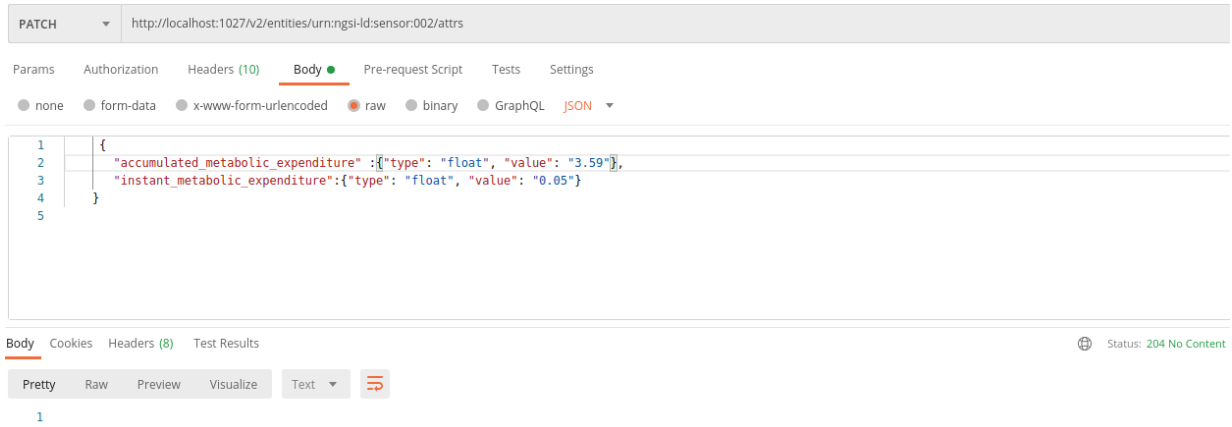

Figura 57. Escenario publicación 1.1. Prueba 1.

2. Agente IoT 1000 actualiza los atributos *accumulated_metabolic_expenditure* e *instant_metabolic_expenditure* de la entidad *urn:ngsi-ld:sensor:004,* cuya entidad pertenece a la actividad física de un paciente no asociado a dicho agente, tampoco está asociado a la camiseta ni organización.

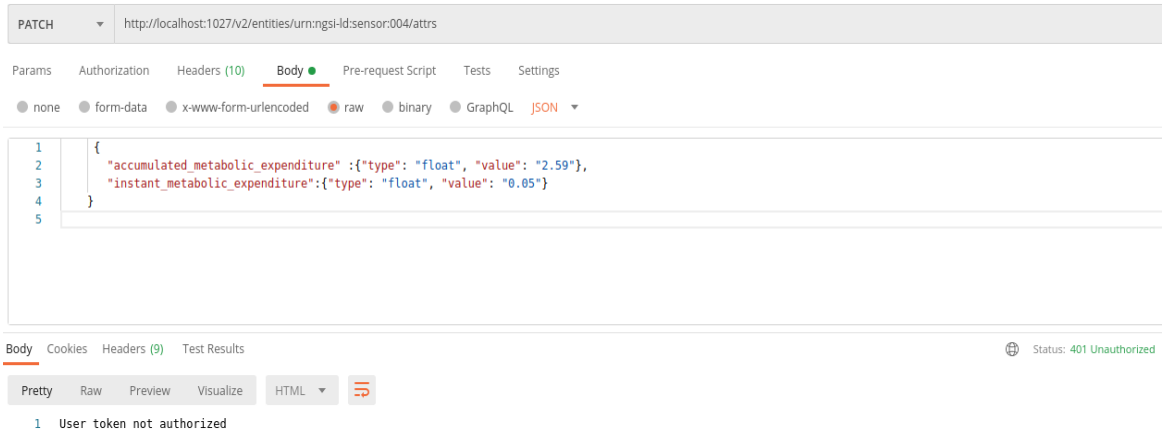

Figura 58. Escenario publicación 1.1. Prueba 2.

3. Un usuario correctamente autenticado, pero que no pertenece a ningún rol pretende actualizar los atributos *accumulated_metabolic_expenditure* e *instant_metabolic_expenditure* de la entidad *urn:ngsi-ld:sensor:004.*

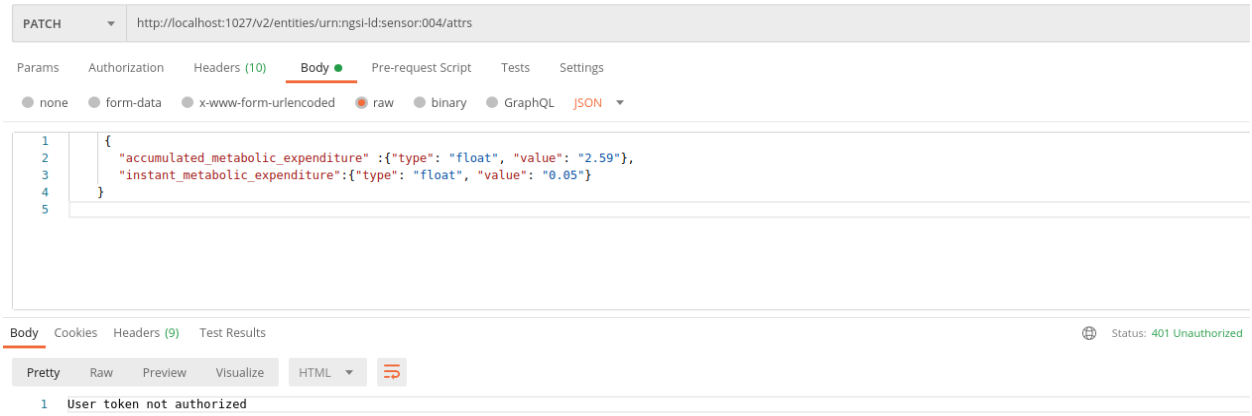

Figura 59. Escenario publicación 1.1. Prueba 3.

4. Agente IoT 1000 actualiza los atributos *accumulated_metabolic_expenditure* e *instant_metabolic_expenditure* de la entidad *urn:ngsi-ld:sensor:002*. Dicha entidad ha sido creada por dicho agente, pero está intentando actualizar fuera de la franja horaria permitida.

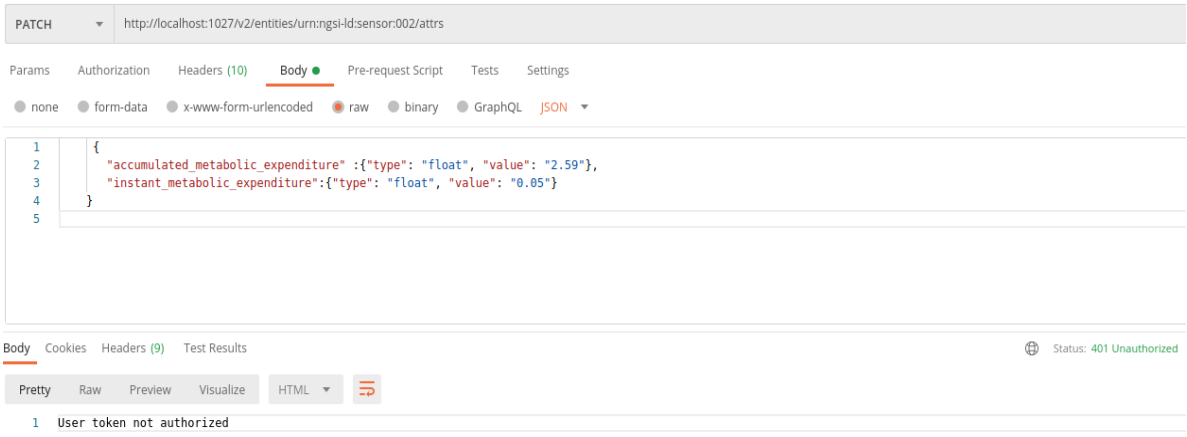

Figura 60. Escenario publicación 1.1. Prueba 4.

3.5.2 Escenario 2: Autorización a usuarios (médicos) para la consulta de la actividad física de los pacientes.

Se quiere asegurar que solo usuarios autorizados puedan consultar información confidencial de los pacientes. Algunos ejemplos de restricción en caso de realizar la operación de consulta:

- Los médicos del Hospital Central solo podrán realizar consultas sobre pacientes pertenecientes a dicha organización. Análogamente ocurrirá con aquellos médicos pertenecientes a la Residencia.
- Los médicos solo podrán realizar consulta en su turno de trabajo, en otro horario se les denegará.
- Si el médico realiza la consulta al recurso /v2/entities, es decir, de manera general y no a una entidad en concreto, obtendrá la actividad física de los pacientes pertenecientes a su organización.

```
async function authorizeAzf(req, res, authToken, userInfo) {
    let array = "";
    var publisher = "";
    var id_patient = "";
    var id_device = "";
    const exec = require('child_process').exec;
    var organization= '';
    var entidades = '';
    var link = '';

    if((req.url != '/v2/entities' && (req.method == 'GET' || req.method == 'PATCH')) || (req.url == '/v2/subscriptions' && req.method == 'POST')){
        if(req.url != '/v2/entities' &&(req.method == 'GET' || req.method == 'PATCH')){

            link = 'curl -X GET  http://localhost:1026'+ req.url;
        }

        exec(link, (err, stdout, stderr) => {
            if (err) {
                log.info(`exec error: ${err}`);
            }
            entidades = stdout;
            if( req.method == 'PATCH'){
                organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
                log.info(organization);
                publisher =  JSON.parse(entidades.toString('utf8'))['publisher']['value'];
                log.info(publisher);
                id_patient =  JSON.parse(entidades.toString('utf8'))['id_patient']['value'];
                log.info(id_patient);
                id_device =  JSON.parse(entidades.toString('utf8'))['id_device']['value'];
                log.info(id_device);

                array = [organization,publisher,id_patient,id_device];
            }
            else{
                organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
            }
        });}
    await sleep(1000);
    if((req.url == '/v2/entities' && req.method == 'GET')&&(userInfo.roles != '')){

        if(userInfo.roles[0].id == 'Medicos_Hosp_Central_turno_mañana' || userInfo.roles[0].id == 'Medicos_Hosp_Central_turno_tarde'){
            organization = 'HospitalCentral';
        }
        if(userInfo.roles[0].id == 'Medicos_Res_Sevilla_turno_tarde' || userInfo.roles[0].id == 'Medicos_Res_Sevilla_turno_mañana'){
            organization = 'ResidenciaSevilla';
        }
    }
```

Figura 61. Escenario consulta. Implementación authorizeAzf.

Una vez autenticado el médico, si el recurso que pretende acceder es una entidad en concreto, es decir, distinto de /v2/entities, se debe realizar una consulta previa a Orion para obtener el atributo *organization* de dicha

entidad para que sea traspasado y evaluado en Authzforce.

Si el recurso fuera precisamente /v2/entities, no haría falta realizar esta consulta previa hacia Orion. Pero para que el mensaje de solicitud hacia Authzforce sea igual tanto para una entidad concreta como para todas las entidades, se ha incluido una serie de condiciones.

Si el recurso al que se pretende acceder es /v2/entities y el rol del usuario solicitante es uno de los siguientes*:*

- *Medicos_Hosp_Central_turno_mañana.*
- *Medicos_Hosp_Central_turno_tarde.*
- *Medicos_Res_Sevilla_turno_mañana.*
- *Medicos_Res_Sevilla_turno_tarde.*

se asigna el valor de la variable organization como HospitalCentral o ResidenciaSevilla.

En ambos recursos, se envía la variable *organization* como argumento de la función checkPermissions para que posteriormente sea enviada como argumento de la función getRESTPolicy. En la función checkPermissions se ha nombrado la variable que caracteriza a la organization como *peticionget.*

```
const getRESTPolicy = function (roles, action, resource, appId,req,peticionget,tabla) {

    log.info("Checking authorization to roles", roles, "to do ", action, " on ", resource, "and app ", appId);
    if((action == 'POST' && resource == '/v2/entities')||(action == 'PATCH')){
    }
    else{
        var url = '';
        if( action == 'POST'){
             url = JSON.parse(req.body.toString('utf8'))['notification']['http']['url'];
        }

        const  XACMLPolicy = {
        "Request":{
            "xmlns":"urn:oasis:names:tc:xacml:3.0:core:schema:wd-17",
            "CombinedDecision": "false",
            "ReturnPolicyIdList":"false",
            "Attributes":[
                {
                    "Category":"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject",
                    "Attribute":[
                    ]
                },
                {
                    "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
                    "Attribute":[
                        {
                            "AttributeId":"urn:oasis:names:tc:xacml:1.0:resource:resource-id",
                            "IncludeInResult": "false",
                            "AttributeValue":{
                                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                                "$t": appId
                            }
                        },
                        {
                            "AttributeId":"urn:thales:xacml:2.0:resource:sub-resource-id",
                            "IncludeInResult": "false",
                            "AttributeValue":{
                                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                                "$t": escapeXML(resource)
                            }
                        }
                    ]
```

Figura 62. Escenario de consulta. Implementación de getRESTPolicy.

En la Figura 62, se puede ver parte de la función getRESTPolicy perteneciente al archivo azf.js. Tal y como se ha nombrado antes, la organización perteneciente a la entidad que pretende acceder el usuario se obtiene mediante el parámetro peticionget de la función getRESTPolicy.

En este caso, se entraría en el else, ya que action va a ser igual a GET al estar en el escenario de consulta. La variable url no será necesaria para este escenario, será utilizada para la suscripción. Se indican algunos de los parámetros que van a ser enviados a Authzforce, ya que a partir de los cuales este llevará a cabo la evaluación

de la autorización. En esta figura, los parámetros indicados serían la aplicación en la que se está trabajando y el recurso al que pretende acceder.

```
                },
                {
                    "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:action",
                    "Attribute":{
                        "AttributeId":"urn:oasis:names:tc:xacml:1.0:action:action-id",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": action
                        }
                    }
                },
                 {
                    "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
                     "Attribute":{
                        "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:organization",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": peticionget
                        }
                    }
                 },
                  {
                    "Category":"urn:oasis:names:tc:xacml:3.0:attribute-category:environment",
                     "Attribute":{
                        "AttributeId":"urn:oasis:names:tc:xacml:1.0:environment:url",
                        "IncludeInResult": "false",
                        "AttributeValue":{
                            "DataType":"http://www.w3.org/2001/XMLSchema#string",
                            "$t": url
                        }
                    }
                }
            ]
        }
    };
```

Figura 63. Escenario de consulta. Implementación de getRESTPolicy.

En la Figura 63 podemos observar que también se enviará el tipo de acción que realiza el usuario, la organización a la que pertenece y la url, que en este caso se enviará vacía.

```
    // create Attribute only roles is not empty because XACML schema requires that an Attribute has at least one AttributeValue
if(roles.length > 0) {
       XACMLPolicy.Request.Attributes[0].Attribute[0] =
                        {
                           "AttributeId":"urn:oasis:names:tc:xacml:2.0:subject:role",
                           "IncludeInResult": "false",
                           "AttributeValue": [
                               // One per role
                               // {
                               // "DataType":"http://www.w3.org/2001/XMLSchema#string",
                               // "$t":"Manager"
                               // }
                           ]
                        };

      for (const i in roles) {
        XACMLPolicy.Request.Attributes[0].Attribute[0].AttributeValue[i] = {
            //"AttributeId":"urn:oasis:names:tc:xacml:2.0:subject:role",
            //"IncludeInResult": "false",
            //"AttributeValue":{
                "DataType":"http://www.w3.org/2001/XMLSchema#string",
                "$t": roles[i]
            //}
        };
      }
}
```

Figura 64. Escenario de consulta. Implementación de getRESTPolicy.

Por último, se mandará también el rol al que pertenece el usuario. Mediante este atributo podremos diferenciar en Authzforce los médicos de turno de mañana y tarde del Hospital Central, y también los médicos de turno de mañana y tarde de la Residencia.

Una vez enviado el mensaje de solicitud de autorización, Authzforce lo evaluará con sus respectivas políticas. Para este escenario se ha implementado la política de la Figura 65.

```
<Policy PolicyId="escenario_sanitario_consulta_informacion" Version="1.0"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
<Description>Politicas para la consulta de información por parte de los medicos</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">escenario_sanitario</AttributeValue>
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
         </Match>
      </AllOf>
   </AnyOf>
</Target>
<Rule RuleId="Reglas_Medicos_Hospital_Central" Effect="Permit">
   <Description>Reglas medicos del Hospital Central</Description>
   <Target>
     <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:3.0:function:string-starts-with">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/v2/entities</AttributeValue>
               <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
               AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
               DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
            </Match>
         </AllOf>
      </AnyOf>
```

```
   <AnyOf>
      <AllOf>
         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">GET</AttributeValue>
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
         </Match>
      </AllOf>
   </AnyOf>
</Target>
```

Figura 65. Escenario de consulta. Política.

En la primera parte de la política se comprueba si la aplicación con la que está trabajando el usuario es *escenario_sanitario.* Además, en esta política va a haber dos reglas, una para los médicos del Hospital Central y otra regla para los médicos de la Residencia. Vamos a detallar la regla de los médicos del Hospital Central ya que para la Residencia será igual, pero variando los valores que van a ser comparados con los atributos *organization* y *role*. Además, también se variará la franja horaria en función la organización y el turno a los que pertenece el usuario.

El tipo de acción que se va a trabajar en esta política será GET, pero para el recurso se ha optado por una función denominada string-starts-with explicada en el Apartado 2, que permite que el recurso sea todo aquel que empiece por /v2/entities. Mediante esto conseguimos que los médicos puedan acceder tanto a /v2/entities como /v2/entities/entity-id excepto si incumple las siguientes condiciones de la política.

```
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Medicos_Hosp_Central_turno_mañana</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
      AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
      DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
         <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">15:00:00</AttributeValue>
      </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
       </Apply>
      </Apply>
```

```
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
         <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Medicos_Hosp_Central_turno_tarde</AttributeValue>
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
         AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
            <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
            DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
            </Apply>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">15:00:00</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">21:00:00</AttributeValue>
         </Apply>
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
         </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
          </Apply>
         </Apply>
         </Apply>
   </Condition>
</Rule>
```

Figura 66. Escenario de consulta. Política.

En esta regla va a haber una sentencia OR donde se ha de cumplir que el médico tenga rol *Medicos_Hosp_Central_turno_mañana*, que el valor de organization sea HospitalCentral y además que el horario de consulta esté en el periodo de 9:00:00 a 15:00:00.

Alternativamente, puede darse que el médico tenga rol *Medicos_Hosp_Central_turno_tarde*, que el valor de organization sea HospitalCentral y que además el horario de consulta esté entre las 15:00:00 y las 21:00:00, en otro caso dicha regla dará como resultado la negación de la autorización. Para la Residencia se ha creado otra regla dentro de política con un contenido similar a la comentada.

Por último, tras la correcta autorización por parte de Authzforce, Wilma va a redireccionar a Orion la petición, pero para este escenario se ha modificado el fichero root.js para la propia redirección.

```
const redirRequest = function(req, res, userInfo) {
    if (userInfo) {
        log.info('Access-token OK. Redirecting to app...');
    } else {
        log.info('Public path. Redirecting to app...');
    }

    const protocol = config.app.ssl ? 'https' : 'http';
    if( req.url == '/v2/entities' && (userInfo.roles[0].id == 'Medicos_Hosp_Central_turno_mañana' ||
                                      userInfo.roles[0].id == 'Medicos_Hosp_Central_turno_tarde') && req.method == 'GET'){
        req.url = '/v2/entities?q=organization:HospitalCentral';

    }
    if( req.url == '/v2/entities' && (userInfo.roles[0].id == 'Medicos_Res_Sevilla_turno_tarde' ||
                                      userInfo.roles[0].id == 'Medicos_Res_Sevilla_turno_mañana') && req.method == 'GET'){
        req.url = '/v2/entities?q=organization:ResidenciaSevilla';
    }
```

Figura 67. Implementación de la función redirRequest.

La modificación consiste en añadir una serie de condiciones para modificar la redirección a Orion dependiendo del recurso que haya indicado el médico al realizar la consulta.

Si el recurso es /v2/entities se debe restringir las entidades mostradas por Orion ya que no sería de gran seguridad mostrar todas las entidades registradas. Ante esto, si el rol del usuario pertenece al HospitalCentral, solo obtendrán aquellas entidades donde el valor organization valga HospitalCentral, lo mismo ocurriría con los médicos de la Residencia.

El esquema del funcionamiento de este escenario se muestra en la Figura 68.

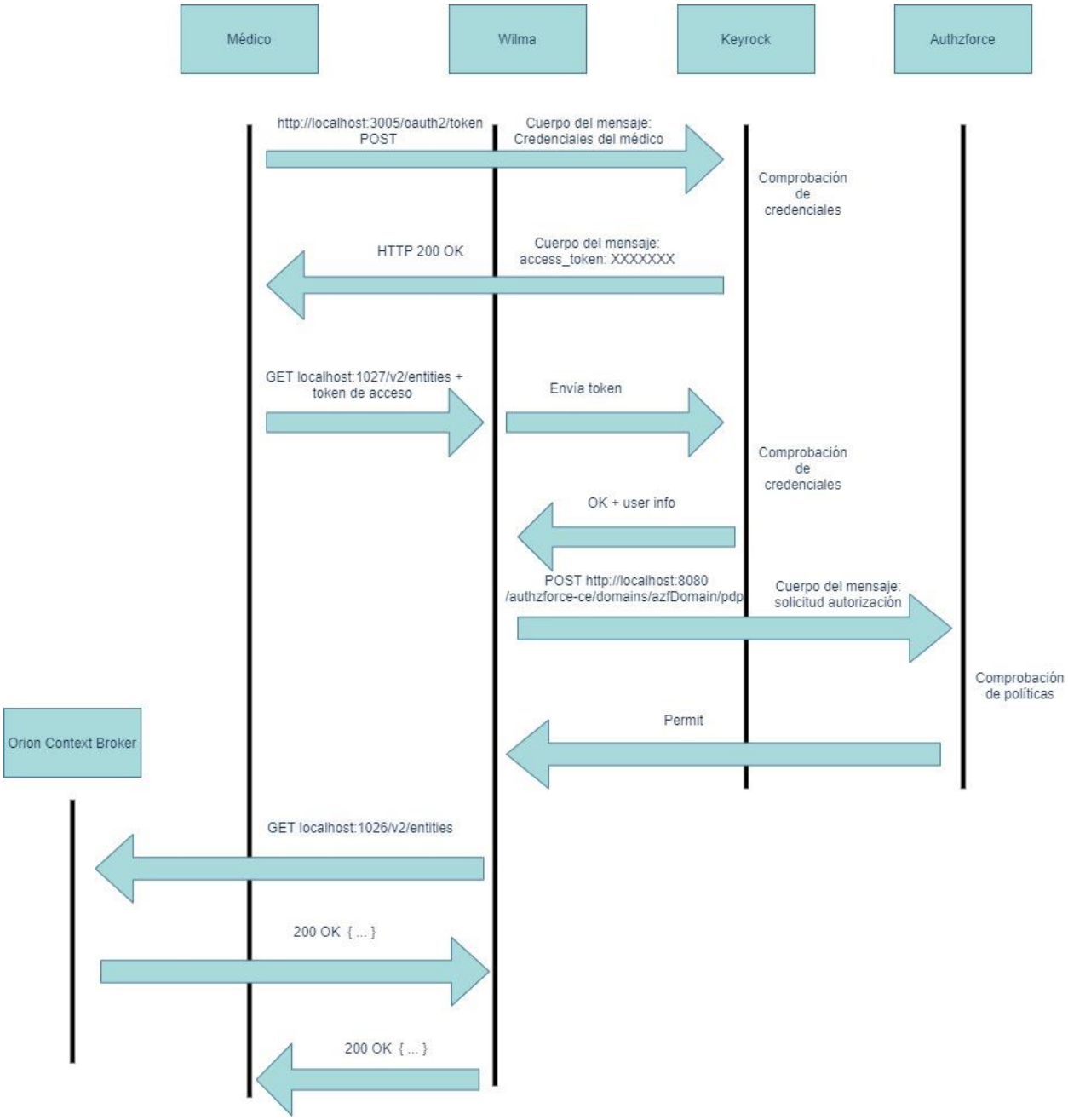

Figura 68. Esquema de funcionamiento del escenario 2.

Pruebas y resultados:

Requisito: En las operaciones GET/DELETE se debe eliminar la cabecera Content-Type, sino obtendremos el siguiente error:

```
{"error":"BadRequest","description":"Orion accepts no payload for GET/DELETE requests. HTTP header Content-Type is thus forbidden"}
```

Figura 69. Error debido a no eliminar la cabecera Content-Type en una consulta.

1. Médico del Hospital Central perteneciente al turno de mañana intenta consultar una entidad perteneciente a dicha organización a las 13:00:00. El médico obtendrá la entidad al cumplir que la entidad tiene como valor de organization HospitalCentral y en una hora permitida para dicho médico.

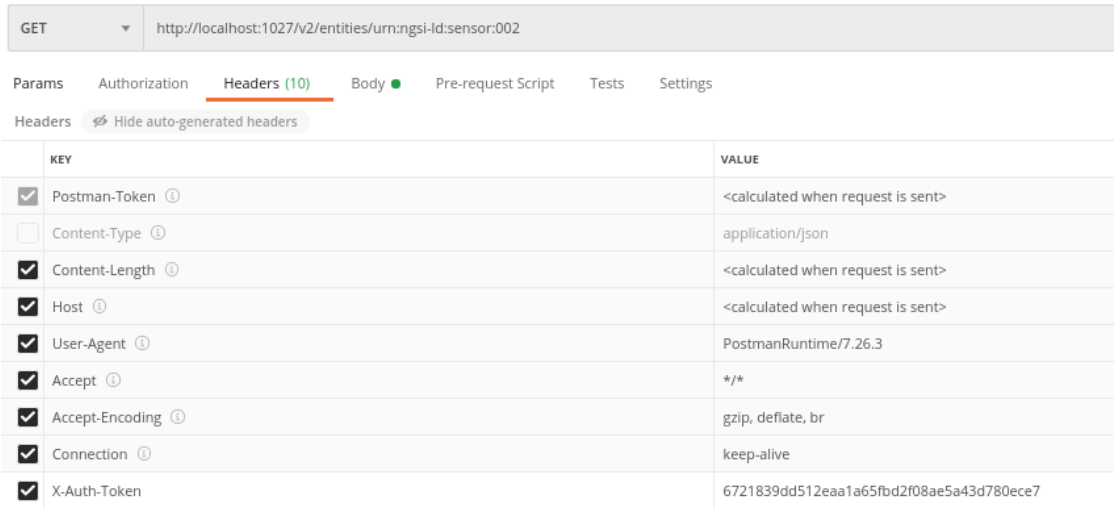

Figura 70. Escenario de consulta. Prueba 1. Petición.


```
[
    {
        "id": "urn:ngsi-ld:sensor:002",
        "type": "ActividadFisica",
        "accumulated_metabolic_expenditure": {
            "type": "StructuredValue",
            "value": {
                "type": "float",
                "value": "4.59"
            },
            "metadata": {}
        },
        "id_device": {
            "type": "Text",
            "value": "00:1E:C0:25:E6:99",
            "metadata": {}
        },
        "id_patient": {
            "type": "Text",
            "value": "123456789F",
            "metadata": {}
        },
        "instant_metabolic_expenditure": {
            "type": "StructuredValue",
            "value": {
                "type": "float",
                "value": "0.07"
            },
            "metadata": {}
        },
        "organization": {
            "type": "Text",
            "value": "HospitalCentral",
            "metadata": {}
        },
        "publisher": {
            "type": "Text",
            "value": "Agente1000",
            "metadata": {}
        },
        "timestamp": {
            "type": "StructuredValue",
            "value": {
                "type": "DateTime",
                "value": "2020-07-24T09:30:23.238Z"
            },
            "metadata": {}
        }
    }
]
```


Figura 71. Escenario de consulta. Prueba 1. Respuesta.

2. Médico del Hospital Central perteneciente al turno de tarde intenta consultar una entidad perteneciente a dicha organización a las 13:05:00. No tendrá acceso al no estar en la franja horaria permitida.

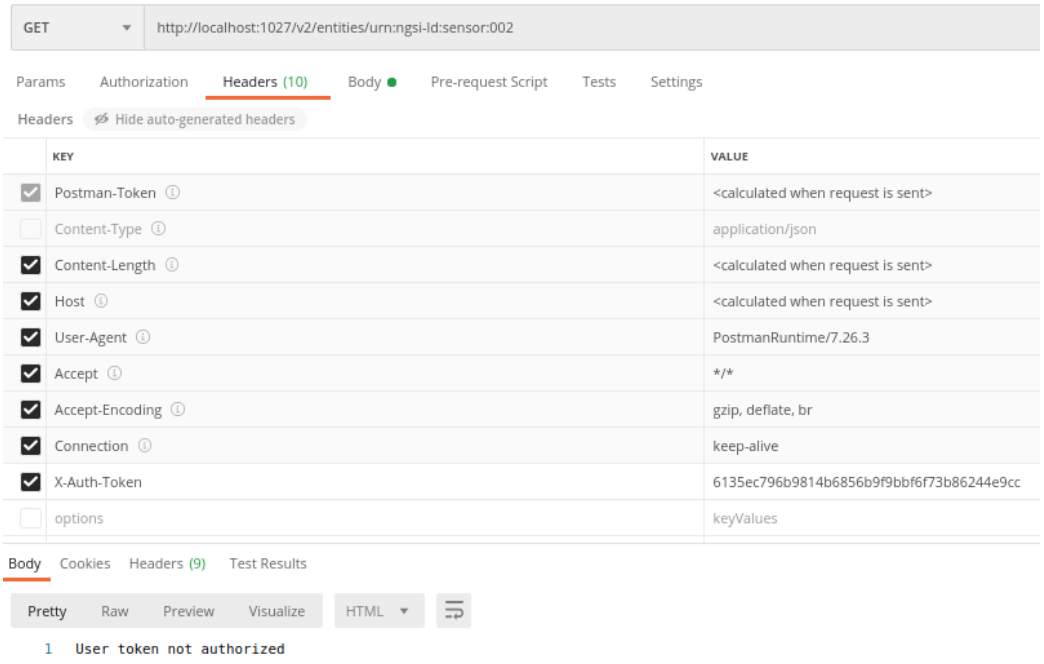

Figura 72. Escenario de consulta. Prueba 2.

3. Médico de la Residencia de Sevilla realiza una consulta de todas las entidades (/v2/entities) a las 13:10:00. Se observará que solo obtiene las entidades con el atributo *organization* con valor ResidenciaSevilla.

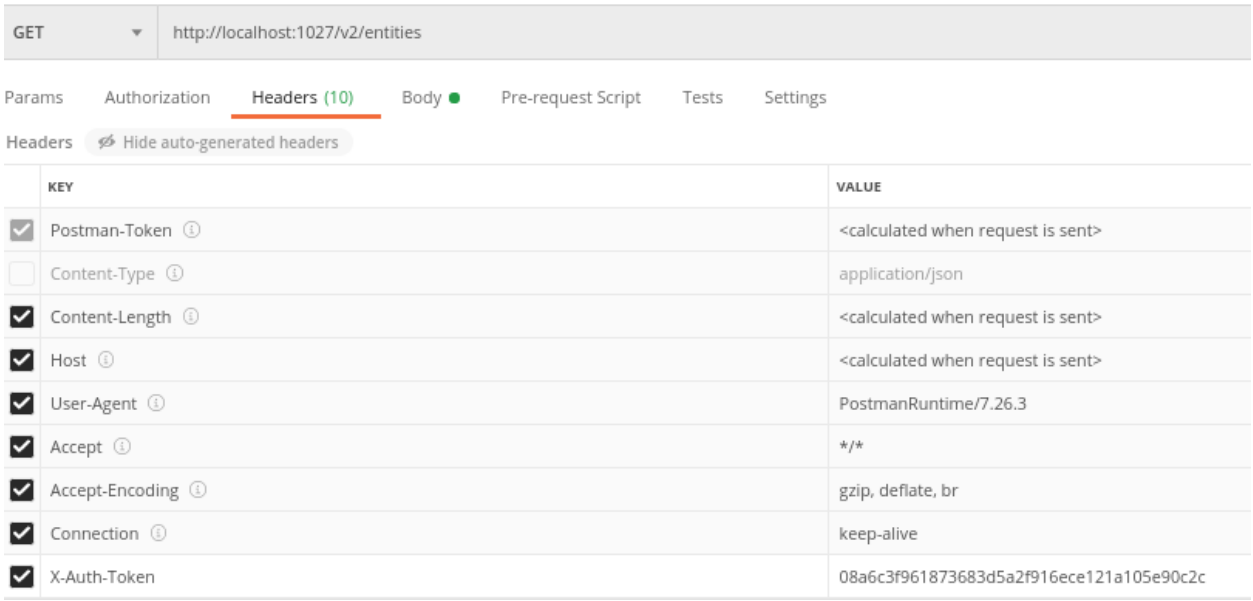

Figura 73. Escenario de consulta. Prueba 3. Petición.

```
[
    {
        "id": "urn:ngsi-ld:sensor:005",
        "type": "ActividadFisica",
        "accumulated_metabolic_expenditure": {
            "type": "StructuredValue",
            "value": {
                "type": "float",
                "value": "5.59"
            },
            "metadata": {}
        },
        "id_device": {
            "type": "Text",
            "value": "11:1E:C0:25:E6:99",
            "metadata": {}
        },
        "id_patient": {
            "type": "Text",
            "value": "222222222A",
            "metadata": {}
        },
        "instant_metabolic_expenditure": {
            "type": "StructuredValue",
            "value": {
                "type": "float",
                "value": "0.05"
            },
            "metadata": {}
        },
        "organization": {
            "type": "Text",
            "value": "ResidenciaSevilla",
            "metadata": {}
        },
        "publisher": {
            "type": "Text",
            "value": "Agente2001",
            "metadata": {}
        },
        "timestamp": {
            "type": "StructuredValue",
            "value": {
                "type": "DateTime",
                "value": "2020-07-24T09:30:23.238Z"
            },
            "metadata": {}
        }
    },
    {
        "id": "urn:ngsi-ld:sensor:004",
        "type": "ActividadFisica",
        "accumulated_metabolic_expenditure": {
            "type": "StructuredValue",
            "value": {
                "type": "float",
                "value": "5.59"
```

```
        },
        "metadata": {}
    },
    "id_device": {
        "type": "Text",
        "value": "11:1E:C0:25:E6:99",
        "metadata": {}
    },
    "id_patient": {
        "type": "Text",
        "value": "111111111A",
        "metadata": {}
    },
    "instant_metabolic_expenditure": {
        "type": "StructuredValue",
        "value": {
            "type": "float",
            "value": "0.05"
        },
        "metadata": {}
    },
    "organization": {
        "type": "Text",
        "value": "ResidenciaSevilla",
        "metadata": {}
    },
    "publisher": {
        "type": "Text",
        "value": "Agente2001",
        "metadata": {}
    },
    "timestamp": {
        "type": "StructuredValue",
        "value": {
            "type": "DateTime",
            "value": "2020-07-24T09:30:23.238Z"
        },
        "metadata": {}
    }
}
]
```


Figura 74. Escenario de consulta. Prueba 3. Respuesta.

4. Médico del Hospital Central realiza una consulta de una entidad perteneciente a la Residencia de Sevilla en su debido horario. Se observará que no estará autorizado ya que no puede consultar nada a no ser que sea de su organización.

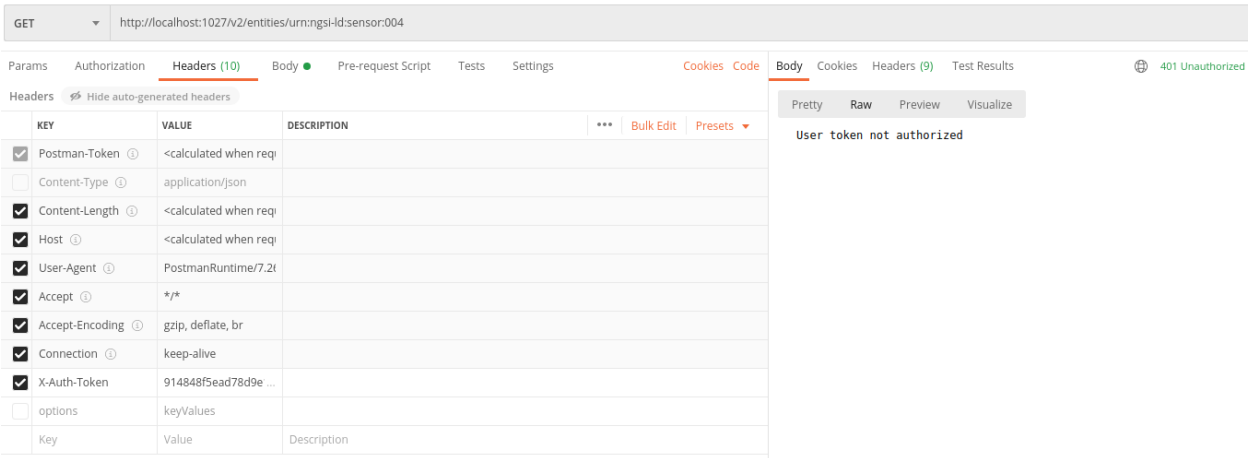

Figura 75. Escenario de consulta. Prueba 4.

3.5.3 Escenario 3: Autorización a los usuarios (médicos) para la suscripción a entidades.

Se quiere asegurar que solo usuarios autorizados se pueden suscribir para recibir notificaciones de alguna modificación de una entidad. Se debe cumplir lo siguiente:

- El médico debe pertenecer a la misma organización que la entidad suscrita.
- La dirección IP indicada para recibir las notificaciones debe estar registrada para dicha organización.

```
async function authorizeAzf(req, res, authToken, userInfo) {
    let array = "";
    var publisher = "";
    var id_patient = "";
    var id_device = "";
    const exec = require('child_process').exec;
    var organization= '';
    var entidades = '';
    var link = '';

    if((req.url != '/v2/entities' && (req.method == 'GET' || req.method == 'PATCH')) || (req.url == '/v2/subscriptions' && req.method == 'POST')){
        if(req.url != '/v2/entities' &&(req.method == 'GET' || req.method == 'PATCH')){

            link = 'curl -X GET  http://localhost:1026'+ req.url;
        }
        else{
            const id = JSON.parse(req.body.toString('utf8'))['subject']['entities'][0]['id'];
            link = 'curl -X GET  http://localhost:1026/v2/entities/'+id+'&options=keyValues';
        }
        exec(link, (err, stdout, stderr) => {
            if (err) {
                log.info(`exec error: ${err}`);
            }
            entidades = stdout;
            if( req.method == 'PATCH'){
                organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
                log.info(organization);
                publisher =  JSON.parse(entidades.toString('utf8'))['publisher']['value'];
                log.info(publisher);
                id_patient =  JSON.parse(entidades.toString('utf8'))['id_patient']['value'];
                log.info(id_patient);
                id_device =  JSON.parse(entidades.toString('utf8'))['id_device']['value'];
                log.info(id_device);

                array = [organization,publisher,id_patient,id_device];
            }
            else{
                organization = JSON.parse(entidades.toString('utf8'))['organization']['value'];
            }
        });}
    await sleep(1000);
```

Figura 76. Escenario de suscripción. Implementación authorizeAzf.

Una vez autenticado el médico, si el recurso que pretende acceder es /v2/subscriptions y la acción es POST, se obtiene del cuerpo del mensaje el id de la entidad a la que se quiere suscribir para luego realizar una consulta a Orion y obtener el valor del atributo *organization* de dicha entidad.

Se envía la variable *organization* como argumento de la función checkPermissions para que posteriormente sea enviada como argumento de la función getRESTPolicy. En la función checkPermissions se ha nombrado la variable que caracteriza a la organization como *peticionget*, es decir, de la misma manera que el escenario de consulta.

```
const getRESTPolicy = function (roles, action, resource, appId,req,peticionget,tabla) {

    log.info("Checking authorization to roles", roles, "to do ", action, " on ", resource, "and app ", appId);
    if((action == 'POST' && resource == '/v2/entities')||(action == 'PATCH')){
    }
    else{
        var url = '';
        if( action == 'POST'){
            url = JSON.parse(req.body.toString('utf8'))['notification']['http']['url'];
        }
```

Figura 77. Escenario de suscripción. Implementación authorizeAzf.

En la Figura 77, se puede ver parte de la función getRESTPolicy perteneciente al archivo azf.js. Tal y como se ha nombrado antes, la organización perteneciente a la entidad que pretende acceder el usuario se obtiene mediante el parámetro *peticionget* de la función getRESTPolicy.

En este caso, se entraría en el else, ya que action va a ser igual a GET al estar en el escenario de consulta. La variable *url* tendrá como valor la dirección donde quiere recibir las notificaciones el usuario. Dicha dirección se obtiene del cuerpo del mensaje.

El mensaje de solicitud hacia Authzforce será igual que el escenario de consulta, pero ahora el valor de la variable url sí tendrá importancia para la decisión de autorización.

Después de enviar el mensaje de solicitud de autorización, Authzforce lo evaluará mediante sus políticas activas. Para este escenario se ha implementado la política mostrada en la Figura 78.

```
<Policy PolicyId="escenario_sanitario_suscripcion" Version="1.0" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
  <Description>Politicas para la suscripcion por parte de los medicos</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">escenario_sanitario</AttributeValue>
              <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
              DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
           </Match>
        </AllOf>
     </AnyOf>
  </Target>
  <Rule RuleId="Reglas_Medicos_Hospital_Central_suscripcion" Effect="Permit">
     <Description>Reglas medicos del Hospital Central suscripcion</Description>
     <Target>
       <AnyOf>
          <AllOf>
             <Match MatchId="urn:oasis:names:tc:xacml:3.0:function:string-starts-with">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/v2/subscriptions</AttributeValue>
                <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
             </Match>
          </AllOf>
       </AnyOf>
       <AnyOf>
      <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">POST</AttributeValue>
               <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Condition>
```

Figura 78. Escenario de suscripción. Política.

En la primera parte de la política se comprueba si la aplicación en la que está trabajando el usuario es *escenario_sanitario.* Además, en esta política va a haber dos reglas, una para los médicos del Hospital Central y otra regla para los médicos de la Residencia. Vamos a detallar la regla de los médicos del Hospital Central ya que para la Residencia será igual, pero variando los valores que van a ser comparados con los atributos *url, organization y role*. Además, también se variará la franja horaria en función de a qué organización pertenezca y turno le pertenece el usuario.

El tipo de acción y el recurso al que se pretende acceder van a ser igual para las dos reglas, POST y /v2/subscriptions.

Se van a mostrar en la Figura 79 las condiciones usadas en esta regla.

En esta regla va a haber una sentencia OR donde se ha de cumplir una de las siguientes opciones:

- El médico tenga rol *Medicos_Hosp_Central_turno_mañana*, que el valor de organization sea HospitalCentral, que la dirección sea la registrada en dicha organización y además que el horario de consulta esté dentro del periodo de 9:00:00 a 15:00:00.
- Alternativamente, que; el médico tenga rol *Medicos_Hosp_Central_turno_tarde*, que el valor de organization sea HospitalCentral, que la dirección sea la registrada en dicha organización y que además el horario de consulta esté comprendido entre las 15:00:00 y las 21:00:00.

En otro caso dicha regla dará como resultado la negación de la autorización. Para la Residencia se ha creado otra regla para dicha política, muy similar a esta indicada.

```
<Condition>
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
       <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Medicos_Hosp_Central_turno_mañana</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
      AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
      DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
         <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">15:00:00</AttributeValue>
      </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
      </Apply>
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
       </Apply>
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
```

```
      DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
   </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
    </Apply>
           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.1:1028/subscriptions</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.2:1028/subscriptions</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.3:1028/subscriptions</AttributeValue>
    </Apply>
    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:url" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
  </Apply>
   </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
           <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Medicos_Hosp_Central_turno_tarde</AttributeValue>
   <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
   AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
</Apply>
 <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
      <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
      Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
      DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"></AttributeDesignator>
      </Apply>
```

```
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">15:00:00</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">21:00:00</AttributeValue>
        </Apply>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" >
           <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
           AttributeId="urn:oasis:names:tc:xacml:1.0:environment:organization"
           DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
        </Apply>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HospitalCentral</AttributeValue>
         </Apply>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.1:1028/subscriptions</AttributeValue>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.2:1028/subscriptions</AttributeValue>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://172.18.1.3:1028/subscriptions</AttributeValue>
         </Apply>
         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:url" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
       </Apply>
        </Apply>
        </Apply>
  </Condition>
</Rule>
```

Figura 79. Escenario de suscripción. Política.

Por último, tras la correcta autorización por parte de Authzforce, Wilma va a redireccionar a Orion la petición. En caso de haber alguna actualización de uno o varios atributos de una entidad, Authzforce no puede intervenir. Esto se debe a que Orion manda directamente la notificación a dicha dirección, sin pasar por

Wilma. Así que la única solución era restringir las direcciones a las que puede recibir un usuario de una organización en concreto.

A continuación, como resumen se va a exponer el esquema de funcionamiento del escenario de suscripción.

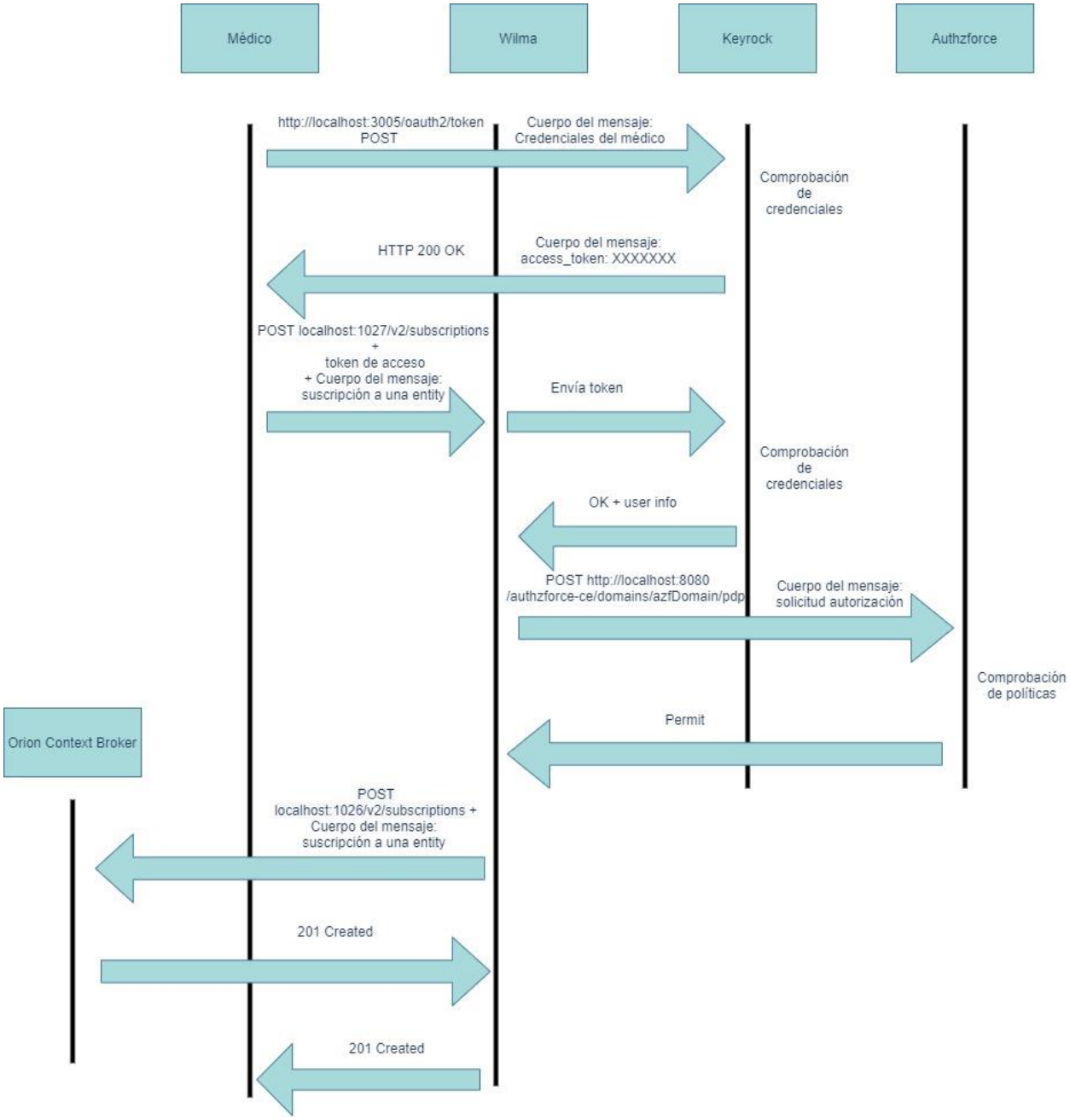

Figura 80. Esquema de funcionamiento del escenario 3.

Pruebas y resultados:

1. Médico de turno de tarde del Hospital Central realiza una operación de suscripción a la entidad urn:ngsi-ld:sensor:002(perteneciente a la organización HospitalCentral), indicando que la dirección a la que pretende recibir notificaciones es http://172.18.1.1:1028/subscriptions, a las 16:30:00.

   En el terminal donde previamente se ha configurado un servicio para recibir peticiones en la dirección 172.18.1.1, puerto 1028 y recurso subscriptions, recibimos la siguiente notificación. Esto se debe a que uno de los agentes IoT perteneciente al Hospital Central ha realizado una operación PATCH sobre la entidad urn:ngsi-ld:sensor:002. Se puede observar dicho servicio en la Figura 82.

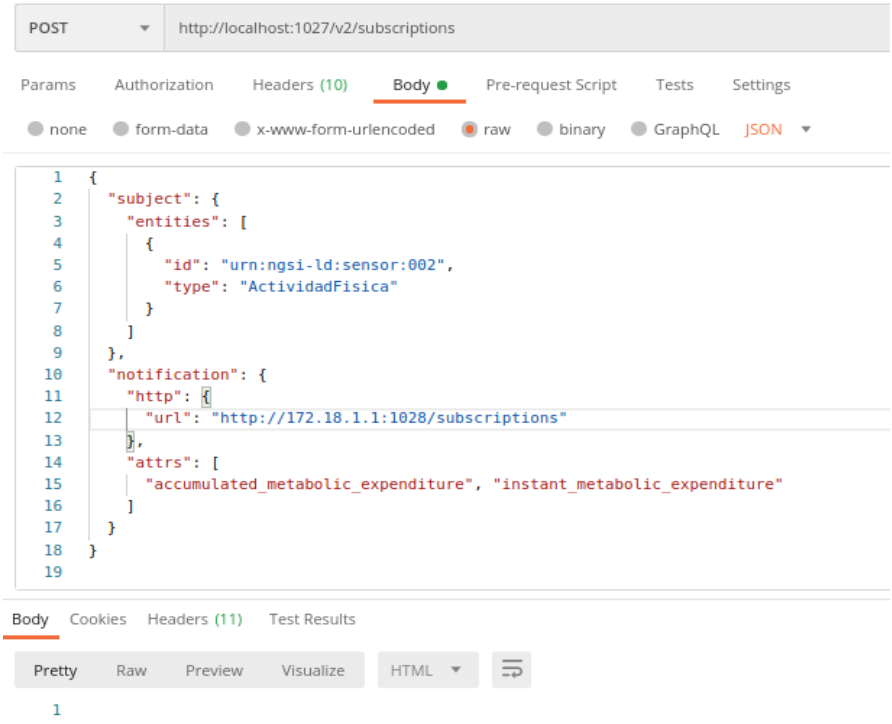

Figura 81. Escenario suscripción. Prueba 1.

Figura 82. Escenario suscripción. Prueba 1.

La puesta en marcha de este servicio se explicará con detalle en el Anexo C.

2. Médico de turno de tarde del Hospital Central realiza una operación de suscripción a la entidad urn:ngsi-ld:sensor:002(perteneciente a la organización HospitalCentral), indicando que la dirección a la que pretende recibir notificaciones es http://172.18.1.20:1028/subscriptions, a las 16:30:00. Dicha dirección no está registrada para el Hospital Central.

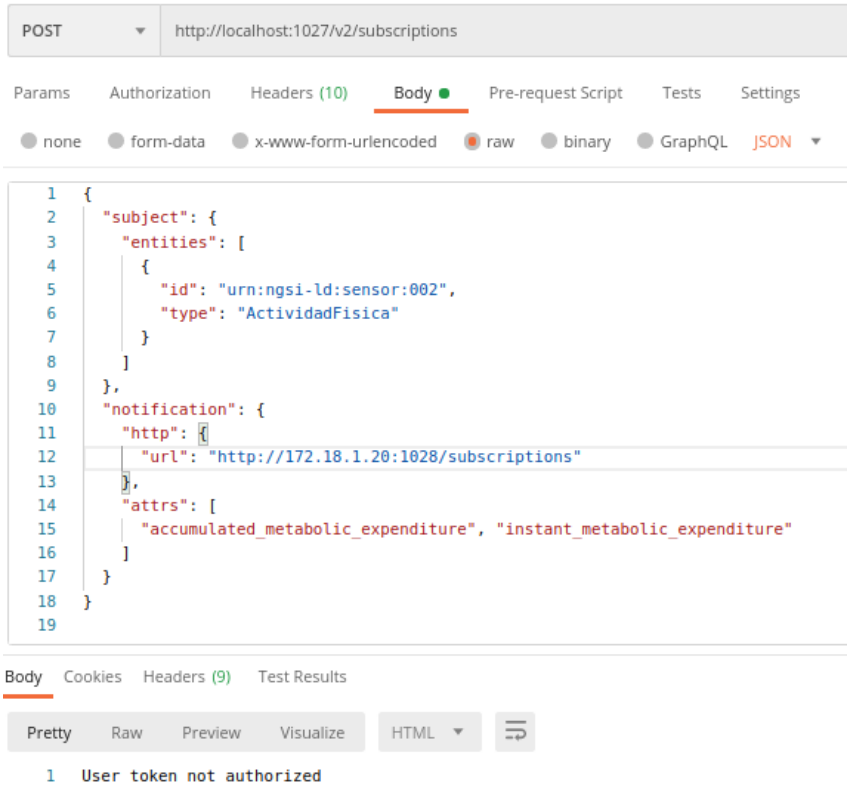

Figura 83. Escenario suscripción. Prueba 2.

3. Médico de turno de mañana del Hospital Central realiza una operación de suscripción a la entidad urn:ngsi-ld:sensor:002(perteneciente a la organización HospitalCentral), indicando que la dirección a la que pretende recibir notificaciones es http://172.18.1.1:1028/subscriptions, a las 16:30:00. Dicha hora no se le está permitido a dicho médico.

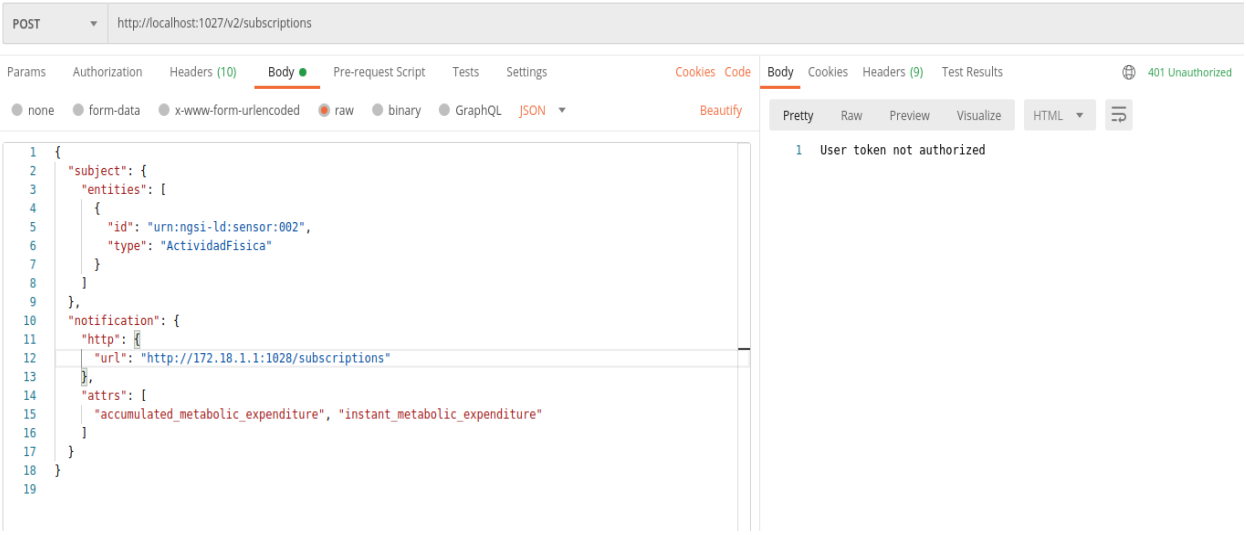

Figura 84. Escenario suscripción. Prueba 3.

4. Médico de turno de tarde de la Residencia de Sevilla realiza una operación de suscripción a la entidad urn:ngsi-ld:sensor:002(perteneciente a la organización HospitalCentral), indicando que la dirección a la que pretende recibir notificaciones es http://172.18.1.4:1028/subscriptions, a las 16:30:00. No tiene acceso a dicha entidad.

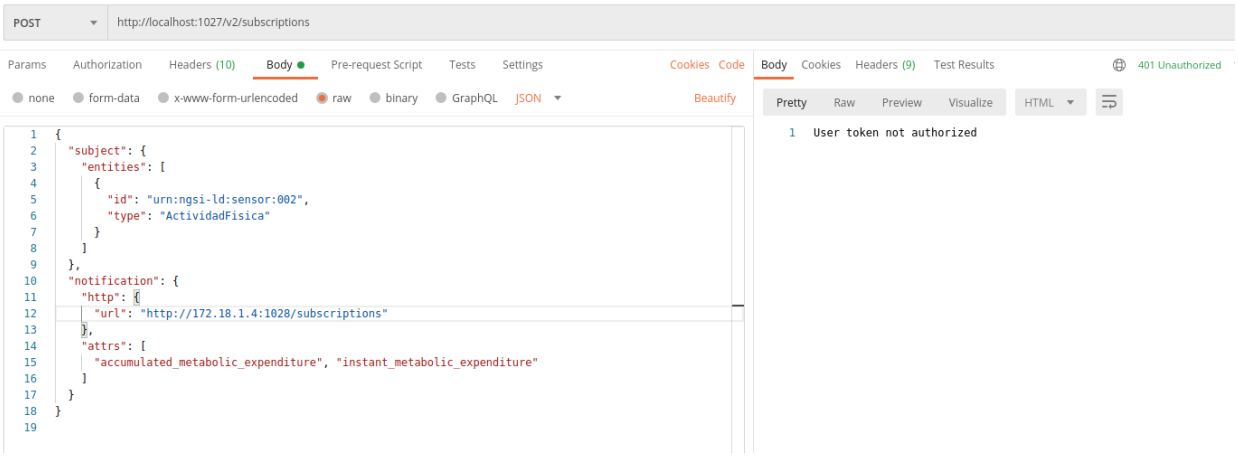

Figura 85. Escenario suscripción. Prueba 4.

3.5.4 Escenario 4. Autorización para el administrador del sistema.

Para finalizar, se ha creado un usuario Administrador que puede consultar cualquier entidad previamente registrada y, además, podrá observar todas las suscripciones creadas a cualquier hora. La solicitud de autorización hacia Authzforce será la misma que en el escenario de consulta, pero los únicos atributos que tendrán importancia serán el recurso y la acción; url y peticionget estarán vacíos ya que no serán evaluados en Authzforce al tratarse de un usuario Administrador.

En la Figura 86, se puede comprobar en la política mostrada, que la directiva <Target> tiene un mayor papel que la directiva <Condition>, ya que esta sólo evalúa si el rol del usuario es *Administrador*.

En la sentencia <Target>, se ha indicado dos directivas <AnyOf>, tal y como se explicó, se sabe que deben cumplirse ambas. En el primer <AnyOf> se evalúa si la acción que realiza el usuario es GET, en otro caso, dicha política denegaría la autorización.

En el segundo <AnyOf> aparecen dos <AllOf> de modo que debe cumplirse una de las dos sentencias. La primera comprueba si el recurso al que se accede es /v2/entities, y el segundo comprueba si es /v2/subscriptions. Al igual que en el primer <AnyOf> si no se cumple ningún <AllOf>, la política denegaría la autorización. Es necesario resaltar que, si otra política permite la autorización, el acceso sería viable

```
<Policy PolicyId="escenario_sanitario_administrador" Version="1.0" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
    <Description>Politicas para el administrador</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">escenario_sanitario</AttributeValue>
                <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
             </Match>
          </AllOf>
       </AnyOf>
    </Target>
    <Rule RuleId="Reglas_Administrador" Effect="Permit">
       <Description>Regla para el administrador</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">GET</AttributeValue>
                   <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                   AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                </Match>
             </AllOf>
          </AnyOf>
```

```
          <AnyOf>
             <AllOf>
               <Match MatchId="urn:oasis:names:tc:xacml:3.0:function:string-starts-with">
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/v2/subscriptions</AttributeValue>
                   <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                   AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                </Match>
             </AllOf>
                <AllOf>
                 <Match MatchId="urn:oasis:names:tc:xacml:3.0:function:string-starts-with">
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/v2/entities</AttributeValue>
                   <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                   AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                </Match>
             </AllOf>
          </AnyOf>
       </Target>
               <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
             <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Administrador</AttributeValue>
             <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
             AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
             DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
          </Apply>
                  </Condition>
     </Rule>
 </Policy>
</PolicySet>'
```

Figura 86. Escenario para el administrador. Política.

Pruebas y resultados:

1. El administrador quiere obtener todas las suscripciones activas del sistema. En esta prueba solo había una suscripción.

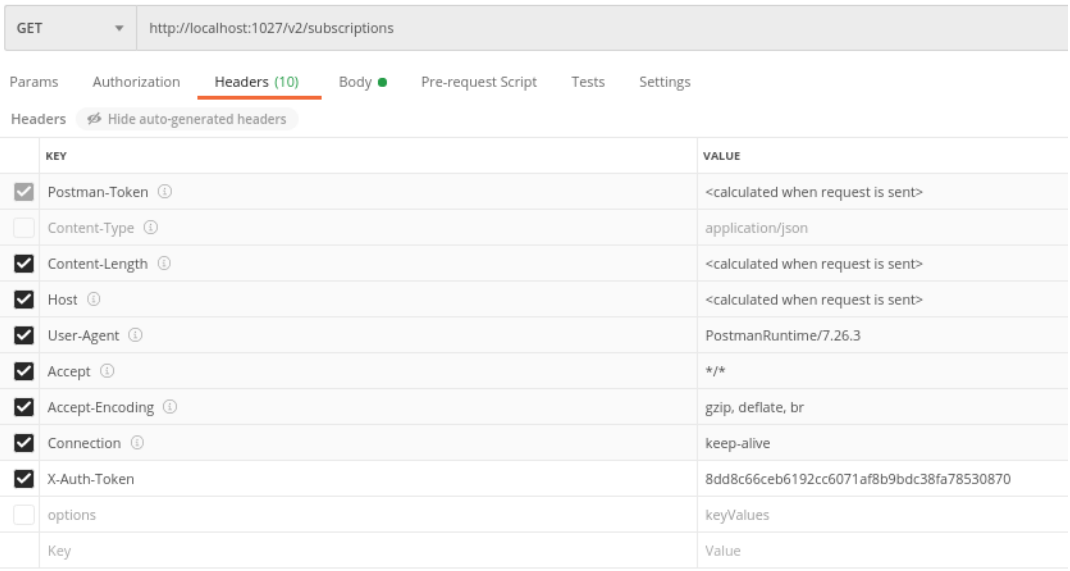

Figura 87. Escenario para el administrador. Prueba 1. Petición.

```
{
    "id": "5f53780cd8ee5b14c726f4b6",
    "status": "active",
    "subject": {
        "entities": [
            {
                "id": "urn:ngsi-ld:sensor:002",
                "type": "ActividadFisica"
            }
        ],
        "condition": {
            "attrs": []
        }
    },
    "notification": {
        "timesSent": 1,
        "lastNotification": "2020-09-05T11:35:40.00Z",
        "attrs": [
            "accumulated_metabolic_expenditure",
            "instant_metabolic_expenditure"
        ],
        "onlyChangedAttrs": false,
        "attrsFormat": "normalized",
        "http": {
            "url": "http://172.18.1.1:1028/subscriptions"
        },
        "lastSuccess": "2020-09-05T11:35:40.00Z",
        "lastSuccessCode": 200
    }
}
```


Figura 88. Escenario para el administrador. Prueba 1. Respuesta.

2. El administrador quiere obtener todas las entidades registradas en el sistema.

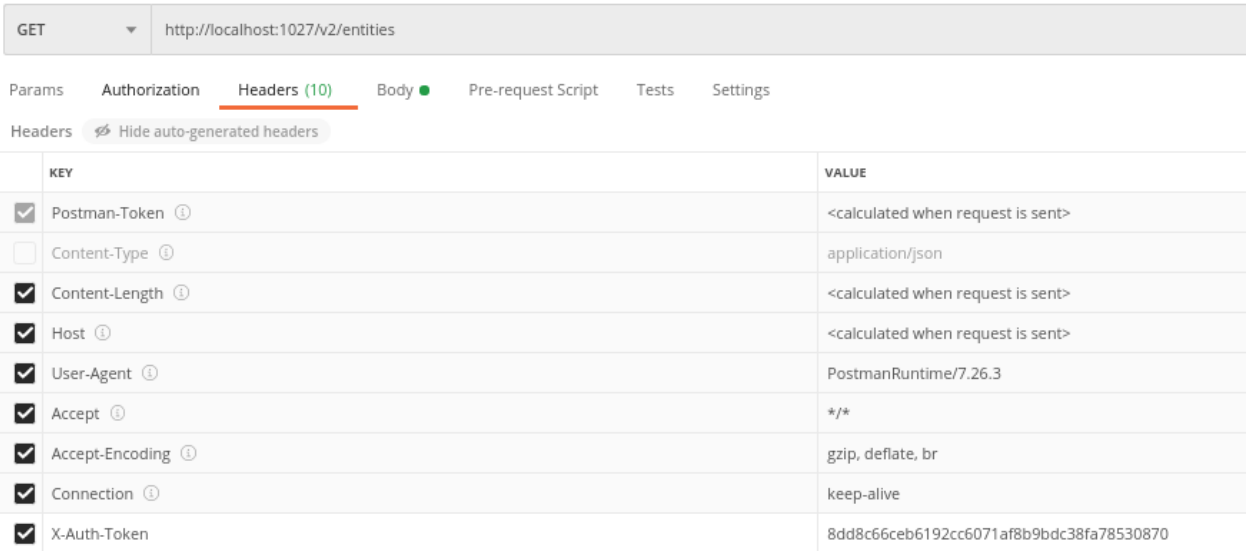

Figura 89. Escenario para el administrador. Prueba 2. Petición.

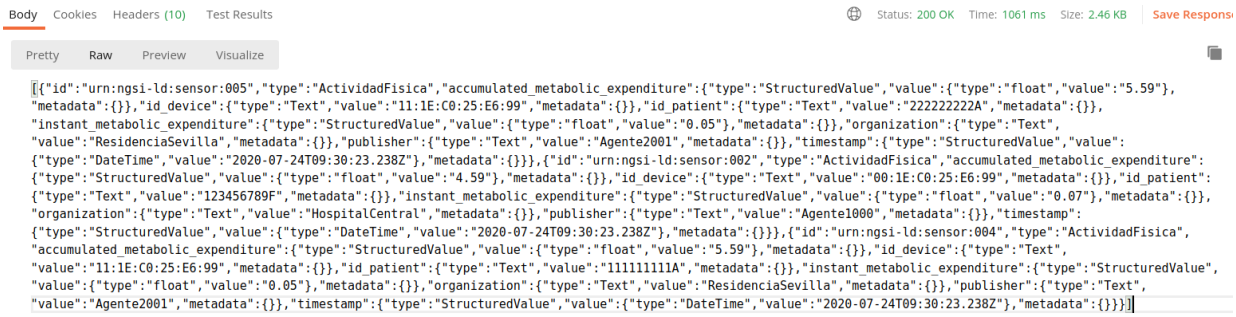

Figura 90. Escenario para el administrador. Prueba 2. Respuesta.

4 CONCLUSIONES Y LÍNEAS FUTURAS

En la realización de este proyecto se ha conseguido establecer una serie de niveles de seguridad para poder controlar el acceso a la información de los pacientes, y, además, controlar la publicación por parte de los agentes IoT.

Uno de los aspectos más importantes para conseguir el objetivo del proyecto ha sido entender muy bien Authzforce y Wilma, saber qué información enviaba Wilma a Authzforce, cómo la recibía Authzforce, qué respuesta daba Authzforce y cómo la redireccionaba Wilma a Orion.

Cabe destacar que Authzforce aún está en crecimiento y tiene muchas limitaciones para trabajar con él, pero gracias a las modificaciones realizadas tanto en Wilma como Authzforce se ha conseguido con éxito la implementación de los diferentes escenarios sanitarios planteados. Los inconvenientes que se han dado en el uso de Authzforce para implementar el mecanismo de seguridad en el sistema han sido:

- No permite una conexión directa con Orion, por lo que no se puede restringir directamente, mediante Authzforce, la consulta de alguna entidad con respecto a los valores de los atributos que contengan dichas entidades a la que desea acceder un determinado usuario. Esto lo hemos logrado gracias a que Wilma permite enviar los atributos de una entidad concreta a Authzforce, para que este pueda evaluarlos. Para ello, se ha modificado Wilma para que consulte los atributos necesarios en Orion antes de hacer la petición a Authzforce.
- En numerosas ocasiones he encontrado errores en la verificación de las políticas ante la presencia de tabulaciones, debido a que los tomaba como caracteres. Una posible solución podría ser el desarrollo de una interfaz gráfica que permita indicar precisamente los errores en la definición de las políticas. Otra alternativa sería separar la funcionalidad del PAP de Authzforce, e implementarla a otro componente que permita una mejor visión de errores en la declaración de las políticas.
- La creación de los dominios, políticas y puesta en marcha de las políticas se ha realizado mediante el terminal, y resultaba incómodo la mayoría de las veces. Un punto de mejora futura posible sería implementar una interfaz gráfica que permita crear políticas y ponerlas en funcionamiento.

Sobre el escenario de suscripción, una posible mejora futura sería poder obtener diferentes suscripciones en función del valor de un atributo en concreto de la entidad suscrita. Esta propuesta se debe a que en el sistema planteado no se ha permitido que los médicos consulten el recurso /v2/subscriptions porque Orion devuelve todas las suscripciones existentes y no tiene sentido que los médicos del Hospital Central sepan qué suscripciones hay en la Residencia.

Con respecto a Keyrock, una posible mejora, sería alargar el tiempo de vida de un token proporcionado por este componente ya que cada poco tiempo caduca el token y hay que actualizarlo.

Un planteamiento de cara al futuro del sistema sería implementar un Servicio Web, para que los médicos puedan acceder mediante una interfaz gráfica que permita visualizar las actividades físicas de los distintos pacientes y realizar estadísticas sobre los diferentes datos que proporcione la camiseta al medir por ejemplos las calorías perdidas de un paciente.

El mecanismo de control de acceso desarrollado ha sido aplicado al caso de uso de intervenciones de actividad física, pero podría utilizarse en otros escenarios sanitarios. Para poder usar el mecanismo en cualquier aplicación sanitaria es necesario primero tener instalado los componentes descritos en este trabajo. Por consiguiente, se debe plantear y describir las entidades específicas del dominio que se van a utilizar, con el fin de saber a quién se debe permitir el acceso al sistema y qué tipo de información se va a transmitir. Una vez definidas las entidades, se deben registrar aquellos usuarios o agentes que vayan a acceder al sistema, y clasificarlos en función del rol que les pertenezca. Por último, se deberían definir y poner en funcionamiento las políticas que permitan acceder al sistema a aquellos usuarios o agentes autorizados en función de una serie de condiciones, como por ejemplo el horario de acceso, el rol que tenga el usuario, etc.

ANEXO A: INSTALACIÓN Y CONFIGURACIÓN DE DOCKER

Para el despliegue de los distintos componentes usados de FIWARE se ha utilizado Docker y que con un simple fichero de configuración y docker-compose, es posible preparar el entorno de forma eficiente.

A continuación, se muestran los pasos a seguir para instalar Docker y Docker-compose.

Primero, actualizamos la máquina a la última versión usando los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade
```

En este paso debemos crear el nuevo protocolo de HTTPS. Ingresa el siguiente comando en la línea de comando:

```
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

Ahora debemos importar el comando mientras usamos la clave GPG correcta con el comando Curl. Para esto solo debemos escribir el siguiente comando:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Habiendo llegado a este punto necesitaremos un comando adicional para agregar el repositorio Docker APT.

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Por último, procederemos a la instalación completa de Docker mediante el siguiente comando:

```
sudo apt-get install docker-ce
```

Para la instalación de docker-compose debemos ejecutar los siguientes comandos:

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Comprobamos si se ha instalado correctamente mediante los siguientes comandos:

```
salas@linux:~$ docker-compose --version
docker-compose version 1.21.2, build a133471
salas@linux:~$ docker --version
Docker version 19.03.12, build 48a66213fe
```

Figura 91. Comprobación de la instalación de Docker y Docker-Compose.

ANEXO B: INSTALACIÓN Y CONFIGURACIÓN DE ORION, AUTHZFORCE, WILMA Y KEYROCK.

Para el despliegue de todos los componentes FIWARE utilizados, se ha seguido la guía indicada por Git-Hub [13] y se ha personalizado el fichero de configuración, eliminando los componentes no necesarios y añadiendo la configuración adecuada al sistema planteado.

Todos los ficheros relativos a la instalación y el despliegue de los componentes de FIWARE se encuentran en la carpeta escenario_sanitario. La estructura de las carpetas y archivos más importantes es la siguiente:

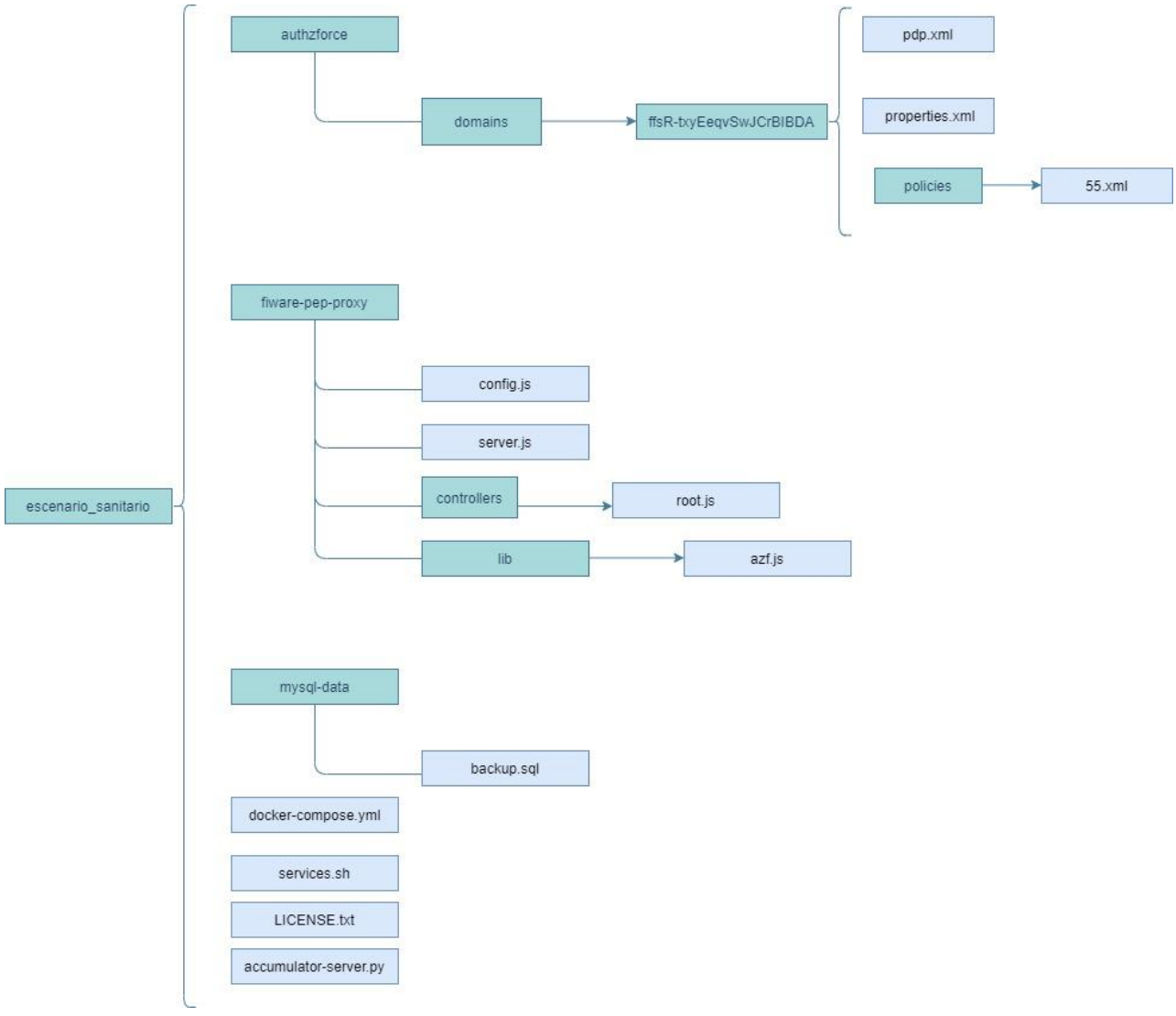

Figura 92. Estructura de la carpeta escenario_sanitario

El fichero ejecutable services.sh contiene todas las órdenes para el despliegue de los componentes del sistema.

```
case "${command}" in
        "help")
        echo "usage: services [create|start|stop]"
        ;;
    "start")
                stoppingContainers
                echo -e "Starting seven containers \033[1;34mOrion\033[0m, \033[1;31mKeyrock\033[0m,
                        \033[1;31mAuthZforce\033[0m and \033[1mMongoDB\033[0m and \033[1mMySQL\033[0m databases."
                echo -e "- \033[1;34mOrion\033[0m is the context broker"
                echo -e "- \033[1;31mKeyrock\033[0m is an Identity Management Front-End"
                echo -e "- \033[1;31mAuthZforce\033[0m is a XACML Authorization Server"
                startContainers false
                waitForOrion
                waitForKeyrock
                displayServices
                echo -e "Now open \033[4mhttp://localhost:3000\033[0m"
                ;;
        "stop")
                stoppingContainers
                ;;
        "create")
                echo "Pulling Docker images"
                docker-compose --log-level ERROR -p fiware pull
                ;;
        *)
                echo "Command not Found."
                echo "usage: services [create|start|stop|stop]"
                exit 127;
                ;;
```

Figura 93. Services.sh.

A continuación, se describen las más importantes:

```
startContainers () {
        echo ""
        export IDM_HTTPS_ENABLED="$1"
        docker-compose --log-level ERROR -f docker-compose.yml -p fiware up -d --remove-orphans
        echo ""
}

stoppingContainers () {
        echo "Stopping containers"
        docker-compose --log-level ERROR -f docker-compose.yml  -p fiware down -v --remove-orphans
        echo ""

}
```

Figura 94. Funciones startContainers y stoppingContainers.

- stoppingContainers: elimina todos los posibles contenedores que estuviesen previamente en la máquina, para evitar duplicidades y hacer un despliegue limpio.
- startContainers: ejecuta el archivo docker-compose.yml indicado por parámetros, arrancando los contenedores que contienen los componentes del proyecto: Orion, Authzforce y Keyrock.

```
displayServices () {
        echo ""
        docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}" --filter name=fiware-*
        echo ""
}
```

Figura 95. Función displayServices.

- displayServices: Muestra una tabla indicando el estado de los componentes una vez que se han puesto en marcha.

El fichero Docker-compose.yml recoge toda la configuración de los distintos componentes y su interconexión, y se comentará en el siguiente apartado.

Fichero de despliegue Docker-compose.yml

Docker-Compose permite desplegar a través de un simple comando una aplicación contenida en múltiples contenedores, relacionados entre sí en la misma máquina. Toda la configuración de cada componente se recogerá en un fichero YAML.

En este proyecto el fichero Docker-compose.yml es el que contiene toda la información necesaria para el despliegue. A continuación, se muestra la configuración de cada uno de los contenedores y se destacarán los campos más importantes.

- Orion Context Broker:

```
orion:
  image: fiware/orion:2.4.0
  hostname: orion
  container_name: fiware-orion
  depends_on:
    - mongo-db
  networks:
    default:
      ipv4_address: 172.18.1.9
  expose:
    - "1026"
  ports:
    - "1026:1026"
  command: -dbhost mongo-db -logLevel DEBUG
  healthcheck:
    test: curl --fail -s http://localhost:1026/version || exit 1
```

Figura 96. Configuración de Orion en el fichero Docker-compose.yml.

- image: se indica la imagen Docker a descargar junto con la versión requerida.
- hostname: define el nombre dentro de la red creada.
- depends_on: indica las dependencias de arranque de servicios. Antes de poner en marcha este contenedor, se iniciarán las dependencias. Para cerrar los contenedores, se cierran antes los que dependen de otros.
- networks: establece la subred donde se encontrará el contenedor.
- expose: establece puertos que solo estarán disponibles dentro de los servicios definidos, no en la máquina donde se ejecuta.
- ports: puertos que se van a abrir de cara al exterior, para que sean accesibles desde fuera
- healthcheck: establece un comando para determinar si el contenedor está correctamente desplegado

- Keyrock:

```
keyrock:
  image: fiware/idm:7.8.1
  container_name: fiware-keyrock
  hostname: keyrock
  networks:
    default:
      ipv4_address: 172.18.1.5
  depends_on:
    - mysql-db
    - authzforce
  ports:
    - "3005:3005"
  environment:
    - DEBUG=idm:*
    - IDM_DB_HOST=mysql-db
    - IDM_DB_PASS=secret
    - IDM_DB_USER=root
    - IDM_HOST=http://localhost:3005
    - IDM_PORT=3005
    - IDM_ADMIN_USER=fernando_admin_aplicacion
    - IDM_ADMIN_EMAIL=fernando-the-admin@tfg.com
    - IDM_ADMIN_PASS=test
    - IDM_PDP_LEVEL=advanced
    - IDM_AUTHZFORCE_ENABLED=true
    - IDM_AUTHZFORCE_HOST=authzforce
    - IDM_AUTHZFORCE_PORT=8080
  healthcheck:
    test: curl --fail -s http://localhost:3005/version || exit 1
```

Figura 97. Configuración de Keyrock en el fichero Docker-compose.yml.

- image: se utiliza la versión del contenedor 7.8.1
- depends_on: depende de MySQL, para almacenar información sobre credenciales, aplicaciones registradas, usuarios registrados, …
- ports: Keyrock está disponible a través del puerto 3005, tanto para la conexión con el PEP Proxy para solicitar tokens de acceso.
- IDM_DB_USER: especifica el usuario con acceso a la base de datos.
- IDM_ADMIN_USER: nombre del administrador de Keyrock.
- IDM_ADMIN_EMAIL: email con el que acceder a la configuración de Keyrock
- IDM_ADMIN_PASS: contraseña definida para acceder a la configuración de Keyrock junto con el email.
- IDM_PDP_LEVEL: especifica que tipo de nível va a ser el PDP, con advanced indicamos que Authzforce va a actuar como PDP, y no Keyrock.
- IDM_AUTHZFORCE_ENABLED: Indicamos si la conexión Keyrock y Authzforce está o no habilitada.

- Authzforce:

```
authzforce:
  image: fiware/authzforce-ce-server:release-8.1.0
  hostname: authzforce
  container_name: fiware-authzforce
  networks:
    default:
      ipv4_address: 172.18.1.12
  ports:
    - "8080:8080" # localhost:8080
  volumes:
    - ./authzforce/domains:/opt/authzforce-ce-server/data/domains
  healthcheck:
    test: curl --fail -s http://authzforce:8080/authzforce-ce/version || exit 1
```

Figura 98. Configuración de Authzforce en el fichero Docker-compose.yml.

- volumes: En el primer argumento indicamos la ruta donde queremos que se almacenen las políticas, el segundo argumento es la ruta donde se monta el archivo o directorio en el contenedor.

- Base de datos utilizadas: MongoDB y MySQL:

```
# Databases
mongo-db:
  image: mongo:3.6
  hostname: mongo-db
  container_name: db-mongo
  ports:
    - "27017:27017" # localhost:27017
  expose:
    - "27017"
  networks:
    - default
  command: --bind_ip_all --smallfiles
  volumes:
    - mongo-db:/data

mysql-db:
  restart: always
  image: mysql:5.7
  hostname: mysql-db
  container_name: db-mysql
  expose:
    - "3306"
  ports:
    - "3306:3306" # localhost:3306
  networks:
    default:
      ipv4_address: 172.18.1.6
  environment:
    - "MYSQL_ROOT_PASSWORD=secret"
    - "MYSQL_ROOT_HOST=172.18.1.5" # Allow Keyrock to access this database
  volumes:
    - mysql-db:/var/lib/mysql
    - ./mysql-data:/docker-entrypoint-initdb.d/:ro # Preload Keyrock Users
```

Figura 99. Configuración de MongoDB y MySQL en el fichero Docker-compose.yml.

- volumes (en MySQL): se especifica el fichero que va a exponer el estado inicial de la base de datos. Se explicará con detalle dicho estado inicial en el siguiente apartado.

- PEP Proxy Wilma:

Va a ser el único componente instalado mediante código fuente.

Primero debemos instalar nodejs y npm para hacer arrancar el servidor de Wilma.

```
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash –
sudo apt-get install nodejs
sudo apt install npm
```

Es habitual que salga el error: Error: *Cannot find module 'xmlhttprequest*, para arreglarlo solo debemos agregar dicho módulo mediante el siguiente comando:

```
npm install xmlhttprequest
```

Comprobamos si se han instalado correctamente, mediante los siguientes comandos:

```
salas@linux:~$ node -v
v13.14.0
salas@linux:~$ npm -v
6.14.4
```

Figura 100. Comprobación de las versiones de node y npm.

Una vez instalados node.js y npm procedemos a clonar un repositorio de Github [14] con los archivos necesarios para el despliegue de Wilma.

La configuración de PEP-Proxy Wilma erá la siguiente:

```
// Used only if https is disabled
config.pep_port = process.env.PEP_PROXY_PORT || 1027;

// Set this var to undefined if you don't want the server to listen on HTTPS
config.https = {
  enabled: toBoolean(process.env.PEP_PROXY_HTTPS_ENABLED, false),
  cert_file: 'cert/cert.crt',
  key_file: 'cert/key.key',
  port: process.env.PEP_PROXY_HTTPS_PORT || 443,
};

config.idm = {
  host: process.env.PEP_PROXY_IDM_HOST || 'localhost',
  port: process.env.PEP_PROXY_IDM_PORT || 3005,
  ssl: toBoolean(process.env.PEP_PROXY_IDM_SSL_ENABLED, false),
};

config.app = {
  host: process.env.PEP_PROXY_APP_HOST || 'localhost',
  port: process.env.PEP_PROXY_APP_PORT || '1026',
  ssl: toBoolean(process.env.PEP_PROXY_APP_SSL_ENABLED, false), // Use true if the app server listens in https
};

config.organizations = {
  enabled: toBoolean(process.env.PEP_PROXY_ORG_ENABLED, false),
  header: process.env.PEP_PROXY_ORG_HEADER || 'fiware-service'
}

// Credentials obtained when registering PEP Proxy in app_id in Account Portal
config.pep = {
  app_id: process.env.PEP_PROXY_APP_ID || 'escenario_sanitario',
  username: process.env.PEP_PROXY_USERNAME || 'pep_proxy_00000000-0000-0000-0000-000000000000',
  password: process.env.PEP_PASSWORD || 'test',
  token: {
    secret: process.env.PEP_TOKEN_SECRET || '', // Secret must be configured in order validate a jwt
  },
  trusted_apps: [],
```

Figura 101. Configuración de PEP Proxy Wilma.

```
config.authorization = {
  enabled: toBoolean(process.env.PEP_PROXY_AUTH_ENABLED, true),
  pdp: process.env.PEP_PROXY_PDP || 'authzforce', // idm|authzforce
  azf: {
    protocol: process.env.PEP_PROXY_AZF_PROTOCOL || 'http',
    host: process.env.PEP_PROXY_AZF_HOST || 'localhost',
    port: process.env.PEP_PROXY_AZF_PORT || 8080,
    custom_policy: process.env.PEP_PROXY_AZF_CUSTOM_POLICY || undefined, // use undefined to default policy checks (HTTP verb + path).
  },
};
```

Figura 102. Configuración de PEP Proxy Wilma.

En config.pep_port, se ha configurado PEP Proxy Wilma para que escuche peticiones en el puerto 1027.

Luego indicamos que el IDM está ubicado en el puerto 3005 con dirección "localhost", concretamente en la parte de config.idm.

En config.app indicamos que Orion Context Broker esta ubicado en el puerto 1026 con la dirección 127.0.0.1.

En config.pep se configura Wilma para que pueda acceder a Keyrock, se indica el id de la aplicación registrada en Keyrock, el nombre y contraseña del PEP Proxy registrado en Keyrock. Estos valores se almacenan en MySQL al arrancarse el sistema gracias al archivo de .sql que explicaremos más adelante.

Por último, se ha modificado en el archivo config.js, config.authorization para que Wilma reconozca a Authzforce como el PDP del sistema, indicando respectivamente en qué puerto está ubicado.

Para iniciar PEP-Proxy Wilma debemos ejecutar el siguiente comando en el directorio donde estén creados los ficheros de configuración de Wilma:

```
sudo node server
```

- **Estado inicial de la base de datos MySQL**

Cuando ponemos en marcha los contenedores, se crea una instancia de la base de datos MySQL, donde se va a almacenar aplicaciones, organizaciones, los tokens, claves, usuarios permitidos relativos a Keyrock.

Para facilitar el estado inicial del sistema, se ha creado un fichero de órdenes .sql que introduce en la base de datos MySQL información inicial sobre la aplicación, usuarios registrados, sobre el dominio que se va a trabajar Authzforce, etc…

Se va a explicar algunas de las tablas creadas para el estado inicial del sistema.

- Se ha introducido el id de dominio de Authzforce junto con el id de PolicySet en la tabla Authzforce de la base de datos. Esta tabla será necesaria cuando Wilma vaya a consultar con Authzforce, ya que debe verificar antes si hay algún dominio creado.

```
LOCK TABLES `authzforce` WRITE;
/*!40000 ALTER TABLE `authzforce` DISABLE KEYS */;
INSERT INTO `authzforce` VALUES
('ffsR-txyEeqvSwJCrBIBDA','f8194af5-8a07-486a-9581-c1f05d05483c',2,'escenario_sanitario');
/*!40000 ALTER TABLE `authzforce` ENABLE KEYS */;
UNLOCK TABLES;
```

Figura 103. Tabla authzforce en MySQL.

- Se ha registrado un PEP Proxy en la aplicación con una serie de parámetros que serán necesarios para la conectividad entre Wilma y Keyrock.

```
LOCK TABLES `pep_proxy` WRITE;
/*!40000 ALTER TABLE `pep_proxy` DISABLE KEYS */;
INSERT INTO `pep_proxy` VALUES ('pep_proxy_00000000-0000-0000-0000-000000000000','e9f7c64ec2895eec281f8fd36e588d1bc762bcca',NULL,'escenario_sanitario');
/*!40000 ALTER TABLE `pep_proxy` ENABLE KEYS */;
UNLOCK TABLES;
```

Figura 104. Tabla pep_proxy en MySQL.

- Se han creado una serie de roles que serán necesarios para especificar la función que tiene cada usuario en el sistema.

```
LOCK TABLES `role` WRITE;
/*!40000 ALTER TABLE `role` DISABLE KEYS */;
INSERT INTO `role` VALUES
('Medicos_Hosp_Central_turno_mañana','Medicos',0,'escenario_sanitario'),
('Medicos_Hosp_Central_turno_tarde','Medicos',0,'escenario_sanitario'),
('Medicos_Res_Sevilla_turno_mañana','Medicos',0,'escenario_sanitario'),
('Medicos_Res_Sevilla_turno_tarde','Medicos',0,'escenario_sanitario'),
('Agente_IoT_Hospital_Central','Agentes_IoT',0,'escenario_sanitario'),
('Agente_IoT_Residencia_Sevilla','Agentes_IoT',0,'escenario_sanitario'),
('Administrador','Admin',0,'escenario_sanitario'),
('provider','Provider',1,'idm_admin_app'),('purchaser','Purchaser',1,'idm_admin_app');
```

Figura 105. Tabla role en MySQL.

- Se han creado una serie de usuarios. Se podrían dividir en 8 conjuntos:
  - Un Administrador del sistema.
  - Dos médicos de turno de mañana perteneciente al Hospital Central de Sevilla.
  - Dos médicos de turno de tarde perteneciente al Hospital Central de Sevilla.
  - Dos médicos de turno de mañana perteneciente a la Residencia de Sevilla.
  - Dos médicos de turno de tarde perteneciente a la Residencia de Sevilla.
  - Dos agentes IoT pertenecientes al Hospital Central de Sevilla.
  - Dos agentes IoT pertenecientes a la Residencia de Sevilla.
  - Un agente IoT malicioso y un usuario maliciso.

```
LOCK TABLES `user` WRITE;
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
INSERT INTO `user` VALUES
('fernando_admin_aplicacion','Fernando', 'Fernando is the admin',NULL,'default',0,'fernando-the-admin@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,1,NULL,NULL,
0,NULL),
('Jose_Medico_Hospital_Central','Jose','Jose is a medician',NULL,'default',0,'jose_hospital_central@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL),
('Pablo_Medico_Hospital_Central','Jose','Pablo is a medician',NULL,'default',0,'pablo_hospital_central@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Fernando_Medico_Hospital_Central','Fernando','Fernando is a medician',NULL,'default',0,'fernando_hospital_central@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Maria_Medico_Hospital_Central','Maria','Maria is a medician',NULL,'default',0,'maria_hospital_central@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Ana_Medico_Residencia_Sevilla','Ana','Ana is a medician',NULL,'default',0,'ana_residencia_sevilla@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL),
('Sergio_Medico_Residencia_Sevilla','Sergio','Sergio is a medician',NULL,'default',0,'sergio_residencia_sevilla@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Rafael_Medico_Residencia_Sevilla','Rafael','Rafael is a medician',NULL,'default',0,'rafael_residencia_sevilla@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Marta_Medico_Residencia_Sevilla','Marta','Marta is a medician',NULL,'default',0,'marta_residencia_sevilla@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Agente_IoT_1000','Agente IoT 1000','Agente IoT',NULL,'default',0,'agente_iot_hospital_central_1000@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL),
('Agente_IoT_1001','Agente IoT 1001','Agente IoT',NULL,'default',0,'agente_iot_hospital_central_1001@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL),
('Agente_IoT_2000','Agente IoT 2000','Agente IoT',NULL,'default',0,'agente_iot_residencia_sevilla_2000@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Agente_IoT_2001','Agente IoT 2001','Agente IoT',NULL,'default',0,'agente_iot_residencia_sevilla_2001@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',
1,0,NULL,NULL,0,NULL),
('Agente_IoT_malicioso','Agente IoT malicioso','Agente IoT',NULL,'default',0,'agente_iot_malicioso@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL),
('Usuario_malicioso','Usuario malicioso','Usuario malicioso',NULL,'default',0,'usuario_malicioso@tfg.com','89e48c55e4e4b3b86141fb15f5e6abf70f8c32c0', 'fbba54b6750b16e8', '2020-08-10 11:41:14',1,0,NULL,NULL,
0,NULL);
```

Figura 106. Tabla user en MySQL.

- Se ha registrado una aplicación a la que pertenecerán todos los usuarios citados anteriormente con ciertos permisos para poder consultar, publicar y suscribir a las entidades creadas en Orion.

```
INSERT INTO `oauth_client` VALUES
('escenario_sanitario','Aplicacion para escenarios sanitarios',
  'FIWARE Sanitary Application','escenario_sanitario_secret',
  'http://localhost:3000','http://localhost:3000/login',NULL,'default','authorization_code,implicit,password,client_credentials,refresh_token','code',NULL,NULL,NULL,'bearer', NULL);
```

Figura 107. Tabla oauth_client en MySQL.

ANEXO C: INSTALACIÓN DE ACCUMULATOR-SERVER.

Para la recepción de notificaciones una herramienta muy útil proporcionada por FIWARE es el Accumulator Server. Es un servidor que escucha las notificaciones que le llegan y las imprime por la salida estándar. De tal modo que se podrán obtener los mensajes de notificación que provienen de Orion.

Para instalar el Accumulator Server es suficiente con descargar el código Python proporcionado por el repositorio de Telefónica I+D[15].

Para la ejecución de este servicio debemos tener instalado Python y dos librerías:

```
sudo apt install python
pip install Flask==1.0.2
pip install pyOpenSSL==19.0.0
```

Para ejecutar dicho servicio debemos ejecutar el siguiente comando:

```
./accumulator-server.py --port 1028 --url /subscriptions --host 172.18.1.1 --pretty-print -v
```

Una vez en ejecución, si se desea que la aplicación notifique al Accumulator Server, basta con introducir como URL de notificación de la suscripción la siguiente URL: http://172.18.1.1:1028/subscriptions.

Para crear direcciones IP de una determinada interfaz de red bastaría con ejecutar el siguiente comando:

```
sudo ip addr add DIRECCION_A_CREAR/MASCARA dev INTERFAZ_DE_RED
```

Un ejemplo sería:

```
sudo ip addr add 172.18.1.2/24 dev br-32daa60f40bc
```

Esto será útil para simular el escenario de suscripción, ya que cada organización tiene registrada una serie de direcciones IPs.

La Figura 108 muestra una notificación recibida por el Accumulator Server con los atributos que se han modificado, el identificador de la entidad, tipo de la misma y el id de la suscripción.

```
POST http://172.18.1.1:1028/subscriptions
Fiware-Servicepath: /
Content-Length: 271
X-Auth-Token: 6651941d5c3dcc287cbacbaec9461cd74704e109
User-Agent: orion/2.4.0-next libcurl/7.29.0
Ngsiv2-Attrsformat: normalized
Host: 172.18.1.1:1028
Accept: application/json
Content-Type: application/json; charset=utf-8
Fiware-Correlator: ae8adfec-eb9f-11ea-adf6-0242ac120109

{
    "data": [
        {
            "accumulated_metabolic_expenditure": {
                "metadata": {},
                "type": "float",
                "value": "2.59"
            },
            "id": "urn:ngsi-ld:sensor:002",
            "instant_metabolic_expenditure": {
                "metadata": {},
                "type": "float",
                "value": "0.05"
            },
            "type": "ActividadFisica"
        }
    ],
    "subscriptionId": "5f4d17218064f4ca472b444d"
}
```

Figura 108. Ejemplo de una notificación recibida.

ANEXO D: INSTALACIÓN DE POSTMAN.

Para la simulación de los agentes IoT y los médicos del sistema se ha utilizado Postman. Dicho componente va a permitir simular diversas peticiones a Wilma. La instalación de este servicio debe ser de la siguiente manera:

```
sudo apt install snapd
sudo snap install postman
```

El funcionamiento de Postman consiste en primero escribir la dirección web de la API a la que se desea acceder y el tipo de petición que se va a realizar. Luego, es importante incluir las cabeceras necesarias y el contenido del mensaje necesario para la petición de información. Una vez realizado esto, se enviará la petición al servidor de la API, en este caso sería PEP Proxy Wilma, el cuál la recibe y devuelve una respuesta al usuario, siendo esta mostrada en la interfaz de Postman.

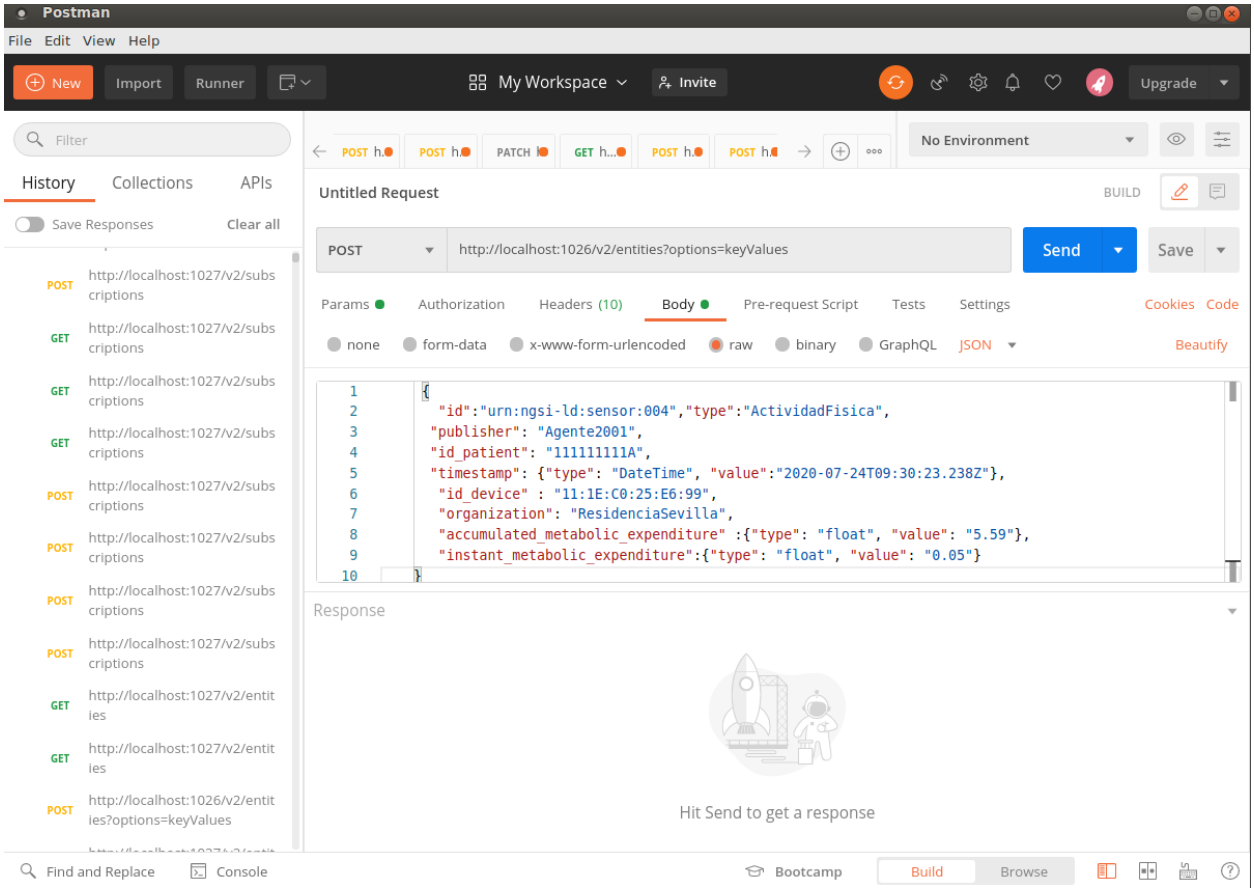

Figura 109. Interfaz gráfica de Postman.

REFERENCIAS

[1] Ciberseguridad y resiliencia para hospitales inteligentes. [En línea] Available: https://www.enisa.europa.eu/publications/cyber-security-and-resilience-for-smart-hospitals

[2] Página web oficial de FIWARE. [En línea] Available: https://www.fiware.org

[3] Lenguaje de marcado de control de acceso extensible. [En línea] Available http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[4] Orion Context Broker (OCB). [En línea] Available: https://fiwaretraining.readthedocs.io/es_MX/latest/ecosistemaFIWARE/ocb

[5] Identity Manager – Keyrock. [En línea] Available: https://fiware-idm.readthedocs.io/en/latest/#identity-manager-keyrock

[6] Wilma-PEP Proxy. [En línea] Available: https://fiware-academy.readthedocs.io/en/latest/security/wilma/index.html

[7] Authzforce. [En línea] Available: https://authzforce-ce-fiware.readthedocs.io/en/latest/

[8] Postman [En línea] Available: https://docs.postmanecho.com/?version=latest

[9] Docker [En línea] Available: https://www.redhat.com/es/topics/containers/what-is-docker

[10] Docker-Compose [En línea] Available: https://docs.docker.com/compose/

[11] MongoDB [En línea] Available: https://www.mongodb.com/es

[12] MySQL [En línea] Available: https://www.mysql.com/

[13] Repositorio GitHub para la instalación de los componentes y ayuda para entender el funcionamiento entre ellos. [En línea] Available: https://github.com/FIWARE/tutorials.Administrating-XACML

[14] Repositorio PEP-Proxy Wilma. [En línea] Available: https://github.com/FIWARE/tutorials.PEP-Proxy

[15] Repositorio para la instalación de Accumulator Server [En línea] Available: https://github.com/telefonicaid/fiware-orion/blob/master/scripts/accumulator-server.py