Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Gestor de consentimientos para el procesamiento de información sanitaria en el marco de la GDPR y conforme al estándar HL7 FHIR

Autor: Francisco José Ruiz Gómez

Tutor: Jorge Calvillo Arbizu

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

Sevilla, 2021

Proyecto Fin de Carrera
Grado en Ingeniería de las Tecnologías de Telecomunicación

Gestor de consentimientos para el procesamiento de información sanitaria en el marco de la GDPR y conforme al estándar HL7 FHIR

Autor:
Francisco José Ruiz Gómez

Tutor:
Jorge Calvillo Arbizu
Profesor Ayudante Doctor

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Gestor de consentimientos para el procesamiento de información sanitaria en el marco de la GDPR y conforme al estándar HL7 FHIR

Autor: Francisco José Ruiz Gómez

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, 2021

El Secretario del Tribunal

A mi familia. Ellos saben quiénes son.

Agradecimientos

Escribo estas líneas pensando que se trata del final de una larga etapa y no puedo evitar pensar en todo lo que me ha traído hasta este punto. No se trata del cierre de mi etapa universitaria, sino de un camino que empezó mucho antes, sin siquiera saber que lo estaba recorriendo, en el que he vivido todo tipo de momentos con una única constante, mi familia y mis amigos, todos aquellos con los que me crucé en algún momento y que por un motivo u otro, decidimos que sería mejor recorrerlo juntos.

En primer lugar, agradecer a mi padre y a mi madre, quienes siempre han estado ahí para lo que fuera, un consejo, un abrazo, una discusión, una broma... para absolutamente todo. Y por supuesto, a mi hermano, quien está para lo que necesite y con el que he compartido un sinfín de momentos.

En segundo lugar, agradecer a mis amigos de siempre: Iván, Josema, Javi, Antonio, Alberto, Álvaro y Sergio. En buenos o malos momentos, siempre los elegiría a ellos. No hemos estado juntos desde pequeños pero lo estaremos hasta que seamos viejos.

En tercer lugar, a la familia que me encontré en Sevilla. Igna, Patricio y Pablo, con quienes hice bromas un día y sin saber por qué se convirtieron en hermanos. Si hiciera otras cuatro carreras, querría hacer cada una de ellas con vosotros.

Por último y no por ello menos importante, gracias a todos los profesores que me han enseñado y ayudado a llegar a este momento, desde mis maestros de infantil hasta el último profesor de universidad. Y en especial, agradecer a Jorge Calvillo, mi tutor en este proyecto, quien confió en mí para este trabajo y me ha ayudado en todo momento.

Finalmente, quiero plasmar mis recuerdos en estas líneas: aquellas carreras mientras jugaba en el colegio, las bromas en el instituto, los ratos en la calle con amigos, los efímeros años de bachillerato, las tardes de trabajos en la universidad o el rato del bocadillo de papas con mayonesa. En todos estos momentos estaba con personas que de una forma u otra han contribuido en mi crecimiento personal y forman parte de mí. Gracias a todos.

Francisco José Ruiz Gómez

Sevilla, 2021

Resumen

Actualmente podemos encontrar numerosos y variados sistemas que emplean TIC (tecnologías de información y comunicaciones) en cada área de trabajo. Concretamente, en el sector sanitario, podemos ver que los datos sanitarios de un mismo individuo pueden estar almacenados en muchos sistemas (los cuales, en general, no comparten información) y usarse de forma diferente. Es decir, no existe un punto de intercambio entre sistemas que facilite el paso de los datos independientemente de la forma en la que estos datos hayan sido almacenados. Por ello, la comunicación en este ámbito suele ser compleja incluso sin entrar en la burocracia pertinente para el paso de datos entre distintas organizaciones sanitarias, por ejemplo. Además, para realizar cualquier acción sobre los datos sanitarios es necesario el consentimiento previo del paciente para ello de forma que estén restringidos y controlados tal y como indica el Reglamento General de Protección de Datos (GDPR). Existe la posibilidad de que cada sistema gestione los consentimientos aunque esto solo agravaría el problema de la interoperabilidad entre sistemas que usen información sanitaria. De esta forma, llegamos a la conclusión de que sería muy conveniente disponer de un único sistema con el que gestionar todos los consentimientos de los pacientes relacionados con información sanitaria. Así, cualquier acceso a estos datos necesitaría ser solicitado y aprobado por el ciudadano en un único sistema.

Para ello, tendríamos que tener en cuenta el Reglamento General de Protección de Datos (Reglamento 2016/679) por el que el Parlamento Europeo, el Consejo de la Unión Europea y la Comisión Europea tienen la intención de reforzar y unificar la protección de datos dentro de la Unión Europea (UE).

Por tanto, el objetivo de este trabajo es el diseño y desarrollo de un gestor de consentimientos que funcione bajo el marco del GDPR.

Para el almacenamiento de los consentimientos adoptaremos el último estándar desarrollado por Health Level Seven(HL7), empresa responsable de crear algunos de los protocolos de comunicación más utilizados hoy en día en el ámbito sanitario. Se trata del estándar Fast Healthcare Interoperability Resources, FHIR por sus siglas.

Dado el gran uso del teléfono móvil que hacemos hoy en día, consideramos que resultaría muy útil disponer de una aplicación móvil con la que el usuario pueda gestionar sus consentimientos médicos y recibir peticiones de tratamiento de sus datos sanitarios. Además, dada la pandemia que estamos viviendo, se hace muy conveniente el uso de una aplicación móvil para cualquier gestión para la que hasta hace nada requeríamos de papel. Todo ello nos da sustento a un trabajo que no podría estar más situado en el contexto actual.

Así, tendremos una aplicación móvil de la que un ciudadano o agente solicitante hará uso para gestionar estos consentimientos. A través del estándar FHIR, se hará uso de diferentes recursos recogidos en el estándar y aplicados en el ámbito sanitario para comunicarse con un servicio web el cual se encargará de obtener los datos necesarios de una base de datos y hacérselos llegar al usuario de la aplicación.

Finalmente, llegamos a la conclusión de que es necesario capturar qué ciudadano permite realizar una acción sobre determinados datos personales a un agente sanitario el cual solicita el consentimiento. Para ello, se usan un total de 5 recursos comprendidos en la especificación FHIR así como la API que proporciona conocida como HAPI FHIR. Tiene gran importancia el recurso *Consent* ya que será el que almacene la información presente en el consentimiento. Además de esto, el uso de la aplicación móvil proporciona una forma muy cómoda de gestión para el usuario y posibilita una continuación en el desarrollo del sistema. Por otro lado, el trabajo con los recursos FHIR puede llegar a complicarse ya que es necesario conocer la estructura de los mismos para poder trabajar con ellos correctamente. Además, el estándar usado se encuentra en continuo desarrollo por lo que la implementación conseguida podría verse obsoleta en un futuro.

Abstract

Nowadays, we can find numerous and varied systems that use ICT (information and communication technologies) in each area of work. Specifically, in the health sector, we can see that health data from the same individual can be stored in many systems (which, in general, do not share information) and used differently. In other words, there is no interchange point between systems to facilitate the passage of data regardless of how this data has been stored. Therefore, communication in this area is often complex even without entering into the relevant bureaucracy for the passage of data between different health organisations, for example. In addition, to carry out any action on health data, the prior consent of the patient is required in order to do so in such a way that they are restricted and controlled as indicated in the General Data Protection Regulation (GDPR). It is possible for each system to manage consents although this would only aggravate the problem of interoperability between systems using health information. In this way, we conclude that it would be highly desirable to have a single system with which to manage all patient consents related to health information. Thus, any access to this data would need to be requested and approved by the citizen in a single system.

To do so, we would have to take into account the General Data Protection Regulation (Regulation 2016/679) whereby the European Parliament, the Council of the European Union and the European Commission intend to strengthen and unify data protection within the European Union (EU).

Therefore, the objective of this work is the design and development of a consent manager that works under the GDPR framework.

For the storage of consents, we will adopt the latest standard developed by Health Level Seven (HL7), a company responsible for creating some of the most widely used communication protocols today in the healthcare field. This is the Fast Healthcare Interoperability Resources standard, FHIR.

Given the large use of the mobile phone that we make today, we consider that it would be very useful to have a mobile application with which the user can manage their medical consents and receive requests for treatment of their health data. In addition, given the pandemic we are experiencing, it is very convenient to use a mobile application for any management for which until now we did not require paper. All this gives us sustenance to a job that could no longer be situated in the current context.

Thus, we will have a mobile application that an applicant citizen or agent will use to manage these consents. Through the FHIR standard, different resources collected in the standard and applied in the healthcare field will be used to communicate with a web service which will be responsible for obtaining the necessary data from a database and making them available to the user of the application.

Finally, we conclude that it is necessary to capture which citizen allows an action on certain personal data to a health agent who requests the consent. For this, a total of 5 resources covered by the FHIR specification are used as well as the API it provides known as HAPI FHIR. The Consent resource is of great importance as it will be the one that stores the information present in the consent. In addition to this, the use of the mobile app provides a very comfortable form of management for the user and enables a continuation in the development of the system. On the other hand, the work with FHIR resources can become complicated as it is necessary to know the structure of the resources in order to work with them correctly. In addition, the standard used is in continuous development so that the implementation achieved could become obsolete in the future.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

GDPR	General Data Protection Regulation (Reglamento General de Protección de Datos)
FHIR	Fast Healthcare Interoperability Resources
HL7	Health Level Seven
UE	Unión Europea
AEPD	Agencia Española de Protección de Datos
STU	Standard for Trial Use
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
CASE	Computer Aided Software Engineering
POM	Project Object Model
API	Application Programming Interfaces
DAO	Data Access Object
CU	Caso de uso
TIC	Tecnologías de Información y Comunicaciones
SDK	Software Development Kit
DEX	Dalvik Executable

1 INTRODUCCIÓN

La grandeza nace de pequeños comienzos.

- Sir Francis Drake -

En este primer capítulo se abordará la motivación, el propósito y los objetivos que se pretenden alcanzar con la realización de este trabajo así como el ámbito, alcance y límites puestos en la investigación. Además, se mostrará la metodología empleada para el desarrollo del mismo y se expondrá el plan de trabajo seguido.

1.1 Motivación y objetivos

El GDPR tiene por objetivo dar control a los ciudadanos sobre sus datos personales que están en poder de empresas o terceros y simplificar el entorno regulador de los negocios internacionales unificando la regulación dentro de la UE [1]. De esta forma, confiere una serie de derechos y deberes sobre nuestros datos que podemos ejercer en cualquier momento, siempre que la entidad se adapte correctamente al marco. Así, los derechos que podemos encontrar en el GDPR [2] son:

- Derecho de acceso
- Derecho de rectificación
- Derecho de oposición
- Derecho de supresión("al olvido")
- Derecho a la limitación del tratamiento
- Derecho a la portabilidad
- Derecho a no ser objeto de decisiones individuales automatizadas
- Derecho de información
- Derechos Schengen

Hoy en día, los consentimientos en el ámbito sanitario suele ser recogidos cuando firmamos un determinado papel, lo cual puede resultarle tedioso al paciente. Además, junto a estos consentimientos se guarda numerosa información sobre el paciente que puede ser almacenada de forma indefinida sin su explícito consentimiento. A esto se suman otras situaciones indebidas como que ciertos agentes y organizaciones pueden acceder a datos de pacientes sin que estos sean conscientes y mucho menos lo hayan previamente permitido o que los datos sean traspasados a terceros. Todo esto es lo que el GDPR trata de evitar en el cualquier ámbito incluido el sanitario. A través de los derechos que expone, da la posibilidad al ciudadano de controlar los datos personales

que son almacenados o compartidos por otros. Por ello, es necesario establecerlo como marco regulatorio de los datos personales y es lo que se hará en el desarrollo del proyecto con el objetivo de gestionar de qué datos podrá disponer un agente a través de consentimientos previos.

Tras esto, definimos el objetivo global del proyecto: diseñar, desarrollar y desplegar un sistema que funcione como gestor de consentimientos bajo el GDPR y que use el estándar FHIR [3]. Este objetivo lo desglosamos en los siguientes:

- Comprensión del marco del GDPR para el tratamiento de datos personales
- Estudio del estándar FHIR para el almacenamiento y comunicación de información sanitaria
- Diseño de la correspondencia entre la GDPR y el estándar FHIR
- Diseño, desarrollo, implantación y pruebas de una aplicación móvil para la gestión de los consentimientos a través de su conexión con un servicio web.
- Diseño, desarrollo, implantación y pruebas de un servicio web que acceda a una base de datos FHIR para la extracción de información y haga de intermediario entre esta y la aplicación móvil.

En cuanto a FHIR, se trata de un estándar usado en el ámbito sanitario para el almacenamiento e intercambio de información sanitaria de amplio uso que evoluciona constantemente. Periódicamente se publican nuevas versiones del mismo ya que se trata de una especificación abierta. Entonces, trataremos de usar esta especificación en un gestor de consentimientos que se mueva bajo el GDPR, es decir, trabajaremos con unos consentimientos relacionados con el ámbito sanitario solicitados por unos usuarios (organizaciones sanitarias o profesionales) y aceptados, rechazados o pospuestos por otros (ciudadanos y pacientes). Esto permitirá gestionar los consentimientos para los que hasta ahora teníamos que pasar por el tedioso papeleo a través de un dispositivo móvil.

En la Figura 1-1 se muestra el esquema del sistema que vamos a diseñar, desarrollar y desplegar. Se trata de una aplicación móvil que enviará y recibirá información a un servicio web el cual se conectará con una base de datos FHIR para obtener o cargar la información de consentimientos.

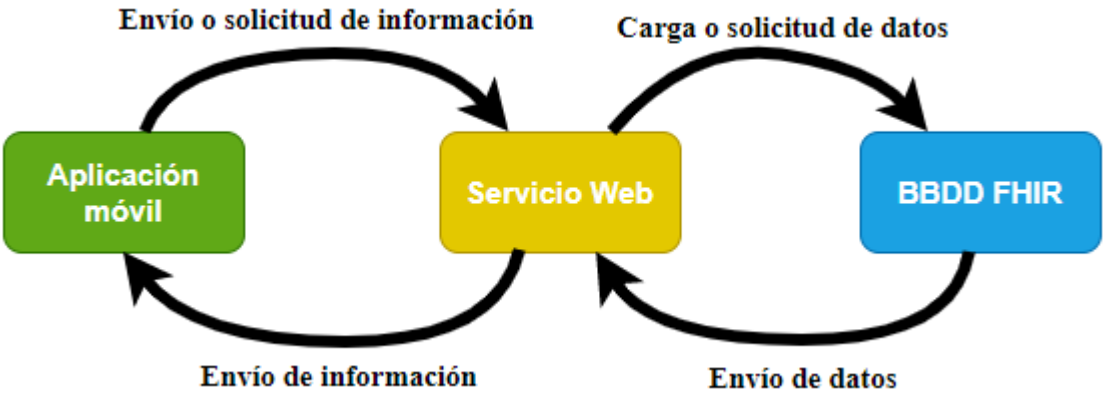

Figura 1-1. Funcionamiento general del sistema

1.2 Metodología empleada

Se propone una metodología incremental para la realización del trabajo. Esta metodología se caracteriza por evitar el compromiso total con los requisitos y permitir la incorporación de cambios. De esta forma, se crean versiones intermedias incrementales hasta que se finalice el proyecto. Por lo tanto, estamos apostando por la acomodación de los cambios y la realimentación al contar con estas versiones.

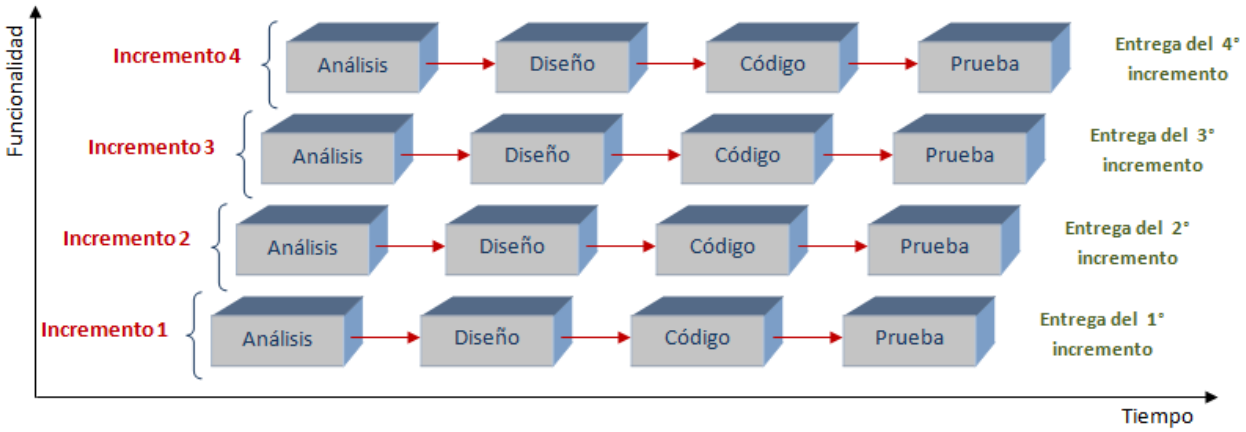

Figura 1-2. Metodología incremental

La retroalimentación nos permitirá aprovechar los conocimientos aprendidos y la experiencia frente a problemas que nos encontremos en cada fase de forma que nos sea más sencillo progresar en cada iteración. Además, detectaremos los errores en cada prueba. Con ello evitaremos acarrear estos problemas, trabajando con un código más limpio y enfrentarnos a algún error para el que nos pudiera complicar el encontrar su solución.

1.3 Plan de trabajo

Antes de comenzar con el diseño y desarrollo del sistema, es necesario realizar una serie de puntos relacionados con la comprensión de lo que queremos lograr:

1. Estudio y comprensión del marco del GDPR y los puntos que debemos tener en cuenta para no violar sus restricciones.
2. Estudio y comprensión del estándar FHIR y su API.
3. Estudio y modelado de la base de datos que queremos implementar para el proyecto.
4. Estudio y comprensión de los recursos FHIR ofrecidos por el estándar para cubrir los objetos de información que queremos usar.
5. Estudio de la conectividad entre la aplicación móvil y el servicio web a desarrollar, así como entre entre el servicio y la base de datos que queremos usar.

Los siguientes puntos responden a una metodología iterativa de tal forma que se han repetido en ciclos para la implemetación final de cada una de las funcionalidades que queremos lograr en el proyecto:

6. Diseño e implementación de los objetos de información o recursos FHIR en la base de datos necesarios para el trabajo.
7. Creación de clases y métodos en el servicio web tanto para la obtención de los datos de la base de

datos como para el envío de información a la aplicación móvil.

8. Creación de clases y métodos en la aplicación móvil tanto para el envío como la recepción de información intercambiada con el servicio web.
9. Implementación del código de los dos puntos anteriores.
10. Realización de pruebas y evaluación de los resultados obtenidos.
11. Corrección del código implementado si tras la evaluación se considerase oportuno.

Estas tareas han sido cuantificadas en un número de horas estimadas necesarias para completar cada punto que se muestran en la Tabla 1-1 junto a las horas que finalmente se necesitaron para la realización de ellas.

Tabla 1–1. Planificación temporal

Tareas	Horas estimadas	Horas reales
Estudio y comprensión del marco del GDPR y los puntos que debemos tener en cuenta para no violar sus restricciones	10	8
Estudio y comprensión del estándar FHIR y su API	20	16
Estudio y modelado de la base de datos que queremos implementar para el proyecto	5	5
Estudio y comprensión de los recursos FHIR ofrecidos por el estándar para cubrir los objetos de información que queremos usar	5	6
Estudio de la conectividad entre la aplicación móvil y el servicio web a desarrollar, así como entre entre el servicio y la base de datos que queremos usar	8	6
Diseño e implementación de los objetos de información o recursos FHIR en la base de datos necesarios para el trabajo	5	4
Creación de clases y métodos en el servicio web tanto para la obtención de los datos de la base de datos como para el envío de información a la aplicación móvil	5	4

Implementación del código de los dos puntos anteriores	60	75
Realización de pruebas y evaluación de los resultados obtenidos	50	50
Corrección del código implementado si tras la evaluación se considerase oportuno	100	110
Total de horas	**268 horas estimadas**	**284 horas reales**

2 ESTADO DEL ARTE

El placer más noble es el júbilo de comprender.

- Leonardo da Vinci -

A continuación, vamos a situar el contexto sanitario y técnico así como el reglamento que tenemos que respetar en la realización del trabajo. Abordaremos los derechos que marca el GDPR ya que debemos tenerlos en cuenta para el diseño e implementación del sistema. Además, se tratará la interoperabilidad en el ámbito sanitario y el estándar FHIR.

2.1. El marco del GDPR

El Reglamento General de Protección de Datos (GDPR) (Reglamento 2016/679) es un reglamento por el que el Parlamento Europeo, el Consejo de la Unión Europea y la Comisión Europea tienen la intención de reforzar y unificar la protección de datos dentro de la Unión Europea (UE).

En primer lugar, hay que señalar que el marco del GDPR sobre el que nos situamos comprende únicamente a los datos personales. Por ello, definimos lo que es un dato personal. Tal y como señala en el artículo 4 del GDPR [3], se entiende por "dato personal" toda la información sobre una persona física identificada o identificable, considerando identificable toda persona cuya identidad pueda determinarse, directa o indirectamente, mediante un identificador, sea un nombre un dato sobre la localización. De esta forma, el GDPR expone una serie de derechos que los ciudadanos pueden ejercer ante el responsable del tratamiento de sus datos personales. Estos tienen una serie de características en común:

- El plazo estipulado para contestar a las solicitudes es de un mes. Sin embargo, si la complejidad y la cantidad de solicitudes lo impiden, se puede ampliar 2 meses más.
- El responsable tiene el deber de informarte sobre los medios para hacer uso de tus derechos. Además, estos medios tienen que ser accesibles y no se le pueden negar al ciudadano aunque decida usar otros medios.
- Si la solicitud se realiza electrónicamente la información se aportará por el mismo medio a menos de que el solicitante pida que se haga por otro medio.

En las siguientes subsecciones veremos en detalle cada uno de los derechos que el ciudadano puede ejercer.

2.1.1 Derecho de acceso

Es el derecho por el cual el ciudadano puede dirigirse al responsable del tratamiento para saber si se están usando sus datos o no. En caso afirmativo, puede pedir la siguiente información:

- Una copia de los datos personales que se están tratando.
- El objetivo de por el cual se van usar los datos.
- La categoría de los datos.
- A quién se le han comunicado o se le van a comunicar los datos personales, haciendo énfasis en organizaciones internacionales o países terceros.
- El tiempo previsto durante el cual se van a conservar los datos. Si no fuera posible, cuál es el criterio usado en el cálculo de este periodo.
- El derecho a: la rectificación o supresión de los datos personales, así como la limitación del tratamiento o la oposición a tal tratamiento.
- El derecho a presentar una reclamación a la Autoridad de Control.
- El origen de los datos en caso de que no se haya obtenido directamente del ciudadano.
- La existencia de decisiones tomadas de forma automática así como la elaboración de un perfil en base a tales decisiones. En este caso, también se puede solicitar información sobre la lógica aplicada, la importancia y las consecuencias previstas del tratamiento para el solicitante. La elaboración de perfiles se define como cualquier forma de tratamiento de datos que se centre en evaluar aspectos personales para analizar o predecir temas concretos como el rendimiento laboral, el estado de salud o los intereses personales, entre otros.
- Si se transfieren datos a terceros países u organizaciones internacionales, está en tu derecho ser informado de las garantías adecuadas sobre las que se realiza la transferencia.

2.1.2 Derecho de rectificación

A través de este derecho se puede conseguir la rectificación de los datos personales que sean incorrectos inmediatamente. Además, dependiendo del objetivo del tratamiento, también se tiene derecho a tener los datos completos. Esto quiere decir que si no estuvieran completos deberán completarse incluso a través de una declaración adicional.

Es necesario señalar los datos que se quieren corregir y cómo. Si fuera necesario, esta solicitud tendrá que ir acompañada de un justificante del error o inexactitud de tus datos.

2.1.3 Derecho de oposición

Tal y como podemos imaginar, este derecho consiste en negarse a que los datos personales sean usados. Sin embargo, este derecho solo puede ejercerse en las siguientes situaciones:

- Si el objetivo del tratamiento tiene un interés público o legítimo incluida la elaboración de perfiles, el responsable no podrá tratar los datos excepto si acredita tener motivos de peso que prevalezcan sobre las libertades, intereses y derechos del solicitante o en caso de formulación, la defensa o ejercicio de reclamaciones.
- Si el objetivo del tratamiento es la mercadotecnia directa incluida la elaboración de perfiles, los datos no serán usados independientemente de los motivos del responsable.

2.1.4 Derecho de supresión

Se podrá hacer uso de este derecho cuando se dé alguna de las siguientes situaciones:

- Los datos personales ya no son necesarios para el objetivo por el que se tomaron en un primer momento.
- Si para el uso de los consentimientos el responsable se basó en el consentimiento del ciudadano y este es retirado teniendo que cuenta que el tratamiento de los datos no se apoye en otro motivo legítimo.
- Si la oposición se debe a alguna de las siguientes circunstancias:
    - El interés del tratamiento era legítimo o público y no se han mantenido otros motivos para reclamar el uso de los datos.
    - Los datos personales son objetivo de mercadotecnia directa incluyendo la elaboración de perfiles.
- Los datos han sido usados ilícitamente.
- La supresión de los datos se solicita para cumplir con una obligación legal establecida en el Derecho de la Unión o de los Estados miembros que se aplique al responsable.
- Si la obtención de los datos se ha logrado a través de la oferta de servicios de la sociedad de la información.

El GDPR conecta este derecho con el "derecho al olvido" por el cual si un responsable filtra al exterior los datos personales está en la obligación de informar a otros usuarios que estén haciendo uso de los datos de que deben eliminar los datos y toda conexión a ellos. Sin embargo, no siempre se puede ejercer este derecho. No se podrá hacer uso de él si el tratamiento de los datos es indispensable para la libertad de expresión e información, el cumplimiento de una obligación legal o un fin con interés público, el ejercicio de poderes públicos que ostenta el responsable, por razones de interés público, en el ámbito de la salud pública o con un objetivo de archivo de interés público, investigación científica o histórica o fines estadísticos, o para la formulación, el ejercicio o la defensa de reclamaciones.

2.1.5 Derecho a la limitación del tratamiento

Mediante este derecho se puede limitar el tratamiento de los datos. Cabe destacar las dos vertientes que tiene:

- Se puede solicitar la suspensión del tratamiento:
    - Cuando se impugnen los datos personales y su verificación se pueda hacer en un plazo determinado.
    - Cuando haya oposición al tratamiento que el responsable quiere realizar bajo un interés legítimo o público mientras este verifica que sus motivos prevalecen sobre los del ciudadano.
- Se puede solicitar la conservación de los datos:
    - Cuando el tratamiento sea ilícito, haya oposición a la supresión de los datos y en su lugar se solicita la limitación de su uso.
    - Cuando ya no se necesiten los datos personales para cumplir el objetivo por el cual que han usado pero el ciudadano los solicite para la formulación, el ejercicio o la defensa de reclamaciones.

2.1.6 Derecho a la portabilidad

A través de este derecho se consigue que, cuando un tratamiento se realice mediante automatismos, el control sobre los datos personales sea reforzado. Así, el interesado recibirá sus datos con un formato estructurado para la lectura mecánica e interoperable y de uso común. Además, se podrá transmitir a otro responsable si el tratamiento de los datos se fundamenta en el consentimiento o en el marco de la ejecución de un contrato.

Sin embargo, este derecho no es ejecutable cuando el tratamiento tenga un interés público o sea indispensable para el ejercicio de poderes públicos que ostente el responsable.

2.1.7 Derecho a no ser objeto de decisiones individuales automatizadas

Con este derecho se evita ser objeto de decisiones basadas en el tratamiento de los datos que tengan efectos jurídicos sobre el interesado o que le afecte de forma importante. La elaboración de perfiles queda incluida como fundamento de estas decisiones. Sin embargo, existen situaciones en las que este derecho no puede ejercerse. Estas son:

- Cuando la decisión automática sea necesaria para la realización de un contrato entre el responsable y el ciudadano
- Cuando el uso de los datos se base en un consentimiento otorgado por el ciudadano

Hay que destacar que para estos dos supuestos el responsable debe garantizar que el ciudadano pueda expresar su punto de vista, impugnar la decisión y conseguir la intervención humana.

2.1.8 Derecho de información

El responsable del tratamiento debe tener en cuenta este derecho a la hora de recopilar los datos personales. Para cumplir con él, el GDPR recomienda usar dos capas en la solicitud de la información de tal forma que:

- En la 1ª capa, se da información básica de manera resumida cuando se obtienen los datos personales mediante el mismo medio y en el mismo momento.
- En la 2ª capa, se da el resto de la información a través del medio más adecuado para la compresión y presentación.

Así, la información que se solicitaría por capas sería:

- **1ª capa:**
    - Quién es el responsable
    - Descripción resumida de los objetivos que se persiguen con el tratamiento de datos
    - Base jurídica
    - Previsiones de cesiones y transferencias a otros países, si hubiera.
    - Referencia a los derechos del ciudadano

- **2ª capa:**
    - Datos de contacto del responsable así como la identidad y los datos del representante y los del delegado de protección de datos si existiesen.
    - Descripción ampliada de los objetivos que se persiguen con el tratamiento de datos además de los plazos o criterios para la conservación de los datos. También se deben señalar las decisiones automatizadas, los perfiles y la lógica aplicada.
    - Base jurídica en detalle para los casos de interés público, legítimo o de obligación legal. Además, habrá que señalar si es obligatorio facilitar los datos y las consecuencias de no hacerlo.
    - Destinatario o destinatarios. Además, habrá que señalar qué decisiones de adecuación, garantías, normas corporativas vinculantes o situaciones específicas se pueden aplicar.
    - Cómo se pueden ejercer todos los derechos mencionados en las anteriores subsecciones además del derecho a retirar un consentimiento otorgado y el derecho a reclamar ante la Autoridad de Control.

Por otro lado, si los datos no se han obtenidos directamente del ciudadano, se le indicará la siguiente información además de las mencionadas en las capas:

- **1ª capa:**
    - El origen del que han obtenido los datos personales.

- **2ª capa:**
    - Información detallada de la fuente de los datos
    - Categoría de los datos que se traten

Toda esta información debe aportarse en un plazo razonable como puede ser un mes excepto si:

- Los datos personales deben usarse para comunicarse con el afectado, hasta el primer contacto.
- Los datos van a ser comunicados a otro interesado, hasta el primer contacto.

2.2 Interoperabilidad en el ámbito sanitario

Según la Real Academia Española (RAE), el término “interoperabilidad” se define como la “habilidad de dos o más sistemas o de sus componentes para utilizarse de forma conjunta e intercambiable” [5]. De esta forma, podemos discernir que el concepto de interoperabilidad en el ámbito sanitario consiste en la capacidad de que distintos sistemas sanitarios trabajen de forma conjunta intercambiando información entre ellos.

Dicho esto, podemos usar la siguiente definición formal de “interoperabilidad en el ámbito sanitario”: capacidad de los sistemas de información y de los procedimientos a los que éstos dan soporte, de compartir datos y posibilitar el intercambio de información y conocimiento entre ellos [6]. Esta interoperabilidad se hace muy necesaria hoy en día ya que cada organización puede tener sus sistemas implementados de muchas

formas distintas por lo que se complica una posible comunicación entre ellos. Por ello, surge la necesidad de un marco de referencia para esta información de forma que exista un punto en común para hacer posible esta comunicación. Como solución tenemos los estándares. Estos estándares nos darán un denominador común para los componentes usados en el ámbito sanitario y para la forma de transmitirlos de modo que sean útiles para terceros.

Existen tres grupos de estándares de interoperabilidad dependiendo del objetivo. Son los estándares de mensajería, terminología y documentos. Nosotros nos vamos a centrar en uno en concreto del grupo de mensajería: el estándar FHIR. Este hereda las mejores características de sus predecesores: HL7 V2 [7], HL7 V3 [8] y CDA [9].

2.2.1 El estándar FHIR

Se trata de un estándar de interoperabilidad creado para facilitar el intercambio de información sanitaria entre proveedores de atención sanitaria, pacientes, cuidadores, pagadores, investigadores y cualquier otro agente involucrado en el ecosistema sanitario. Se compone de dos partes: un modelo de contenido en forma de recursos y una especificación para el intercambio de estos recursos en forma de interfaces RESTful en tiempo real, así como mensajes y documentos [10]. Actualmente, la versión más reciente que tenemos es la R4. Esta especificación es una mezcla entre el estándar de nivel "Normative" y "Standard for Trial Use" (STU).

Por otro lado, FHIR se describe como una especificación "RESTful" ya que usa el protocolo REST para la comunicación. De esta forma, se usarán mensajes HTTP request/response para el envío y recepción de información la cual se transmitirá dentro de estructuras JSON o XML. Además, la API RESTful describe un conjunto de operaciones conocidas como interacciones para manejar los diferentes recursos. En particular, destaca el uso de la interacción *Search* ya que esta permitirá buscar el recurso que queramos [11].

En cuanto a la seguridad de FHIR, hay que señalar que no se trata de un protocolo de seguridad ni define ninguna funcionalidad relacionada con la seguridad. No obstante, sí define una serie de protocolos de intercambio y modelos de contenido que necesitan ser utilizados con varios protocolos para un uso seguro. De manera resumida, indica los siguientes puntos [12]:

- Relojes sincronizados usando NTP/SNTP
- El intercambio de datos debe cifrado usando TLS
- Usuarios o clientes deben autenticarse
- Control de autorización/acceso
- Uso de firmas digitales
- Cuidado con los archivos adjuntos
- Permite el uso de etiquetas relacionadas con la seguridad
- Políticas de gestión de datos
- Cuidado con el uso de los elementos narrativos
- Se deben validar todas las entradas recibidas

Para una correcta comprensión de FHIR, debemos describir varios conceptos sobre lo que se basa la especificación como son los recursos, dominio de recursos y extensiones.

2.2.1.1 El recurso FHIR

La especificación FHIR trabaja con lo que denominamos recurso. Se trata de una entidad que tiene un identificador conocido, identifica un concepto del ámbito sanitario definido por la especificación, contiene un conjunto de datos estructurados y tiene una versión identificada. Por lo tanto, el recurso será el objeto usado para el intercambio y almacenamiento de los datos con el objetivo de tratar un amplio rango de problemas relacionados con la salud tanto para el ámbito clínico como para el administrativo.

Todos los recursos comparten una serie de elementos y propiedades. Estos son:

- Un identificador con el que podremos recuperar el recurso a posteriori. Este podrá ser una "Uniform Resource Locator" (URL) que cambiará a medida que cambie la localización del recurso o podrá formar parte del mismo recurso de tal forma que no cambiará.
- Metadatos para describir al propio recurso.
- Puede tener un elemento de lenguaje que especifica el idioma que tendrán los datos del recurso. Si este se usa, también deberá especificarse en el elemento narrativo.
- Una referencia a una guía de implementación que describa cómo se está usando el recurso.

La Figura 2-1 muestra los campos comunes a los recursos FHIR.

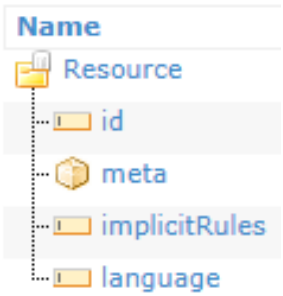

Figura 2-1. Recurso FHIR

Además de los elementos anteriores, los recursos poseen otros elementos que almacenan la información. Cada tipo de recurso tendrá una serie de campos que aportarán la información por la que usamos un recurso determinado. Por ejemplo, tendremos los recursos *Patient* o *Consent* para la información relacionada con un paciente y la relacionada con un consentimiento respectivamente.

Estos campos se podrán completar con un tipo de dato concreto [13]. La especificación define 4 categorías en las que clasifica estos tipos:

- Simples o primitivos, como string o integer
- Complejos, como CodeableConcept o ContactPoint
- Metadatos, como Contributor o UsageContext

- De uso especial, como Reference o Extension

En la Figura 2-2 se pueden observar los tipos de cada categoría:

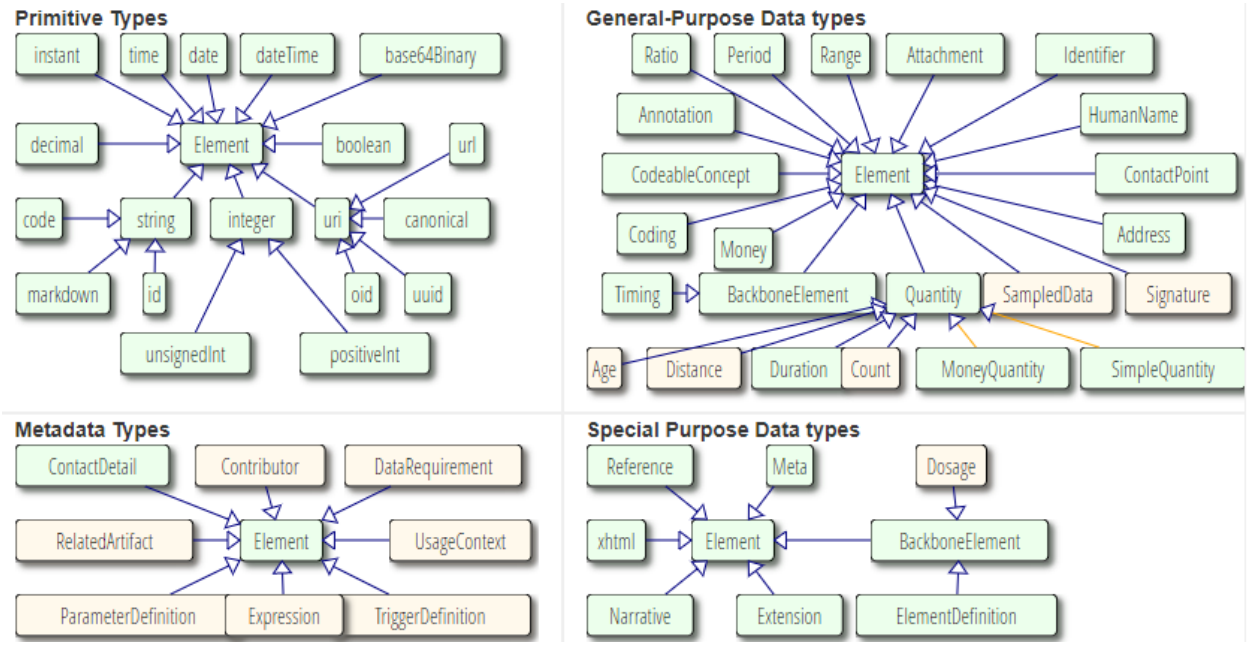

Figura 2-2. Tipos de datos en los recursos FHIR

También es muy interesante conocer los dominios de recursos. Se trata de un recurso que tiene una representación legible en formato "extensible hypertext markup language" (XHTML) del contenido del recurso. Además, puede contener otros recursos relacionados y tener extensiones. Realmente, este recurso es una extensión del recurso base [14]. Cabe señalar que todos los recursos que usaremos son de este tipo.

2.2.1.2 Extensiones de recursos

El intercambio de recursos conforme a la especificación se basa en requisitos comunes que han sido acordados en todo el sistema sanitario. Sin embargo, es habitual usar otros requisitos que se escapan a esos acuerdos previos. Por lo tanto, sería necesario implementarlos y esto complicaría enormemente su uso ya que habría que implementar todos los requisitos que fueran apareciendo. Como solución, estos requisitos se implementarán a través de las extensiones [15].

Una extensión no es más que un elemento hijo de un recurso que añade información al propio recurso. De esta forma, cada implementador podrá añadir la información que necesite a los recursos que use sin necesidad de cambiar la especificación.

En la Figura 2-3 vemos un recurso *Patient* con extensiones en formato JavaScript Object Notation (JSON) usado en el proyecto.

```
{\n
  "resourceType": "Patient",\n
  "extension": [ {\n
    "url": "localhost:8080/extensión/patient/usuario",\n
    "valueString": "usuario1"\n
  }, {\n
    "url": "localhost:8080/extensión/patient/tarjsanitaria",\n
    "valueInteger": 1\n
  } ],\n
  "modifierExtension": [ {\n
    "url": "localhost:8080/extensión/patient/codmensaje",\n
    "valueInteger": 200\n
  } ],\n
  "identifier": [ {\n
    "id": "01234567A",\n
    "system": "http://localhost:8080/TFGREST/ciud/01234567A"\n
  } ],\n
  "name": [ {\n
    "text": "Usuario"\n
  } ],\n
  "telecom": [ {\n
    "system": "phone",\n
    "value": "123456789",\n
    "use": "mobile"\n
  } ]\n
}
```


Figura 2-3. Recurso *Patient* con extensiones

Podemos observar que tenemos dos tipos de extensiones: *"extension"* y "*modifierExtension"*. La primera de ella es la extensión tal cual la hemos explicado anteriormente. En cambio, la segunda es un tipo de extensión que puede cambiar el significado del recurso en sí. También usaremos este tipo de extensión.

2.2.1.3 Recursos usados

Tras abordar el estándar FHIR y estudiar los recursos, concluimos que para el gestor de consentimientos necesitaremos hacer uso de tres tipos de recursos: *Patient*, *Practictioner* y *Consent*.

A. Recurso Patient

Se trata del recurso que identifica a un paciente tal y como su nombre indica. Contiene información relacionada con el ciudadano [16]. Por ello, será usado cuando haya que enviar información de un ciudadano. En la Figura 2-4 tenemos la estructura del recurso con todos los campos de información que el estándar aporta.

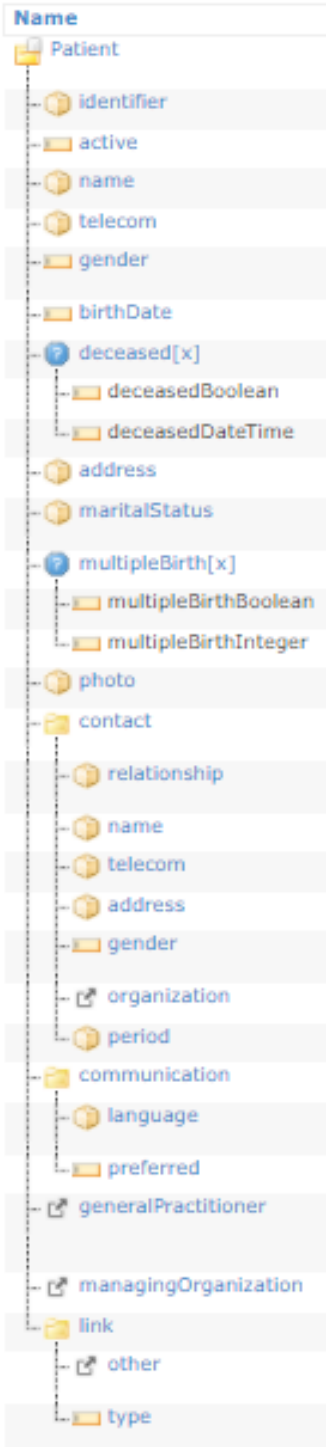

Figura 2-4. Estructura del recurso Patient

Como se puede ver, tenemos campos como *name*, para el nombre del paciente; *gender*, para el género del paciente; o *contact* para identificar un contacto del paciente entre otros. Sin embargo, no vamos a usar todos estos campos, solo los que nos resulten útiles para alcanzar nuestro objetivo (así como los que sean obligatorios). Además, usaremos extensiones y modificadores de extensión para añadir información adicional que no se contemple en el estándar.

A destacar el campo *identifier* que usaremos para almacenar el DNI del ciudadano e identificarlo.

B. Recurso Practitioner

Este recurso cubre a todas las personas que están involucradas en la atención sanitaria y sus servicios relacionados. Comprende desde dentistas o dietistas hasta médicos o enfermeros [17]. Será usado cuando haya que enviar información sobre el agente sanitario que solicitará los consentimientos. En la Figura 2-6 se muestra la estructura del recurso.

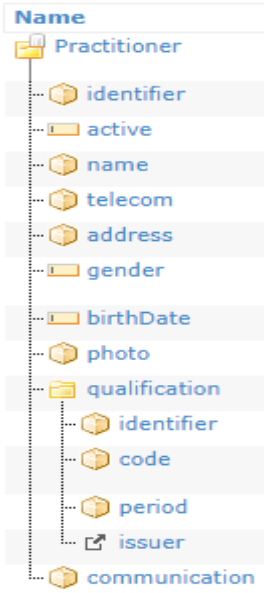

Figura 2-6. Estructura del recurso Practitioner

Tenemos varios campos que también hemos visto en el recurso *Patient* y otros como el campo *qualification* para indicar la cualificación del agente sanitario. Al igual que en el caso del paciente, solo usaremos los campos del recurso que nos son útiles.

En este recurso usamos el campo *identifier* para almacenar la contraseña que usa el agente solicitante para acceder al gestor de consentimientos. Además, podremos usarla para identificarlo.

C. Recurso Organization

Este recurso comprende a un grupo de personas u organizaciones que tienen un objetivo común. Puede ser utilizado para compartir información de varias organizaciones o simplemente como apoyo para otros recursos [18]. En nuestro caso, será usado para recoger la información de la organización a la que pertenece el agente sanitario. En la Figura 2-7 se muestra su estructura.

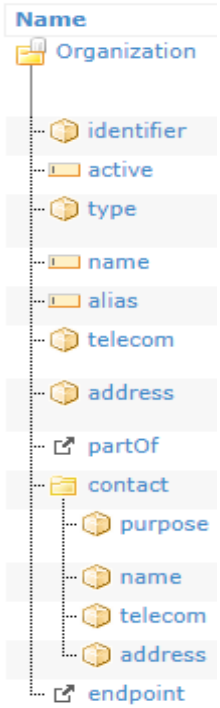

Figura 2-7. Estructura del recurso Organization

Tenemos varios campos en los que podemos almacenar información. Por ejemplo, en *alias* podremos guardar un nombre con el que llamamos comúnmente a la organización o en *address* podremos encontrar la dirección de la misma

D. Recurso PractitionerRole

Este recurso describe el rol que una persona involucrada en la atención sanitaria o sus servicios relacionados realiza para una organización [19]. Nos servirá para identificar a qué departamento pertenece esta persona. Además, gracias a sus campos, podremos relacionar esta información con la propia persona y con la organización a la que pertenece. En la Figura 2-8 vemos su estructura:

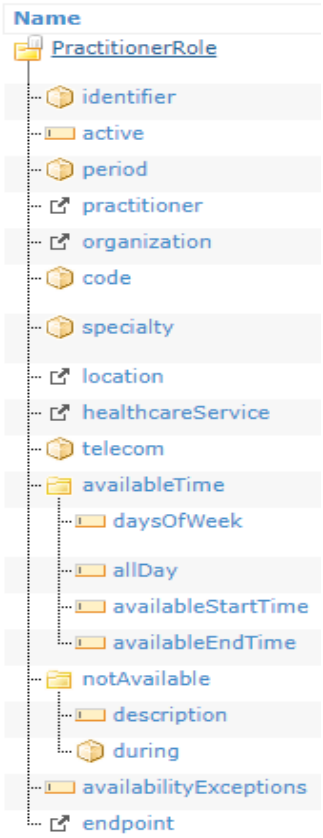

Figura 2-8. Estructura del recurso PractitionerRole

Con los campos *practitioner* y *organization* haremos referencia a esos recursos. Además, tenemos otros campos como *code* que indicará qué rol tiene el profesional o *specialty* que concreta su especialidad.

E. Recurso Consent

El agente solicitante solicitará el consentimiento de los ciudadanos para poder acceder a los datos. Para esto, se usará el recurso *Consent* ya que es el idóneo para lograr este objetivo debido a sus campos [18]. En las Figuras 2-9 y 2-10 tenemos la estructura del recurso.

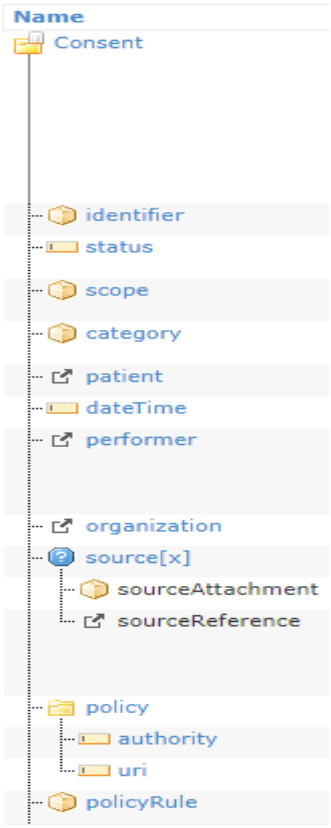

Figura 2-9. Estructura del recurso Consent I

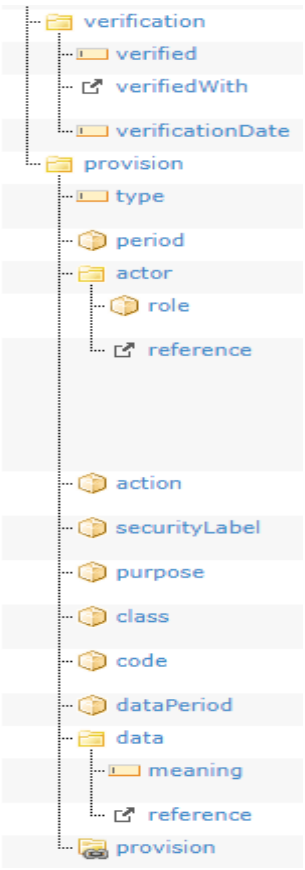

Figura 2-10. Estructura del recurso Consent II

Llegado este punto, tenemos que destacar el campo *patient*. Este es de un tipo especial de datos: Reference [19]. Se caracteriza porque, como indica su nombre, es una referencia a otro recurso. De esta forma, el campo *patient* hará referencia al paciente al cual se le pidió el consentimiento. Lo mismo ocurrirá con los campos *performer* u *organization* entre otros. También tenemos otros campos como *status* para señalar el estado del consentimiento o *action* dentro de *provision* para indicar qué se quiere hacer con los datos una vez el consentimiento sea otorgado por ejemplo.

Para este recurso hemos usado las extensiones para cumplir con el GDPR. Como ya señalamos en los derechos que confiere el mismo, en la solicitud de los consentimientos se deben añadir ciertos campos para no violar el marco regulatorio como durante cuánto tiempo van a usarse los datos privados del ciudadano.

2.3 Materiales

Para el diseño e implementación del gestor de consentimientos hemos usado varias herramientas distintas. Cada una de ellas se describe en las siguientes subsecciones.

2.3.1 Eclipse 2019-12

Se trata de un entorno de desarrollo integrado compuesto por un conjunto de herramientas de programación de código abierto [15]. Se ha usado para desarrollar el servicio web así como para desplegar un servidor Tomcat. Además, hemos usado los frameworks de **Spring** [21] e **Hibernate** [22] para facilitar el desarrollo. En la Figura 2-12 vemos el entorno:

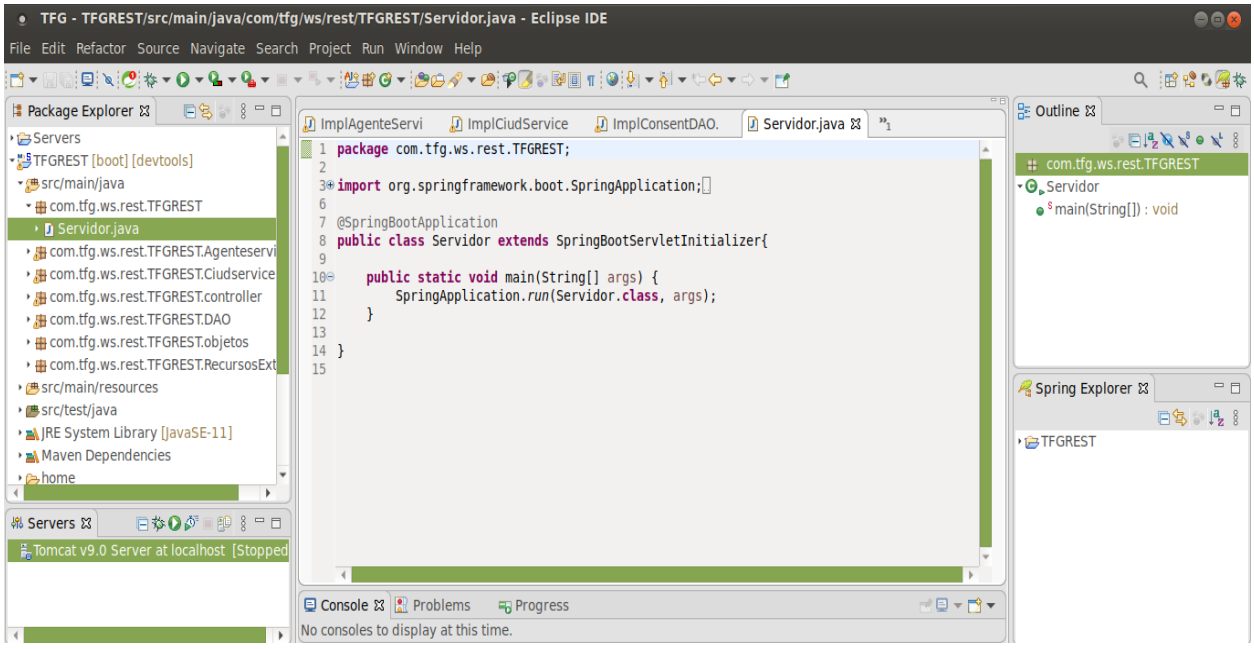

Figura 2-12. Entorno de desarrollo integrado Eclipse

2.3.2 Apache Tomcat v9

Tipo de servidor usado para el despliegue de aplicaciones desarrolladas en Java. Se despliega el servicio web en él a través de Eclipse [23]. En la Figura 2-13 vemos su ubicación en Eclipse donde podremos arrancarlo, reiniciarlo o pararlo cuando queramos:

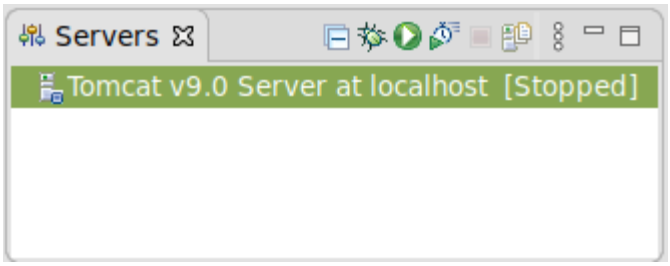

Figura 2-13. Servidor Tomcat en Eclipse

2.3.3 Apache Maven

Se trata de una herramienta Computer Aided Software Engineering (CASE) usada en la gestión de proyectos de software y para la comprensión basada en un *Project Object Model* (POM) [24]. Con esta herramienta describimos el proyecto y añadimos dependencias y plugins de forma sencilla gracias a su motor de búsqueda. Este motor obtiene los softwares de un repositorio en el cual podemos encontrar varias versiones de ellos.

2.3.4 PostgreSQL

Es un sistema de gestión de bases de datos relacionales orientado a objetos y de código abierto [25]. Se ha usado para implementar la base de datos que almacena información de los agentes y los ciudadanos así como para realizar consultas tanto en fase de diseño como de implementación (Figura 2-14).

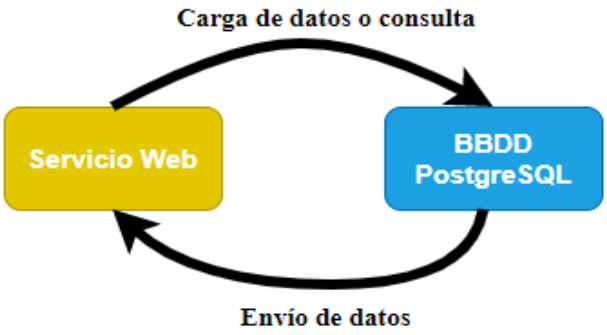

Figura 2-14. Funcionamiento entre el servicio y la base de datos de PostgreSQL

2.3.5 Magic Draw UML

Es una herramienta CASE compatible con el estándar UML [26]. Nos ha servido para modelar todo el servicio web, desde los objetos usados hasta la funcionalidad conseguida.

2.3.6 Android Studio

Herramienta usada en el desarrollo de aplicaciones Android [27]. Con ella se ha realizado la implementación de un cliente Android FHIR que contactará con el servicio web.

Figura 2-15. Funcionamiento entre la aplicación móvil y el servicio web

2.3.7 HAPI FHIR

Application Programming Interfaces (API) de código abierto usada para la implementación de FHIR [28]. La hemos implementado tanto en el servicio web como en la aplicación móvil desarrollada. A destacar que Android tiene dependencias FHIR para facilitar el uso de la especificación.

2.3.8 HAPI FHIR Public Test Server

La API usada también aporta un servidor público para pruebas así como una base de datos FHIR pública [29]. Se ha usado para hacer pruebas con los consentimientos. En cuando a la base de datos, como queremos que cualquier agente solicitante pueda acceder a los consentimientos, no podrá ser de uso privado. Por ello, usaremos esta para almacenar los consentimientos de forma que la gestión sea de forma abierta e interoperable gracias al estándar. En la Figura 2-16 vemos el funcionamiento en conjunto con el servicio web.

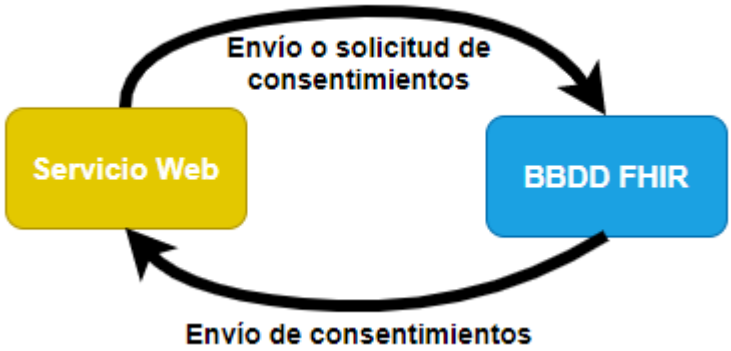

Figura 2-16. Funcionamiento entre el servicio web y la base de datos FHIR

3 RESULTADOS

La ciencia puede divertirnos y fascinarnos, pero es la Ingeniería la que cambia el mundo.

- Isaac Asimov -

En este capítulo veremos los resultados obtenidos en la realización del trabajo. Empezaremos el modelado de la base de datos que mantiene a nuestros pacientes y responsables y continuaremos con una visión detallada de los recursos empleados indicando qué campos se han utilizado y las extensiones añadidas. Después, se explicarán los patrones usados en el proyecto y veremos los casos de uso que se darán en el sistema. Finalmente, veremos a nivel de código cómo se ha implementado el servicio web y la aplicación móvil para logar el correcto funcionamiento del sistema.

3.1 Casos de uso

El sistema presenta una serie de casos de uso (CU) que hemos identificado. En estos casos de uso tenemos dos posibles actores como ya hemos comentado. Estos serían el agente solicitante y el ciudadano los cuales serán usuarios de la aplicación móvil. En la Figura 3-1 tenemos una representación de ambos actores.

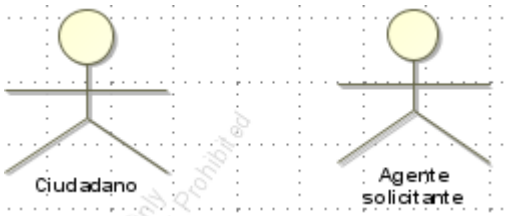

Figura 3-1. Actores en los casos de uso

En las siguientes tablas tenemos una breve descripción de cada usuario.

Tabla 3–1. Usuario Ciudadano

Actor-01	Ciudadano
Descripción	Será el usuario que reciba las solicitudes de los consentimientos
Comentarios	Podrá aceptar, rechazar, revocar o incluso dejar pendientes tales solicitudes. También podrá consultar los consentimientos aceptados y rechazados previamente.

Tabla 3–2. Usuario Agente Solicitante

Actor-02	Agente solicitante
Descripción	Será el usuario que pida el consentimiento a los ciudadanos y vea su estado
Comentarios	Antes de ello, debe rellenar tal solicitud

Cada actor tiene unos casos de uso ya que, como venimos comentando, dispondrá de diferentes funciones dependiendo del rol con el que accedan. Por ello, en la Figura 3-2 vemos diferentes casos de uso asociados de los actores.

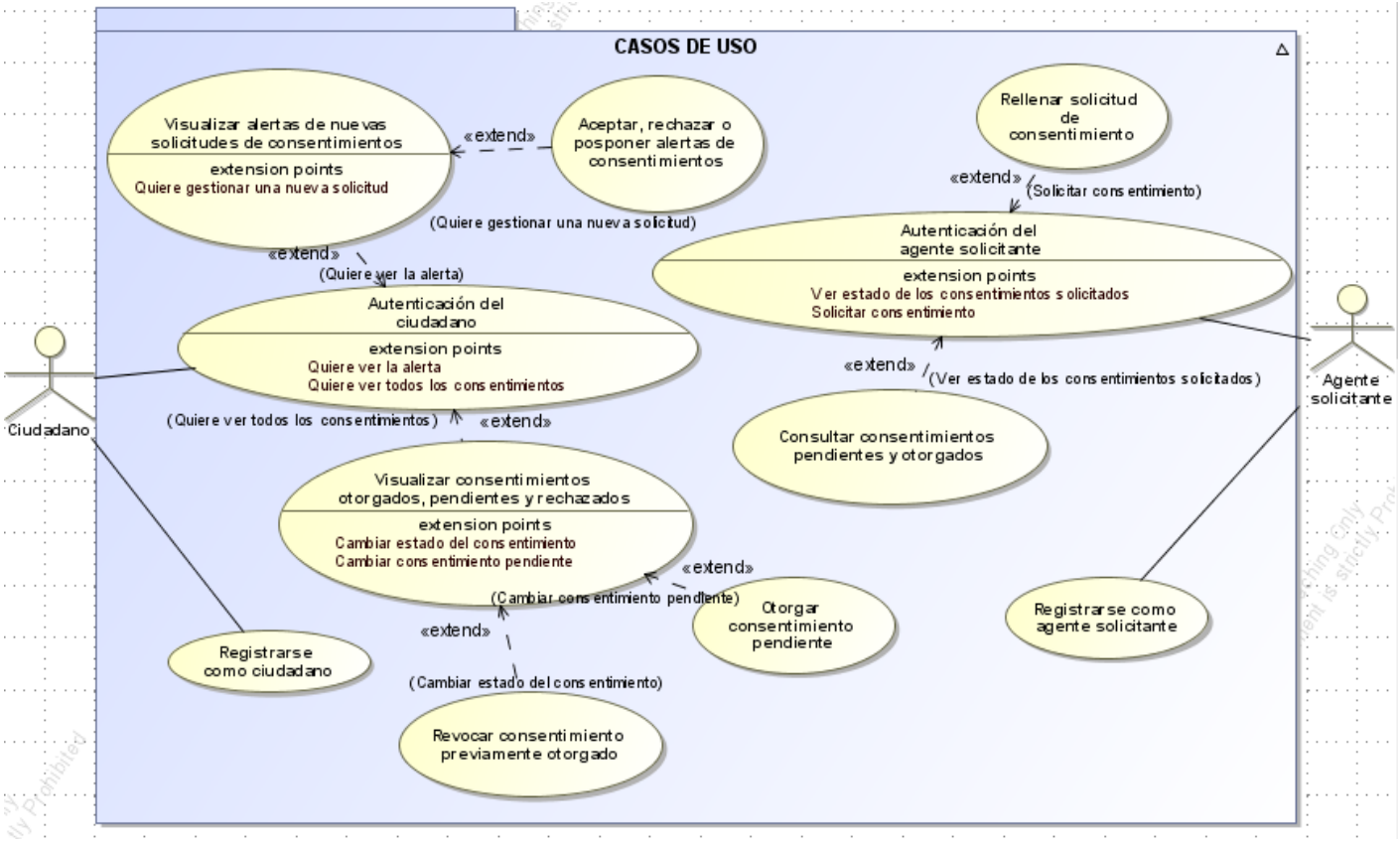

Figura 3-2. Casos de uso

En las siguientes subsecciones veremos cada caso de uso en detalle.

3.1.1 Caso de uso 01: Autenticación del ciudadano

Este caso de uso tiene lugar cuando un usuario intenta acceder como ciudadano. En la Tabla 3-3 tenemos una descripción del mismo.

Tabla 3–3. CU-01

CU-01	Autenticación del ciudadano
Precondición	Ninguna
Descripción	El ciudadano tendrá que autenticarse en la aplicación a través de un login
Postcondición	El ciudadano ha accedido a la aplicación
Comentarios	Los datos con el que el ciudadano ingresa deben estar almacenados en una base de datos

Además de esta tabla, tenemos la Figura 3-3 la cual representa el diagrama de actividad del caso de uso descrito.

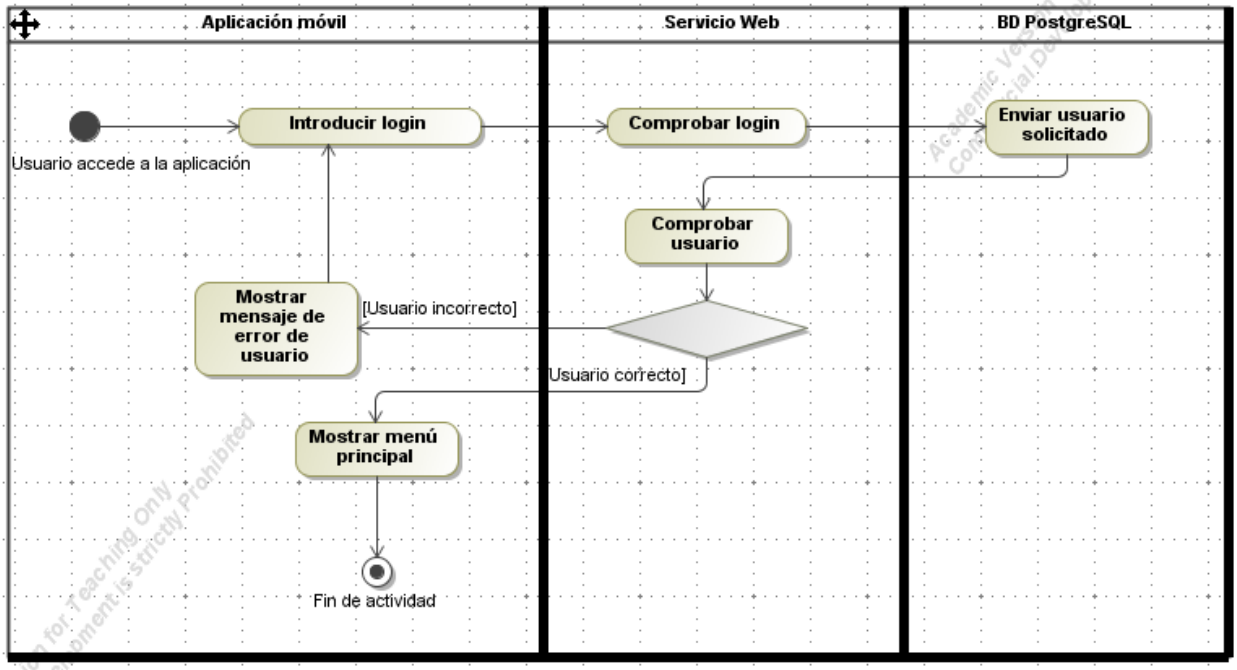

Figura 3-3. Diagrama de actividad del CU-01

3.1.2 Caso de uso 02: Visualizar avisos de nuevas solicitudes de consentimientos

Cuando el ciudadano se ha autenticado, tiene lugar el momento de ver los avisos. Se trata de CU-02 es cual está descrito en la Tabla 3-4.

Tabla 3–4. CU-02

CU-02	Visualizar avisos de nuevas solicitudes de consentimientos
Precondición	El ciudadano se ha autenticado
Descripción	Una vez dentro de la aplicación, el ciudadano ve los avisos de las nuevas solicitudes que haya recibido
Postcondición	Ve una lista de los avisos que ha recibido
Comentarios	Tales avisos son producidos por el envío de una solicitud por parte del agente solicitante

De igual forma, su diagrama de actividad está representado en la Figura 3-4.

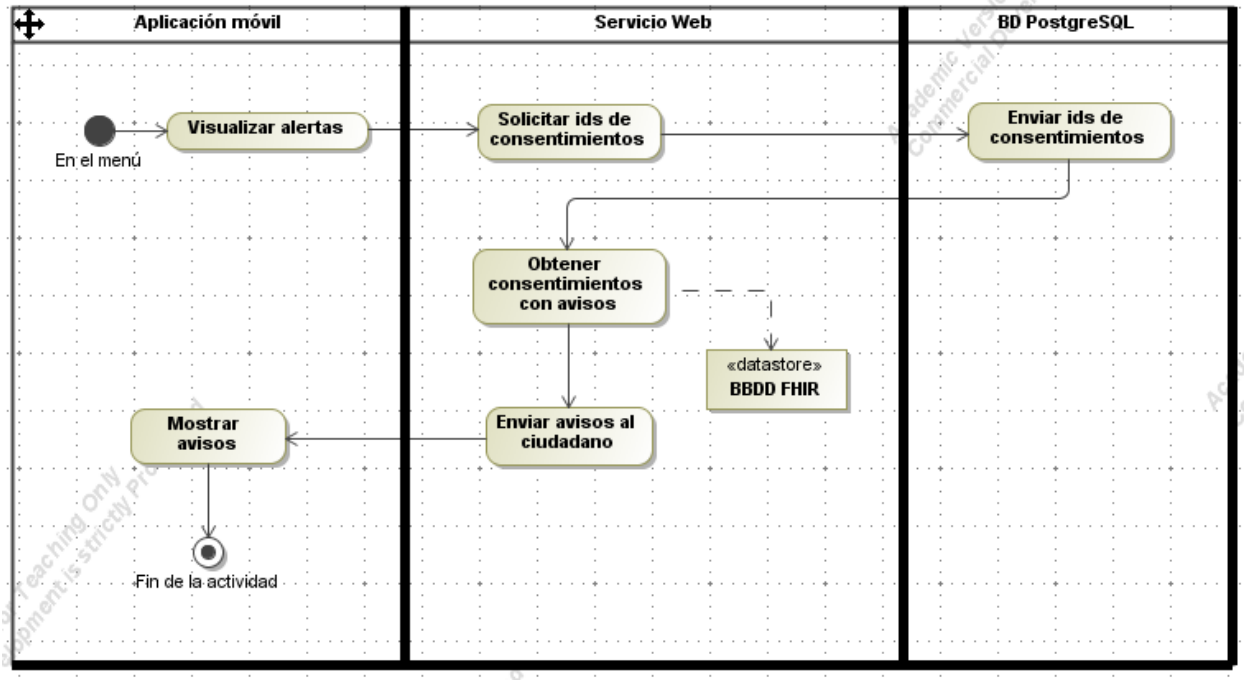

Figura 3-4. Diagrama de actividad del CU-02

3.1.3 Caso de uso 03: Gestionar consentimiento

Cuando el ciudadano vea los avisos puede seleccionar una para gestionarla.

Tabla 3–5. CU-03

CU-03	Aceptar, rechazar o posponer avisos de consentimientos
Precondición	El ciudadano ha seleccionado un aviso
Descripción	El ciudadano gestionará el aviso recibido de la forma que elija
Postcondición	Se ha procesado el aviso y se vuelven a ver todos los avisos
Comentarios	La aviso se produce por la llegada de una solicitud de consentimiento

Al igual que con los demás casos de uso, tenemos el diagrama de actividad.

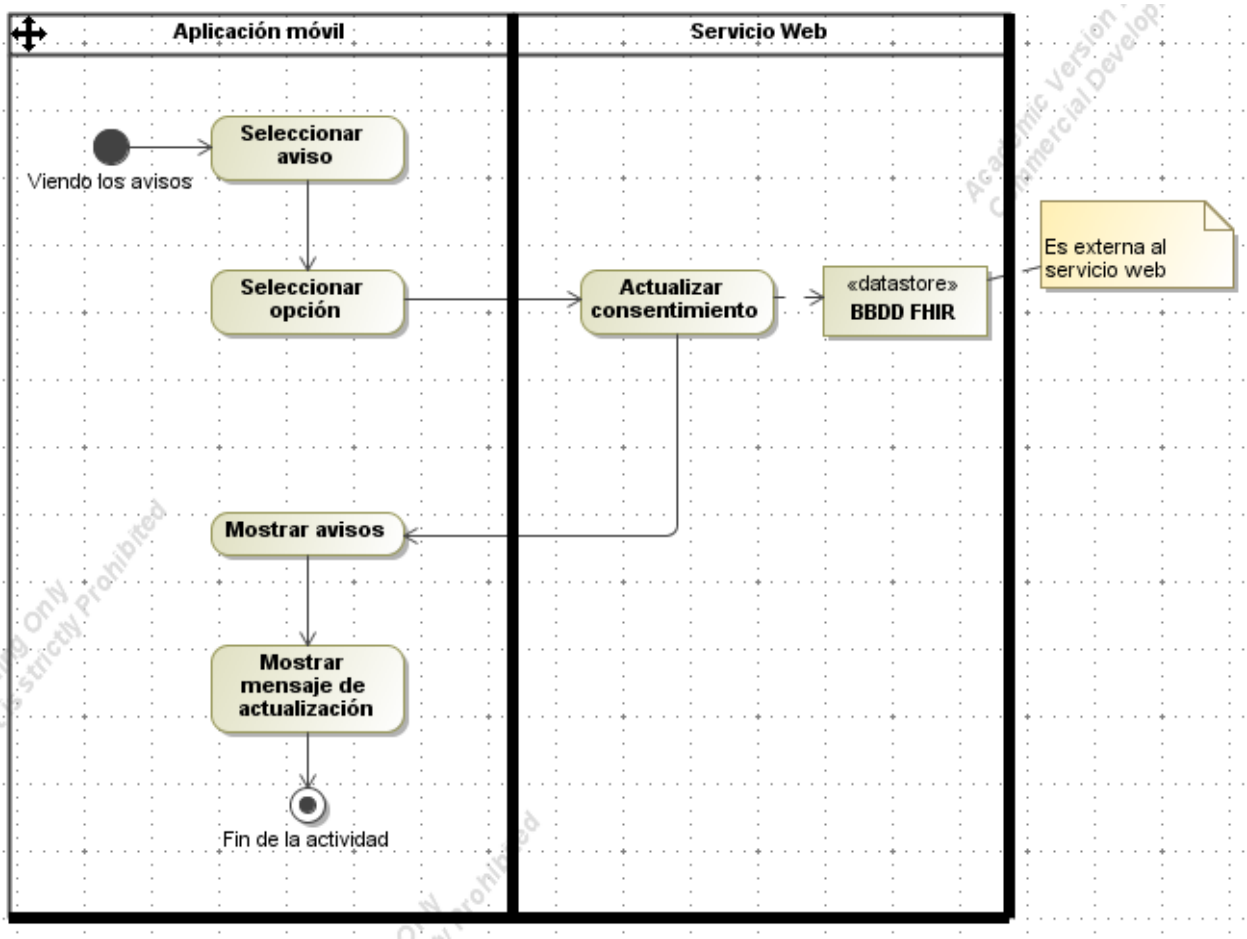

Figura 3-5. Diagrama de actividad del CU-03

3.1.4 Caso de uso 04: Visualizar consentimientos

Si el usuario no quiere pararse en los avisos o prefiero ver todas las solicitudes juntas, también podrá hacerlo al pasar al siguiente menú.

Tabla 3–6. CU-04

CU-04	Visualizar consentimientos otorgados, pendientes y rechazados
Precondición	El ciudadano se ha autenticado y ha saltado los avisos
Descripción	El ciudadano selecciona un estado y los consentimientos en ese estado son mostrados de forma resumida
Postcondición	Ninguna
Comentarios	Los estados serán los ya nombrados: otorgados, pendiente y rechazado

Su respectivo diagrama de actividad se encuentra en la Figura 3-6.

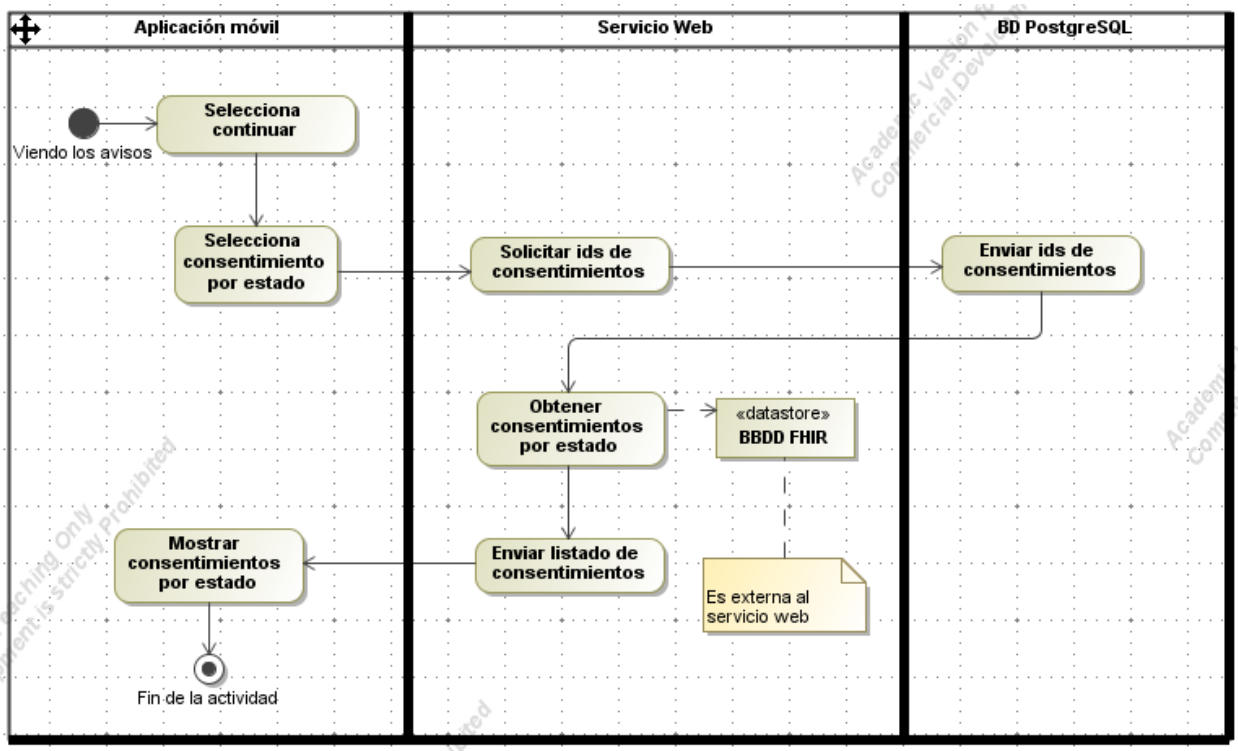

Figura 3-6. Diagrama de actividad del CU-04

3.1.5 Caso de uso 05: Revocar consentimiento previamente otorgado

El ciudadano tiene la posibilidad de revocar la aceptación de una solicitud de consentimiento que previamente había otorgado.

Tabla 3–7. CU-05

CU-05	Revocar consentimiento previamente otorgado
Precondición	Se ha seleccionado un consentimiento que había sido otorgado
Descripción	El ciudadano revoca un consentimiento que había otorgado anteriormente
Postcondición	Ninguna
Comentarios	Después, podrá volver a otorgar el consentimiento revocado si se trata de un error humano

Su correspondiente diagrama de actividad se puede ver en la Figura 3-7.

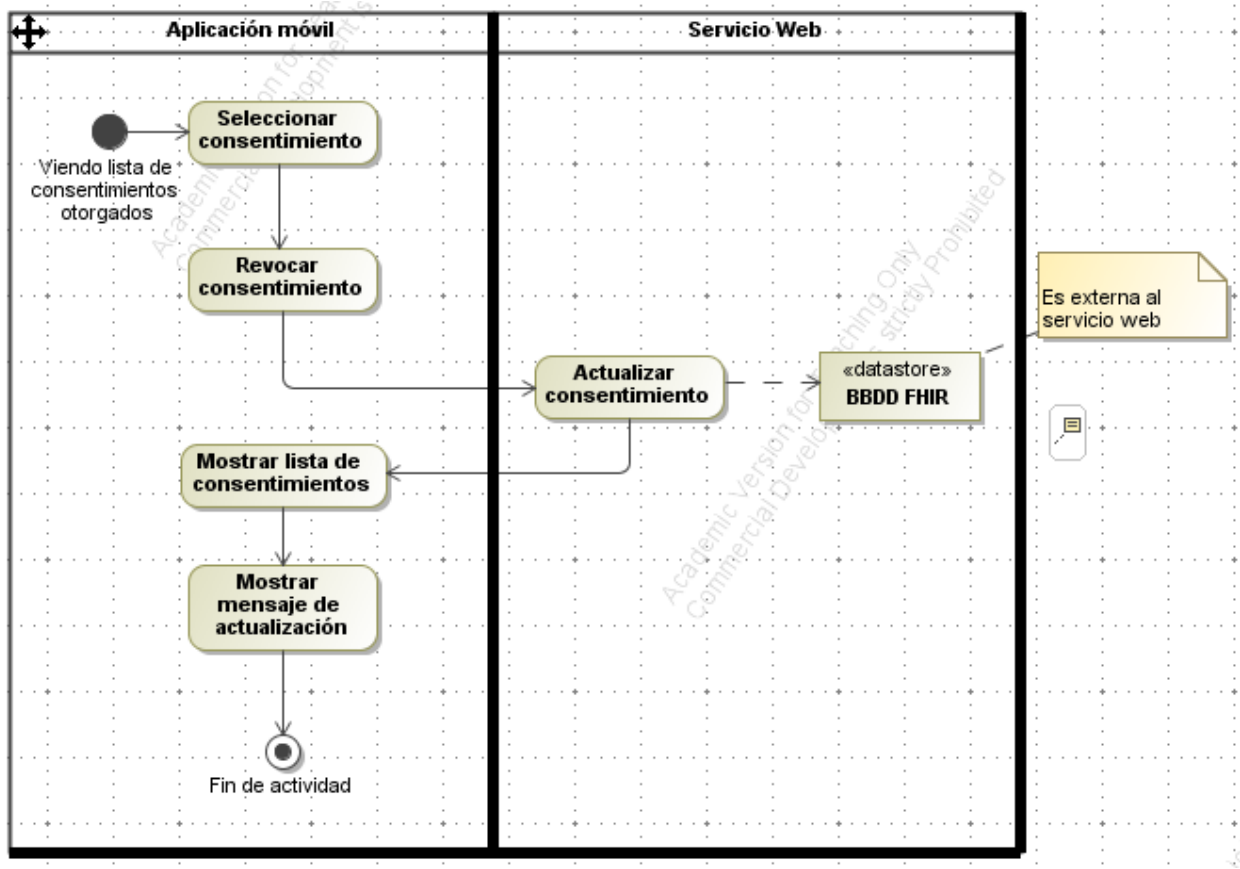

Figura 3-7. Diagrama de actividad del CU-05

3.1.6 Caso de uso 06: Otorgar consentimiento pendiente

Cuando el ciudadano esté viendo sus consentimientos pendientes podrá seleccionarlos para otorgar el consentimiento al solicitante.

Tabla 3–8. CU-06

CU-06	Otorgar consentimiento pendiente
Precondición	Se ha seleccionado un consentimiento que fue pospuesto
Descripción	El ciudadano otorgará un consentimiento que fue pospuesto cuando le llegó el aviso
Postcondición	Ninguna
Comentarios	Modificará su estado

En la Figura 3-8 tenemos su diagrama de actividad.

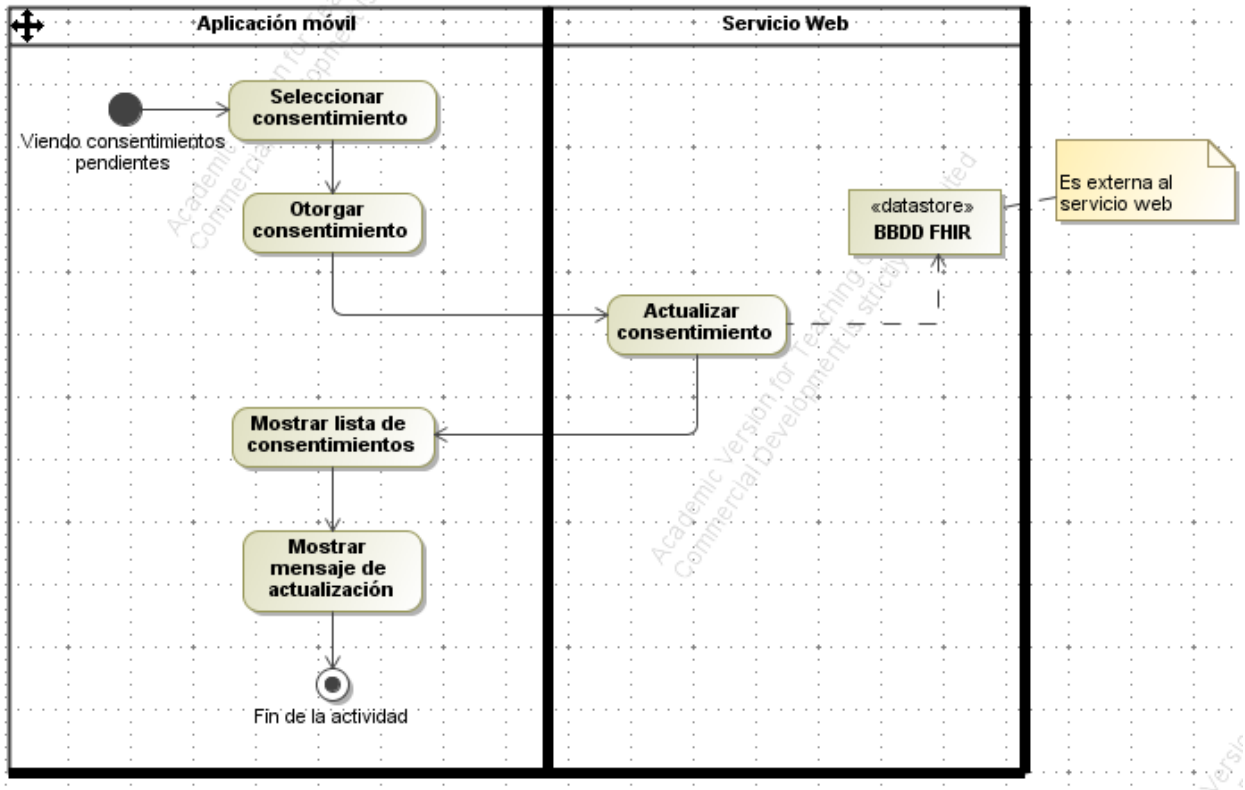

Figura 3-8. Diagrama de actividad del CU-06

3.1.7 Caso de uso 07: Autenticación del agente solicitante

Al igual que con el ciudadano, el agente solicitante deberá autenticarse para acceder a la aplicación móvil tal y

como se indica en la Tabla 3-9.

Tabla 3–9. CU-07

CU-07	Autenticación del agente solicitante
Precondición	Ninguna
Descripción	El agente tendrá que autenticarse en la aplicación a través de un login
Postcondición	El agente ha accedido a la aplicación
Comentarios	Los datos con el que el agente ingresa deben estar almacenados en una base de datos

Su diagrama de actividad es el mismo que el de la Figura 3-3.

3.1.8 Caso de uso 08: Solicitar consentimiento

Este caso de uso muestra la situación por la que un agente solicitante usaría la aplicación: solicitar un consentimiento a un paciente.

Tabla 3–10. CU-08

CU-08	Rellenar solicitud de consentimiento
Precondición	El agente se ha autenticado
Descripción	El agente rellenará la solicitud del consentimiento con los datos que considere oportuno
Postcondición	La solicitud se ha enviado
Comentarios	Una vez enviada, no se puede modificar tal solicitud. Puede enviarse a un único ciudadano o a todos.

Como en los demás casos, tenemos su diagrama de actividad en la Figura 3-9.

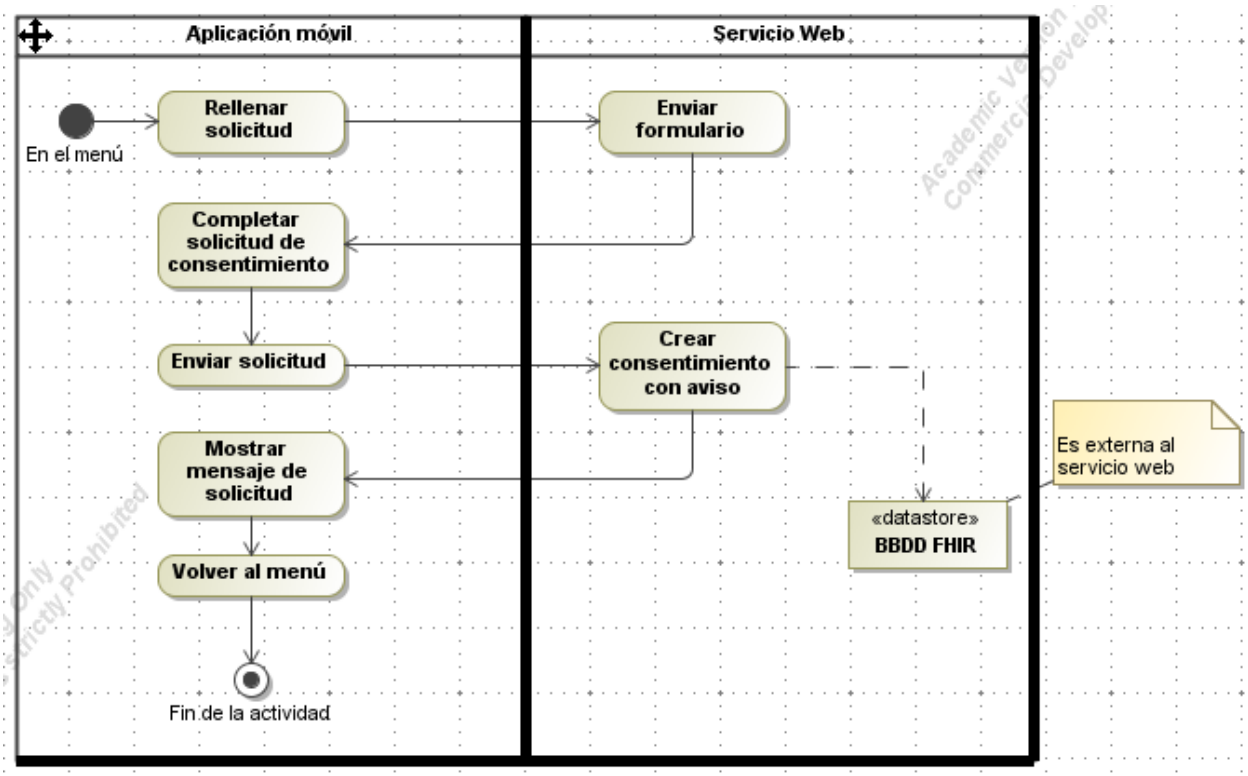

Figura 3-9. Diagrama de actividad del CU-08

3.1.9 Caso de uso 09: Consultar consentimientos

Al igual que un ciudadano, el agente solicitante podrá consultar los consentimientos que haya solicitado en función del estado en el que se encuentren.

Tabla 3–11. CU-09

CU-09	Consultar consentimientos otorgados, pendientes y rechazados
Precondición	El agente se ha autenticado
Descripción	El agente verá una lista de los consentimientos en función de su estado
Postcondición	Ninguna
Comentarios	Los estados serán los ya nombrados: otorgados, pendiente y rechazado

Su diagrama de actividad se puede ver en la Figura 3-10.

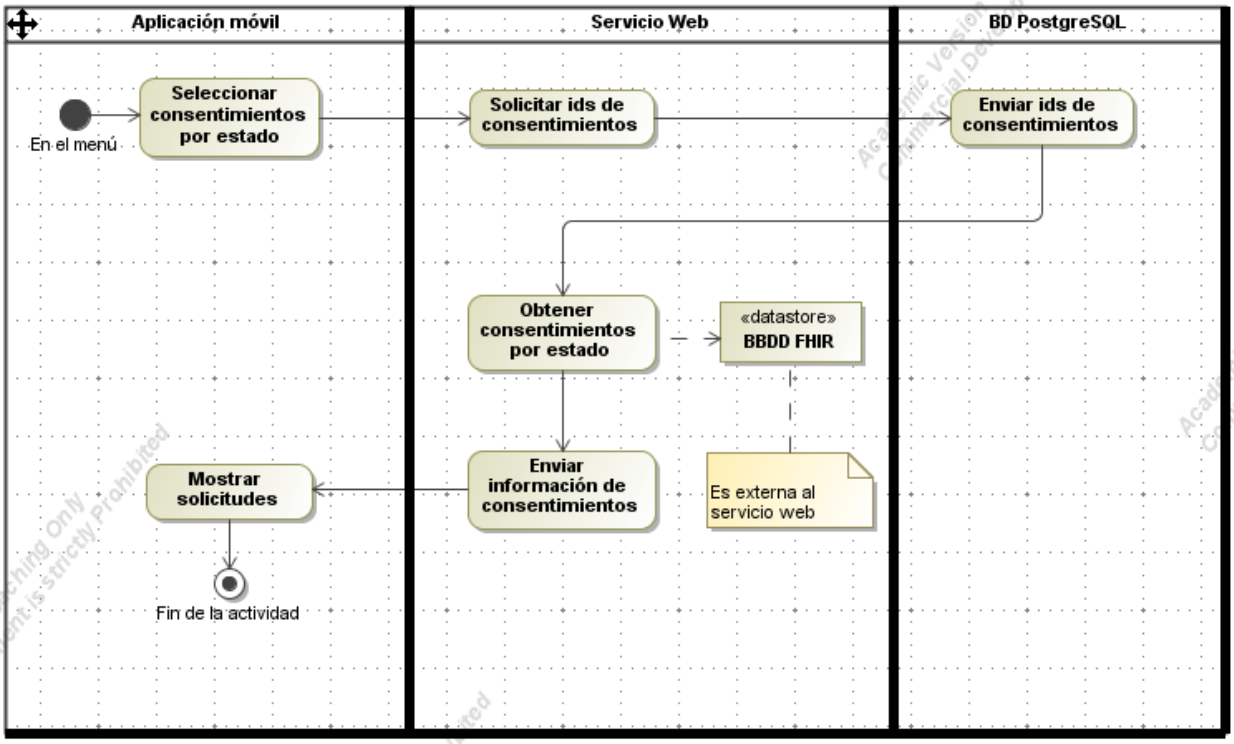

Figura 3-10. Diagrama de actividad del CU-09

3.1.10 Caso de uso 10: Registro como ciudadano

Para usar la aplicación móvil, el ciudadano deberá registrarse.

Tabla 3–12. CU-10

CU-10	Registrarse como ciudadano
Precondición	Ninguna
Descripción	El ciudadano se registrará con sus datos personales. La contraseña tendrá que ser su DNI obligatoriamente.
Postcondición	Ninguna
Comentarios	El nombre de usuario debe tener 8 letras

Esta acción tiene un diagrama de actividad asociado que se muestra en la Figura 3-11.

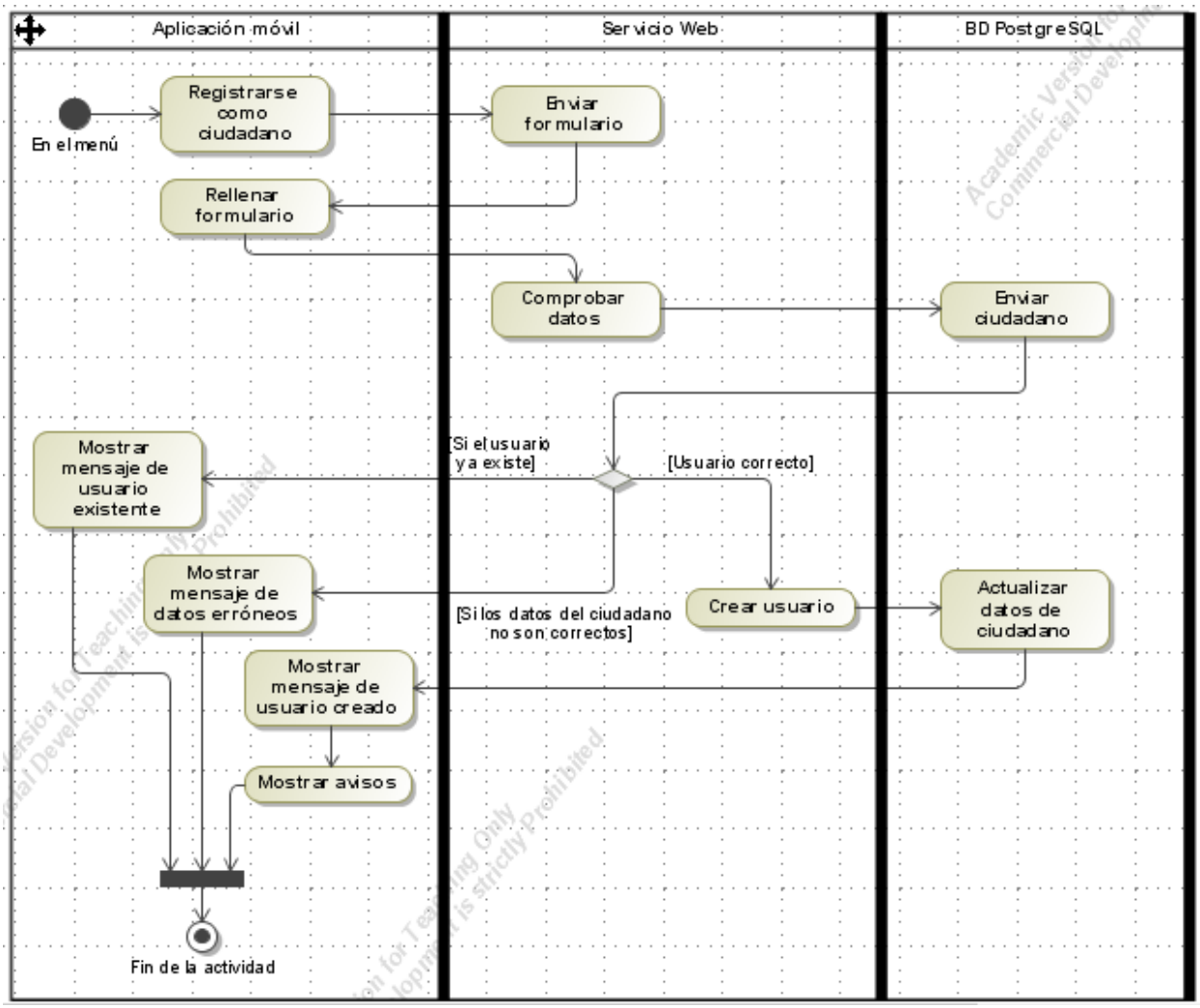

Figura 3-11. Diagrama de actividad del CU-10

3.1.11 Caso de uso 11: Registro como agente solicitante

Al igual que con el ciudadano, el agente sanitario debe registrarse para usar la aplicación móvil.

Tabla 3–13. CU-11

CU-11	Registrarse como agente solicitante
Precondición	Ninguna
Descripción	El agente solicitante se registrará con los datos aportados por la administración del hospital. La contraseña tendrá que ser uno de estos datos obligatoriamente.
Postcondición	Ninguna
Comentarios	El nombre de usuario debe tener 8 letras

Su diagrama de actividad es muy parecido al de la Figura 3-11.

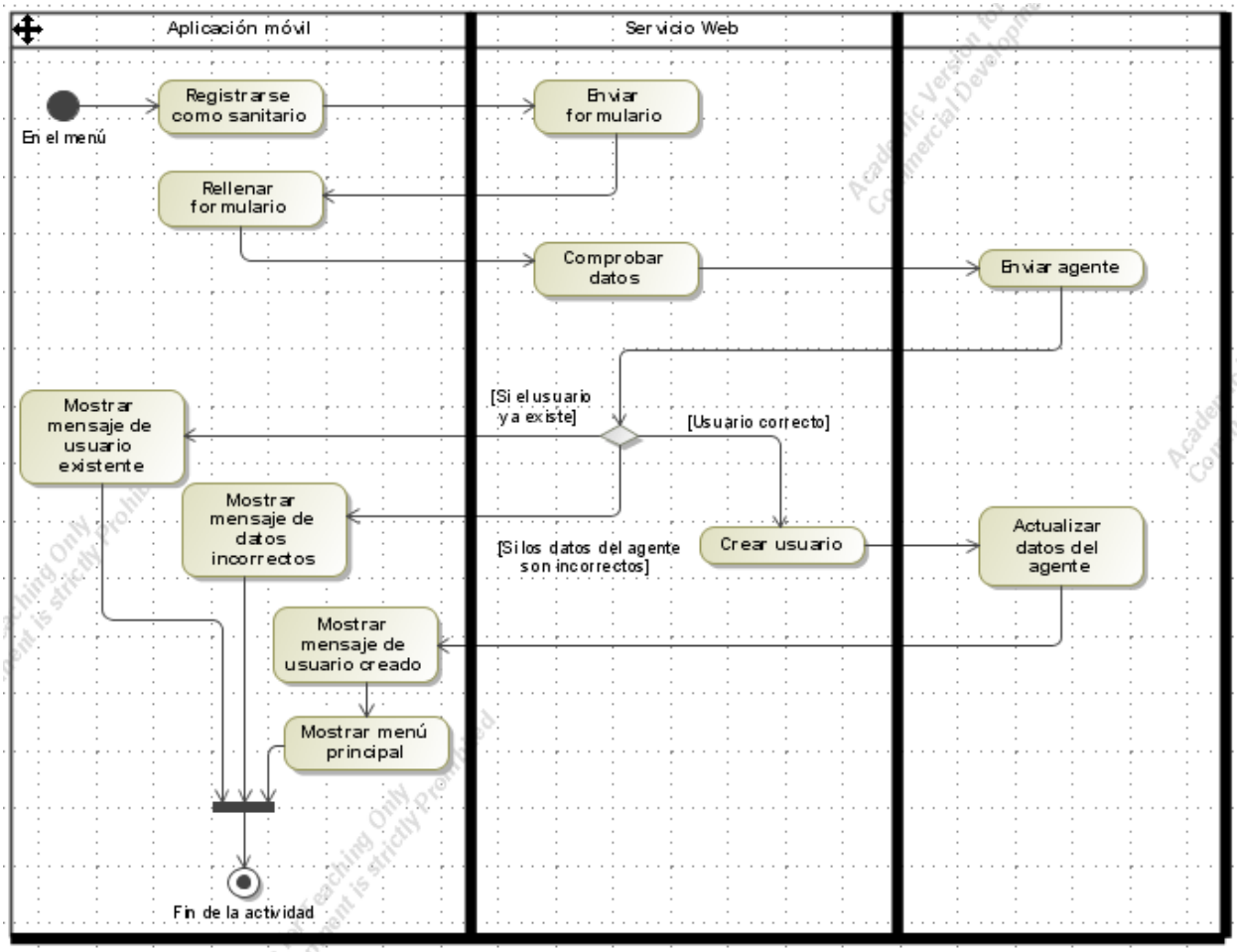

Figura 3-12. Diagrama de actividad del CU-11

3.1.12 Caso de uso 12: Eliminar solicitud de consentimiento

Puede darse la situación en la que el agente solicitante envíe una solicitud de consentimiento y se haya equivocado en algún campo. Por ello, se le da la posibilidad de eliminar la solicitud.

Tabla 3–14. CU-12

CU-12	Eliminar solicitud de consentimiento
Precondición	El agente ha seleccionado una solicitud
Descripción	El agente eliminará la solicitud que creó anteriormente
Postcondición	La solicitud desaparece del listado
Comentarios	Desaparecerá de la base de datos

Como en los demás CU, vemos su diagrama de actividad en la Figura 3-13.

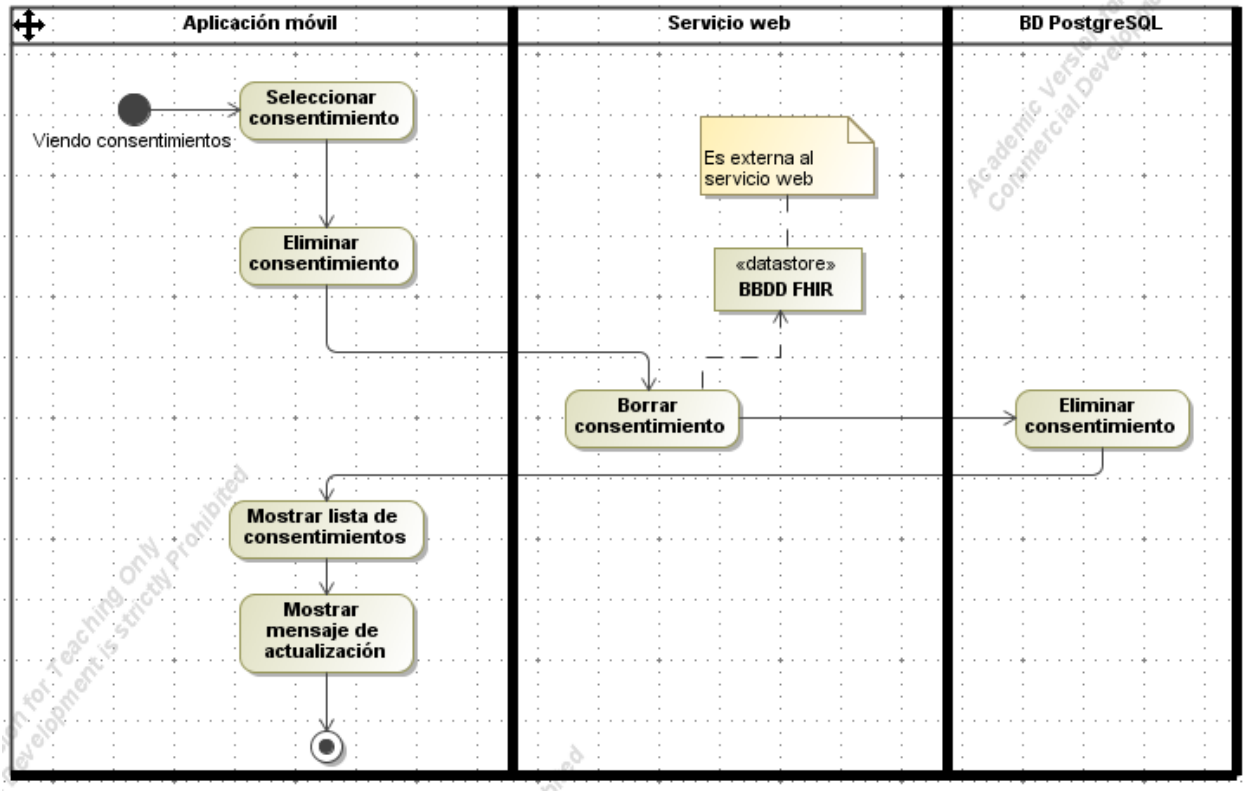

Figura 3-13. Diagrama de actividad del CU-12

3.2 Correspondencia GDPR y recursos FHIR

Ahora pasaremos a ver los campos de los recursos usados en detalle. Con ello, veremos cuáles hemos usado y las extensiones que añadimos.

3.2.1 Recurso *Patient*

Como se ha visto, uno de los recursos usados es el recurso *Patient*. Aunque este tiene varios campos solo usaremos aquellos que nos resultan útiles así como los que sean obligatorios. Estos son:

- *Identifier*: se trata del identificador del paciente. En este campo almacenaremos el DNI del mismo.
- *Name*: campo usado para guardar el nombre del ciudadano.
- *Telecom*: se trata del contacto del paciente. Se usará para el número de teléfono.
- *Extension*: como extensiones hemos usado dos campos. Estos son:
    - *Usuario*: cuando el paciente se registra es necesario usar un nombre de usuario que será el login con el que acceda a la aplicación en futuras ocasiones. Para guardar este login hemos usado esta extensión de recurso.
    - *Tarjsanitaria*: tal y como señalamos anteriormente, el ciudadano tendrá un DNI y un número de tarjeta sanitaria como es habitual. Por lo tanto, necesitaremos esta extensión para poder hacer uso del mismo.

3.2.2 Recurso *Practitioner*

Del recurso *Practitioner* usamos varios campos y en el sistema. Ahora pasamos a ver cada uno de sus campos:

- *Identifier*: identificador del agente. En este campo almacenaremos su DNI.
- *Name*: campo usado para guardar el nombre del agente solicitante.
- *Extension*: como extensiones hemos usado varios campos. Estos son:
    - *Usuario*: el agente solicitante también se registra por lo que es necesario usar un nombre de usuario que será el login con el que acceda a la aplicación en futuras ocasiones al igual que en caso del paciente.
    - *Contra*: se usará para almacenar la contraseña con la que el agente sanitario accede a la aplicación móvil.
    - *Codigo*: se trata del código que es necesario aportar cuando el agente se registra en el sistema

3.2.3 Recurso *Organization*

Cuando el agente se registre en la aplicación tendrá que indicar a qué organización pertenece. Para ello usaremos los siguientes campos de este recurso:

- *Identifier*: como identificador usaremos el DNI del agente que se relaciona con esta organización.
- *Name*: guardaremos el nombre de la organización a la que pertenece el agente solicitante.

3.2.4 Recurso *PractitionerRole*

Este recurso cubre la información relacionada con el tipo de trabajo que el agente realiza para una organización. De esta forma, nos servirá para indicar en qué departamento se encuentra dentro de una organización. Además, nos permitirá relacionar el recurso *Practitioner* con el recurso *Organization* a través de los campos *Reference* del mismo. Usamos los siguientes campos:

- *Identifier*: como identificador usaremos el DNI del agente.
- *Code*: guardaremos el departamento al que pertenece el agente solicitante.
- *Organization*: almacenará la referencia a la organización a la que pertenece el agente.
- *Practitioner*: almacenará la referencia al agente sanitario.

3.2.5 Recurso *Consent*

Podríamos decir que el recurso *Consent* es el recurso más importante del sistema. A continuación detallaremos los campos usados y sus extensiones:

- *Status*: este campo nos indica el estado en el que se encuentra el consentimiento. Los valores que puede tener están predefinidos por el estándar y son: *draft*, si el consentimiento está pendiente; *active*, si ha sido aceptada la solicitud; o *rejected*, si la solicitud ha sido rechazada por el paciente.
- *Scope*: en este campo se almacenará el motivo por el que se va a solicitar el consentimiento.
- *Category*: tal y como su nombre indica, se usará este campo para la categoría de los datos personales a los que se quiere acceder.
- *Patient*: este campo es en realidad una referencia a un paciente. De esta forma, identificaremos al ciudadano al que se le solicita el consentimiento mediante la referencia.

- *Performer*: este campo guarda una referencia a un agente sanitario al agente que solicita el consentimiento.
- *Organization*: es otra referencia a otro recurso. En este caso, apuntará a la organización a la que pertenece el agente y en la que están guardados los datos, ya sean en formato físico o digital.
- *Provision*: añade más información para las restricciones de la política del recurso
    - *Actor*: referencia al agente sanitario que hará uso de los datos a los que se quiere acceder. Este puede ser el agente solicitante o un tercero por lo que se debe dejar claro quién los usará.
    - *Action*: campo que indica la acción a realizar sobre los datos una vez el consentimiento sea aceptado. Esta podrá ser "Acceso", "Lectura", "Modificación" y "Envío".
- *Extension*: como extensiones hemos usado varios campos. Estos son:
    - *Data*: indica a qué datos se quiere acceder.
    - *Duration*: indica durante cuánto tiempo se quiere acceder a los datos. Pasado este plazo, el consentimiento no tendrá validez.
    - *Terms*: campo que informa de condiciones extras en la solicitud de los datos.
    - *Notice*: este campo servirá para comprobar si el consentimiento que le aparece al ciudadano es nuevo o ya se revisó antes.

En las Figuras 3-14, 3-15 y 3-16 tenemos un ejemplo de un consentimiento usado en el sistema:


```
{\n
  "consentimientos": [\n
\t{\n
  "resourceType": "Consent",\n
  "id": "2440551",\n
  "meta": {\n
    "versionId": "1"\n
  },\n
  "extension": [ {\n
    "url": "localhost:8080/extensión/consent/datos",\n
    "valueString": "Resultados"\n
  }, {\n
    "url": "localhost:8080/extensión/consent/duracion",\n
    "valueString": "10 dias"\n
  }, {\n
    "url": "localhost:8080/extensión/consent/cond",\n
    "valueString": "Ninguna"\n
  }, {\n
    "url": "localhost:8080/extensión/consent/aviso",\n
    "valueBoolean": false\n
  } ],\n
  "status": "draft",\n
  "scope": {\n
    "text": "Cita medica"\n
  },\n
  "category": [ {\n
    "text": "Oncologia"\n
  } ],\n
```


Figura 3-14. Ejemplo de consentimiento I

```
"patient": {\n
  "reference": "http://hapi.fhir.org/Patient",\n
  "type": "Patient",\n
  "identifier": {\n
    "value": "76543201A"\n
  }\n
},\n
"dateTime": "2021-07-22T19:55:35+02:00",\n
"performer": [ {\n
  "reference": "http://hapi.fhir.org/Practitioner",\n
  "type": "Practitioner",\n
  "identifier": {\n
    "value": "012345678"\n
  }\n
} ],\n
"organization": [ {\n
  "reference": "http://hapi.fhir.org/Organization",\n
  "type": "Organization",\n
  "identifier": {\n
    "value": "012345678"\n
  },\n
  "display": "Valme"\n
} ],\n
```


Figura 3-15. Ejemplo de consentimiento II


```
"provision": {\n
  "actor": [ {\n
    "reference": {\n
      "reference": "http://hapi.fhir.org/Practitioner",\n
      "type": "Practitioner",\n
      "identifier": {\n
        "value": "012345678"\n
      }\n
    }\n
  } ],\n
  "action": [ {\n
    "coding": [ {\n
      "code": "use"\n
    } ]\n
  } ]\n
}\n
```


Figura 3-16. Ejemplo de consentimiento III

3.3 Modelado de la base de datos de PostgreSQL

En el escenario que planteamos disponemos de ciertos datos tanto de los pacientes como de los agentes sanitarios antes de que hagan uso del sistema. Para el caso del ciudadano, tendríamos un DNI y un número de tarjeta sanitaria tal y como vemos en la Figura 3-17.

```
gestorconsentimientos=> select * from ciudadanos;
 nombre |    dni    | usu | tarjsanitaria | telefono
--------+-----------+-----+---------------+----------
        | 76543201A |     |             1 |
        | 76543201B |     |             2 |
        | 76543201C |     |             3 |
        | 76543201D |     |             4 |
```

Figura 3-17. Tabla de Ciudadanos

Cuando el ciudadano quiera registrarse, estos datos tendrán que estar previamente en la base de datos. Entonces, será actualizada como se muestra en la Figura 3-18.

```
gestorconsentimientos=> select * from ciudadanos;
  nombre  |    dni    |   usu    | tarjsanitaria | telefono
----------+-----------+----------+---------------+-----------
          | 76543201B |          |             2 |
          | 76543201C |          |             3 |
          | 76543201D |          |             4 |
 usuario1 | 76543201A | usuario1 |             1 | 123456789
```

Figura 3-18. Tabla de Ciudadanos actualizada

Con ello, los ciudadanos solo podrían registrarse una vez en el sistema. Para el caso del agente sanitario, los datos con los que registrarse serían dados por la administración correspondiente, siendo este el hospital en el caso de un médico por ejemplo. Estos serían una contraseña y un código para el registro además de que estos estarían repartidos por departamentos. Quedaría como en la Figura 3-19.

```
gestorconsentimientos=> select * from agentes;
 nombre | dni |  contra   | usu | hospital |    depart     | codigo
--------+-----+-----------+-----+----------+---------------+--------
        |     | A76501234 |     | Valme    | Traumatologia | 1A
        |     | B76501234 |     | Valme    | Cardiologia   | 1B
        |     | C76501234 |     | Macarena | Urologia      | 2A
        |     | D76501234 |     | Rocio    | Pediatria     | 3A
```

Figura 3-19. Tabla de Agentes

El agente se registrará y el usuario se actualizará con sus datos (Figura 3-20).

```
gestorconsentimientos=> select * from agentes;
  nombre  |    dni    |  contra   |   usu    | hospital |    depart     | codigo
----------+-----------+-----------+----------+----------+---------------+--------
          |           | B76501234 |          | Valme    | Cardiologia   | 1B
          |           | C76501234 |          | Macarena | Urologia      | 2A
          |           | D76501234 |          | Rocio    | Pediatria     | 3A
 agente11 | 012345678 | A76501234 | agente11 | Valme    | Traumatologia | 1A
```

Figura 3-20. Tabla de Agentes actualizada

Finalmente, también hemos incluido una tabla para los consentimientos. Sin embargo, en esta tabla tendremos una columna para identificar al ciudadano y otra para identificar al agente de tal forma que el consentimiento

pueda asociarse a ellos. Además, se guardará el identificador del consentimiento que se crea en la base de datos FHIR para poder recuperarlo cuando lo necesitemos. Vemos cómo quedaría en la Figura 3-21.

```
gestorconsentimientos=> select * from consentimientos;
 id | contraag  |  dniciud  | id_consent
----+-----------+-----------+------------
 41 | C76501234 | 76543201C | 2385046
 43 | C76501234 | 76543201C | 2385049
 44 | C76501234 | 76543201D | 2385050
 45 | C76501234 | 76543201C | 2385064
 46 | C76501234 | 76543201D | 2385065
```

Figura 3-21. Tabla de Consentimientos

3.4 Patrones

Como en cualquier desarrollo de software, se han utilizado patrones para facilitar la programación del sistema. Los patrones que se han usado son el patrón Singleton y el patrón Data Access Object (DAO) los cuales se explican en las siguientes subsecciones.

3.4.1 Singleton

Este patrón nos garantiza que una clase tenga una única instancia y nos da un punto global de acceso a ella [30]. Ha sido implementado en la aplicación móvil con el objetivo de tener una única aplicación móvil en funcionamiento de forma que un mismo usuario no pueda realizar más de una solicitud en el mismo momento. De esta forma, solo tendremos una instancia de la misma. El código usado para su implementación puede verse en la Figura 3-22.

```
@Override
public void onCreate() {
    super.onCreate();
    mRequestQueue = Volley.newRequestQueue( context: this);
    sInstance = this;
    ctx = FhirContext.forR4();
    ctx.getRestfulClientFactory().setServerValidationMode(ServerValidationModeEnum.NEVER);
}

public synchronized static Aplicacion getInstance() { return sInstance; }

public RequestQueue getRequestQueue() { return mRequestQueue; }
```

Figura 3-22. Implementación del patrón Singleton

En el método *onCreate()* se instancian tanto la cola como la propia instancia de la aplicación. Además, se usarán los métodos *getInstace()* y *getRequestQueue()* para comprobar si estas han sido creadas anteriormente o no.

3.4.2 DAO

Usamos este patrón para separar la lógica de negocio de la lógica del acceso a los datos de forma que la parte DAO se encargará de consultar, insertar, eliminar o actualizar los datos mientras que la otra parte nos servirá para gestionar los datos obtenidos [31]. En la Figura 3-23 vemos un diagrama el cual representa los componentes que intervienen en este proceso.

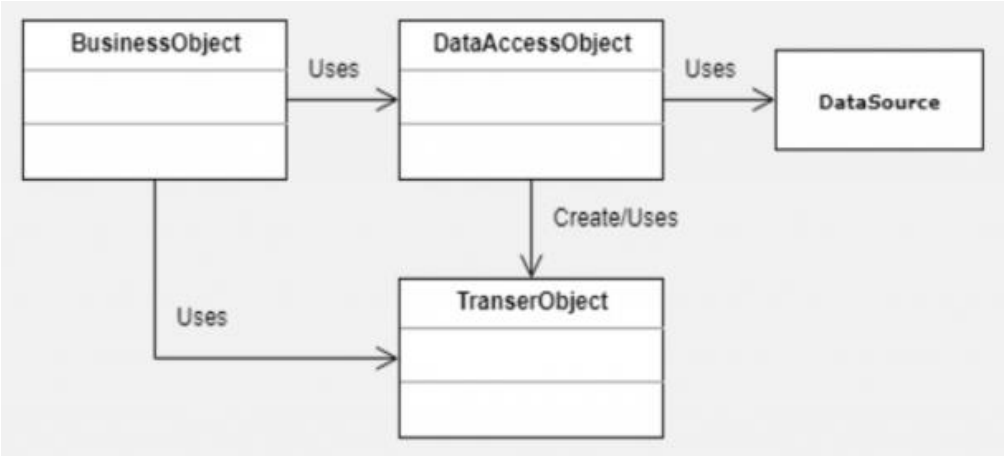

Figura 3-23. Diagrama del patrón DAO

A continuación se explica qué representa cada campo:

- BussinessObject: esta clase representa la lógica de negocio del sistema.
- DataAccessObject: se trata de la lógica de acceso a los datos del sistema.
- DataSource: representa la base de datos
- TransferObject: se trata de un objeto que implementa el patrón el cual pasa la información entre el BussinessObject y el DAO

Para el servicio web se ha hecho una separación muy parecida. En la Figura 3-24 podemos ver los ficheros que serían la lógica de negocio.

Figura 3-24. Ficheros Java correspondientes a la capa de negocio

Sus métodos usarán la capa DAO para el acceso a los datos. Como podemos ver, tenemos dos paquetes de servicios: el correspondiente al servicio del agente y el del servicio del ciudadano. Cada uno de ellos tendrá sus propios métodos ya que tendrán funciones diferentes. En la Figura 3-25 tenemos los ficheros DAO que se conectarán con la base de datos.

Figura 3-25. Ficheros Java de la capa DAO

Como podemos observar, tenemos tres ficheros DAOs que conectarán con la base de datos además de las interfaces correspondientes. Esto se debe a que usaremos cada una de ellas dependiendo del tipo de acceso que queramos realizar: sobre agentes, sobre ciudadanos o sobre consentimientos.

3.5 Implementación del servicio web

En esta sección se verá cómo hemos logrado esta implementación. Empezaremos viendo la estructura del servicio junto a los paquetes en los que se ha dividido para después entrar en el código usado para logar su correcto funcionamiento.

3.5.1 Estructura

El servicio está dividido en varios paquetes. Cada uno de ellos tiene una función distinta y funcionan en conjunto a través de llamadas entre clases. Para ello, se usarán las interfaces de las mismas. En la Figura 3-26 vemos todos los paquetes existentes.

Figura 3-26. Paquetes del servicio web

El primer paquete solo sirve para arrancar el servicio. Para ello, usaremos la anotación *SpringBootApplication* sobre nuestra clase principal tal y como vemos en la Figura 3-27. Es entonces cuando comenzarán a moverse los engranajes de nuestro sistema.

```
@SpringBootApplication
public class Servidor extends SpringBootServletInitializer{
```

Figura 3-27. Clase principal del servicio con su anotación de Spring

3.5.1.1 Paquete Controller

En el paquete Controller tenemos los controladores del servicio web: uno para el ciudadano y otro para el agente solicitante.

A través de las anotaciones de Spring usadas se llamará a uno u otro en función de la URL a la que se lance la petición desde la aplicación móvil. Para señalar que es un controlador usamos la anotación *RestController* sobre la clase que usemos como tal. Además, usaremos la anotación *RequestMapping()* para indicar que dependiendo de cómo comience la ruta usada en la URL se activará un controlador u otro. Podemos ver en la Figura 3-28 que las llamadas a "\agente" serán gestionadas por el del agente mientras que en la Figura 3-29 vemos que las llamadas a "\ciud" son gestionadas por el controlador del ciudadano.

```
@RestController
@RequestMapping("/agente")
public class AgenteRestController {
```

Figura 3-28. Controlador del agente con anotaciones

```
@RestController
@RequestMapping("/ciud")
public class CiudRestController {
```

Figura 3-29. Controlador del ciudadano con anotaciones

Una vez el controlador reciba la petición tendrá que decidir qué método se lanza. Para ello usaremos las anotaciones *GetMapping()*, *PutMapping()* y *PostMapping()*. Dependiendo del método de petición usado se usará un método u otro de forma que si la petición se ha realizado con un "GET" solo los métodos con la primera anotación podrán responder. Además de esto, la ruta será determinante para diferenciar entre métodos que puedan responder al mismo método de petición.

```
@GetMapping("/acceder")
public String accederCiud(
```

Figura 3-30. Ejemplo de método con anotación *GetMapping()*

```
@PutMapping("/consentimiento/modificar")
public String modificarConsent(
```

Figura 3-31. Ejemplo de método con anotación *PutMapping()*

```
@PostMapping("/solicitud/{contra}")
public String solconsent(
```

Figura 3-32. Ejemplo de método con anotación *PostMapping()*

También se han usado anotaciones para recoger datos pasados a los métodos que son usados como variables. Así, usaremos la anotación *RequestParam()* para coger datos adjuntados en la URL. Como ejemplo tenemos la Figura 3-33. Se pasaría una contraseña en la URL indicada como "contra=dato" y se guardaría en un *String* llamado *contra*.

```
public String accederAgente(@RequestParam(value = "contra") String contra){
```

Figura 3-33. Ejemplo de método con anotación *RequestParam()*

Ciertos datos también pueden ser pasados sobre la URL, es decir, sin ser adjuntados de tal forma que la URL sea "http://ip/agente/dato". Para ello usamos otra anotación: *PathVariable()*. Además, la información puede ser pasada en el cuerpo de la petición. Para coger estos datos sería necesario usar la anotación *RequestBody*. En la Figura 3-34 tenemos ambas anotaciones.

```
@PutMapping("/reg/{contra}")
public String regAg(@PathVariable(value = "contra") String contra, @RequestBody String practicante)
```

Figura 3-34. Ejemplo de método con anotación *PathVariable()* y *RequestBody*

Como vemos en este ejemplo, se tomaría el dato que ocupe la posición de "{contra}" y se guardaría en la variable *contra* mientras que el cuerpo pasado en la consulta se guardaría en *practicante*.

Existe una anotación que hemos usado en casi todos los ficheros de los paquetes ya que nos sirve para conectarse entre ellos. Esta es la anotación *Autowired*. Con esta anotación se crean enlaces entre ficheros de tal forma que desde este paquete se puedan hacer llamadas a los paquetes AgenteService y CiudService. Desde estos podremos hacer las respectivas llamadas al paquete DAO.

```
@Autowired
private CiudService ciudService;
```

Figura 3-35. Ejemplo de *Autowired* con CiudService

```
@Autowired
private AgenteService agenteService;
```

Figura 3-36. Ejemplo de Ejemplo de *Autowired* con AgenteService

3.5.1.2 Paquete Agenteservice/Ciudservice

En este paquete tenemos las clases que harán de servicios. Estos se encargarán de conectarse con la capa DAO, hacer las solicitudes de datos y gestionar lo que reciba de esa capa. Usaremos la anotación *Service* para señalar qué clases funcionan como tal.

```
@Service
public class ImplAgenteService implements AgenteService {
```

Figura 3-37. Ejemplo de *Service* en el servicio del agente

```
@Service
public class ImplCiudService implements CiudService {
```

Figura 3-38. Ejemplo de *Service* en el servicio del ciudadano

Evidentemente, estas clases tendrán sus métodos para dotar de todas las funciones necesarias al servicio. Se verán en la siguiente subsección.

3.5.1.3 Paquete DAO

Este será el paquete en el que se encuentren las clases que acceden a la base de datos. Tendrán la anotación *Repository* la cual funciona como marcador para cualquier clase que cumpla esta función. Como trabajamos con varios recursos, existirá una clase para cada uno de ellos.

```
@Repository
public class ImplAgenteDAO implements InterfazAgenteDAO {
```

Figura 3-39. Ejemplo de *Repository* en AgenteDAO

```
@Repository
public class ImplCiudDAO implements InterfazCiudDAO {
```

Figura 3-40. Ejemplo de *Repository* en CiudDAO

```
@Repository
public class ImplConsentDAO implements InterfazConsentDAO {
```

Figura 3-41. Ejemplo de *Repository* en ConsentDAO

3.5.1.4 Paquete objetos

En este paquete se encuentran los objetos que usamos para la base de datos PostgreSQL. Tendremos los objetos Agentes, Ciudadanos y Consentimientos.

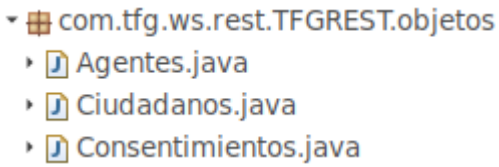

Figura 3-42. Objetos usados en la BBDD PostgreSQL

Sobre los consentimientos solo tendremos el DNI del ciudadano al que se le solicita el consentimiento, la

contraseña con la que accede el agente y el id del propio consentimiento.

```
@ManyToOne
@JoinColumn(name="contra")
private Agentes agentes;

@ManyToOne
@JoinColumn(name="dni")
private Ciudadanos ciudadanos;

@Column(name = "id_consent", nullable = false)
private String idconsent;
```

Figura 3-43. Objetos usados en la BBDD PostgreSQL

3.5.1.5 Paquete RecursosExt

En este paquete tenemos los recursos extendidos que hemos usado. Estos son Consen, Paciente y Practicante.

Figura 3-44. Recursos extendidos

Para extenderlos es necesario señalar qué recurso extienden para que además de los atributos que se extiendan tenga los del recurso original.

```
@ResourceDef(name = "Consent")
public class Consen extends Consent {
```

Figura 3-45. Extensión de recurso *Consent*

Usaremos tres tipos de anotaciones para las extensiones:

- *Child*: indica el nombre de la extensión
- *Description*: breve descripción de para qué se usa la extensión
- *Extension*: tiene varios parámetros
    - *url*: dirección del recurso
    - *isModifier*: si puede modificar el recurso
    - *definedLocally*: si está definida localmente

```
/** * This is a basic extension, with a DataType value (in this case, StringDt) */
//Atributo extendido Duración
@Child(name = "duracion")
@Description(shortDefinition = "Indica cuánto tiempo estarán los datos disponibles")
@Extension(url = "localhost:8080/extensión/consent/duracion", isModifier = false, definedLocally = true)
private StringDt duracion;
```

Figura 3-46. Atributo de recurso extendido

Obviamente, para trabajar con estos campos necesitaremos los respectivos métodos *Getters* y *Setters*.

```
public StringDt getDuracion() {
    return duracion;
}

public void setDuracion(StringDt duracion) {
    this.duracion = duracion;
}
```

Figura 3-47. Métodos *Get* y *Set* del atributo *Duración*

Finalmente, en la Figura 3-48 vemos un esquema de cómo se conectan los paquetes entre sí a través de la URL a la que se hace la petición y las llamadas a los métodos.

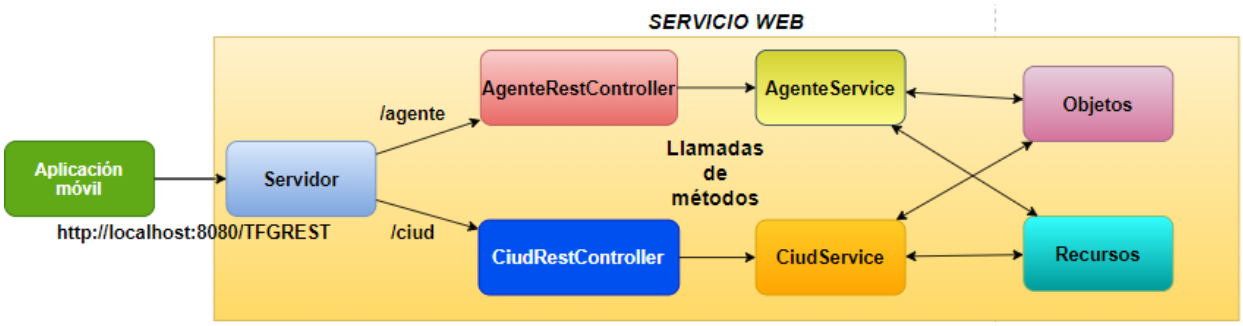

Figura 3-48. Esquema de interconexiones entre paquetes

3.5.2 Código implementado en el servicio web

En esta subsección veremos los métodos implementados en el servicio web. Antes de ello veremos el fichero pom.xml. Como ya comentamos, usamos este fichero para cargar las dependencias necesarias. Para poder trabajar con la especificación FHIR necesitamos añadir sus dependencias.

```
    <!-- https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-structures-r4 -->
<dependency>
    <groupId>ca.uhn.hapi.fhir</groupId>
    <artifactId>hapi-fhir-structures-r4</artifactId>
    <version>5.1.0</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-base -->
<dependency>
    <groupId>ca.uhn.hapi.fhir</groupId>
    <artifactId>hapi-fhir-base</artifactId>
    <version>5.1.0</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-client -->
<dependency>
    <groupId>ca.uhn.hapi.fhir</groupId>
    <artifactId>hapi-fhir-client</artifactId>
    <version>5.1.0</version>
</dependency>
```

Figura 3-49. Dependencias FHIR en el fichero pom.xml

Estas dependencias son descargadas de un repositorio Maven en el que podemos encontrar todo tipo de dependencias [34]. Además de estas, usamos otras necesarias para usar las anotaciones de Spring y las consultas a PostgreSQL.

3.5.2.1 Servidor.java

En primer lugar, tenemos el fichero que arranca el servicio web. Esta es su única labor, lanzar la aplicación de Spring para poder recibir las peticiones.

```
public class Servidor extends SpringBootServletInitializer{

    public static void main(String[] args) {
        SpringApplication.run(Servidor.class, args);
    }

}
```

Figura 3-50. Clase Servidor

3.5.2.2 AgenteRestController.java

Este fichero hará las funciones del controlador para las llamadas del agente sanitarios. Así, en este fichero encontraremos todos los métodos necesarios para que este pueda desempeñar todas sus funciones.

En la Tabla 3-15 tenemos el método usado para que el agente pueda acceder a la aplicación móvil.

Tabla 3–15. Método “accederAgente” del controlador del agente

Método	accederAgente
Descripción	Método que invoca el agente cuando intenta iniciar sesión. Llama al método “accederAgente” de la capa de servicio.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

También serán necesarios varios métodos para registrar al agente en el sistema.

Tabla 3–16. Método “regAgente” del controlador del agente

Método	regAgente
Descripción	Método que invoca el agente cuando se quiere registrar en el sistema. Llama al método “regAgente” de la capa de servicio.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Practicante (String, obligatorio): datos con los que el agente sanitario quiere registrarse en la aplicación los cuales son pasados como String
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Tabla 3–17. Método “reghospital” del controlador del agente

Método	reghospital
Descripción	Método que invoca el agente cuando se quiere registrar en el sistema. Llama al método “reghospital” de la capa de servicio para guardar el hospital u organización al que pertenece el agente.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Hosp (String, obligatorio): datos sobre el hospital al que pertenece el agente
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Tabla 3–18. Método “regdepart” del controlador del agente

Método	regdepart
Descripción	Método que invoca el agente cuando se quiere registrar en el sistema. Llama al método “regdepart” de la capa de servicio para guardar el departamento al que pertenece el agente.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Depart (String, obligatorio): datos sobre el departamento al que pertenece el agente
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Como venimos comentado, el agente solicitará consentimientos a los ciudadanos para poder usar sus datos personales con alguna finalidad. Cuando haga esto, llamará al método “solconsent”.

Tabla 3–19. Método “solconsent” del controlador del agente

Método	solconsent
Descripción	Método que se invoca cuando el agente crea una solicitud de consentimiento. Llama al método “solconsent” para crear esa solicitud.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener su DNI.
	Consentim (String, obligatorio): solicitud de consentimiento creada por el agente y pasada como String
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

De igual forma, este agente podrá ver los consentimientos que haya solicitado según el estado en el que se encuentre.

Tabla 3–20. Método “getconsentestado” del controlador del agente

Método	getconsentestado
Descripción	Método que invoca el agente para obtener los consentimiento que estén en un determinado estado y que él ha solicitado. Llama al método “getconsentestado” de la capa de servicio para obtener estos consentimientos.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener su DNI.
	Estado (String, obligatorio): estado de los consentimientos solicitados
Respuesta	Lista (String): se trata de una lista con todos los consentimientos que ha creado y que se encuentran en el estado indicado pasado como String.

Cuando se obtengan los consentimientos también será necesario identificar el hospital en el que se encuentran almacenados los datos. Para ello se usará el método “gethospital” el cual se muestra en la Tabla 3-21.

Tabla 3–21. Método “gethospital” del controlador del agente

Método	gethospital
Descripción	Método que se invoca para obtener el hospital al que pertenece el agente. Llama al método “getHospital” de la capa de servicio para obtener el hospital u organización.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener su DNI.
Respuesta	Hospital (String): se trata del nombre del hospital al que pertenece el agente

Otra función que puede realizar el agente es la de eliminar un consentimiento. Puede darse la situación en la que este se haya equivocado al completar el formulario del consentimiento o que simplemente ya no necesite ese consentimiento.

Tabla 3–22. Método “eliminarConsent” del controlador del agente

Método	eliminarConsent
Descripción	Método que invoca el agente para eliminar un consentimiento que solicitó previamente. Llama al método “eliminarConsent” de la capa de servicio para ejecutar este borrado.
Parámetros de entrada	Consentimiento (String, obligatorio): consentimiento que el agente quiere eliminar pasado como String
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

3.5.2.3 CiudRestController.java

También tenemos el fichero que hace las funciones del controlador para las llamadas del ciudadano. De esta forma, este fichero contendrá todos los métodos necesarios para que este pueda desempeñar todas sus funciones.

Cuando un ciudadano quiera acceder a la aplicación móvil se llamará al método que se muestra en la Tabla 3-23.

Tabla 3–23. Método "accederCiud" del controlador del ciudadano

Método	accederCiud
Descripción	Método que invoca el ciudadano cuando intenta iniciar sesión. Llama al método "accederCiud" de la capa de servicio.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano que lo usa a modo de contraseña para acceder a la aplicación
Respuesta	Cod (String): se trata de códigos como "200" o "300" que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

A diferencia del registro del agente, para un ciudadano solo necesitaremos un método.

Tabla 3–24. Método "regCiud" del controlador del ciudadano

Método	regCiud
Descripción	Método que invoca el ciudadano cuando se quiere registrar en el sistema. Llama al método "regCiud" de la capa de servicio.
Parámetros de entrada	Dni (String, obligatorio): DNI que el agente usa para acceder a la aplicación como contraseña
	Paciente (String, obligatorio): datos con los que el ciudadano quiere registrarse en la aplicación los cuales son pasados como String
Respuesta	Cod (String): se trata de códigos como "200" o "300" que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Evidentemente, el ciudadano necesitará ver los consentimientos que le han solicitado. Esto lo podrá hacer en función del estado en el que se encuentre el consentimiento ya que este podrá otorgarlos, rechazarlos o posponerlos.

Tabla 3–25. Método “getconsentestado” del controlador del ciudadano

Método	getconsentestado
Descripción	Método que invoca el ciudadano para obtener los consentimiento que estén en un determinado estado y que le haya llegado. Llama al método “getconsentestado” de la capa de servicio para obtener estos consentimientos.
Parámetros de entrada	Dni (String, obligatorio): DNI con el que el ciudadano accede a la aplicación.
	Estado (String, obligatorio): estado de los consentimientos solicitados
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Como hemos comentado, el ciudadano podrá modificar el estado de los consentimientos según decida concederlos o no.

Tabla 3–26. Método “modificarConsent” del controlador del ciudadano

Método	modificarConsent
Descripción	Método que invoca el ciudadano para modificar el estado del consentimiento. Llama al método “modificarConsent” de la capa de servicio para realizar la modificación.
Parámetros de entrada	Estado (String, obligatorio): estado al que quiere cambiarse el consentimiento.
	Consentimiento (String, obligatorio): consentimiento al que quiere cambiársele el estado.
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Cuando el ciudadano accede a la aplicación ve las nuevas solicitudes de consentimientos que le han llegado. Estas tendrán el aviso que hemos comentado activado. Obtendremos estos consentimientos mediante el método “getconsentAvisos”.

Tabla 3–27. Método “getconsentAvisos” del controlador del ciudadano

Método	getconsentAvisos
Descripción	Método que se invoca cuando el ciudadano accede a la aplicación para obtener las solicitudes de consentimientos nuevas. Llama al método “getavisosCiud” de la capa de servicio.
Parámetros de entrada	Dni (String, obligatorio): DNI con el que el ciudadano accede a la aplicación.
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

Una vez el ciudadano vea estos nuevos consentimientos, el aviso de estos tendrá que se desactivado. Para ello usamos el método mostrado en la Tabla 3-28.

Tabla 3–28. Método “actualizarAviso” del controlador del ciudadano

Método	actualizarAviso
Descripción	Método que se invoca cuando el ciudadano ve un consentimiento por primera vez. Llama al método “modificarConsent” de la capa de servicio para realizar la modificación.
Parámetros de entrada	Consentimiento (String, obligatorio): consentimiento al que se le modificará su aviso.
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

De igual forma que con el agente, necesitaremos identificar el hospital en el que se encuentran almacenados los datos. Para ello se usará el método “gethospital.

Tabla 3–29. Método “gethospital” del controlador del ciudadano

Método	gethospital
Descripción	Método que se invoca para obtener el hospital al que pertenece el agente. Llama al método “getHospital” de la capa de servicio para obtener el hospital u organización.
Parámetros de entrada	Dni (String, obligatorio): DNI del agente con el que se obtendrá su hospital.
Respuesta	Cod (String): se trata de códigos como “200” o “300” que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

El ciudadano tiene que saber quién le ha solicitado el consentimiento. Por ello, su nombre deberá aparecer en

la información del consentimiento cuando el ciudadano esté viéndolo. Esto se consigue con el método "getNombreAg" que se muestra en la Tabla 3-30.

Tabla 3–30. Método "getNombreAg" del controlador del ciudadano

Método	getNombreAg
Descripción	Método que se invoca para obtener el nombre del agente a partir de su DNI. Llama al método "getNombre" de la capa de servicio para obtenerlo.
Parámetros de entrada	Dni (String, obligatorio): DNI del agente que solicitó el consentimiento. Sirve para obtener su nombre.
Respuesta	Cod (String): se trata de códigos como "200" o "300" que se usan de igual forma que el protocolo HTTP: uno indicará que el acceso es correcto, otro que la contraseña es incorrecta, etc.

3.5.2.4 ImplAgenteService.java

Todos los métodos del controlador implementado para los agentes llamarán a métodos de esta clase ya que este fichero hace las labores de capa de servicio para el agente. De esta forma, la mayor parte de la lógica necesaria para el servicio web por parte del agente las encontraremos aquí.

Cuando el agente quiera acceder, el método usado en el controlador llamará al método "accederAgente" mostrado en la Tabla 3-31.

Tabla 3–31. Método "accederAgente" del servicio del agente

Método	accederAgente
Descripción	Método que llama al método "accederAgente" de la capa DAO para obtener los datos del agente. Comprueba que la información obtenida sea correcta y decide el código a enviar.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
Respuesta	Cod (String): código "400" cuando el agente no existe.
	Cod (String): código "300" cuando el agente no se ha registrado pero la contraseña existe para un futuro usuario.
	Cod (String): código "200" cuando el agente accede correctamente.

Como se indicó en otro apartado anterior, se llamará a varios métodos cuando el agente quiera registrase. Estos a su vez tendrán que llamar a los métodos indicados de la capa de servicio.

Tabla 3–32. Método “regAgente” del servicio del agente

Método	regAgente
Descripción	Método que comprueba si el agente ya está registrado o siquiera existen los datos con los que quiere registrase. Además, llama al método “actualizarAg” de la capa DAO y decide el código a enviar.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Practicante (String, obligatorio): datos con los que el agente sanitario quiere registrarse en la aplicación los cuales son pasados como String
Respuesta	Cod (String): código “400” cuando el agente no existe.
	Cod (String): código “200” cuando el agente no se ha registrado pero la contraseña existe para un futuro usuario.
	Cod (String): código “500” cuando el código usado para registrarse es erróneo.
	Cod (String): código “300” cuando el agente se registra correctamente.
	Cod (String): código “100” cuando ha habido algún otro tipo de error

Tabla 3–33. Método “reghospital” del servicio del agente

Método	reghospital
Descripción	Método que comprueba si el nombre del hospital es correcto y llama al método “actualizarHospital” de la capa DAO. Además, decide el código a enviar.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Hosp (String, obligatorio): datos sobre el hospital al que pertenece el agente
Respuesta	Cod (String): código “600” cuando el nombre del hospital es incorrecto.
	Cod (String): código “200” cuando el hospital es correcto

Tabla 3–34. Método “regdepart” del servicio del agente

Método	regdepart
Descripción	Método que comprueba si el departamento es correcto y llama al método “actualizarDepart” de la capa DAO. Además, decide el código a enviar.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
	Depart (String, obligatorio): datos sobre el departamento al que pertenece el agente
Respuesta	Cod (String): código “600” cuando el departamento es incorrecto.
	Cod (String): código “200” cuando el departamento es correcto.

Cuando el agente quiera solicitar un consentimiento se llama al método “solconsent”.

Tabla 3–35. Método “solconsent” del servicio del agente

Método	solconsent
Descripción	Método que llamará al método “crearconsent” para crear el la solicitud de consentimiento. Obtendrá el DNI del agente a partir de su login. Comprobará si es una solicitud realizada a un único ciudadano o a todos y creará el consentimiento o consentimientos en consecuencia. Además, decidirá el código a enviar.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener su DNI.
	Consentimiento (String, obligatorio): solicitud de consentimiento creada por el agente y pasada como String
Respuesta	Cod (String): código “500” cuando hay un error en la creación de solicitudes de consentimientos de todos los ciudadanos.
	Cod (String): código “200” cuando se han creado correctamente solicitudes de consentimientos de todos los ciudadanos o de alguno en concreto.
	Cod (String): código “400” cuando la solicitud de consentimiento se le quiere enviar un ciudadano que no existe.
	Cod (String): código “100” cuando la solicitud de consentimiento se le quiere enviar un ciudadano cuyos datos existen pero no está registrado.

Para ver el listado de los consentimientos creados por el agente se llamará al método “getconsentestado” el cual podemos encontrar en la Tabla 3-36.

Tabla 3–36. Método “getconsentestado” del servicio del agente

Método	getconsentestado
Descripción	Método que llamará al método “getconsent” para obtener los consentimientos creados por el agente en un determinado estado. Para ello, obtendrá la contraseña del agente a partir de su login y con ella obtendrá los ids de los consentimientos con los que podremos identificarlos.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener su DNI.
	Estado (String, obligatorio): estado de los consentimientos solicitados
Respuesta	Listaconsentimientos (List<Consen>): se trata de una lista con todos los consentimientos que ha creado y que se encuentran en el estado indicado.

También se deberá identificar el hospital al que pertenece el agente sanitario.

Tabla 3–37. Método “gethospital” del servicio del agente

Método	gethospital
Descripción	Método que se invoca para obtener el hospital al que pertenece el agente. Llama al método “getAgLogin” de la capa DAO para obtener el hospital u organización.
Parámetros de entrada	Login (String, obligatorio): login con el que el agente accede a la aplicación. Sirve para obtener los datos del agente.
Respuesta	Hospital (String): se trata del nombre del hospital al que pertenece el agente

Al eliminar un consentimiento será necesaria otra gestión en la capa de servicio. De esta se encargará el método “eliminarConsent” mostrado en la Tabla 3-38.

Tabla 3–38. Método “eliminarConsent” del servicio del agente

Método	eliminarConsent
Descripción	Método que llama al método “eliminarConsent” de la capa DAO para eliminar el consentimiento y decidirá el código enviado en función de la respuesta al borrado.
Parámetros de entrada	Consentimiento (String, obligatorio): consentimiento que el agente quiere eliminar pasado como String
Respuesta	Cod (String): código “400” cuando ha ocurrido un error en la eliminación del consentimiento.
	Cod (String): código “200” cuando el consentimiento se ha borrado correctamente.

3.5.2.5 ImplCiudService.java

Los métodos del controlador implementado para los ciudadanos llamarán a métodos de esta clase para las labores de capa de servicio del ciudadano. Así, tal y como ocurre con el caso del agente, la mayor parte de la lógica necesaria para el funcionamiento del servicio web para el ciudadano se encuentra en esta clase.

El ciudadano tendrá su propio método en la capa de servicio para acceder a la aplicación móvil.

Tabla 3–39. Método "accederCiud" del servicio del ciudadano

Método	accederCiud
Descripción	Método que llama al método "accederCiud" de la capa DAO para obtener los datos del ciudadano. Comprueba que la información obtenida sea correcta y decide el código a enviar.
Parámetros de entrada	Dni (String, obligatorio): DNI que el ciudadano usa para acceder a la aplicación
Respuesta	Cod (String): código "400" cuando el ciudadano no existe.
	Cod (String): código "300" cuando el ciudadano no se ha registrado pero se puede registrar en el futuro.
	Cod (String): código "200" cuando el ciudadano accede correctamente.

Como ya comentamos, solo necesitará un método en esta capa para realizar el registro.

Tabla 3–40. Método “regCiud” del servicio del ciudadano

Método	regCiud
Descripción	Método que comprueba si el ciudadano ya está registrado o siquiera existen los datos con los que quiere registrase. Además, llama al método “actualizarCiud” de la capa DAO y decide el código a enviar.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano usado para acceder a la aplicación
	Paciente (String, obligatorio): datos con los que el ciudadano quiere registrarse en la aplicación los cuales son pasados como String
Respuesta	Cod (String): código “400” cuando el ciudadano no existe.
	Cod (String): código “200” cuando el ciudadano no se ha registrado pero se puede registrar en el futuro.
	Cod (String): código “500” cuando el número de la tarjeta sanitaria usado para registrarse es erróneo.
	Cod (String): código “300” cuando el ciudadano se registra correctamente.
	Cod (String): código “100” cuando ha habido algún otro tipo de error

El ciudadano obtendrá los consentimientos que se le han solicitado de forma muy parecida a como el agente ve los que ha solicitado. Para ello se usa el método mostrado en la Tabla 3-41.

Tabla 3–41. Método “getconsentestado” del servicio del ciudadano

Método	getconsentestado
Descripción	Método que llamará al método “getconsent” para obtener los consentimientos solicitados al ciudadano en un determinado estado. Para ello, se obtendrán los ids de los consentimientos con los que podremos identificarlos a través del DNI del ciudadano.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano usado para acceder a la aplicación
	Estado (String, obligatorio): estado de los consentimientos solicitados
Respuesta	Listaconsentimientos (List<Consen>): se trata de una lista con todos los consentimientos que se le han solicitado al ciudadano y que se encuentran en el estado indicado.

Para cambiar el estado de los consentimientos se llamará al método “modificarConsent”.

Tabla 3–42. Método “modificarConsent” del servicio del ciudadano

Método	modificarConsent
Descripción	Método que llamará al método “modificarConsent” de la capa DAO para cambiar el estado del consentimiento al seleccionado por el usuario. Además, decidirá el código a enviar.
Parámetros de entrada	Estado (String, obligatorio): estado al que cambiará el consentimiento
	Consentimiento (String, obligatorio): consentimiento que el ciudadano quiere modificar
Respuesta	Cod (String): código “200” cuando el estado del consentimiento se ha modificado correctamente.
	Cod (String): código “400” cuando ha habido un error al modificar el estado del consentimiento.

Otra modificación que se realiza sobre el consentimiento es la del aviso. Como se ha comentado, cuando este sea visto por primera vez por el ciudadano tendrá que desactivarse.

Tabla 3–43. Método “actualizarAviso” del servicio del ciudadano

Método	actualizarAviso
Descripción	Método que llamará al método “actualizarAviso” de la capa DAO para eliminar el aviso en el consentimiento. Además, decidirá el código a enviar.
Parámetros de entrada	Consentimiento (String, obligatorio): consentimiento al que se le quiere quitar el aviso
Respuesta	Cod (String): código “200” cuando el aviso del consentimiento se ha eliminado correctamente.
	Cod (String): código “400” cuando ha habido un error al eliminar el aviso del consentimiento.

Antes de cambiar el aviso tendrán que verse los consentimientos nuevos. Para ello se usa el método “getavisosCiud” mostrado en la Tabla 3-44.

Tabla 3–44. Método “getavisosCiud” del servicio del ciudadano

Método	getavisosCiud
Descripción	Método que llamará al método “getconsent” para obtener los consentimientos solicitados al ciudadano. Después filtrará por aquellos que tengan el aviso activado. Para ello, se obtendrán los ids de los consentimientos con los que podremos identificarlos a través del DNI del ciudadano.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano usado para acceder a la aplicación
Respuesta	Listaconsentimientos (List<Consen>): se trata de una lista con todos los consentimientos que se le han solicitado al ciudadano y que tengan el aviso activado.

Será necesario saber cuál es el hospital en el que se encuentran los datos.

Tabla 3–45. Método “gethospital” del servicio del ciudadano

Método	gethospital
Descripción	Método que se invoca para obtener el hospital al que pertenece el agente. Llama al método “getAgDNI” de la capa DAO para obtener el hospital u organización a partir del DNI del agente.
Parámetros de entrada	Dni (String, obligatorio): DNI del agente que servirá para identificar el hospital al que pertenece.
Respuesta	Hospital (String): se trata del nombre del hospital al que pertenece el agente

Por último, el ciudadano necesitará saber quién le ha solicitado el consentimiento. Esto se logrará a través del método “getNombre”.

Tabla 3–46. Método “getNombre” del servicio del ciudadano

Método	getNombre
Descripción	Método que se invoca para obtener el nombre del agente. Llama al método “getAgDNI” de la capa DAO para obtener la información del agente.
Parámetros de entrada	Dni (String, obligatorio): DNI del agente que servirá para obtener su información.
Respuesta	Agentes (Agentes): objeto implementado con la información del agente

3.5.2.6 ImplAgenteDAO.java

Como ya comentamos, la capa DAO se encargará de realizar todas las llamadas necesarias a las bases de datos. En este fichero encontraremos todos los métodos usados en consultas que tratan datos de los agentes.

De esta forma, para comprobar si el acceso a la aplicación móvil es correcto necesitaremos un método que realice una llamada.

Tabla 3–47. Método "accederAgente" de la capa DAO del agente

Método	accederAgente
Descripción	Método que lanzará una consulta a la base de datos PostgreSQL para obtener los datos del agente con el que se quiere acceder. Para identificarlo se usará la contraseña con la que quiere acceder.
Parámetros de entrada	Contra (String, obligatorio): contraseña que el agente usa para acceder a la aplicación
Respuesta	Agente (Agentes): devuelve el objeto implementado para los agentes.

Cuando un agente sanitario se registre tendremos que actualizar los datos de la base de datos PostgreSQL así como crear los recursos necesarios en la base de datos FHIR.

Tabla 3–48. Método "actualizarAg" de la capa DAO del agente

Método	actualizarAg
Descripción	Método que se encarga de crear el recurso *Practitioner* extendido en la base de datos FHIR y actualiza la de PostgreSQL con el nuevo agente.
Parámetros de entrada	Prac (Practicante, recurso *Practitioner* extendido): recurso usado para el agente sanitario con los datos con los que quiere registrarse en la aplicación.
	Agente (Agentes): objeto implementado para los agentes en la base de datos PostgreSQL.
Respuesta	De tipo Void, no envía respuesta.

Tabla 3–49. Método "actualizarHospital" de la capa DAO del agente

Método	actualizarHospital
Descripción	Método que se encarga de crear el recurso *Organization* en la base de datos FHIR.
Parámetros de entrada	Hospital (Recurso *Organization*): recurso usado para la organización a la que pertenece el agente sanitario.
Respuesta	De tipo Void, no envía respuesta.

Tabla 3–50. Método “actualizarDepart” de la capa DAO del agente

Método	actualizarDepart
Descripción	Método que se encarga de crear el recurso *PractitionerRole* en la base de datos FHIR.
Parámetros de entrada	Depart (Recurso *PractitionerRole*): recurso usado para el departamento al que pertenece el agente sanitario y conectarlo tanto con él como con su organización.
Respuesta	De tipo Void, no envía respuesta.

Dependiendo de la situación, tendremos que obtener los datos del agente en función de su login o de su DNI. Para ello tenemos los métodos mostrados en la Tabla 3-51 y la Tabla 3-52.

Tabla 3–51. Método “getAgLogin” de la capa DAO del agente

Método	getAgLogin
Descripción	Método que lanzará una consulta a la base de datos PostgreSQL para obtener los datos del agente con el que se quiere acceder a partir de su login.
Parámetros de entrada	Login (String, obligatorio): login que usa el agente para acceder a la aplicación
Respuesta	Agente (Agentes): devuelve el objeto implementado para los agentes.

Tabla 3–52. Método “getAgDNI” de la capa DAO del agente

Método	getAgDNI
Descripción	Método que lanzará una consulta a la base de datos PostgreSQL para obtener los datos del agente con el que se quiere acceder a partir de su DNI.
Parámetros de entrada	Dni (String, obligatorio): DNI del agente sanitario con el que para acceder a la aplicación
Respuesta	Agente (Agentes): devuelve el objeto implementado para los agentes.

3.5.2.7 ImplCiudDAO.java

En este fichero encontraremos los métodos usados en consultas que tratan datos de los ciudadanos. A diferencia del caso del agente, tendremos menos métodos que en otros ficheros.

Así, como en el caso del agente, tenemos un método para el acceso del ciudadano.

Tabla 3–53. Método “accederCiud” de la capa DAO del ciudadano

Método	accederCiud
Descripción	Método que lanzará una consulta a la base de datos PostgreSQL para obtener los datos del ciudadano con el que se quiere acceder. Para identificarlo se usará su DNI.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano usado para acceder a la aplicación
Respuesta	Ciud (Ciudadanos): devuelve el objeto implementado para los ciudadanos.

También necesitaremos actualizar los datos del ciudadano cuando este se registre en el sistema.

Tabla 3–54. Método “actualizarCiud” de la capa DAO del ciudadano

Método	actualizarCiud
Descripción	Método que se encarga de crear el recurso *Patient* extendido en la base de datos FHIR y actualiza la de PostgreSQL con el nuevo ciudadano registrado.
Parámetros de entrada	Paciente (Paciente, recurso *Patient* extendido): recurso usado para el ciudadano con los datos con los que quiere registrarse en la aplicación.
	Ciud (Ciudadanos): objeto implementado para los ciudadanos en la base de datos PostgreSQL.
Respuesta	De tipo Void, no envía respuesta.

Por último, cuando un agente quiera enviar la misma solicitud de consentimiento a todos los ciudadanos registrados necesitará una lista de ellos. Para ello se usará el método mostrado en la Tabla 3-55.

Tabla 3–55. Método “obtenerciuds” de la capa DAO del ciudadano

Método	obtenerciuds
Descripción	Método que lanza una consulta para obtener una lista de todos los ciudadanos registrados
Parámetros de entrada	Void, no recibe parámetros.
Respuesta	Listaciuds (List<Ciudadanos>): lista con todos los ciudadanos que se han registrado.

3.5.2.8 ImplConsentDAO.java

Los métodos usados para realizar cualquier llamada a las bases de datos que se relacionen con los consentimientos se encuentran en este fichero.

Cuando se quiere crear un consentimiento se tendrá que lanzar la petición a la base de datos FHIR junto a los datos del consentimiento. Para ello usaremos el método "crearconsent".

Tabla 3–56. Método "crearconsent" de la capa DAO de los consentimientos

Método	crearconsent
Descripción	Método que se encarga de crear un recurso *Consent* extendido en la base de datos FHIR y de guardar el ID del consentimiento en la base de datos PostgreSQL para identificarlo.
Parámetros de entrada	Agdni (String, obligatorio): DNI del agente para asociarlo al ID del consentimiento
	Ciuddni (String, obligatorio): DNI del ciudadano para asociarlo al ID del consentimiento
	Consentimiento(Consen, recurso *Consent* extendido): consentimiento que se quiere guardar
Respuesta	De tipo Void, no envía respuesta.

Los consentimientos serán recuperados a través de sus IDs. Se muestra el método usado para ello en la Tabla 3-57.

Tabla 3–57. Método "getconsent" de la capa DAO de los consentimientos

Método	getconsent
Descripción	Método que lanza una consulta a la base de datos FHIR para obtener un consentimiento a partir de su ID.
Parámetros de entrada	Id (String, obligatorio): ID del consentimiento a consultar.
Respuesta	Response (recurso *Bundle*): recurso que actúa de contenedor de otros recursos.

Antes de obtener los consentimientos será necesario obtener sus IDs. Estos son guardados en la base de datos PostgreSQL cuando son creados tal y como se ha comentado. Tanto los ciudadanos como los agentes podrán obtenerlos. La única diferencia es que el ciudadano usará su DNI para identificar los consentimientos asociados al mismo mientras que el agente sanitario usará su contraseña.

Tabla 3–58. Método “getidsconsentC” de la capa DAO de los consentimientos para los ciudadanos

Método	getidsconsentC
Descripción	Método que lanza una consulta a la base de datos PostgreSQL para obtener los IDs de los consentimientos asociados a un ciudadano. Para ello, usaremos su DNI.
Parámetros de entrada	Dni (String, obligatorio): DNI del ciudadano asociado a consentimientos.
Respuesta	Listaconsent (List<Consentimientos>): lista de objetos implementados para los consentimientos.

Tabla 3–59. Método “getidsconsentA” de la capa DAO de los consentimientos para los agentes

Método	getidsconsentA
Descripción	Método que lanza una consulta a la base de datos PostgreSQL para obtener los IDs de los consentimientos asociados a un ciudadano. Para ello, usaremos su contraseña.
Parámetros de entrada	Contra (String, obligatorio): contraseña del agente asociada a consentimientos.
Respuesta	Listaconsent (List<Consentimientos>): lista de objetos implementados para los consentimientos.

Lógicamente, para eliminar un consentimiento será necesario utilizar otro método. Este será el método “eliminarConsent” mostrado en la Tabla 3-60.

Tabla 3–60. Método “eliminarConsent” de la capa DAO de los consentimientos

Método	eliminarConsent
Descripción	Método que lanzará una petición a la base de datos FHIR para eliminar un consentimiento seleccionado y a otra a la base de datos PostgreSQL para borrar su ID almacenado.
Parámetros de entrada	Consentimiento (recurso *Consent* extendido): consentimiento que se quiere eliminar.
Respuesta	Outcome (MethodOutcome): respuesta recibida del servidor FHIR

De igual forma, serán necesarios otros métodos para modificar el estado de los consentimientos en función de lo que decida el ciudadano así como para quitarles el aviso una vez este los vea.

Tabla 3–61. Método “modificarConsent” de la capa DAO de los consentimientos

Método	modificarConsent
Descripción	Método que lanzará una petición a la base de datos FHIR para modificar su estado
Parámetros de entrada	Consentimiento (recurso *Consent* extendido): consentimiento que se quiere modificar.
Respuesta	De tipo Void, no envía respuesta.

Tabla 3–62. Método “actualizarAviso” de la capa DAO de los consentimientos

Método	actualizarAviso
Descripción	Método que lanzará una petición a la base de datos FHIR para quitar el aviso del consentimiento
Parámetros de entrada	Consentimiento (recurso *Consent* extendido): consentimiento al que se le quiere quitar el aviso.
Respuesta	De tipo Void, no envía respuesta.

3.6 Implementación de la aplicación móvil

Será en la aplicación móvil donde realmente veamos los resultados del trabajo realizado ya que será la herramienta mediante la que los usuarios se registren, soliciten los consentimientos y los gestionen. Como venimos comentando, las funciones para un ciudadano y para un agente sanitario serán diferentes. Por lo tanto, lo que le aparezca a cada uno en la aplicación tendrá ciertas diferencias.

3.6.1 Configuración previa

La aplicación móvil ha sido desarrollada en Android Studio. Para el correcto uso del entorno es necesario realizar alguna configuración del gradle de la aplicación. Habrá que indicar el kit de desarrollo de software (SDK) mínimo que usa la aplicación así como el que se pretende usar por defecto. Se trata de un conjunto de herramientas que permiten crear aplicaciones. Además de esto, el gradle está limitado en cuanto al número de métodos que puede cargar. Al superar los 65.536 métodos se produce un error de compilación que indica que se ha alcanzado el límite de la arquitectura de compilación de Android. Este número representa la cantidad total de referencias que el código puede invocar en un mismo archivo de código de bytes Dalvik Executable (DEX) [35]. Cuando añadimos las dependencias FHIR excedemos esta limitación de tal forma que no podríamos arrancar la aplicación. Podremos deshabilitar este límite a través de la configuración conocida como “multidex” la cual permite que la aplicación compile y lea varios archivos DEX. Por ello, tendremos que eliminar esta limitación mediante la línea de código “multiDexEnabled true”. Todo esto se puede apreciar en la Figura 3-51.

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    defaultConfig {
        applicationId "com.dam.gestorconsentimientos"
        minSdkVersion 16
        targetSdkVersion 29
        multiDexEnabled true
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
```

Figura 3-51. Configuración del gradle

Además de esto, tendremos que incluir una dependencia de biblioteca de compatibilidad la cual pasa a formar parte del archivo DEX de la aplicación y administra el acceso a los archivos DEX adicionales. Esta dependencia se muestra en la Figura 3-52.

```
implementation 'com.android.support:multidex:1.0.3'
```

Figura 3-52. Dependencia Multidex

De forma adicional, tendrá que incluirse un método en la clase con la que se arranca la aplicación el cual habilita multidex.

```
@Override
protected void attachBaseContext(Context base) {
    super.attachBaseContext(base);
    MultiDex.install( context: this);
}
```

Figura 3-53. Instalación de Multidex

Esta dependencia se suma a las de la propia aplicación de Android. Además, la especificación FHIR aporta una librería para Android con la que podremos usar sus recursos. También habrá que excluir explícitamente ciertas librerías que usan contenedores prohibidos en Android [36].

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.annotation:annotation:1.0.2'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
    implementation 'com.android.volley:volley:1.1.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'ca.uhn.hapi.fhir:hapi-fhir-android:5.1.0'
    implementation 'ca.uhn.hapi.fhir:hapi-fhir-structures-r4:5.1.0'
    implementation 'com.android.support:multidex:1.0.3'

}

configurations {
    all*.exclude group: 'org.codehaus.woodstox'
    all*.exclude group: 'org.apache.httpcomponents'
}
```

Figura 3-54. Dependencias y exclusiones de la aplicación

Hay otro detalle que debemos tener en cuenta para el correcto funcionamiento de la aplicación. Como esta lanzará peticiones a nuestro servicio web necesitará tener permiso para acceder a Internet. Para ello, tendremos que otorgar este permiso en el fichero AndroidManifest.xml. Este archivo describe información esencial de la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play [37]. En la Figura 3-55 se muestra la línea de código que permite el uso de Internet.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Figura 3-55. Permiso de Internet en el AndroidManifest.xml

3.6.2 Funcionamiento

En esta subsección veremos finalmente cómo funciona el sistema completo a través de los resultados que se muestran en la aplicación móvil. Así, lo primero que nos encontraremos será un primer layout de la aplicación el cual corresponderá con la pantalla de login.

Figura 3-56. Pantalla de login de la aplicación móvil para un ciudadano

Esta sería la pantalla que aparece cuando se accede a la pantalla de login. Por defecto, sería la correspondiente a un ciudadano ya que se le solicita tanto su usuario como su DNI para acceder al sistema. Como se puede ver, podemos elegir entre entrar como ciudadano o como agente solicitante. Cuando seleccionamos hacerlo como agente se nos pedirá la contraseña con la que accede el agente en vez del DNI. Además, en ambas situaciones tendremos en la parte inferior la dirección URL de la AEPD a la que podemos acceder para conocer los derechos.

Figura 3-57. Pantalla de login de la aplicación móvil para un agente

Si hacemos click en el botón de iniciar sesión con los datos correctos accederemos al sistema mientras que si pulsamos el otro botón veremos la pantalla de registro.

De la misma manera, podremos elegir entre registrarnos como ciudadanos y como agentes. En el primero de ellos se solicitará el nombre completo, un usuario de 8 letras, el DNI, el número de la tarjeta sanitaria y el número de teléfono.

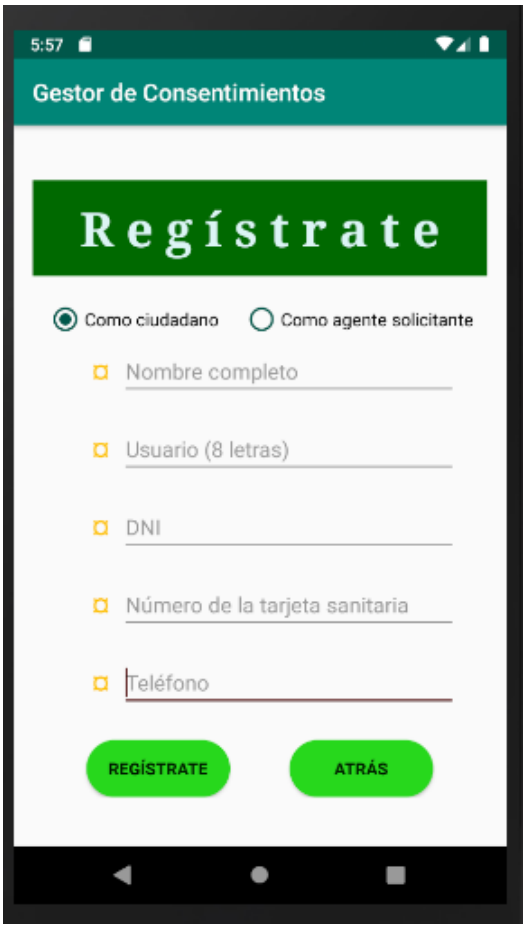

Figura 3-58. Pantalla de registro de la aplicación móvil para un ciudadano

Completamos estos datos para registrarnos en el sistema. No podemos olvidar que debe existir el DNI y el número de tarjeta sanitaria para poder registrarse. Si hubiera algún error en el registro se muestran los siguientes mensajes:

- "Persona inexistente", si el DNI no existe.
- "Usuario ya registrado", si como indica el propio mensaje, se ha registrado alguien con ese DNI y tarjeta sanitaria.
- "Número de tarjeta sanitaria incorrecto", si el número introducido no coincide con el asociado al DNI.
- "Error al registrarse", si ocurre cualquier otro tipo de error.

De esta forma, procedemos a realizar un registro de ciudadano:

Figura 3-59. Registro de ciudadano

Cuando el usuario se registre correctamente se le mostrará un mensaje indicando que lo ha hecho y pasará a una pantalla en la que podrá seleccionar el tipo de consentimiento que quiere ver según su estado.

Figura 3-60. Menú de estados de ciudadano

Llegado a este punto, cerraremos sesión para ver el registro como un agente sanitario. A este se le solicitarán más campos que a un ciudadano. Estos serán el nombre completo, un usuario de 8 letras, la contraseña con la que accede, el DNI, el hospital u organización al que pertenece, el departamento al que corresponde y un código. Para nuestro escenario suponemos que este usuario es dado por la administración de una organización de tal forma que para que este empleado se registre necesitará una contraseña y un código dados por la misma.

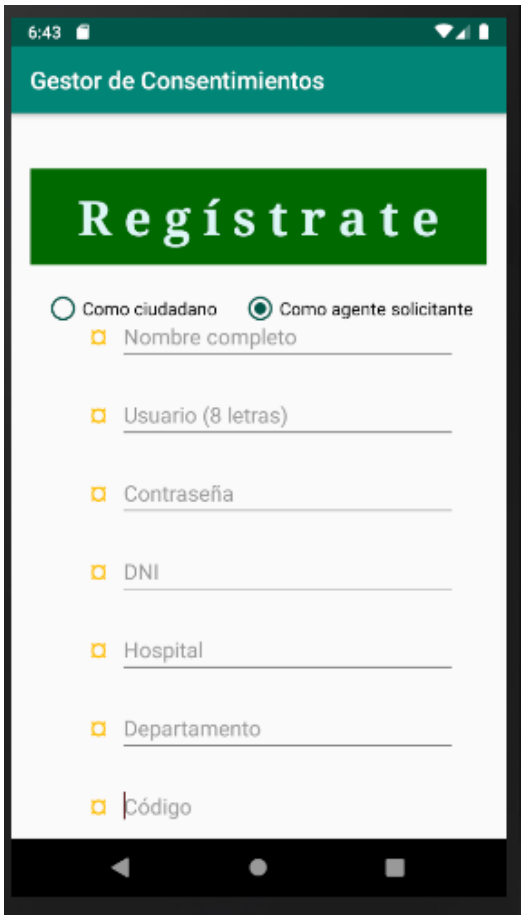

Figura 3-61. Pantalla de registro de la aplicación móvil para un agente sanitario

Completamos los datos. Si hubiera algún error en el registro se muestran los siguientes mensajes:

- "Persona inexistente", si la contraseña no existe en el sistema.
- "Usuario ya registrado", si como indica el propio mensaje, se ha registrado alguien con esa contraseña y código.
- "Código de agente incorrecto", si el código introducido no coincide con asociado a la contraseña.
- "Hospital incorrecto para contraseña y código", si este no coindice con el que corresponderían la contraseña y el código.
- "Departamento incorrecto para contraseña y código", si este no coindice con el que corresponderían la contraseña y el código.
- "Error al registrarse", si ocurre cualquier otro tipo de error.

Así, procedemos a realizar un registro de agente:

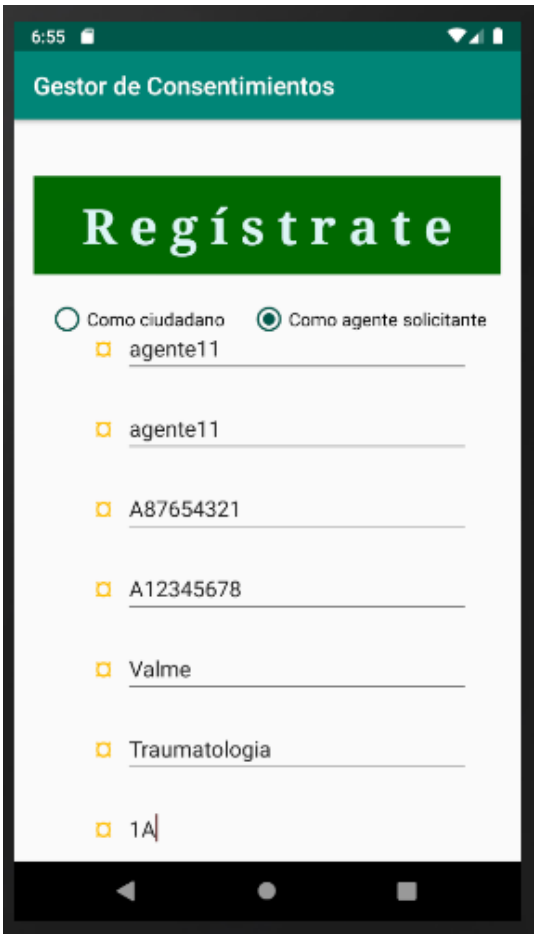

Figura 3-62. Registro de agente

Cuando el agente se registre correctamente se le mostrará un mensaje indicando que lo ha hecho y pasará a una pantalla en la que podrá seleccionar el tipo de consentimiento que quiere ver según su estado al igual que en el caso del ciudadano.

Figura 3-63. Menú de estados de agente

Puede darse la situación en la que un agente no tenga consentimientos solicitados en un estado determinado así como que al ciudadano no le aparezcan solicitudes. En este caso se muestra una imagen como la mostrada en la Figura 3-64.

Figura 3-64. Muestra de inexistencia de consentimientos en un estado determinado

Llegado este punto, el agente solicitante podrá solicitar un consentimiento a uno o varios ciudadanos. Para ello pulsará el botón “Nueva solicitud”. Entonces, se le mostrará una nueva pantalla en la que se le solicitará el usuario de los datos, la ubicación de los mismos, la categoría de los datos y los propios datos. Además, deberá seleccionar el tipo de acción a realizar y si quiere solicitar el consentimiento a un único ciudadano o a todos los que estén registrados. Por último, tendrá que indicar el motivo, durante cuánto tiempo se quiere disponer de los datos y de forma opcional las condiciones si hubiera.

Figura 3-65. Solicitud de consentimiento I

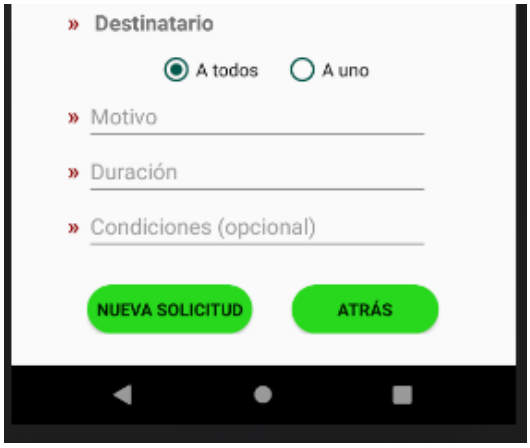

Figura 3-66. Solicitud de consentimiento II

Si se selecciona un único destinatario nos aparecerá un campo más para indicar el ciudadano.

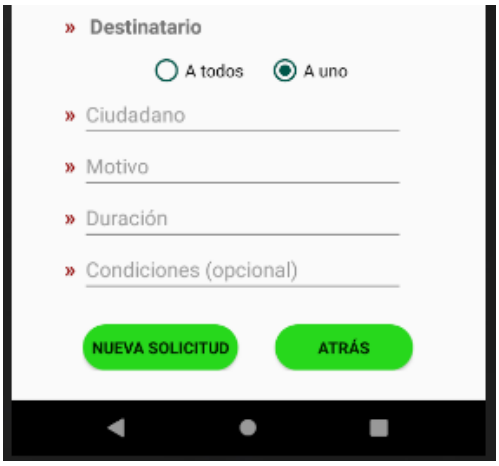

Figura 3-67. Solicitud de consentimiento III

En cuanto a las acciones podemos elegir entre las que se muestran en la Figura 3-68.

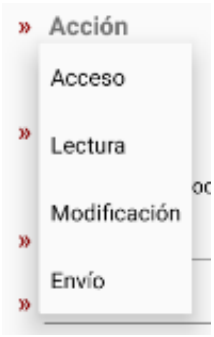

Figura 3-68. Desplegable del campo Acción

Procederemos a solicitar un consentimiento al ciudadano que se registró en este ejemplo. Para ello, seleccionaremos un único destinatario y pondremos su DNI. Además, la acción a realizar será de “Lectura”.

Figura 3-69. Consentimiento a solicitar I

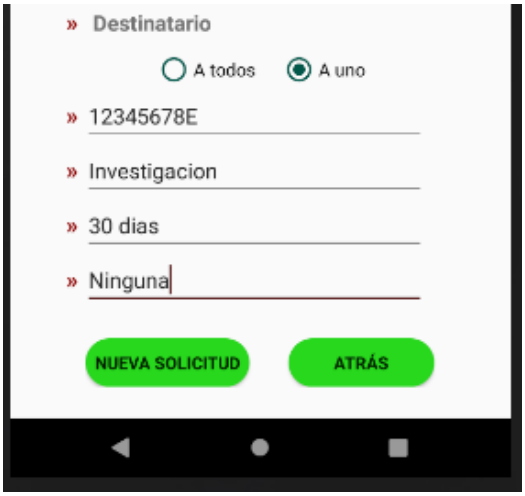

Figura 3-70. Consentimiento a solicitar II

Creamos este consentimiento y el agente será devuelto al menú de consentimientos por estados. Entonces, como el consentimiento acaba de crearse, deberá aparecerle al pulsar en los consentimientos pendientes.

Figura 3-71. Lista de consentimientos pendientes

Puede darse el caso de que haya algún error al crear el consentimiento. En este caso, se muestran los siguientes mensajes dependiendo del caso:

- "Ciudadano elegido inexistente", si el DNI del ciudadano no existe en el sistema.
- "Ciudadano elegido no registrado", si como indica el propio mensaje, no se ha registrado nadie con ese DNI.

Una vez que hemos solicitado un consentimiento pasamos a la parte del ciudadano. En primer lugar, este tendrá que acceder al sistema. Cuando lo haga se le mostrará una pantalla con los consentimientos nuevos ya que estos se mostrarán como si fuese un aviso. Como se ha solicitado un consentimiento al usuario registrado, esta solicitud deberá aparecerle. Cuando la vea tendrá dos opciones: continuar al menú antes mostrado o hacer click en el consentimiento. Si hace lo segundo la aplicación le llevará a otra pantalla en la que verá toda la información del consentimiento. Al ver el consentimiento por primera vez se eliminará este aviso de forma que ya solo lo verá en el menú, nunca al momento de acceder.

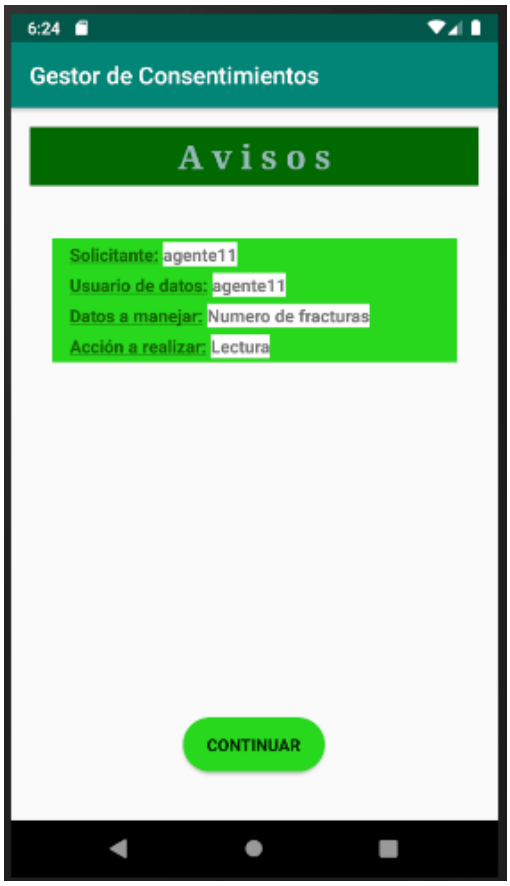

Figura 3-72. Consentimientos en avisos

Una vez el ciudadano accede al consentimiento tiene varias opciones. Podrá posponer la decisión de aceptar o no el consentimiento de tal forma que no cambiará su estado. Por el contrario, si lo otorga o rechaza, cambiará a su respectivo estado. Así, para poder verlo tendrá que seleccionar el estado al que haya cambiado el consentimiento.

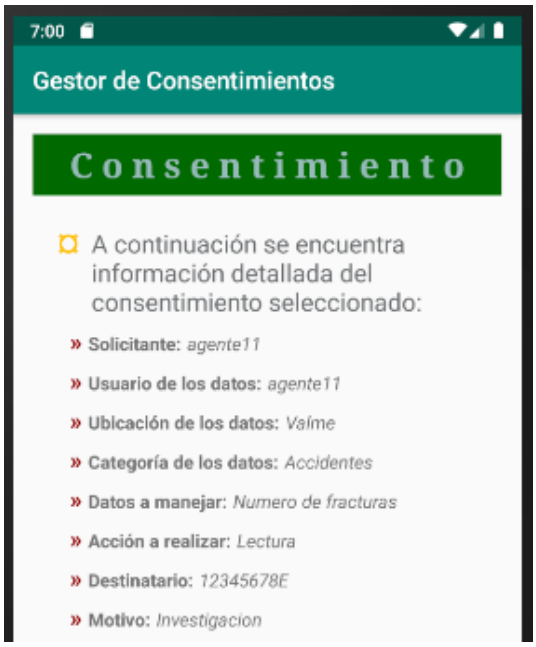

Figura 3-73. Información del consentimiento solicitado para ciudadano I

Figura 3-74. Información del consentimiento solicitado para ciudadano II

Como ya se ha visto el consentimiento, lo que aparecerá ahora en los avisos será lo mostrado en la Figura 3-75.

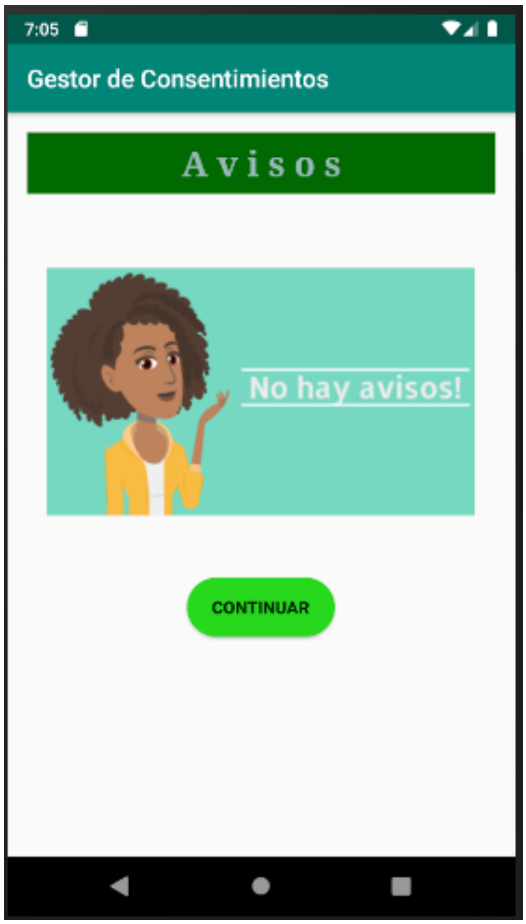

Figura 3-75. Consentimientos sin avisos

Como hemos comentado, los consentimientos podrán ser otorgados o rechazados. En las Figuras 3-76 y 3-77 se muestra cómo aparecerán estos consentimientos dependiendo del estado.

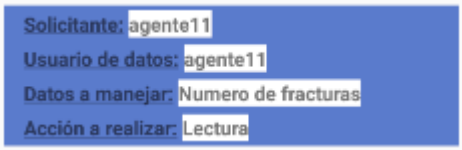

Figura 3-76. Muestra de consentimiento otorgado

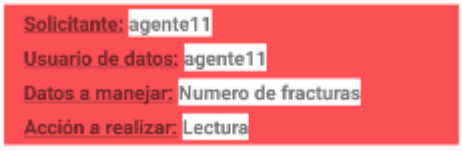

Figura 3-77. Muestra de consentimiento rechazado

En el caso del agente verá otros datos del consentimiento en este adelanto. Vemos un ejemplo en la Figura 3-78.

Figura 3-78. Ejemplo de consentimiento rechazado para agente

Ahora volvemos a la parte del agente para ver otra funcionalidad más. Cabe la posibilidad de que el agente se equivoque en algún dato al rellenar la solicitud de consentimiento o simplemente que quiera borrar un consentimiento porque ya no lo necesita. Por ello, se le da la posibilidad de eliminar un consentimiento. Tendrá que estar viendo la información del consentimiento para pulsar el botón habilitado para este objetivo.

Figura 3-79. Información del consentimiento para agente I

Figura 3-80. Información del consentimiento para agente II

Si eliminamos el consentimiento no aparecerá más y tampoco habrá forma de recuperarlo.

Por último, veremos un ejemplo de una solicitud de consentimiento a todos los ciudadanos registrados. Para ello, se registra otro usuario previamente. Como indicamos anteriormente, seleccionaremos como destinatario "A todos".

Figura 3-81. Selección de destinatario

De esta forma, ahora aparecerán tantos consentimientos pendientes como el número de ciudadanos que se hayan registrado.

Figura 3-82. Lista de consentimientos pendientes tras solicitar a todos los ciudadanos registrados

4 CONCLUSIONES Y LÍNEAS FUTURAS

No basta tener un buen ingenio, lo principal es aplicarlo bien.

- René Descartes -

Para este proyecto se han estudiado dos puntos necesarios para el diseño, desarrollo e implementación del gestor de consentimientos. Uno de ellos ha sido el GDPR del cual se ha identificado la información que debe estar presente en la solicitud de un consentimiento para entrar en su marco legal. Así, llegamos a la conclusión de que es necesario capturar qué ciudadano permite realizar una acción sobre determinados datos personales a un agente. Además, deben darse facilidades para que el ciudadano que use el gestor pueda ejercer cualquiera de sus derechos. El otro punto ha sido la especificación HL7 FHIR la cual ha aportado elementos muy útiles para trabajar sobre el contexto sanitario en el que se mueve nuestro sistema.

La especificación ha facilitado el modelado de los datos gracias a los recursos que define. De esta forma, identificamos la información necesaria para el funcionamiento del sistema y la adaptamos a los recursos existentes. Además, gracias a las extensiones disponibles añadimos la información que no esté predefinida en el recurso y que sea necesaria para el gestor. Por ello, nos han sido suficientes cinco recursos del estándar para contemplar todos los datos que requiere el sistema.

Esta especificación también aporta una API de código abierto llamada HAPI FHIR la cual puede implementarse tanto en un servidor o servicio web como en aplicaciones Android. Esta API facilita la conexión a servidores FHIR, el trabajo con los recursos y la creación de los mismos, entre otras funciones. De esta forma, nos ha permitido almacenar los recursos en un servidor público que creamos en la aplicación móvil para recuperarlos cuando sea necesario.

Por otro lado, debemos tener en cuenta que también presenta aspectos negativos. Este estándar se encuentra en continuo desarrollo por lo que la información obtenida en la actualidad podrá verse obsoleta pasado un tiempo debido a las nuevas versiones del mismo. Además, es necesario conocer la estructura de los recursos a usar. Esto provoca que deban estudiarse previamente para que trabajar con ellos no sea demasiado complejo en ciertos casos además de que su uso en varios elementos de un sistema debe ajustarse para el correcto funcionamiento.

4.1 Futuras líneas de desarrollo

Al finalizar el proyecto se alcanzó el punto en el cual cumplimos con los objetivos que se propusieron al inicio: diseñar, desarrollar e implementar un gestor de consentimientos de información sanitaria que trabaje bajo el marco del GDPR y conforme al estándar FHIR. No obstante, se presenta numerosas posibilidades en

cuanto a ampliaciones y mejoras:

- **Incluir mecanismos de seguridad en el sistema:** la seguridad es un punto importante en cualquier sistema y más aún en el contexto sanitario. En nuestro sistema no se ha implementado ningún mecanismo más allá de una simple validación de los datos introducidos por los usuarios ya que no es uno de los objetivos del trabajo. Sin embargo, sería recomendable implementar los mecanismos que propone el estándar FHIR. Estos son:
    - Intercambio de datos cifrado mediante TLS/SSL.
    - Autenticación de usuarios así como control de autorización/acceso de los usuarios
    - Uso de firmas digitales

- **Adición de todos los recursos FHIR:** como se ha comentado, solo hemos usado los cinco recursos que han sido necesarios para lograr el objetivo del proyecto. Sería interesante incluir todos los recursos de los que dispone la especificación y con ellos añadir nuevas funcionalidades al sistema. Además, se podrían aprovechar todos los campos de los recursos de forma que la información con la que se trabaje sea mucho más completa.

- **Implementación de nuevas funcionalidades:** los nuevos recursos que se podrían añadir darían pie a otras funcionalidades. Además, se podrían añadir otras nuevas relacionadas con los consentimientos como el número de veces que se ha accedido al consentimiento o a los datos a los que la aceptación de este da acceso, un registro de los cambios de estado que han sufrido los consentimientos o de los consentimientos que han sido eliminados por el profesional así como otro registro para almacenar la fecha y hora del momento en el que se ha accedido al consentimiento.

REFERENCIAS

[1] «Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.,» [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673. [Último acceso: 3 Mayo 2021].

[2] «Ejerce tus derechos | AEPD,» [En línea]. Available: https://www.aepd.es/es/derechos-y-deberes/conoce-tus-derechos. [Último acceso: 20 Abril 2021].

[3] «HL7 FHIR Resource,» [En línea]. Available: https://www.hl7.org/fhir/resource.html. [Último acceso: 25 Mayo 2021].

[4] P. Europeo, «Boletín Oficial del Estado,» 27 Abril 2016. [En línea]. Available: https://www.boe.es/doue/2016/119/L00001-00088.pdf. [Último acceso: 19 Junio 2021].

[5] «Real Academia Española,» [En línea]. Available: https://dpej.rae.es/lema/interoperabilidad. [Último acceso: 4 Mayo 2021].

[6] L. Muñoz Fernández y S. Gallego Riestra, «La Protección de Datos y la interoperabilidad en el ámbito sanitario: dos realidades inseparables,» Segunda ed., vol. XXIX, 2019.

[7] «HL7 International,» HL7 Version 2 Product Suite, [En línea]. Available: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185. [Último acceso: 6 Mayo 2021].

[8] «HL7 International,» HL7 Version 3 Product Suite, [En línea]. Available: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186. [Último acceso: 6 Mayo 2021].

[9] «HL7 International,» CDA® Release 2, [En línea]. Available: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. [Último acceso: 6 Mayo 2021].

[10] «HL7 International. FHIR R4,» [En línea]. Available: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=491. [Último acceso: 11 Mayo 2021].

[11] «HL7 FHIR. RESTful API,» [En línea]. Available: https://www.hl7.org/fhir/http.html. [Último acceso: 18 Mayo 2021].

[12] «HL7 FHIR. Security,» [En línea]. Available: https://www.hl7.org/fhir/security.html. [Último acceso: 18 Mayo 2021].

[13] «HL7 FHIR. Data Types,» [En línea]. Available: https://www.hl7.org/fhir/datatypes.html. [Último acceso: 3 Junio 2021].

[14] «HL7 FHIR. DomainResource,» [En línea]. Available: https://www.hl7.org/fhir/domainresource.html.

[Último acceso: 27 Mayo 2021].

[15] «HL7 FHIR. Extensibility,» [En línea]. Available: https://www.hl7.org/fhir/extensibility.html. [Último acceso: 3 Junio 2021].

[16] «HL7 FHIR. Patient,» [En línea]. Available: hl7.org/fhir/patient.html. [Último acceso: 20 Junio 2021].

[17] «HL7 FHIR. Practictioner,» [En línea]. Available: https://www.hl7.org/fhir/practitioner.html. [Último acceso: 20 Junio 2021].

[18] «HL7 FHIR. Organization,» [En línea]. Available: https://www.hl7.org/fhir/organization.html. [Último acceso: 23 Julio 2021].

[19] «HL7 FHIR. PractitionerRole,» [En línea]. Available: https://www.hl7.org/fhir/practitionerrole.html. [Último acceso: 23 Julio 2021].

[20] «HL7 FHIR. Consent,» [En línea]. Available: https://www.hl7.org/fhir/consent.html. [Último acceso: 20 Junio 2021].

[21] «HL7 FHIR. Reference,» [En línea]. Available: https://www.hl7.org/fhir/references.html. [Último acceso: 22 Junio 2021].

[22] «Eclipse Foundation,» [En línea]. Available: https://www.eclipse.org/downloads/packages/release/2019-12. [Último acceso: 5 Junio 2021].

[23] «Spring,» [En línea]. Available: https://spring.io/. [Último acceso: 5 Junio 2021].

[24] «Hibernate,» [En línea]. Available: https://hibernate.org/. [Último acceso: 5 Junio 2021].

[25] «Apache Tomcat,» [En línea]. Available: https://tomcat.apache.org/download-90.cgi. [Último acceso: 5 Mayo 2021].

[26] «Apache Maven,» [En línea]. Available: https://maven.apache.org/. [Último acceso: 5 Junio 2021].

[27] «PostgreSQL,» [En línea]. Available: https://www.postgresql.org/. [Último acceso: 5 Junio 2021].

[28] «MagicDraw UML,» [En línea]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/. [Último acceso: 5 Junio 2021].

[29] «Android Studio,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 5 Junio 2021].

[30] «HAPI FHIR,» [En línea]. Available: https://hapifhir.io/. [Último acceso: 5 Junio 2021].

[31] «HAPI FHIR Public Test Server,» [En línea]. Available: http://hapi.fhir.org/. [Último acceso: 21 Junio 2021].

[32] E. Gamma, R. Helm, R. Johnson y J. Vlissides, «Patrones de diseño: Elementos de software orientado a objetos reutilizable,» Oviedo, Addison Wesley, 2003, pp. 119-125.

[33] «Oracle. Core J2EE Patterns - Data Access Object,» [En línea]. Available:

https://www.oracle.com/java/technologies/dataaccessobject.html. [Último acceso: 23 Junio 2021].

[34] «Maven Repository,» [En línea]. Available: https://mvnrepository.com/. [Último acceso: 23 Julio 2021].

[35] «Android Studio. Developers. Guía de usuario.,» [En línea]. Available: https://developer.android.com/studio/build/multidex?hl=es. [Último acceso: 26 Julio 2021].

[36] «HAPI FHIR. Android Client.,» [En línea]. Available: https://hapifhir.io/hapi-fhir/docs/android/client.html. [Último acceso: 26 Julio 2021].

[37] «Android Studio. Developers. Descripción general del manifiesto de una app.,» [En línea]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419. [Último acceso: 27 Julio 2021].