Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Definición e implementación del proceso de gestión de consentimientos

Autor: Jose Antonio García Linares

Tutores: Isabel Román Martínez,
Jorge Calvillo Arbizu

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

Sevilla, 2023

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

Definición e implementación del proceso de gestión de consentimientos

Autor:
Jose Antonio García Linares

Tutora:
Isabel Román Martínez
Profesora Colaboradora

Tutor externo:
Jorge Calvillo Arbizu
Profesor Ayudante Doctor

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Definición e implementación del proceso de gestión de consentimientos

Autor: Jose Antonio García Linares

Tutores: Isabel Román Martínez,

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, 2023

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

A continuación, quiero expresar mi profundo agradecimiento a todas las personas e instituciones que han aportado su granito de arena para llevar a cabo el desarrollo de mi Trabajo Fin de Grado y con ello la finalización de esta etapa académica.

En primer lugar, agradecer a mis tutores, Isabel y Jorge, por la dedicación a lo largo de este proyecto, además de todos los consejos y correcciones a lo largo de este camino, que provocaron un incremento de valor del presente proyecto.

En segundo lugar, agradecer a mis familiares y amigos por mostrarme su apoyo a lo largo de mi camino. A pesar de mis imperfecciones y mis cambios de humor, la presencia y el amor incondicional brindado han sido fundamentales para conseguir una de las metas más significativas hasta el momento.

Finalmente, me gustaría agradecer a la Universidad, así como al cuerpo docente, su generosidad al compartir su conocimiento y proporcionar los recursos necesarios, impulsando mi crecimiento intelectual y equipándome con las competencias cruciales de cara a mi futuro profesional.

Jose Antonio García Linares

Sevilla, 2023

Resumen

El sector de la salud ha experimentado una transformación significativa en la gestión de datos y la toma de decisiones clínicas debido al avance tecnológico. Esta evolución ha permitido una recopilación y un intercambio de información sanitaria más eficientes y rápidos. No obstante, este intercambio de datos plantea retos cruciales, especialmente en lo que respecta a la preservación de la privacidad y la salvaguardia de la información confidencial de los pacientes.

La protección de la privacidad y confidencialidad de los datos de salud se erige como una piedra angular para fomentar la confianza y la seguridad del paciente. Es imperativo que los pacientes tengan el control y la capacidad de tomar decisiones respecto al uso de sus datos personales en el ámbito de la atención médica. En este contexto, surge la necesidad de establecer un mecanismo eficaz y confiable para solicitar y registrar el consentimiento de los pacientes en relación con el uso de su información sanitaria.

El propósito fundamental de este Trabajo de Fin de Grado (TFG) consiste en analizar, desarrollar y poner en práctica un proceso eficiente y fiable para la solicitud y el registro del consentimiento de los pacientes en la utilización de sus datos de salud. Se persigue la creación de un marco procedimental que garantice la salvaguardia de la privacidad de los pacientes y promueva una relación médico-paciente basada en la transparencia y el respeto.

La eficaz administración de una entidad de atención sanitaria implica la optimización de los procesos internos. Para alcanzar esta meta, se ha logrado la creación de una aplicación de gestión de procesos empresariales altamente especializada en la gestión de solicitudes de consentimiento de los pacientes en relación con sus datos de salud. Los procesos contemplados comprenden la emisión de solicitudes de consentimiento por parte del personal médico debidamente autorizado, así como la recepción y el procesamiento de las respuestas de confirmación o negación por parte de los pacientes.

En la realización de este desarrollo, se ha empleado un marco tecnológico que incluye las tecnologías jBPM y Spring. Es esencial resaltar que la aplicación se ha diseñado teniendo en cuenta la utilización del estándar FHIR (Fast Healthcare Interoperability Resources). Esto posibilita una integración fluida de la aplicación desarrollada en entornos previamente establecidos que siguen este estándar, asegurando así una interoperabilidad efectiva.

El producto final es una aplicación web completamente funcional que aborda de manera efectiva los desafíos planteados, asegurando la protección de la privacidad de los pacientes y mejorando la eficiencia de los procesos de solicitud y registro de consentimientos en el ámbito de la atención médica.

Abstract

The healthcare sector has undergone a significant transformation in data management and clinical decision-making due to technological advancement. This evolution has enabled more efficient and faster collection and exchange of health information. However, this data exchange poses crucial challenges, especially with regard to the preservation of privacy and the safeguarding of confidential patient information.

Protecting the privacy and confidentiality of health data stands as a cornerstone for fostering patient trust and security. It is imperative that patients have control and the ability to make decisions regarding the use of their personal data in the healthcare setting. In this context, the need arises to establish an effective and reliable mechanism for seeking and recording patients' consent to the use of their health information.

The fundamental purpose of this Final Degree Project (TFG) is to analyzes, develop and implement an efficient and reliable process for requesting and recording patients' consent to the use of their health data. The aim is to create a procedural framework that guarantees the safeguarding of patients' privacy and promotes a doctor-patient relationship based on transparency and respect.

The efficient administration of a healthcare organization involves the optimization of internal processes. To achieve this goal, a business process management application has been developed that is highly specialized in the handling of consent requests from patients in relation to their health data. The processes envisaged include the issuing of consent requests by duly authorized medical staff, as well as the receipt and processing of confirmation or refusal responses from patients.

In the realization of this development, a technological framework including jBPM and Spring technologies has been used. It is essential to highlight that the application has been designed adopting the use of the FHIR standard. This enables a seamless integration of the developed application into previously established environments that follow this standard, thus ensuring effective interoperability.

The final product is a fully functional web application that effectively addresses the challenges posed, ensuring the protection of patient privacy and improving the efficiency of the consent request and registration processes in the healthcare environment.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

TFG	Trabajo Fin de Grado
API	Application Programming Interface
REST	Representational State Transfer
FHIR	Fast Healthcare Interoperability Resource
BPM	Business Process Management
jBPM	Java Business Process Management
BPMN	Business Process Model and Notation
HAPI-FHIR	Healthcare API for Fast Healthcare Interoperability Resource
KIE	Knowledge Is Everything
JPA	Java Persistence API
JTA	Java Transaction API
HL7	Health Level Seven
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
XML	Extensible Markup Language
IDE	Integrated Development Environment
POM	Project Object Model

1 INTRODUCCIÓN

En este capítulo se examinarán diversos aspectos, tales como la motivación que condujo a la realización de este trabajo, los objetivos que se pretenden lograr, la metodología aplicada durante el desarrollo y, por último, se presentará un esquema que reflejará el tiempo invertido en este proyecto.

1.1 Motivación y objetivos

En la era actual de la información, el sector de la salud ha experimentado una revolución en la forma en que se gestionan los datos y se toman decisiones clínicas. El avance de la tecnología ha permitido la recopilación y el intercambio de datos sanitarios de manera más eficiente y rápida. Sin embargo, este intercambio de datos conlleva importantes desafíos, especialmente en lo que respecta a la privacidad y la protección de la información personal sensible de los pacientes.

La protección de la privacidad y la confidencialidad de los datos sanitarios es un aspecto crucial para garantizar la confianza y la seguridad de los pacientes. Es fundamental que los pacientes tengan el control y la capacidad de tomar decisiones sobre el uso de sus datos personales en el ámbito de la salud. En este contexto, surge la necesidad de contar con un mecanismo eficiente y confiable para solicitar y registrar el consentimiento de los pacientes a utilizar su información sanitaria.

El objetivo de este TFG es analizar, desarrollar y poner en práctica un proceso eficiente y confiable para solicitar y registrar el consentimiento de los pacientes en el uso de sus datos sanitarios. Se busca establecer un marco procedimental que asegure la protección de la privacidad de los pacientes y promueva una relación médico-paciente basada en la transparencia y el respeto.

La gestión eficiente de una organización sanitaria implica la optimización de los procesos que en ella se ejecutan, incluidos, por supuesto, los referidos a la gestión de consentimientos. En este sentido, la aplicación del paradigma de gestión de procesos, respaldado por las tecnologías correspondientes, ofrece una solución efectiva. Esta metodología facilita las tareas de diseñar, automatizar, integrar, monitorizar y optimizar de forma continua los procesos de una organización.

En este proyecto en particular, se aplica la gestión de procesos a los procedimientos relacionados con la gestión de consentimientos. La implementación de un sistema de gestión de procesos en este ámbito proporciona numerosas ventajas y objetivos beneficiosos como son:

- Estandarización y optimización de procedimientos: La implementación de un sistema de gestión de procesos permite establecer estándares y optimizar los procedimientos involucrados en la recopilación y uso de datos sanitarios, lo cuál mejora la eficiencia y reduce errores o inconsistencias.
- Identificación y corrección proactiva de brechas de seguridad: Mediante el seguimiento y registros adecuados, es posible identificar de forma temprana posibles brechas de seguridad o incumplimientos en la protección de la privacidad, lo que facilita la adopción de medidas correctivas oportunas.
- Mejora de la trazabilidad y transparencia: El sistema de gestión de procesos permite rastrear el flujo de datos desde su origen hasta su destino final, lo cual contribuye a una mayor transparencia y responsabilidad en la gestión de datos sanitarios.
- Adaptabilidad y agilidad: La gestión de procesos favorece la adaptabilidad y la agilidad ante los cambios normativos o legales relacionados con la protección de datos, permitiendo actualizar y ajustar los procesos de forma oportuna para garantizar el cumplimiento normativo y mantener la confidencialidad de la información.

Al aplicar el paradigma de gestión de procesos en la gestión de consentimientos en una organización sanitaria,

se busca alcanzar una mayor eficiencia, seguridad y responsabilidad en el manejo de los datos, optimizando continuamente los procedimientos y adaptándolos a las necesidades cambiantes.

Los objetivos específicos de este proyecto son:

1. Identificar y modelar los procesos implicados en la gestión de consentimientos.
2. Implementar un sistema de gestión de procesos, utilizando la tecnología jBPM.
3. Integrar la comunicación con sistemas de gestión de datos sanitarios.
4. Diseñar e implementar un servicio, con interfaz conforme al paradigma REST, para la gestión y ejecución de los procesos.
5. Garantizar la seguridad y la privacidad de los datos personales de los pacientes durante todo el proceso de obtención de consentimiento.
6. Realizar pruebas exhaustivas para validar el funcionamiento y la eficacia del sistema implementado.

En conclusión, la gestión de procesos desempeña un papel esencial en la eficiente y segura gestión de datos sanitarios. Al estandarizar los procedimientos, mejorar la seguridad y privacidad, incrementar la trazabilidad y fomentar la adaptabilidad, se promueve la confianza de los pacientes y se asegura un manejo responsable de los datos en el sector de la salud.

1.2 Metodología

La metodología empleada para el desarrollo de este trabajo se basa en un enfoque incremental. Esta metodología combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos, lo que nos permite lograr un crecimiento progresivo de la funcionalidad del producto.

El enfoque incremental se basa en dividir el proyecto en iteraciones o incrementos. Cada incremento representa una fase del desarrollo, y su finalización proporciona una retroalimentación que nos permite aprender y aprovechar los conocimientos adquiridos para progresar en la siguiente iteración. [1]

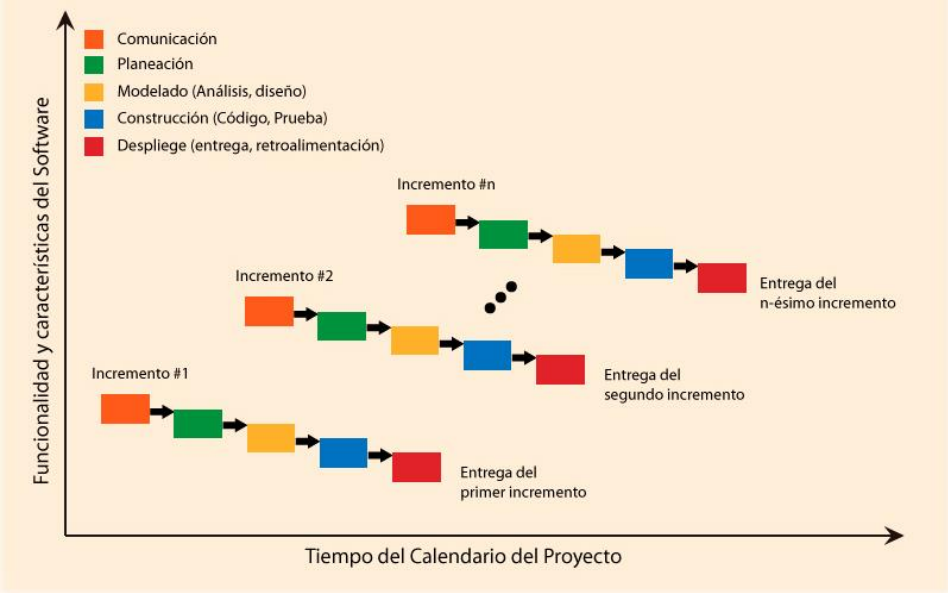

Figura 1. Metodología incremental

La Figura 1 muestra una representación visual de este enfoque incremental. A medida que avanzamos a través de los incrementos, podemos observar cómo se desarrolla el proyecto en cada fase, con entregas planificadas y una mejora continua en la funcionalidad.

Esta metodología nos brinda flexibilidad para realizar ajustes y cambios a medida que avanzamos, evitando un compromiso total con los requisitos iniciales. La retroalimentación constante que obtenemos en cada incremento nos permite detectar errores y problemas, lo que nos ayuda a trabajar con un código más limpio y a

encontrar soluciones más efectivas.

En resumen, la metodología incremental empleada en este trabajo se basa en el aprendizaje continuo y la retroalimentación constante. A través de entregas incrementales y la mejora iterativa, buscamos lograr un progreso gradual y satisfacer los objetivos establecidos.

1.3 Plan de trabajo

La elaboración de un plan de trabajo adecuado es esencial para llevar a cabo el proyecto de manera ordenada y eficiente, permitiendo cumplir con los objetivos establecidos y obtener resultados satisfactorios.

A lo largo de este plan de trabajo, se han definido una serie de etapas que se han diseñado de manera cuidadosa y estratégica para abordar los diferentes aspectos y requisitos del proyecto. Cada etapa tiene sus propias características y actividades específicas que se llevarán a cabo de manera secuencial. A continuación, se describirán brevemente cada una de ellas, resaltando las tareas que se seguirán para completar cada etapa:

1. Documentación: Se trata de la fase en la que se estudia el contexto y los fundamentos teóricos relacionados con el proyecto. Las tareas correspondientes a esta etapa son:
    - Estudio del estándar FHIR
    - Estudio de los recursos FHIR que mejor se adaptan a lo que se quiere realizar en el proyecto
    - Estudio del framework jBPM
2. Diseño e implementación de la estructura del proyecto: Fase en la que se han visto las posibles opciones de implementación y estructura del producto final. Se compone de las siguientes tareas:
    - Diseño de la estructura del proyecto
    - Estudio de la base de datos HAPI-FHIR
    - Estudio de la librería HAPI-FHIR
    - Estudio de la librería KIE Server
3. Implementación del código: Etapa en la que se han elegido las herramientas, el entorno de desarrollo y lenguaje de programación para desarrollar todas las funcionalidades que se desean en el proyecto.
4. Prueba y evaluación de los resultados obtenidos: Fase en la que se busca detectar y corregir los errores ocasionados a la hora de ejecutar el código, así como testear que el funcionamiento y los resultados obtenidos son los esperados.
5. Desarrollo de la memoria: Etapa en la que se documenta todo lo que se ha ido realizando en el proyecto y se explica su funcionamiento.

En resumen, el plan de trabajo de este TFG sigue una estructura organizada y secuencial, en la cual las etapas de implementación del código y prueba/evaluación de resultados se repetirán varias veces a lo largo del proyecto. Esto nos permitirá optimizar el tiempo asegurando una ejecución eficiente y efectiva del proyecto, permitiendo alcanzar los objetivos establecidos.

En la Tabla 1 se puede observar todas las etapas seguidas a lo largo del proyecto con la especificación de las horas estimadas y las horas que realmente nos ha llevado completar cada una de ellas.

Tabla 1. Plan de trabajo

Etapas	Horas estimadas	Horas reales
Documentación	30	50
Diseño e implementación de la estructura del proyecto	25	40
Implementación del código	80	100
Prueba y evaluación de los resultados obtenidos	50	70
Desarrollo de la memoria	30	50
Total de horas	**195**	**310**

2 ESTADO DE LA TÉCNICA

En el presente capítulo, se llevará a cabo la contextualización de la parte teórica del proyecto. En primera instancia, se establecerá una definición precisa de gestión de procesos, y se describirá el framework utilizado para su implementación. Posteriormente, se procederá a detallar el estándar FHIR, y los recursos que se han utilizados en el desarrollo del proyecto relacionados con este estándar. Por último, se proporcionará una explicación sobre los materiales empleados a lo largo de la ejecución del proyecto.

2.1 Gestión de procesos

2.1.1 Introducción a BPM

La gestión de procesos de negocio (BPM) es una disciplina empresarial que se enfoca en optimizar y mejorar los procesos internos de una organización para lograr resultados más eficientes y efectivos [2]. Un proceso en este contexto se define como una secuencia estructurada y coordinada de actividades interrelacionadas, diseñadas con el propósito de alcanzar un objetivo específico dentro de una organización. Estas actividades pueden involucrar recursos humanos, tecnológicos y otros activos organizativos, y están vinculadas por reglas y lógica predefinidas. Un proceso puede abarcar múltiples departamentos, roles y sistemas, y generalmente cruza las fronteras funcionales dentro de una organización.

Se trata de un enfoque estratégico que busca identificar, analizar, diseñar, implementar, controlar y mejorar los procesos de una organización con el objetivo de aumentar su productividad, reducir costos, mejorar la calidad de los productos y servicios, y aumentar la satisfacción del cliente.

BPM crea el puente entre los analistas de negocios, los desarrolladores y los usuarios finales al ofrecer funciones y herramientas de administración de procesos de una manera que les gusta tanto a los usuarios de negocios como a los desarrolladores. Al adoptar un enfoque centrado en los procesos, las organizaciones pueden identificar y eliminar ineficiencias, simplificar tareas, automatizar actividades repetitivas, mejorar la comunicación y la colaboración entre departamentos, y agilizar los flujos de trabajo.

En esta introducción general a BPM, se han destacado los conceptos y principios fundamentales de la gestión de procesos de negocio. A continuación, pasaremos a explorar una plataforma específica de BPM, jBPM, que ofrece herramientas y funcionalidades para la implementación de procesos empresariales de manera efectiva.

2.1.2 jBPM: Un framework para la gestión de procesos de negocio

jBPM es un framework de código abierto y una suite[1] de gestión de procesos empresariales (BPM) que proporciona una plataforma robusta y flexible para el modelado, ejecución y monitorización de procesos de negocio. Escrito en Java y basado en la notación estándar de modelado BPMN (Business Process Model and Notation), jBPM se ha convertido en una herramienta ampliamente utilizada en la automatización de procesos empresariales. [3]

La principal finalidad de jBPM es permitir a las organizaciones mejorar la eficiencia operativa y la agilidad empresarial al automatizar y optimizar sus procesos de negocio. Proporciona un enfoque estructurado para gestionar y controlar las actividades empresariales, lo que facilita la colaboración entre los equipos técnicos y los usuarios de negocio.

Una de las fortalezas de jBPM radica en su capacidad para modelar gráficamente los procesos de negocio

[1] Conjunto de software o aplicaciones relacionadas que se ofrecen juntas como un paquete integrado. Estas aplicaciones están diseñadas para trabajar de manera coherente y complementaria, proporcionando funcionalidades específicas o soluciones completas para un determinado propósito.

mediante diagramas. Esto permite una representación visual, clara y comprensible tanto para los equipos técnicos como para los usuarios de negocio, fomentando la colaboración y el entendimiento común.

Además, jBPM ofrece una amplia gama de funcionalidades que facilitan la gestión eficiente de los procesos de negocio. Algunas de estas funcionalidades incluyen:

- Asignación de tareas: Permite asignar tareas correctas a los participantes adecuados dentro de los procesos de negocio, asegurando una correcta ejecución y colaboración entre los actores involucrados.
- Reglas de negocio: Permite definir reglas y condiciones para guiar la ejecución del proceso, lo que garantiza que las decisiones se tomen de acuerdo con las políticas y objetivos empresariales.
- Monitorización en tiempo real: Ofrece capacidades de seguimiento y monitorización en tiempo real de los procesos, al tiempo que te brinda acceso a datos históricos. Esta combinación permite tener una visión actualizada de su estado y desempeño. Esto ayuda a identificar cuellos de botella, detectar problemas y tomar medidas correctivas de manera oportuna.
- Integración con sistemas externos: jBPM se integra fácilmente con sistemas externos, lo que permite la interoperabilidad y la automatización de flujos de trabajo que involucran múltiples sistemas y aplicaciones.
- Adaptabilidad y flexibilidad: jBPM es capaz de adaptarse a cambios y mejoras en los procesos de manera ágil, lo que permite una evolución continua de los mismos. Esto es especialmente útil en entornos empresariales dinámicos donde los requisitos y las condiciones pueden cambiar con frecuencia.

Los componentes principales de jBPM son:

1. Motor central (Core Engine): Es el corazón de jBPM y proporciona la capacidad de ejecutar procesos utilizando la especificación BPMN 2.0. El motor central se puede integrar en una aplicación o funcionar como un servicio independiente. Permite la gestión del ciclo de vida completo de los procesos, desde la creación hasta la ejecución y monitorización.
2. Servicio de tareas humanas (Human Task Service): Este componente permite la asignación y gestión de tareas que deben ser realizadas por actores humanos dentro de los procesos. Proporciona un mecanismo para la colaboración y la interacción entre los participantes, permitiendo la asignación de tareas, la notificación de eventos y la supervisión del progreso.
3. Persistencia y transacciones (Persistence and Transactions): jBPM utiliza la API de persistencia JPA (Java Persistence API) y la API de transacciones JTA (Java Transaction API) para garantizar la integridad de los datos y el control transaccional durante la ejecución de los procesos. Proporciona capacidades de persistencia conectables y asegura que los datos se almacenen correctamente y estén disponibles para su posterior análisis y consulta.
4. Gestión de casos (Case Management): Este componente amplía las capacidades de jBPM para admitir casos de uso más adaptables y flexibles. Permite la gestión de situaciones complejas y variables en tiempo real, donde los usuarios finales tienen el control sobre qué partes del proceso deben ejecutarse. La gestión de casos brinda mayor flexibilidad y adaptabilidad a los procesos comerciales, permitiendo su ajuste según las necesidades específicas del negocio.
5. Diseñador de procesos (Process Designer): Es una herramienta gráfica que permite modelar y diseñar procesos de negocio utilizando notación BPMN. Facilita la creación de diagramas que representan los pasos y la lógica del proceso, lo que mejora la visibilidad y la comprensión de los procesos.
6. Modelador de datos y formularios (Data and Form Modeler): Este componente proporciona un entorno web para la creación y gestión de modelos de datos y formularios asociados a las tareas del proceso. Permite definir la estructura de los datos necesarios para el proceso y diseñar formularios personalizados para interactuar con los usuarios. El modelador de datos y formularios agiliza el desarrollo y la personalización de las interfaces de usuario.
7. Informes y paneles (Reporting and Dashboards): jBPM ofrece capacidades para generar informes y paneles personalizables que permiten el seguimiento y visualización de la ejecución de los procesos. Estas herramientas brindan información en tiempo real sobre el rendimiento y la eficiencia de los

procesos, lo que ayuda en la toma de decisiones y el análisis de la actividad empresarial.

A continuación, se muestra una representación visual, donde se ilustran los componentes mencionados anteriormente y su interacción dentro del framework jBPM.

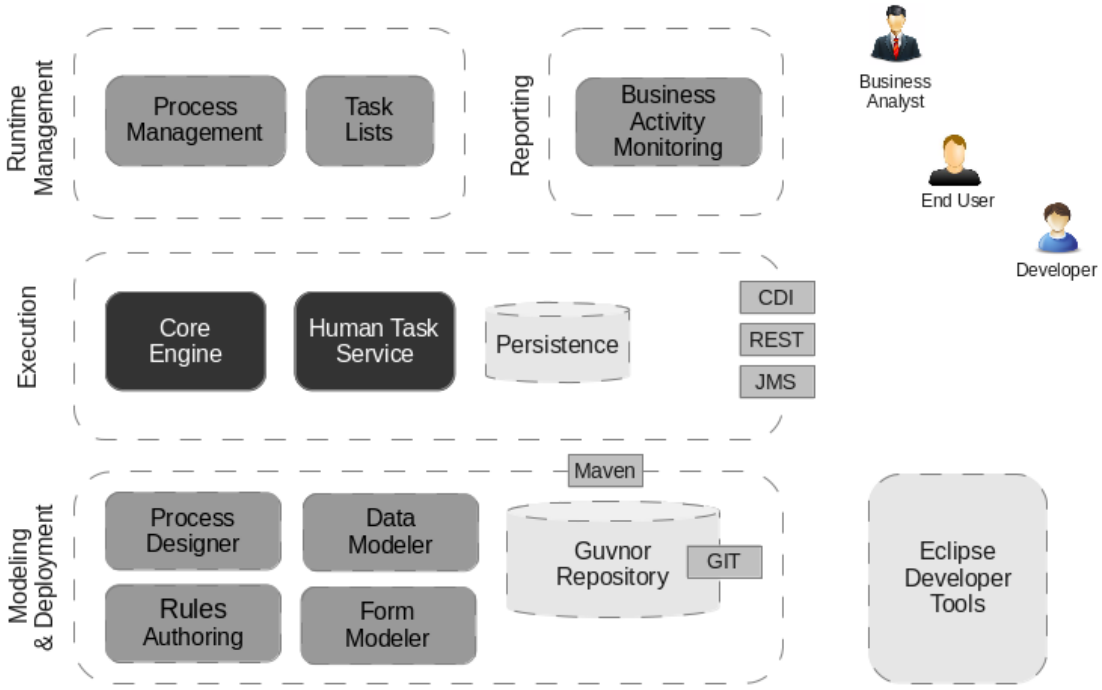

Figura 2. Descripción general de jBPM

En conclusión, jBPM se posiciona como una herramienta poderosa y versátil que ayuda a las organizaciones a automatizar y optimizar sus procesos de negocio, promoviendo la eficiencia, la colaboración y la mejora continua en un entorno empresarial en constante cambio.

2.2 Interoperabilidad en el ámbito sanitario

La interoperabilidad entre sistemas sanitarios es la capacidad de intercambio fluido y coherente de información clínica entre diferentes sistemas y organizaciones de salud. Se refiere a la capacidad de los sistemas y dispositivos de atención médica para compartir, interpretar y utilizar datos de manera efectiva con el objetivo de mejorar la coordinación y calidad de los servicios sanitarios.

La consecución de la interoperabilidad en salud presenta diversos desafíos, entre ellos la falta de estándares comunes de datos, la heterogeneidad de los sistemas de información utilizados por distintas entidades y las preocupaciones relacionadas con la privacidad y seguridad de la información médica. Los avances tecnológicos y la adopción de estándares reconocidos, como HL7 y FHIR, han propiciado un entorno para el intercambio seguro y eficiente de datos en el ámbito sanitario.

Los beneficios de la interoperabilidad en salud son significativos. Al permitir el acceso oportuno y completo a información clínica relevante, se promueve una atención médica más coordinada y segura. Asimismo, contribuye a mejorar la eficiencia en la prestación de servicios, reducir los errores médicos, facilitar la investigación clínica y fomentar la innovación en el desarrollo de tecnologías y tratamientos médicos.

En muchos países, tanto los gobiernos como las organizaciones de salud han reconocido la importancia de la interoperabilidad y la han incorporado como parte de sus estrategias digitales en el ámbito sanitario. Esto implica la implementación de infraestructuras de intercambio de información, la adopción de estándares comunes y la promoción de la colaboración entre los diversos aspectos del sistema de salud.

En resumen, la interoperabilidad en el ámbito sanitario desempeña un papel fundamental para mejorar la calidad, seguridad y eficiencia de la atención médica. Conforme avanza la tecnología y se establecen

estándares internacionales reconocidos, se espera que la interoperabilidad continúe avanzando y contribuya de manera significativa a la transformación digital de los sistemas de salud en todo el mundo.

2.2.1 FHIR

FHIR es un estándar de intercambio de información clínica desarrollado por la organización HL7 [5]. Ha emergido como el referente principal en la interoperabilidad en el ámbito de la salud debido a su enfoque moderno, su capacidad para adaptarse a diferentes tecnologías y su simplicidad de implementación.

Se basa en tecnologías web estándar, como HTTP, RESTful y JSON/XML. Al utilizar estos estándares modernos ampliamente adoptados, FHIR permite una transferencia eficiente y segura de datos de salud entre diferentes sistemas y dispositivos. La combinación de HTTP como protocolo de comunicación y JSON/XML como formato de datos proporciona una base sólida para la interoperabilidad, facilitando la integración con otras tecnologías y sistemas existentes.

La base conceptual de FHIR se sustenta en el concepto de recursos, que representan la unidad básica de interoperabilidad. Estos recursos son representaciones de los diferentes conceptos y elementos del mundo de la atención médica, como pacientes, médicos, observaciones de laboratorio o procedimientos médicos.

FHIR se distingue por sus características comunes, que incluyen un conjunto reducido de propiedades principales ampliamente admitidas por la mayoría de los sistemas. Esto significa que FHIR se diferencia al tener características compartidas, como un conjunto pequeño de propiedades esenciales que son aceptadas por la mayoría de los sistemas. Gracias a esto, se logra una consistencia y compatibilidad en la interpretación y procesamiento de los datos clínicos. En otras palabras, los datos médicos pueden ser entendidos y manejados de manera uniforme y coherente entre diferentes sistemas, lo que facilita su intercambio y uso.

Además, FHIR ofrece mecanismos de extensión que permite agregar nuevas propiedades de manera sencilla, brindando flexibilidad para adaptarse a necesidades específicas de implementación.

Cada recurso FHIR cuenta con una serie de características comunes:

- Un pequeño conjunto de propiedades principales que la gran mayoría de los sistemas soportan actualmente.
- Un mecanismo de extensión que permite a las implementaciones añadir nuevas propiedades de manera sencilla.
- Una identificación a través de la cual puede ser registrado, localizado y recuperado.
- Un componente (elementos narrativos) que permiten una visión legible de los datos almacenados en el recurso.

El modelo de datos de FHIR se fundamenta en la definición de estos recursos, y su utilización puede variar dependiendo de la necesidad. En lugar de una única forma estática, los recursos se adaptan a distintos contextos y aplicaciones, permitiendo flexibilidad en cómo se utilizan para intercambiar información clínica.

2.2.1.1 Tipos de datos

En FHIR, se especifica un conjunto de tipos de datos [] que se utilizan en los elementos de recursos. Estos tipos se organizan en cinco categorías:

1. Tipos abstractos:

   Son representaciones conceptuales de datos que actúan como superclases para definir características y comportamientos comunes entre los tipos derivados.

   Estos tipos establecen una estructura base y definición compartida en la especificación, pero no se utilizan directamente para almacenar o intercambiar datos.

2. Tipos simples/primitivos:

   Son aquellos que no contienen elementos adicionales como subelementos y se caracterizan por tener un único valor.

Los tipos primitivos son utilizados para representar información básica y fundamental, como números, cadenas de texto fechas, booleanos, entre otros. Estos tipos no tienen estructuras internas complejas y su principal propósito es proporcionar un formato estándar para representar y compartir datos de salud.

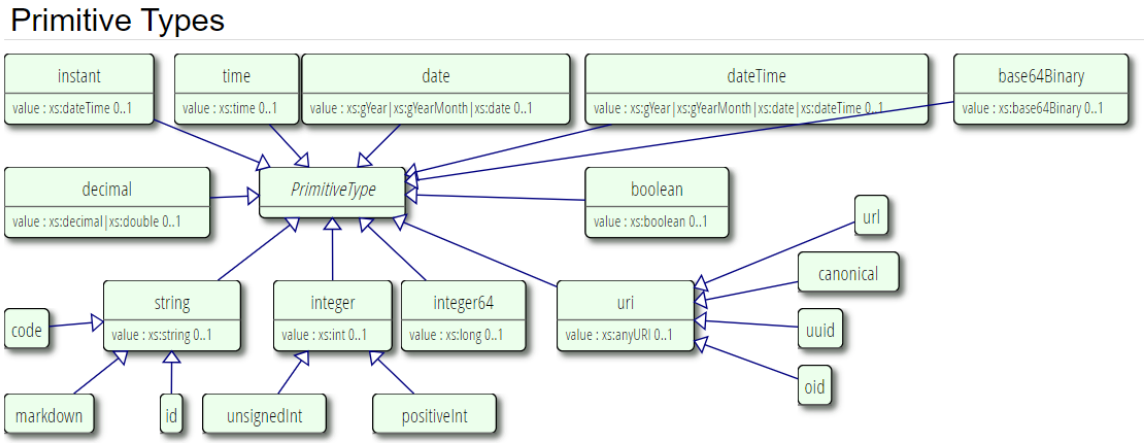

Figura 3. Tipos primitivos de FHIR

Aunque no contienen elementos secundarios, como hijos, pueden tener identificadores y extensiones, lo que les permite agregar información adicional o personalizada a los valores primitivos según sea necesario.

A continuación, se presenta una representación exhaustiva en formato JSON de un tipo primitivo, que incluye un identificador, dos extensiones y un valor:

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

Figura 4. Ejemplo de un tipo primitivo en formato JSON

3. Tipos complejos:

   Son estructuras de datos más avanzadas y sofisticadas que contienen múltiples elementos y propiedades. A diferencia de los tipos primitivos, los tipos complejos pueden tener elementos secundarios y pueden representar información más detallada y estructurada.

   Se utilizan para modelar entidades de datos más complejas, que pueden contener múltiples atributos y subelementos para representar diferentes aspectos y características de la entidad.

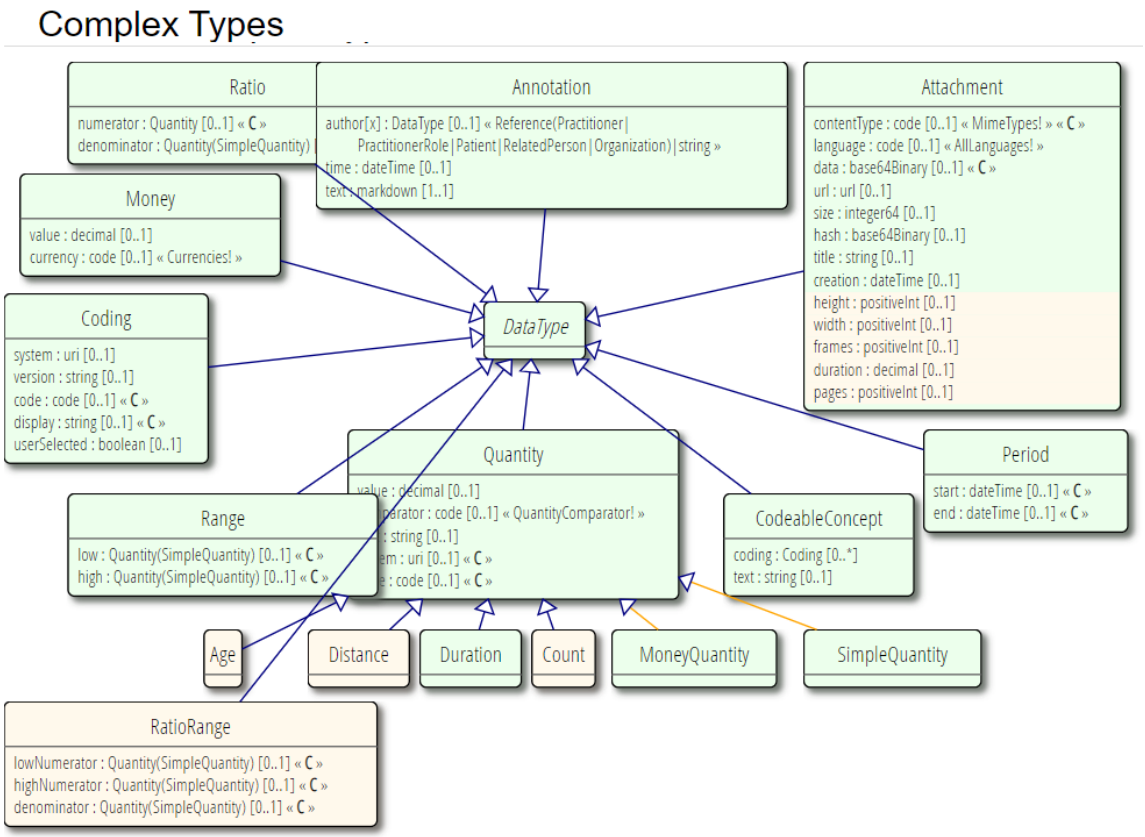

Figura 5. Algunos tipos complejos de FHIR.

En la siguiente figura, se observa una representación en formato JSON de un tipo complejo:

```
"identifier" : {
  "use" : "official",
  "system" : "http://www.acmehosp.com/patients",
  "value" : "44552",
  "period" : {
    "start" : "2003-05-03"
  }
}
```

Figura 6. Ejemplo de un tipo complejo en formato JSON

4. Tipos de metadatos:

   Se utilizan para proporcionar detalles adicionales y contextualizar los recursos en FHIR.

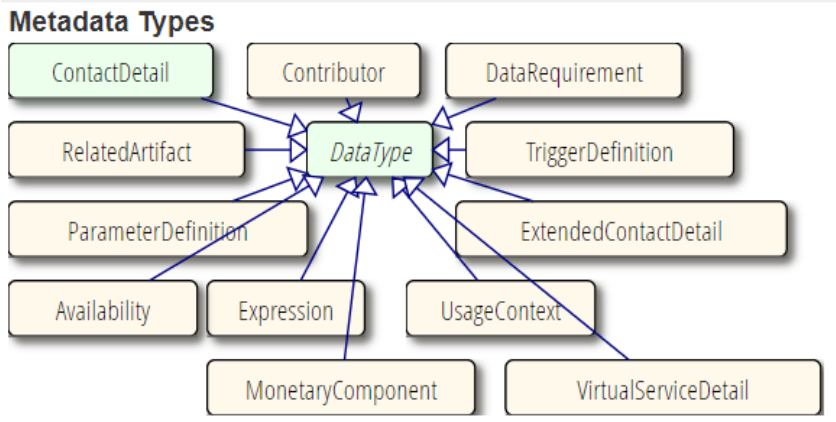

Figura 7. Algunos tipos de metadatos de FHIR

5. Tipos de datos de propósito especial:

   Tipos de datos especializados que capturan información específica de atención médica. Estos tipos permiten una representación precisa y estandarizada de información relevante en el ámbito de la atención médica.

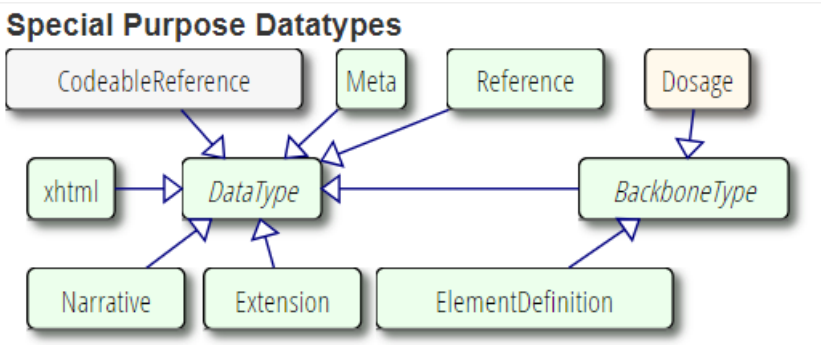

Figura 8. Algunos tipos de propósito especial de FHIR

2.2.1.2 Recursos FHIR

Los recursos FHIR son entidades que se utilizan para el intercambio de datos en el campo de la atención médica. Estos recursos están definidos en la especificación FHIR y se utilizan para resolver una amplia gama de problemas clínicos y administrativos en el ámbito de la atención médica. [7]

Cada recurso FHIR tiene una identidad única, representada por una URL base, que permite su dirección y acceso de manera unívoca. Estos recursos se clasifican en diferentes tipos, cada tipo de recurso tiene una estructura de datos definida específicamente que describe los elementos y propiedades asociados a ese recurso en particular.

Además de la estructura de datos, los recursos FHIR también contienen propiedades comunes a todos ellos. Estas propiedades incluyen:

1. Identidad: Cada recurso FHIR tiene una identidad única representada por una URL base, que se utiliza para abordar y acceder al recurso de manera única.
2. Metadatos: Los metadatos proporcionan información adicional sobre el recurso, como la fecha y hora de creación, la última modificación, el autor o la fuente de los datos.
3. Idioma base: El idioma base es un atributo opcional que se puede utilizar para indicar el idioma en el que se encuentra el contenido del recurso.
4. Reglas implícitas: Los recursos FHIR hacen referencia a las "reglas implícitas", que son los principios y las mejores prácticas que se deben seguir al intercambiar y utilizar los recursos FHIR.

Figura 9. Estructura del recurso FHIR Resource

A continuación, definimos los recursos FHIR que vamos a utilizar en este proyecto:

2.2.1.2.1 Questionnaire

El recurso Questionnaire es una estructura de datos estandarizada que representa un cuestionario utilizado para

recopilar información de pacientes, proveedores de atención médica u otras personas involucradas en el ámbito de la atención médica. Se utiliza para definir y organizar una serie de preguntas que se formularán con el fin de obtener datos relevantes. [8]

Un cuestionario puede estar compuesto por preguntas individuales o agruparse jerárquicamente en grupos y subgrupos, lo que permite una estructura organizada y lógica, Cada pregunta en el cuestionario se define con su texto, opciones de respuesta posibles y restricciones sobre las respuestas permitidas.

El recurso Questionnaire especifica cómo se deben presentar las preguntas, en qué orden deben aparecer y cómo se deben agrupar. También puede contener texto instructivo para guiar al encuestado durante la respuesta al cuestionario.

El objetivo principal del recurso Questionnaire es estandarizar la captura de datos en diferentes sistemas y aplicaciones de atención médica. Proporciona una forma común de comunicar y compartir cuestionarios entre sistemas interoperables Además de capturar datos, los cuestionarios también pueden utilizarse para definir una presentación estandarizada de datos existentes, lo que facilita la visualización y el intercambio de información en un formato predefinido.

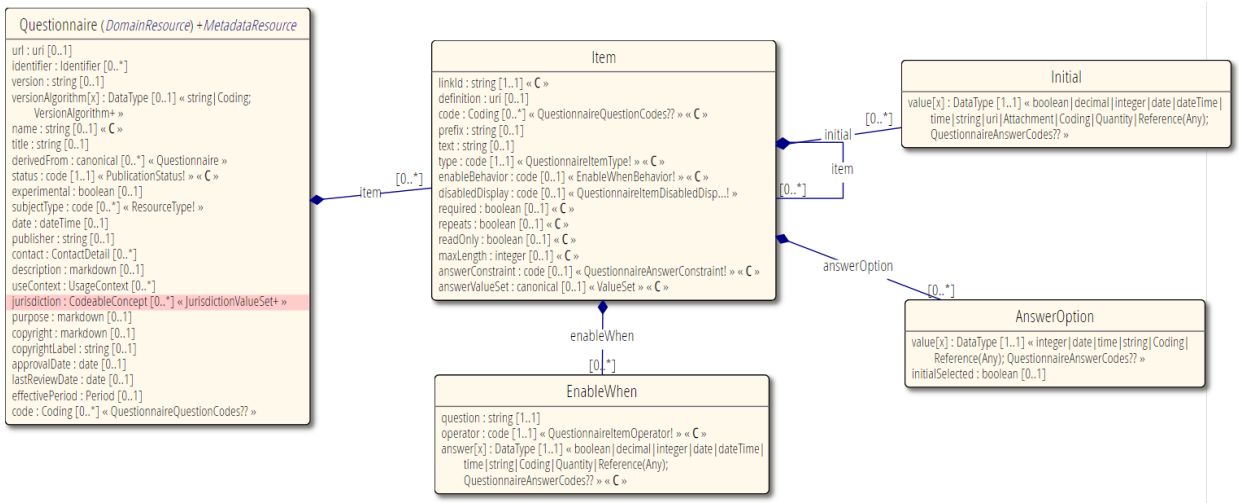

Figura 10. Diagrama UML del recurso Questionnaire

Los resultados de un cuestionario se pueden comunicar utilizando el recurso QuestionnaireResponse, que contiene las respuestas proporcionadas por el encuestado a las preguntas del cuestionario.

2.2.1.2.2 QuestionnaireResponse

El recurso QuestionnaireResponse es una estructura de datos que representa una lista completa o parcial de respuestas a un conjunto de preguntas formuladas en un cuestionario. Este recurso proporciona un registro de las respuestas proporcionadas por un encuestado al responder a un cuestionario específico. [9]

Las respuestas en el QuestionnaireResponse pueden incluirse directamente en el recurso, y además hacer referencia a un recurso Questionnaire que define las preguntas del cuestionario y las restricciones sobre las respuestas permitidas. Esto permite una vinculación explícita entre las respuestas y el cuestionario correspondiente. Además, puede contener información local suficiente para permitir la presentación del cuestionario, lo que proporciona detalles específicos sobre cómo se capturaron los datos, incluyendo el orden de las preguntas y las respuestas dadas.

Cada vez que se completa un cuestionario para un tema o en un momento diferente, se crea una respuesta al cuestionario diferente. Sin embargo, es posible editar o actualizar un conjunto de respuestas ingresadas previamente, lo que permite realizar cambios o agregar información adicional.

Al igual que el recurso Questionnaire, el recurso QuestionnaireResponse se utiliza para comunicar datos provenientes de formularios utilizados en exámenes de historia clínica, cuestionarios de investigación y registros de especialidades clínicas. Estas respuestas registran detalles específicos sobre la captura de datos, como las preguntas realizadas, el orden de las preguntas y las respuestas proporcionadas.

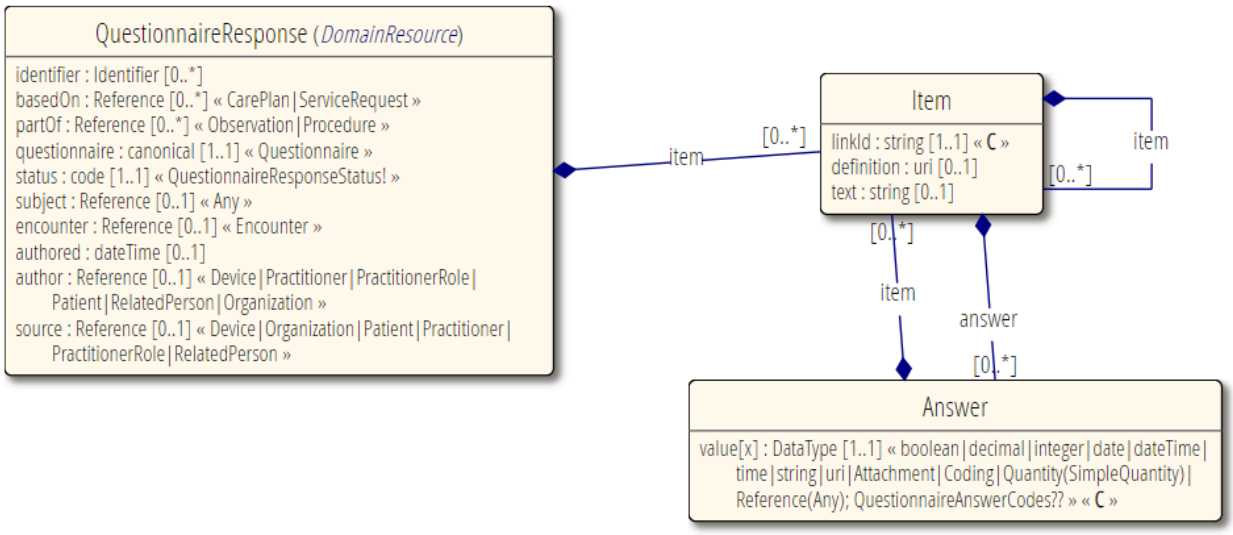

Figura 11. Diagrama UML del recurso QuestionnaireResponse

2.3 Herramientas software

Para llevar a cabo este proyecto, se han utilizado diversas herramientas y programas que han sido fundamentales para su desarrollo. En las siguientes subsecciones se describen cada una de ellas.

2.3.1 Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) ampliamente utilizado y de código abierto. Es una herramienta poderosa para desarrollar software en varios lenguajes de programación, incluyendo Java. Eclipse proporciona una amplia gama de características y funcionalidades que ayudan a los desarrolladores a escribir, depurar y desplegar aplicaciones de manera eficiente.

Figura 12. Logo Eclipse

2.3.2 Business Central

Es una aplicación web que abarca todo el ciclo de vida de los proyectos de BPM, desde su creación hasta la implementación, ejecución y seguimiento. Combina una serie de herramientas basadas en la web en una solución configurable para administrar todos los activos y datos de tiempo de ejecución necesarios para la solución empresarial. [10]

Esca aplicación es compatible con:

- Un servicio de repositorio que almacena los procesos y los artefactos relacionados. Utiliza un repositorio de Git que permite el control de versiones, el acceso remoto a Git (como un sistema de archivos) y el acceso a través de REST.
- Una interfaz de usuario basada en la web que permite administrar los procesos. Está dirigida a usuarios comerciales y también permite la visualización y edición de artefactos. Además, integra editores basados en web, como el diseñador, el modelador de datos y los formularios. También proporciona funciones de categorización, construcción e implementación, entre otras.
- Funciones de colaboración que permiten a múltiples actores, como usuarios comerciales y desarrolladores, trabajar juntos en el mismo proyecto.

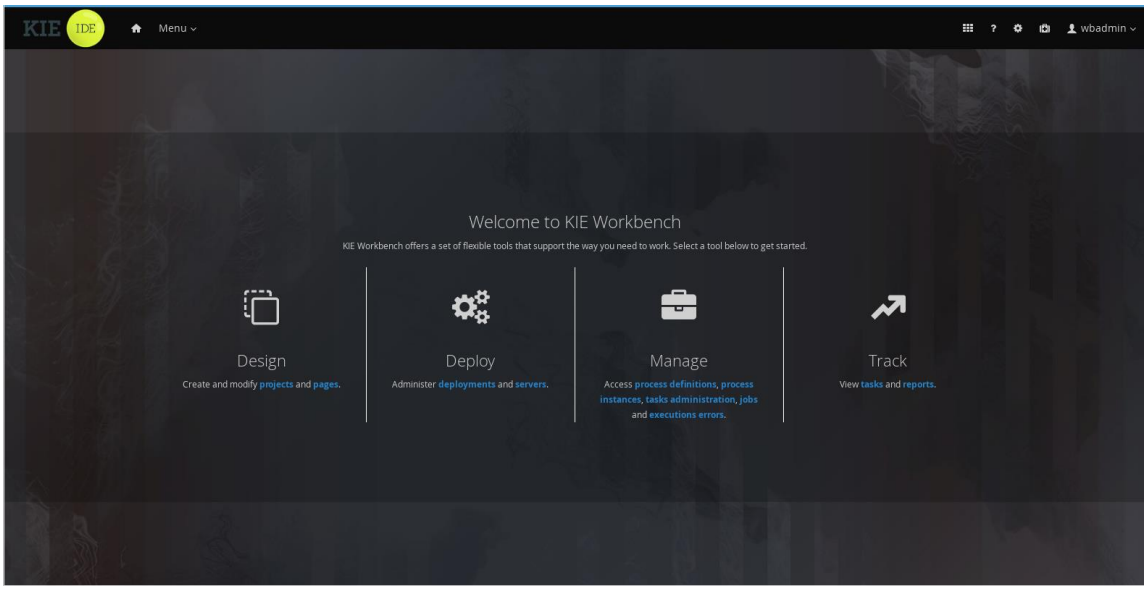

Figura 13. Aplicación Business Central.

2.3.3 Maven

Es una herramienta de gestión y construcción de proyectos software, ampliamente utilizada en el desarrollo de aplicaciones Java, aunque también es compatibles con otros lenguajes de programación. [11]

Maven proporciona un enfoque estructurado y coherente para la construcción de proyectos, la gestión de dependencias, la compilación, las pruebas, la generación de informes y a distribución de software. Su objetivo principal es simplificar el proceso de construcción y manejo de proyectos, al proporcionar un sistema de configuración declarativo basado en archivos XML llamados POM.

El archivo POM describe la estructura del proyecto, sus dependencias, las configuraciones del entorno de compilación, las pruebas y otros aspectos del ciclo de vida del proyecto. Maven se encarga de resolver y descargar automáticamente las dependencias necesarias para el proyecto desde repositorios remotos, lo que facilita la administración de las bibliotecas y evita la necesidad de descargarlas manualmente.

Además, Maven utiliza convenciones estándar para la organización del proyecto y el manejo de la estructura de directorios, lo que facilita la colaboración entre desarrolladores y el intercambio de proyectos. También proporciona una serie de complementos (plugins) predefinidos que amplían su funcionalidad.

Figura 14. Maven

2.3.4 HAPI FHIR

Es una biblioteca de código abierto ampliamente utilizada para el desarrollo de aplicaciones en el ámbito de la salud que se adhieren al estándar FHIR. [13]

HAPI FHIR proporciona una serie de herramientas y utilidades que facilitan la implementación de la especificación FHIR en aplicaciones de salud. Está escrito en Java, HAPI FHIR ofrece una API fácil de usar y un conjunto de funciones que permiten la creación, validación, manipulación y búsqueda de recursos FHIR.

Una de las características más notables de HAPI FHIR es su capacidad para funcionar como servidor FHIR completo. Esto significa que puede implementar un punto final FHIR funcional y servir recursos FHIR a través de una interfaz web. Este enfoque permite a los desarrolladores crear sistemas de información de salud que cumplan con el estándar FHIR y sean accesibles para otros sistemas y aplicaciones.

HAPI FHIR proporciona un servidor de referencia listo para usar. Este servidor se puede utilizar como punto

de partida para construir un servidor FHIR personalizado, ya que ofrece una implementación completa y bien documentada del estándar FHIR. Esto facilita la configuración y personalización según las necesidades específicas del proyecto.

En cuanto al servidor público de HAPI FHIR [14], es una instancia en línea que ofrece una API pública y abierta para acceder a recursos FHIR. Este servidor público es una valiosa herramienta para los desarrolladores y profesionales de la salud, ya que brinda la posibilidad de experimentar, probar y acceder a recursos utilizado el estándar FHIR en un entorno seguro y confiable.

Figura 15. Logo HAPI FHIR

2.3.5 JSON

Es un formato de intercambio de datos ligero y fácil de leer y escribir. Se utiliza para representar datos estructurados en forma de texto, con el objetivo de transmitir información entre diferentes sistemas de manera eficiente. [15]

3 RESULTADOS

En este capítulo se describirá en profundidad todo el proceso que se ha llevado a cabo para la realización del presente trabajo. En primer lugar, se describirá la fase de análisis y definición de requisitos. A continuación, se detallará la implementación.

3.1 Análisis y definición de requisitos

3.1.1 Introducción

La etapa inicial de cualquier proyecto de software incluye la evaluación y la definición de los requisitos, y es una de las más cruciales. Esto implica llevar a cabo un análisis exhaustivo del ámbito del problema para identificar las demandas del usuario final, establecer metas y crear un sistema integral y funcional sin abordar cuestiones de diseño.

3.1.2 Análisis de requisitos

3.1.2.1 Actores del Sistema

Un actor es una entidad externa que tiene interés en interactuar con el sistema. En nuestro caso describe el rol que desempeña un usuario en su interacción con el sistema. En este caso, hay dos tipos de actores los cuáles se describirán a continuación.

Tabla 2. ACT-01: Practitioner

ACT-01	**Practitioner**
Versión	v1.0.0
Dependencias	Restricción de usuario y roles (RN-002)
Descripción	Este actor representa a cualquier personal médico con autorización para solicitar peticiones de consentimiento a los pacientes. Este actor tendrá acceso a todas las funciones del sistema.

Tabla 3. ACT-02: Patient

ACT-02	**Patient**
Versión	v1.0.0
Dependencias	Restricción de usuario y roles (RN-002)
Descripción	Este actor representa a cualquier persona física con acceso únicamente a las peticiones de consentimientos que vayan dirigidas hacia él. .

3.1.2.2 Casos de uso del Sistema

Los casos de uso describen cuál será la secuencia de acciones en las interacciones entre un actor y el sistema software. En las siguientes tablas se podrán visualizar todos los detalles relacionados con los casos de uso detectados.

Tabla 4. CU-001: Inicio de sesión

CU-001	**Inicio de sesión**	
Versión	v1.0.0	
Dependencias	Información de usuario (RINF-001) Practitioner (ACT-01) Patient (ACT-02) Información inicio de sesión de un usuario (RN-001)	
Precondición	El Practitioner (ACT-01) o Patient (ACT-02) accede a la aplicación web	
Descripción	El sistema deberá seguir la siguiente secuencia para el acceso de los usuarios al contenido de la aplicación	
Secuencia	**Paso**	**Acción**
	1	El sistema solicita al Practitioner (ACT-01) o Patient (ACT-02) que inserte sus credenciales
	2	El usuario deberá proporcionar los datos solicitados, DNI y contraseña, presentes en RINF-001 (Información de usuario)
	3	El sistema deberá comprobar si las credenciales son correctas y el usuario ya estaba registrado en el sistema de persistencia
	4	Si la autenticación se ha realizado correctamente
	4.1	El sistema permite el acceso al usuario, mostrando un menú distinto en base a su rol: • Si el usuario es Practitioner (ACT-01), se muestra un menú con la opción de trabajar como Practitioner (ACT-01) o Patient (ACT-02) • Si el usuario no tiene el rol Practitioner (ACT-02), nos mostrará las solicitudes de consentimientos pendientes de confirmación.
	5	Si la autenticación es errónea, el sistema nos redirigirá a la vista principal, la de autenticación mostrando un aviso
Prioridad	Muy alta	

Tabla 5. CU-002: Registrar nuevo usuario

CU-002	**Registrar nuevo usuario**

Versión	v1.0.0	
Dependencias	Información de usuario (RINF-001) Practitioner (ACT-01) Patient (ACT-02) Restricción usuarios y roles (RN-002)	
Precondición	El Practitioner (ACT-01) o Patient (ACT-02) accede a la aplicación web.	
Descripción	Para el registro de un nuevo usuario será necesario seguir la siguiente secuencia	
Secuencia	**Paso**	**Acción**
	1	El sistema solicita al Practitioner (ACT-01) o Patient (ACT-02) que inserte sus credenciales
	2	Si el usuario aún no está registrado en la aplicación, deberá pulsar el botón “Sign up” para registrarse en ella
	3	El sistema mostrará el formulario con los datos, correspondientes al RINF-001 (Información de usuario), que este debe rellenar para poder registrarse correctamente
	4	El usuario deberá proporcionar los datos solicitados
	5	El sistema deberá validar que los datos introducidos son correctos y no existe un usuario con ese identificador
	6	Si el usuario no existe
	6.1	Se persiste el usuario de forma local en la base de datos con rol Practitioner (ACT-01) en caso de haberse seleccionado la opción “Permisos de Practitioner” del RINF-001 (Información de usuario),o rol Patient (ACT-02) en caso contrario
	6.2	El sistema vuelve a la página principal, la de autenticación
	7	Si el usuario ya existe, el sistema se redirigirá a la pantalla de registro mostrando un aviso
Prioridad	Muy alta	

Tabla 6. CU-003: Solicitud de nuevo consentimiento

CU-003	**Solicitud de nuevo consentimiento**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) MetaCuestionnario (RINF-002)

	Información del proceso “solicitudConsentimiento” (RINF-003) Información de instancias de procesos asociadas al Practitioner (RINF-005) Información de instancias de procesos asociadas al Patient (RINF-006) Practitioner (ACT-01) Proceso “solicitudConsentimiento” (RN-010) MetaCuestionario (RN-009)	
Precondición	Una vez realizado el CU-001 (Inicio de sesión), el Practitioner (ACT-01) pulsa el botón “Practitioner”, y luego elige la opción ”Request for consent”.	
Descripción	El sistema debe interaccionar con el motor jBPM siguiendo la siguiente secuencia	
Secuencia	**Paso**	**Acción**
	1	El sistema realizará una petición a la RN-010 (Proceso “solicitudConsentimiento”)
	2	El sistema realizará una petición para iniciar la tarea humana del RN-010 (Proceso “solicitudConsentimiento”)
	3	El sistema mostrará el MetaCuestionario (RN-009) como formulario html
	4	El usuario rellenará los campos de datos del RINF-002 (MetaCuestionario) para la solicitud de consentimiento que quiere realizar y procederá a pulsar enviar.
	5	El sistema recoge los datos proporcionados por el usuario
	6	El sistema completa la tarea humana (cambia su estado)
	7	Se persiste localmente los datos correspondientes al RINF-005 (Información de instancias de procesos asociadas al Practitioner)
	8	Se persiste localmente los datos correspondientes al RINF-006 (Información de instancias de procesos asociadas al Patient)
Prioridad	Alta	

Tabla 7. CU-004: Ver solicitudes de consentimientos enviadas

CU-004	**Ver solicitudes de consentimientos enviadas**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información de instancias de procesos asociadas al Practitioner (RINF-005) Practitioner (ACT-01)
Precondición	Una vez realizado el CU-001 (Inicio de sesión), el Practitioner (ACT-01)

	pulsa el botón “Practitioner”, y luego elige la opción ”Consents Resquested”.	
Descripción	El sistema debe mostrar las solicitudes de consentimientos enviadas por un Practitioner (ACT-01).	
Secuencia	**Paso**	**Acción**
	1	El sistema deberá obtener todos los datos almacenados del RINF-005 (Información de instancias de procesos asociadas al Practitioner)
	2	Mostrará una lista con el título, dato obtenido del RINF-005 (Información de instancias de procesos asociadas al Practitioner), de todas las solicitudes de consentimientos vigentes realizadas por el Practitioner (ACT-01).
Prioridad	Alta	

Tabla 8. CU-005: Ver solicitud de consentimiento enviada

CU-005	**Ver solicitud de consentimiento enviada**	
Versión	v1.0.0	
Dependencias	Información de usuario (RINF-001) MetaCuestionnario (RINF-002) Información de instancias de procesos asociadas al Practitioner (RINF-005) Practitioner (ACT-01)	
Precondición	Una vez ejecutado el CU-004 (Ver solicitudes de consentimientos enviadas), el Practitioner (ACT-01) selecciona una solicitud.	
Descripción	El sistema debe mostrar los datos del MetaCuestionario (RINF-002) que fueron enviados en la solicitud de consentimiento	
Secuencia	**Paso**	**Acción**
	1	Obtenemos todos los datos del RINF-005 (Información de instancias de procesos asociadas al Practitioner) almacenados localmente en el sistema correspondiente al proceso
	2	El sistema deberá mostrar el MetaCuestionario (RINF-002) correspondiente a esa solicitud elegida por el Practitioner (ACT-01)
Prioridad	Alta	

Tabla 9. CU-006: Ver solicitudes de consentimientos recibidas

CU-006	**Ver solicitudes de consentimientos pendientes**

Versión	v1.0.0	
Dependencias	Información de instancias de procesos asociadas al Patient (RINF-006) Practitioner (ACT-01) Patient (ACT-02)	
Precondición	Una vez realizado el CU-001 (Inicio de sesión) NOTA: Si es Practitioner (ACT-01), pulsa el botón "Patient" en el menú principal	
Descripción	El sistema debe mostrar las solicitudes de consentimientos pendientes de un Patient (ACT-02)	
Secuencia	**Paso**	**Acción**
	1	El sistema deberá obtener todos los datos almacenados localmente del RINF-006 (Información de instancias de procesos asociadas al Patient)
	2	Mostrará una lista con el título, dato del RINF-006 (Información de instancias de procesos asociadas al Patient), de todas las solicitudes de consentimientos pendientes de confirmar/denegar que tiene el Patient (ACT-02)
Prioridad	Alta	

Tabla 10. CU-007: Confirmar/Denegar consentimiento

CU-007	**Confirmar/Denegar consentimiento**	
Versión	v1.0.0	
Dependencias	Información de proceso "confirmacionConsentimiento" (RINF-004) Información de instancias de procesos asociadas al Patient (RINF-006) Practitioner (ACT-01) Patient (ACT-02) Proceso "confirmaciónConsentimiento" (RN-011)	
Precondición	Una vez ejecutado el CU-006 (Ver solicitudes de consentimientos recibidas), el Patient (ACT-02) selecciona un elemento de la lista	
Descripción	El sistema deberá seguir los siguientes pasos para mostrar los datos de la solicitud enviada por el Practitioner (ACT-01) añadiendo una nueva opción que permita aceptar o denegar el consentimiento	
Secuencia	**Paso**	**Acción**
	1	El sistema realizará una petición para iniciar la tarea humana de la instancia de la RN-011 (Proceso "confirmaciónConsentimiento")
	2	Mostramos la solicitud de consentimiento enviada por el

		Practitioner (ACT-01) con un campo obligatorio adicional que permite aceptar/denegar el consentimiento
	3	El Patient (ACT-02) deberá aceptar o denegar la solicitud y pulsar enviar
	4	Una vez enviado el formulario, el sistema completará la tarea humana correspondiente a la RN-011 (Proceso "confirmaciónConsentimiento")
	5	Se eliminarán los datos del RINF-006 (Información de instancias de procesos asociadas al Patient) almacenados localmente en el sistema que corresponden a esa instancia
Prioridad	Alta	

3.1.2.3 Requisitos de información

Los requisitos de información son aquellos que describen qué datos deben ser almacenados y gestionados por un sistema para respaldar los procesos de negocio.

Tabla 11. RINF-001: Información de usuario

RINF-001	**Información de usuario**
Versión	v1.0.0
Descripción	El sistema debe manejar y persistir localmente la información correspondiente a sus usuarios.
Datos específicos	• Nombre • Apellidos • Email • DNI • Contraseña • Permisos de Practitioner (De tipo Boolean cuyo valor 1 indica que el usuario también tendrá el rol Practitioner)
Prioridad	Muy alta

Tabla 12. RINF-002: MetaCuestionario

RINF-002	**MetaCuestionario**
Versión	v1.0.0
Descripción	Recurso FHIR Questionnaire en el que se define todos los campos que puede tener un formulario para la solicitud de consentimientos (https://hapi.fhir.org/baseR4/Questionnaire/11149406)
Datos específicos	• Campos obligatorios ○ Título

	○ Pacientes a los que va dirigido el consentimiento ○ Propósito ○ Tiempo de validez ○ Quién/es lo solicitan ○ Acción que se realizará con dicha información • Campos opcionales ○ Periodo de tiempo de la información ○ Tipo de información ○ Tipo de recursos FHIR solicitados ○ Identificador de recursos específicos
Prioridad	Alta

Tabla 13. RINF-003: Información del proceso "solicitudConsentimiento"

RINF-003	**Información del proceso "solicitudConsentimiento"**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001)
Descripción	El proceso "solicitudConsentimiento" deberá manejar los siguientes datos
Datos específicos	• Questionnaire: El MetaCuestionario (RINF-002) • patientList: La lista de pacientes a los que se dirige la solicitud de consentimiento • QuestionnaireResponse: Recurso FHIR QuestionnaireResponse generado una vez que el usuario Practitioner rellena el cuestionario • Title: Titulo de la solicitud de consentimiento • Id_QuestionnaireResponse
Prioridad	Alta

Tabla 14. RINF-004: Información del proceso "confirmacionConsentimiento"

RINF-004	**Información del proceso "confirmacionConsentimiento"**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001)
Descripción	El proceso "confirmacionConsentimiento" deberá manejar los siguientes datos
Datos específicos	• Patient: Identificación del usuario que recibe la solicitud de consentimieto • Title: Título de la solicitud de consentimiento que se recibe • QuestionnaireResponse: Recurso FHIR QuestionnaireResponse generado una vez que el usuario Practitioner rellena el cuestionario • QuestionnaireResponse1: Recurso FHIR QuestionnaireResponse generado una vez que el usuario Patient acepta/deniega el consentimiento
Prioridad	Alta

Tabla 15. RINF-005: Información de instancias de procesos asociadas al Practitioner

RINF-005	**Información de instancias de procesos asociadas al Practitioner**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información del proceso "solicitudConsentimiento" (RINF-003)
Descripción	El sistema deberá guardar la siguiente información de las instancias de los procesos creados asociados al actor Practitioner
Datos específicos	• Id • Practitioner: DNI del usuario • Instance: Id de la instancia del proceso "solicitudConsentimiento" • Title: Título de la solicitud • EndDate: Fecha de vencimiento del consentimiento • QuestionnaireResponse: Id del recurso FHIR QuestionnaireResponse generado una vez el Practitioner rellena la solicitud de consentimiento • Patients: Lista de identificadores de pacientes a los que se dirige la solicitud de consentimiento
Prioridad	Alta

Tabla 16. RINF-006: Información de instancias de procesos asociadas al Patient

RINF-006	**Información de instancias de procesos asociadas al Patient**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información del proceso "confirmacionConsentimiento" (RINF-004)
Descripción	El sistema deberá guardar la siguiente información de las instancias de los procesos creados asociados al actor Patient
Datos específicos	• Id • Patient: DNI del usuario • Instance: Id de la instancia del proceso "confirmacionConsentimiento" • Title: Título de la solicitud
Prioridad	Alta

3.1.2.4 Requisitos de reglas de negocio

Las reglas de negocio establecen normas y restricciones para garantizar la coherencia y el cumplimiento en las operaciones.

Tabla 17. RN-001: Información inicio de sesión de un usuario

RN-001	**Información inicio de sesión de un usuario**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001)
Descripción	El sistema deberá solicitar la siguiente información cuando se quiera proceder a la autenticación de un usuario
Datos específicos	Datos del RINF-001: • DNI • Contraseña
Prioridad	Alta

Tabla 18. RN-002: Restricción de usuarios y roles

RN-002	**Restricción de usuarios y roles**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001)
Descripción	Todos los usuarios tienen el rol de “Patient” pero sólo los usuarios marcados para ello, en el campo “Permisos de Practitioner” del RINF-001, podrán ejercer también el rol de “Practitioner”. El rol será asignado en el momento de la autenticación.
Prioridad	Alta

Tabla 19. RN-003: Identificación única de los usuarios del sistema

RN-003	**Identificación única de los usuarios del sistema**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001)
Descripción	Los usuarios se identifican unívocamente a partir de su DNI, de modo que no podrán registrarse dos usuarios con el mismo DNI, en tal caso, se mostrará un aviso.
Prioridad	Alta

Tabla 20. RN-004: Solicitud de nuevo consentimiento

RN-004	**Solicitud de nuevo consentimiento**

Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información del proceso "solicitudConsentimiento" (RINF-003)
Descripción	Sólo los usuarios con el permiso Practitioner (RINF-001) activo podrán crear solicitudes de consentimientos, y por tanto, crear instancia de los procesos
Prioridad	Alta

Tabla 21. RN-005: Realizar tarea humana del proceso "solicitudConsentimiento"

RN-005	**Realizar tarea humana del proceso "solicitudConsentimiento"**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información del proceso "solicitudConsentimiento" (RINF-003)
Descripción	Sólo los usuarios con el permiso Practitioner (RINF-001) activo podrán realizar la tarea humana del proceso "solicitudConsentimiento", es decir, iniciar y completar la tarea
Prioridad	Alta

Tabla 22. RN-006: Realizar tarea humana del proceso "confirmacionConsentimiento"

RN-006	**Realizar tarea humana del proceso "confirmacionConsentimiento"**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información del proceso "confirmacionConsentimiento" (RINF-004)
Descripción	Sólo el usuario/s al que vaya dirigido la solicitud de consentimiento podrá realizar la tarea humana del proceso "confirmacionConsentimiento", es decir, iniciar y completar la tarea
Prioridad	Alta

Tabla 23. RN-007: Ver solicitudes de consentimientos enviadas

RN-007	**Ver solicitudes de consentimientos enviadas**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información de instancias de procesos asociadas al Practitioner (RINF-005)

Descripción	Sólo los usuarios con el permiso Practitioner (RINF-001) activo podrán ver las solicitudes de consentimientos enviadas por dicho usuario
Prioridad	Alta

Tabla 24. RN-008: Lista de solicitudes de consentimientos enviadas

RN-008	**Lista de solicitudes de consentimientos enviadas**
Versión	v1.0.0
Dependencias	Información de usuario (RINF-001) Información de instancias de procesos asociadas al Patient (RINF-006)
Descripción	Cada día se realizará una comprobación de la lista de solicitudes enviadas (RINF-005) y se compararán que su fecha de validez no sobrepase la fecha actual, en tal caso, se eliminarán los datos almacenados localmente a dicha solicitud.
Prioridad	Alta

Tabla 25. RN-009: MetaCuestionario

RN-009	**MetaCuestionario**
Versión	v1.0.0
Dependencias	MetaCuestionario (RINF-002)
Descripción	Los consentimientos deben seguir un modelo genérico que se encuentra definido en un meta cuestionario específico, el cual está previamente definido en RINF-002 (MetaCuestionario)
Prioridad	Alta

Tabla 26. RN-010: Proceso "solicitudConsentimiento"

RN-010	**Proceso “solicitudConsentimiento”**
Versión	v1.0.0
Dependencias	Información del proceso “solicitudConsentimiento” (RINF-003)
Descripción	El sistema debe definir un proceso para las solicitudes de nuevos consentimientos, cuya información se define en RINF-003 (Información del proceso “solicitudConsentimiento”)
Prioridad	Alta

Tabla 27. RN-011: Proceso "confirmacionConsentimiento"

RN-011	**Proceso “confirmacionConsentimiento”**
Versión	v1.0.0
Dependencias	Información del proceso “confirmacionConsentimiento” (RINF-004)
Descripción	El sistema debe definir un proceso para la confirmación de las solicitudes de consentimientos pendientes, cuya información manejada se define en RINF-004 (Información del proceso “confirmacionConsentimiento”)
Prioridad	Alta

3.1.2.5 Requisitos no funcionales

Los requisitos no funcionales son criterios que describen cómo debe comportarse y desempeñarse el sistema en términos de calidad, seguridad, rendimiento y otros aspectos, en lugar de qué funciones específicas debe realizar

Tabla 28. RNF-001: Gestión de procesos

RNF-001	**Gestión de procesos**
Versión	v1.0.0
Tipo	Técnica
Descripción	La utilización de jBPM para la gestión de procesos en la solicitud y registro del consentimiento de los pacientes en el uso de sus datos sanitarios.

Tabla 29. RNF-002: Autenticación y autorización

RNF-002	**Autenticación y autorización**
Versión	v1.0.0
Tipo	Seguridad
Descripción	Uso de Spring Security, un marco de seguridad para aplicaciones Java basadas en Spring que facilita la autenticación y autorización de usuarios, la gestión de sesiones y la protección contra amenazas comunes, brindando una base sólida para la seguridad en aplicaciones web y servicios REST.

3.2 Implementación

En esta sección se abordará la implementación del proyecto [16], examinando en detalle la estructura y la funcionalidad de cada componente para garantizar el funcionamiento integral y adecuado del sistema.

3.2.1 Estructura

El proyecto está organizado en dos directorios diferentes, y cada uno de ellos cumple una función específica esencial para asegurar el correcto funcionamiento y la integración del sistema. A continuación, se presenta una imagen de la estructura:

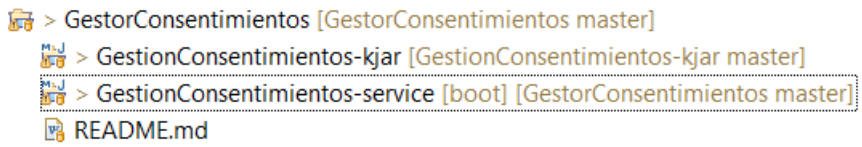

Figura 16. Estructura de directorios

En "GestionConsentimientos-kjar" se encuentran los recursos necesarios para la gestión de procesos (la base de conocimiento que maneja el motor de procesos jBPM), mientras que "GestionConsentimientos-service" alberga los componentes fundamentales para la manipulación de datos, la aplicación y la interfaz de servicio web. Esta división estructural permite una organización eficiente y facilita la administración y mantenimiento del proyecto en su conjunto.

3.2.2 Implementación de los procesos de negocio

Para crear los procesos de negocio necesarios en el proyecto, utilizaremos el workbench Business Central [10], debido a su capacidad para proporcionar una interfaz gráfica que simplifica, entre otras facilidades de interés, la creación de diagramas BPMN.

La base de conocimiento (kjar) incluye la definición de procesos de negocio y otros tipos de activos de empresa. Business Central gestiona esta base de conocimiento como un repositorio git, facilitando así el control de versiones.

Dado que en nuestro caso la aplicación es autónoma, no se ejecuta en el workbench, necesitamos exportar esta base de conocimiento para incluirla en nuestra aplicación. Para ello la carpeta correspondiente (kjar) del proyecto debe poder sincronizarse con el repositorio correspondiente en el workbench, para incluir los activos que especifiquemos en Business Central. El procedimiento que se explica a continuación permitirá esta sincronización posterior, que se verá en el punto 3.2.2.4.

1. Diríjase al directorio "GestiónConsentimientos-kjar" del proyecto
2. Ejecute el comando '`git init`'
3. Ejecute el comando '`git add -A`'
4. Ejecute el comando '`git commit -m "Gestión de consentimientos`' (dentro de las comillas puede colocar cualquier otro mensaje relevante)
5. Inicie sesión en Business Central y vaya al apartado de proyectos
6. Agregue un nuevo espacio de trabajo, en nuestro caso se denomina "tfgv2" (puede asignarle cualquier otro nombre)
7. Dentro del espacio de trabajo creado, seleccione la opción "Agregar proyecto>Importar proyecto" e introduzca la URL de nuestro directorio "GestionConsentimientos-kjar" de la siguiente manera: '`file:///{Ruta}`'

Esto importará el directorio "GestionConsentimientos-kjar" en el workbench Business Central como un nuevo proyecto. A continuación, procederemos a crear los activos de negocio necesarios que serán utilizados por el servicio. En este caso, tendremos dos procesos de negocio.

3.2.2.1 Proceso de negocio “solicitudConsentimiento”

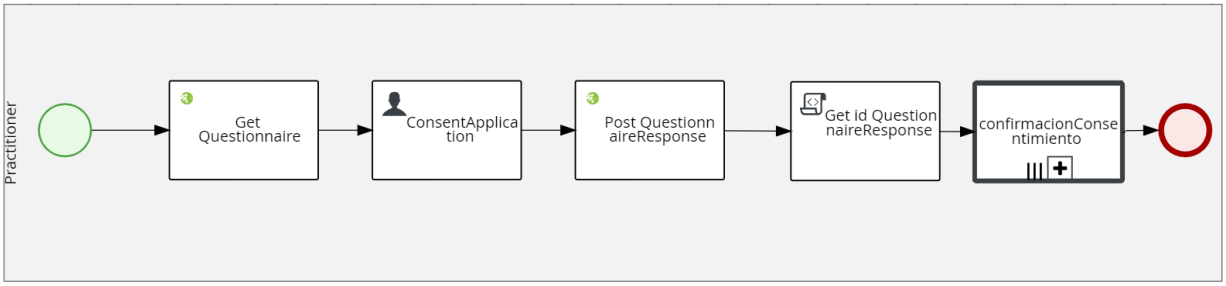

Figura 17. Proceso de negocio “solicitudConsentimiento”

Este es nuestro proceso de negocio correspondiente al Practitioner, que estará compuesto de los siguientes elementos:

1. Get Questionnaire: Esta tarea es una solicitud REST que realiza una petición GET a la URL "https://hapi.fhir.org/baseR4/Questionnaire/11149406" para obtener el MetaCuestionario. El resultado de esta solicitud se almacenará en una variable llamada "questionnaire" de tipo String, como se especificó en RINF-005 (Información de instancias de procesos asociadas al Practitioner).

   Es relevante destacar que el recurso FHIR Questionnaire que se presenta ha sido concebido y desarrollado por nuestra entidad, con el propósito de llevar a cabo las necesarias tareas, y dado que la implementación de un servidor FHIR no constituye el objetivo principal de nuestro proyecto, hemos optado por alojarlo en el servidor público proporcionado por HAPI-FHIR. Es fundamental reconocer que esta decisión conlleva una debilidad potencial, ya que el servidor en cuestión no está bajo nuestra gestión directa, lo que implica que podría estar sujeto a modificaciones sin nuestro control.

2. ConsentApplication: Esta tarea es una actividad humana que debe ser completada por usuarios pertenecientes al grupo Practitioner. La tarea toma el "questionnaire" (que contiene el MetaCuestionario) como entrada y produce las siguientes salidas:

   - "questionnaireResponse" (Tipo String): Contendrá el recurso FHIR QuestionnaireResponse generado por el Practitioner una vez que haya completado el formulario de solicitud de consentimiento.
   - "patientList" (Tipo Object): Contiene una lista de identificaciones (DNIs) de los pacientes a los que se dirige la solicitud de consentimiento.
   - "title" (Tipo String): Contiene el título asignado por el usuario Practitioner a la solicitud de consentimiento.

3. Post QuestionnaireResponse: Esta tarea es una solicitud REST que realiza una petición POST a la URL "https://hapi.fhir.org/baseR4/QuestionnaireResponse", y el campo "ContentData" contendrá la variable "questionnaireResponse" generada anteriormente. Esta tarea se utiliza para persistir el recurso FHIR QuestionnaireResponse en el servidor público HAPI FHIR. Como resultado, se obtendrán varios datos, incluido el ID del QuestionnaireResponse, que se almacenarán en la variable "id_QuestionnaireResponse".

4. Get id QuestionnaireResponse: Este es un script que, a partir de la variable de tipo String "id_QuestionnaireResponse", la convierte en un JSONObject y obtiene el ID del QuestionnaireResponse persistido anteriormente. La variable de tipo String "id_QuestionnaireResponse" se modificará para que tenga el valor de dicho ID.

5. confirmacionConsentimiento: Este es un subproceso reutilizable con múltiples instancias que, al recibir como entrada una colección, creará tantas nuevas instancias del proceso “confirmacionConsentimiento” como elementos tenga la colección proporcionada. En este caso, la colección de entrada será la variable de tipo Object "patientList". Como parámetros de entrada para las nuevas instancias de procesos, se proporcionarán las variables "id_QuestionnaireResponse" y "title".

Este proceso de negocio permite la gestión de consentimientos por parte de los Practitioners y la interacción con recursos FHIR para llevar a cabo estas operaciones de manera eficiente y organizada.

3.2.2.2 Proceso de negocio “confirmacionConsentimiento”

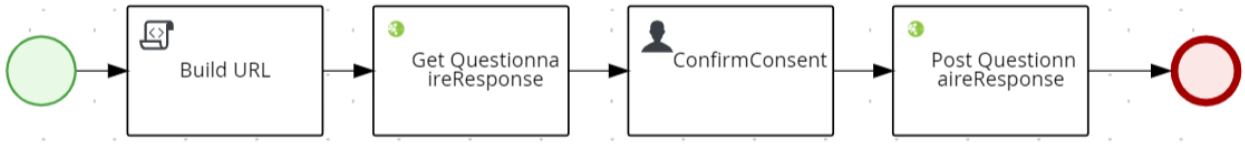

Figura 18. Proceso de negocio “confirmacionConsentimiento”

Este es nuestro proceso de negocio correspondiente al paciente (Patient), que está compuesto por los siguientes elementos:

1. Build URL: Este componente es un script que, utilizando la variable "id_QuestionnaireResponse" (que contiene el ID del recurso FHIR QuestionnaireResponse) proporcionada como parámetro de entrada al proceso, genera una variable de tipo String denominada "URL". El valor de esta variable es "https://hapi.fhir.org/baseR4/QuestionnaireResponse/{ID}", donde "ID" se corresponde con el valor de la variable "id_QuestionnaireResponse".
2. Get QuestionnaireResponse: Esta tarea consiste en una solicitud REST que realiza una petición GET a la URL cuyo valor se encuentra en la variable "URL". Su objetivo es obtener el recurso FHIR QuestionnaireResponse generado por el Practitioner que solicitó el consentimiento. El resultado de esta solicitud se almacena en una variable llamada "questionnaireResponse" de tipo String, conforme a lo especificado en RINF-006.
3. ConfirmConsent: Esta actividad humana debe ser completada por usuarios pertenecientes al grupo Patient. La tarea toma como entrada el "questionnaireResponse" y produce como salida la variable de tipo String "questionnaireResponse1". Esta variable contiene el recurso FHIR QuestionnaireResponse generado por el paciente una vez que haya aceptado o denegado la solicitud de consentimiento.
4. Post QuestionnaireResponse: Esta tarea es una solicitud REST que realiza una petición POST a la URL "https://hapi.fhir.org/baseR4/QuestionnaireResponse". El campo "ContentData" contiene la variable "questionnaireResponse1" generada previamente. La finalidad de esta tarea es persistir el recurso FHIR QuestionnaireResponse en el servidor público HAPI FHIR. Como resultado, se obtienen varios datos, incluyendo el ID del QuestionnaireResponse, que se almacenan en la variable "id_questionnaireResponse1".

Este proceso de negocio permite que los pacientes interactúen con el recurso FHIR QuestionnaireResponse, respondan a las solicitudes de consentimiento y persistan sus respuestas en el servidor FHIR para su posterior uso.

Por tanto, tendremos los siguientes activos de negocio:

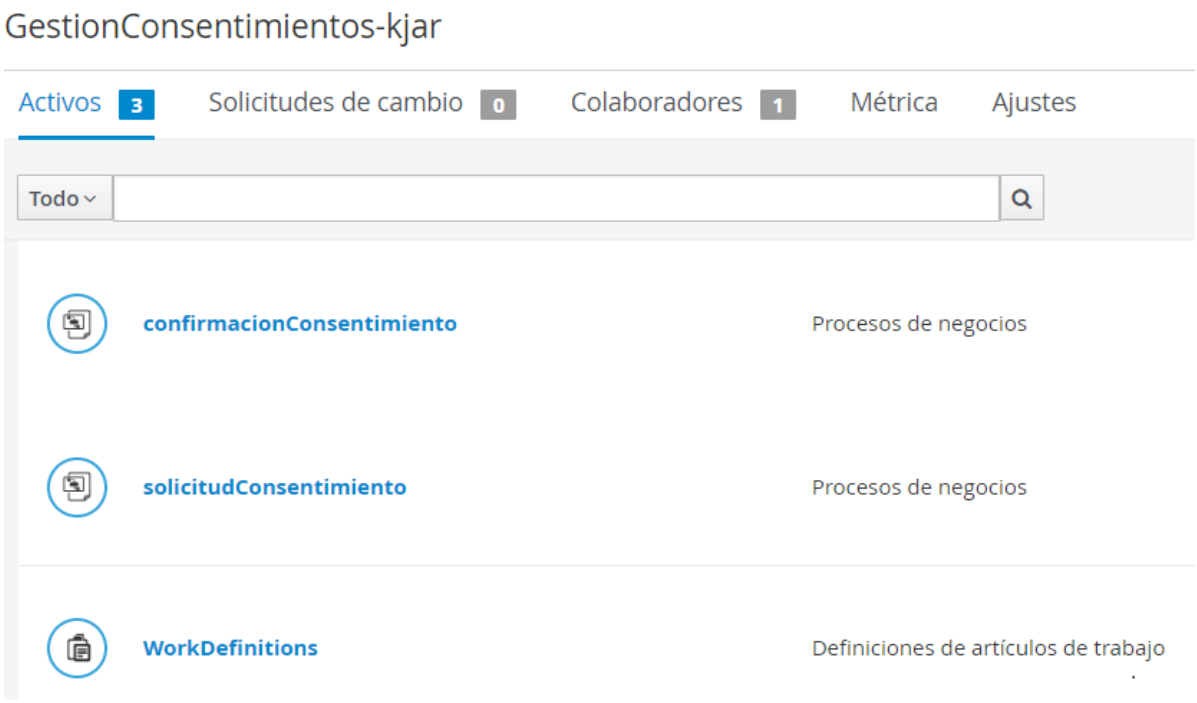

Figura 19. Activos comerciales del proyecto

3.2.2.3 Modificación en la configuración del proyecto

Con el objetivo de asegurar la gestión efectiva de las tareas REST asociadas a los activos de tipo proceso de negocio, es esencial tomar en consideración que en los proyectos de Business Central no se dispone de manera predeterminada de ningún WorkItemHandler para REST. Por lo tanto, se requiere llevar a cabo una breve modificación en la configuración para su creación. A continuación, se detalla el procedimiento para llevar a cabo esta tarea.

1. Acceda al apartado "Ajustes" en el proyecto de Business Central.
2. Dentro de los ajustes, navegue hasta el apartado "Implementaciones" y seleccione "Manejadores de los ítems de trabajo".
3. Agregue un nuevo manejador haciendo clic en la opción correspondiente y asigne los siguientes valores:
    - En el campo "Nombre", ingrese "Rest".
    - En el campo "Valor", introduzca la siguiente expresión: "new org.jbpm.process.workitem.rest.RESTWorkItemHandler()".
4. Haga clic en el botón "Guardar" para confirmar la configuración.

Esta modificación en la configuración permitirá tener disponibles el WorkItemHandler de REST que las tareas se manejen adecuadamente dentro de los activos de negocio relacionados con el proceso de negocio del proyecto.

3.2.2.4 Sincronizar los activos de negocio en el código fuente del proyecto

Hasta este momento, los activos de negocio solamente están disponibles en el workbench de Business Central y no están reflejados en el código fuente del proyecto. Como se comentó en el apartado 3.2.2, necesitamos exportar la base de conocimiento para incluirla en nuestra aplicación. Para ello, seguimos los siguientes pasos:

1. Diríjase al directorio “GestiónConsentimientos-kjar” del proyecto
2. Ejecute el siguiente comando para configurar el origen del repositorio Git:

    “`git remote add origin`

`ssh://wbadmi@localhost:8001/{Espacio}/GestionConsentimientos-kjar`", donde "Espacio" representa el nombre del espacio, en nuestro caso, sería "tfgv2".

3. Ejecute el siguiente comando para realizar una extracción (pull) de la rama "master" desde el origen (origin) del repositorio Git, "`git pull origin master`", y cuando se le solicite, ingrese la contraseña del usuario "wbadmin".

Estos pasos permitirán que los activos sean incorporados en el código fuente del proyecto y estén disponibles para su uso en la aplicación.

3.2.3 Implementación de los servicios

En esta sección se detallará en profundidad el contenido del directorio "GestionConsentimientos-service". En él se encuentra la implementación de la interfaz y el servicio web, el cuál interactúa con el motor de procesos jBPM.

Este directorio está dividido en diferentes paquetes.

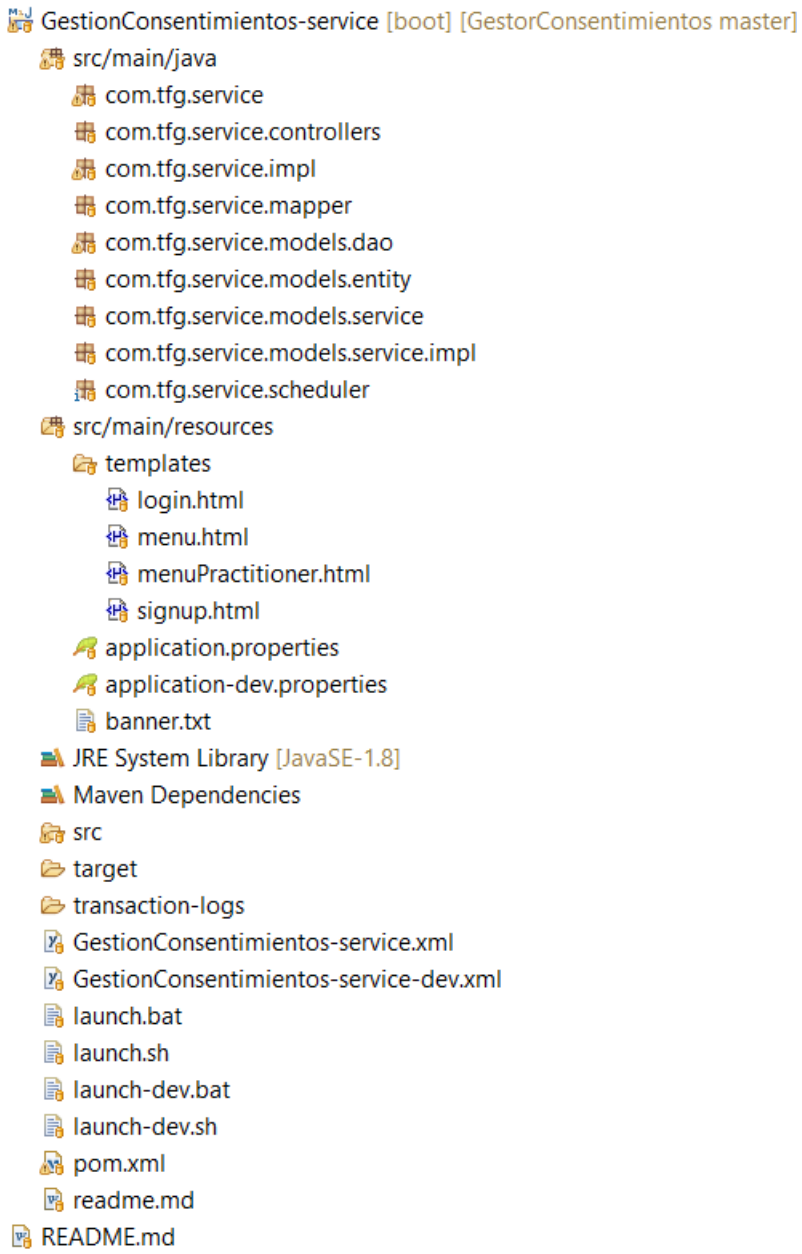

Figura 20. Contenido del directorio GestionConsentimientos-service

3.2.3.1 Paquete com.tfg.service

Dentro de este paquete, se encuentra la clase "`Application`", la cual representa el punto de entrada principal de una aplicación Java basada en Spring Boot. Esta clase anotada con "`@SpringBootApplication`" define el método principal (`main`) que inicia la aplicación

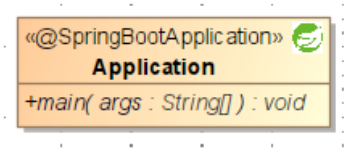

Figura 21. Clase Application

Además de la anterior, también se encuentra la clase "`DefaultWebSecurityConfig`", la cual representa una configuración de seguridad para una aplicación Java basada en Spring Boot [19]. A continuación, se presenta una descripción general de su funcionalidad:

1. Configuración de Seguridad: Esta clase está anotada con "`@Configuration`" y "`@EnableWebSecurity`", lo que la identifica como una configuración de seguridad para la aplicación.
2. Extiende `WebSecurityConfigurerAdapter`, que es una clase base proporcionada por Spring Security para configurar la seguridad web. Esto permite personalizar la seguridad de la aplicación según las necesidades específicas del proyecto.
3. Configuración de Autorización: En el método "`configure(HttpSecurity http)`", se definen las reglas de autorización y autenticación para las rutas de la aplicación. Esto incluye:
    - Configuración de CORS y CSRF.
    - Definición de rutas públicas ("/", "/login/**", "/signup/**") que no requieren autenticación.
    - Restricción de acceso a rutas bajo "/rest/*" a usuarios autenticados.
    - Configuración de inicio de sesión (formLogin) con páginas de inicio de sesión personalizadas y rutas relacionadas.
    - Habilitación de autenticación básica HTTP.
    - Desactivación de la protección contra ataques de fotograma (frameOptions). Estos ataques buscan alterar la apariencia o el contenido de la página web con el propósito de obtener ganancias ilícitas o robar información confidencial.
4. Configuración de Autenticación: En el método "`configureGlobal(AuthenticationManagerBuilder auth)`", se configura el servicio de detalles de usuario ("userDetailsService") y el codificador de contraseñas utilizado para autenticar a los usuarios.
5. Configuración de CORS (Cross-Origin Resource Sharing): Dentro de la clase se define un método anotado con "`@Bean`" denominado "`corsConfigurationSource`", que especifica la configuración CORS para permitir solicitudes desde cualquier origen ("*") y métodos HTTP permitidos.

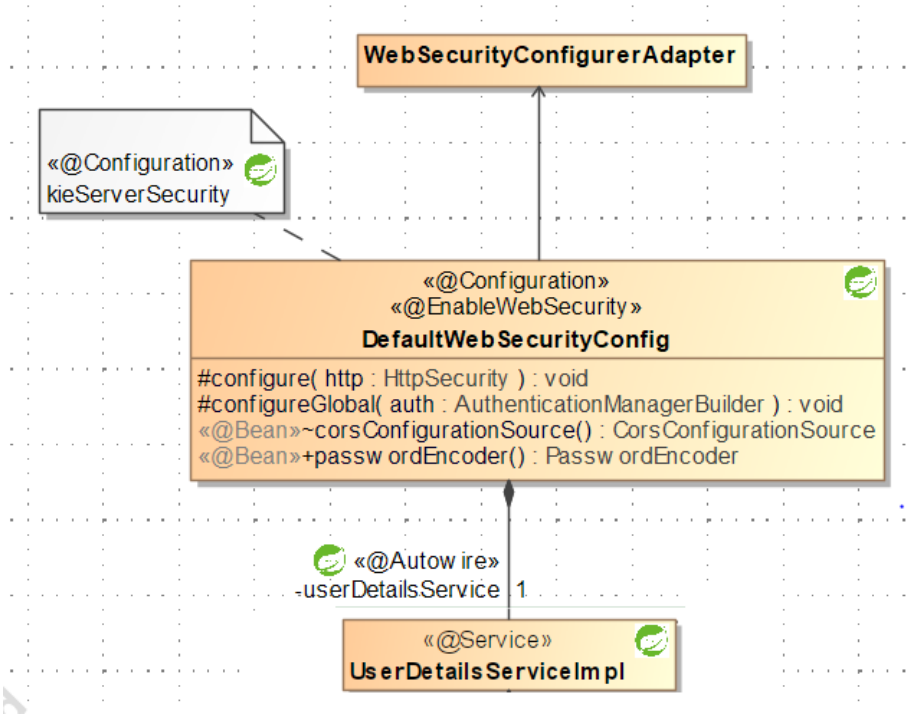

Figura 22. Clase DefaultWebSecurityConfig

Por último, la clase "`KieUtil`" facilita la interacción con el servidor KIE (Knowledge Is Everything) para la gestión de reglas y procesos empresariales. A continuación, se proporciona una descripción general de su funcionalidad:

- Configuración de Conexión: La clase almacena información de configuración, ubicada en el fichero `application.properties`, para conectarse a un servidor KIE. Esta información incluye la URL del servidor, el nombre de usuario y la contraseña necesarios para autenticar la aplicación con el servidor KIE.
- Creación de Clientes KIE: La clase proporciona métodos para obtener diferentes tipos de clientes KIE, como "`ProcessServicesClient`", "`UserTaskServicesClient`", y "`QueryServicesClient`". Estos clientes se utilizan para interactuar con diferentes aspectos del servidor KIE, como la ejecución de procesos, la gestión de tareas y la consulta de datos.
- Configuración de Formato de Marshalling: La clase configura el formato de marshalling (serialización/deserialización) de los datos cuando se comunican con el servidor KIE. En este caso, se configura para utilizar el formato JSON.
- Inicialización de Clientes KIE: Los métodos para obtener clientes KIE utilizan la información de configuración proporcionada para crear una instancia de "`KieServicesClient`" y luego obtener el cliente KIE específico requerido (por ejemplo, `ProcessServicesClient`, `UserTaskServicesClient`, etc.).

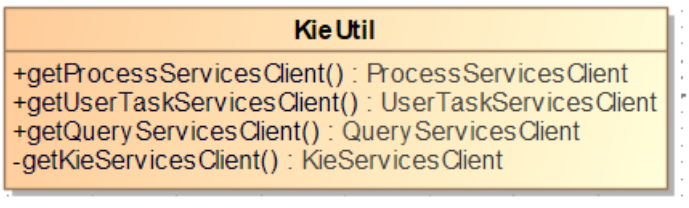

Figura 23. Clase KieUtil

3.2.3.2 Paquete com.tfg.service.models

En este paquete se engloba todo lo relacionado con los modelos manejados por la aplicación.

3.2.3.2.1 Entidades (paquete com.tfg.service.models.entity)

En este contexto, se desarrollan las clases de tipo "Entity". Estas clases desempeñan un papel fundamental en la interacción del sistema con la base de datos, ya que se encargan de mapear las clases Java a las tablas del modelo de datos. Para llevar a cabo este mapeo, se utilizan anotaciones como "`@Entity`" que informan a JPA (Java Persistence API) que se trata de una clase entidad, "`@Id`" para definir la clave primaria, "`@Column`" para describir cada atributo, "`@Table`" para asociar la tabla correspondiente, y "`@GeneratedValue`" para especificar cómo se generan los valores de una columna, generalmente en combinación con la anotación "`@Id`". [21]

Este paquete es esencial para la persistencia y recuperación de datos en el contexto del sistema, garantizando que la información se almacene y se recupere de manera coherente y eficiente en la base de datos H2.

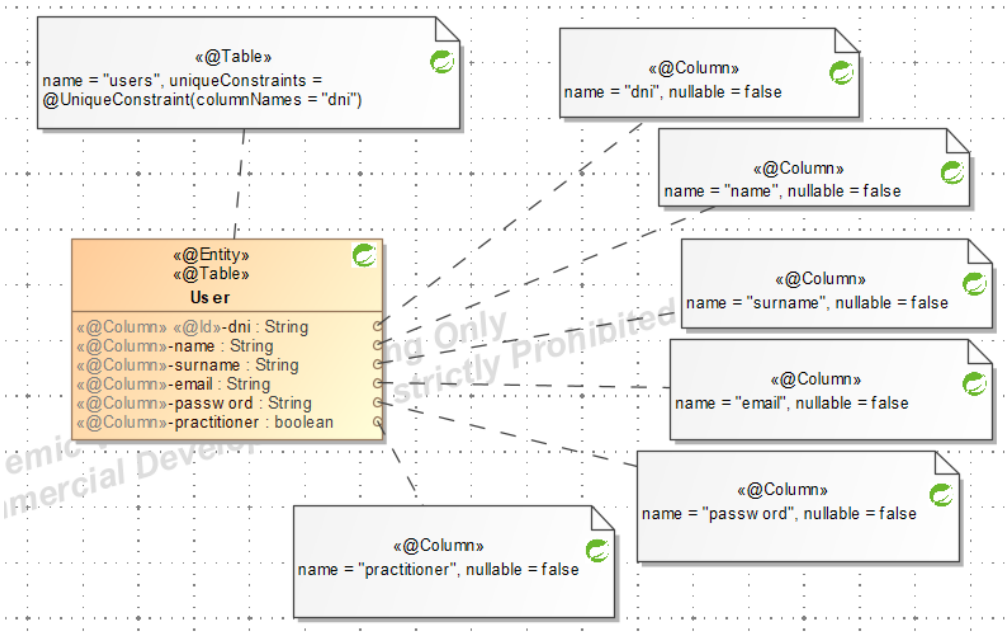

Figura 24. Clase de entidad User

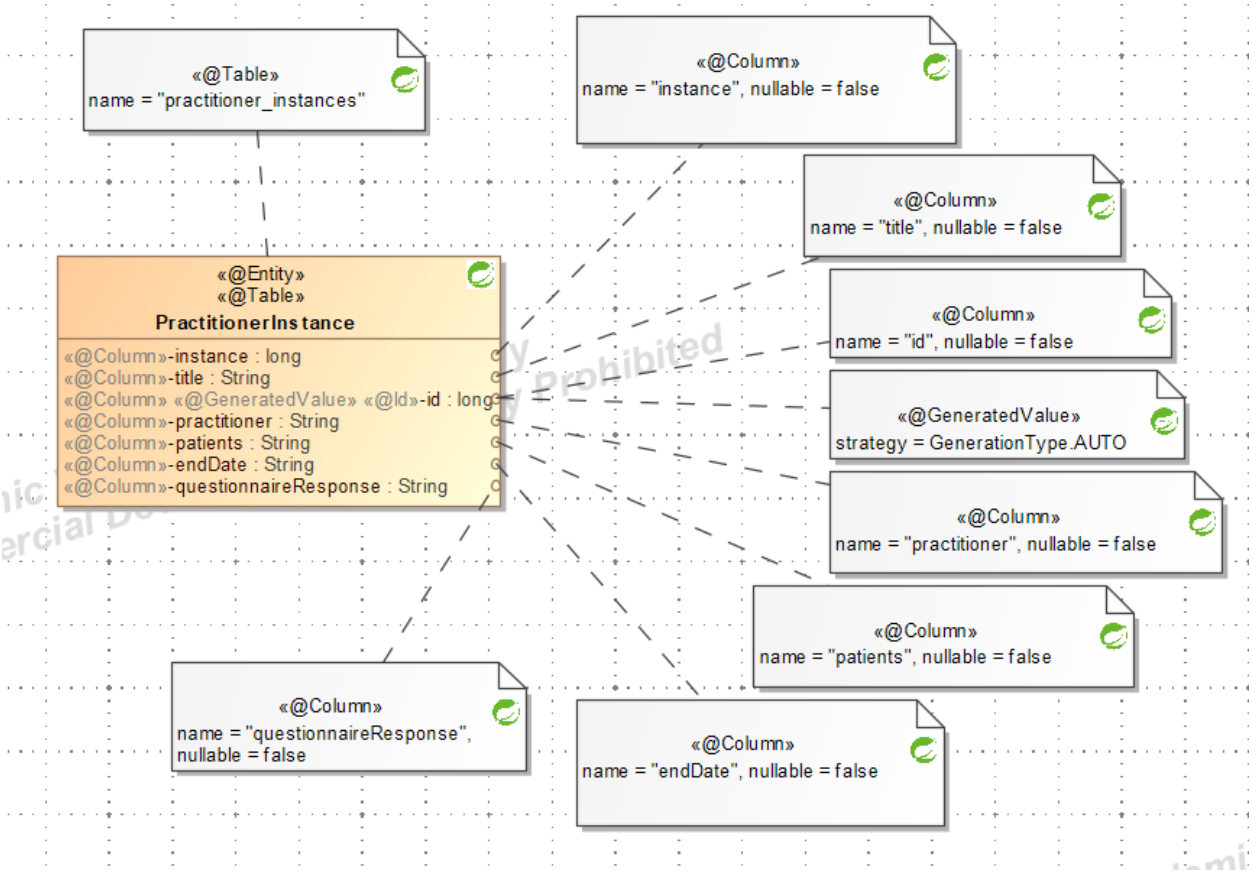

Figura 25. Clase de entidad PractitionerInstance

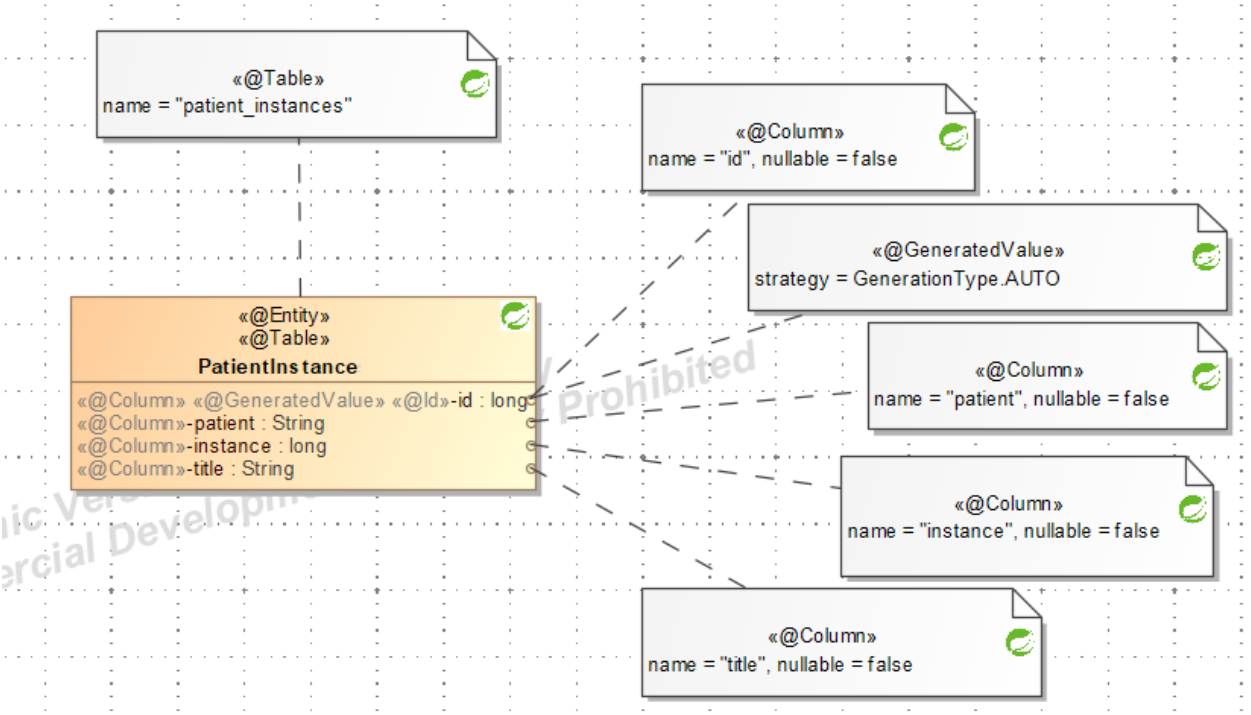

Figura 26. Clase de entidad PatientInstance

3.2.3.2.2 Repositorios (paquete com.tfg.service.dao)

En este paquete se encuentran interfaces que han sido marcadas con la anotación "`@Repository`" y que, a su vez, heredan de la interfaz proporcionada por Spring Data JPA denominada "`JpaRepository`" [23]. Este enfoque, en esencia, simplifica significativamente la interacción con bases de datos relacionales al ofrecer

métodos esenciales para llevar a cabo operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera eficiente.

La notación "`@Repository`" tiene el propósito de indicar que estas interfaces desempeñan el rol de intermediario con la base de datos, encargados de facilitar el acceso y manipulación de información en la base de datos subyacente. Además, Spring Framework asume la responsabilidad de la configuración y la implementación de estas interfaces, lo que conlleva una simplificación substancial en el proceso de desarrollo de aplicaciones.

Este enfoque también sigue el patrón de diseño DAO (Data Access Object), el cual abstrae y separa la lógica de acceso a datos de la lógica de negocio. En este contexto, las interfaces "`@Repository`" se asemejan a los DAOs, proporcionando un medio para realizar operaciones de acceso a datos de manera consistente y reutilizable, promoviendo así una estructura de código modular y fácilmente mantenible.

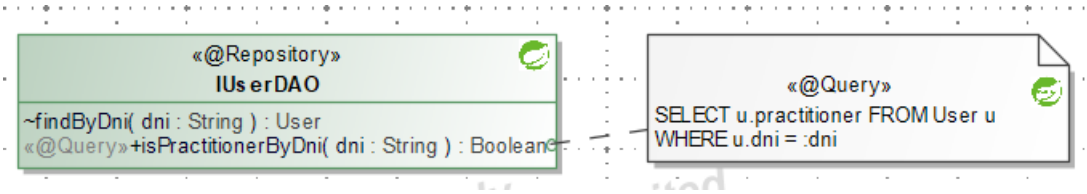

Figura 27. Interfaz IUserDAO

Figura 28. Interfaz IPractitionerInstancesDAO

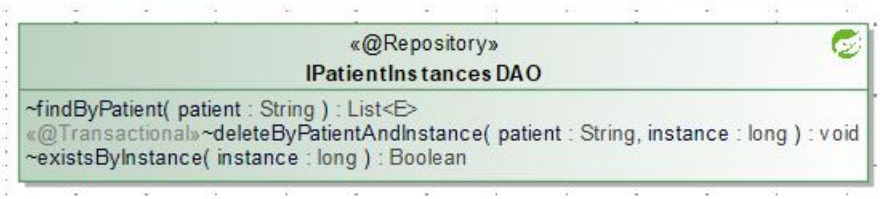

Figura 29. Interfaz IPatientInstancesDAO

3.2.3.2.3 Servicios (paquetes com.tfg.service.models.service y com.tfg.service.models.service.impl)

En este apartado se encuentran las clases identificadas con la anotación "`@Service`". Esta anotación habilita a Spring para encargarse de la creación y gestión de las instancias de dichas clases, simplificando la inyección de dependencias y la administración de transacciones en el contexto de la aplicación.

En nuestro contexto particular, en este paquete específico, se encuentran dos clases de servicio. La primera de ellas se denomina "`UserServiceImpl`" y se encarga de implementar la interfaz "`IUserService`" que nosotros mismos hemos definido. Esta clase se utiliza para llevar a cabo diversas operaciones relacionadas con la entidad "`User`".

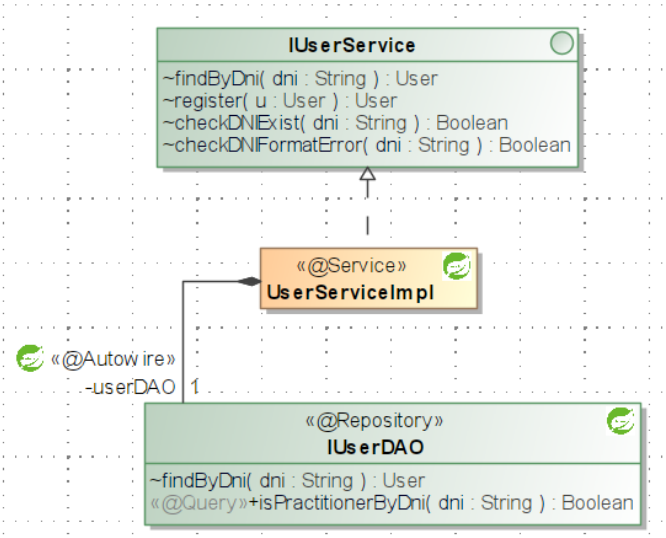

Figura 30. Clase de servicio UserServiceImpl

La segunda clase, llamada "`UserDetailsServiceImpl`", implementa la interfaz "`UserDetailsService`" proporcionada por Spring Security [19]. Esta implementación se utiliza para personalizar el proceso de recuperación de detalles de usuario en el contexto de Spring Security. En concreto, esta clase toma el nombre de usuario proporcionado, lo busca en la base de datos y, en caso de encontrarlo, genera un objeto "`UserDetails`" con la información del usuario correspondiente. En el caso de que el usuario no exista, se lanza una excepción de tipo "`UsernameNotFoundException`". Esta capacidad de personalización resulta fundamental para lograr una autenticación y autorización ajustada a las necesidades específicas de Spring Security.

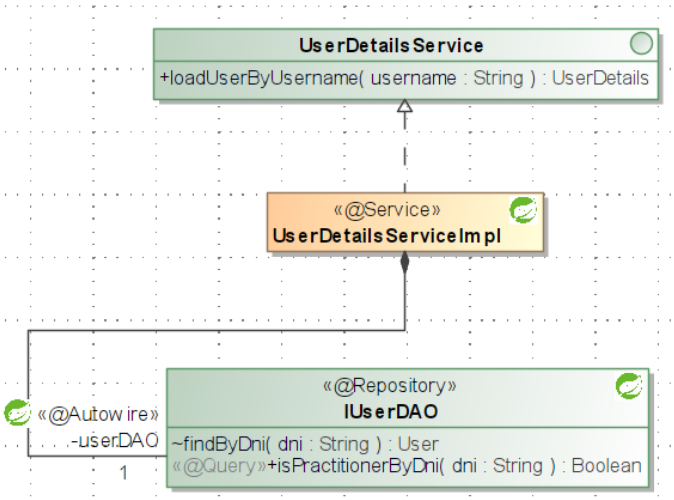

Figura 31. Clase de servicio UserServiceDetailsImpl

3.2.3.3 Paquete com.tfg.service.mapper

En este paquete se localizarán las clases diseñadas para facilitar la conversión de tipos de datos. Con este propósito, se ha creado una interfaz denominada "`IMapper`" que deberá ser implementada por todas las clases desarrolladas en este paquete. Esta interfaz incluye dos parámetros genéricos, "I" que representa la entrada, y "O" que representa la salida. Además, se define un método llamado "`map`" que retornará un resultado de tipo "O" y recibirá un argumento de tipo "I".

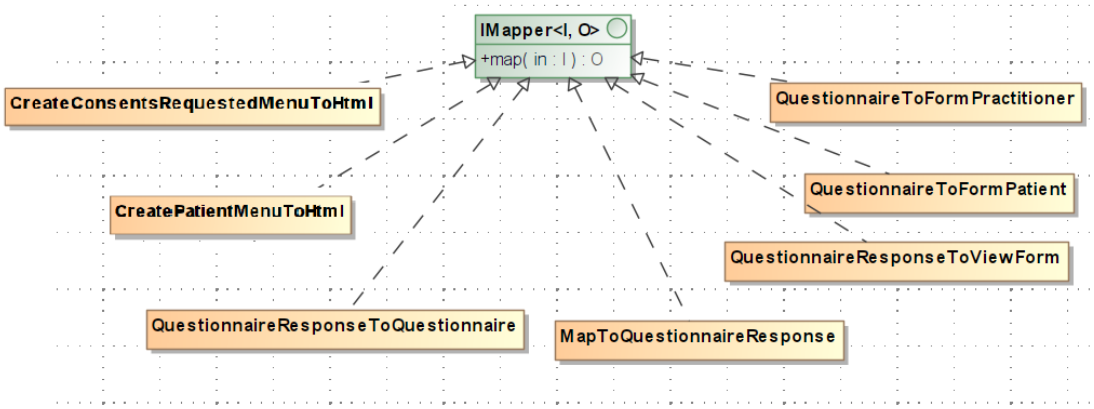

Figura 32. Contenido del paquete com.tfg.service.mapper

Las clases implementadas cumplen las siguientes funciones:

- `QuestionnaireToFormPractitioner`: Esta clase toma como parámetro de entrada un recurso FHIR de tipo Questionnaire que contiene el Metacuestionario y devuelve un String cuyo contenido es una representación en formato HTML que muestra el Metacuestionario como un formulario
- `QuestionnaireToFormPatient`: Recibe como entrada un recurso FHIR de tipo Questionnaire que corresponde al cuestionario solicitado por el Practitioner. Luego, genera un String con una representación en formato HTML que muestra ese cuestionario como un formulario.
- `MapToQuestionnaireResponse`: Esta clase recibe un Map que contiene las respuestas a los formularios mencionados anteriormente y, en función de esto, genera un recurso FHIR QuestionnaireResponse.
- `QuestionnaireResponseToQuestionnaire`: Realiza un mapeo de un recurso FHIR QuestionnaireResponse, que se proporciona como parámetro de entrada, a un recurso FHIR Questionnaire, que es el resultado de la operación.
- `CreatePatientMenuToHtml`: Se le proporciona un Map que contiene las instancias de procesos de solicitudes de consentimientos pendientes para el Patient y las transforma en un String con contenido HTML para su visualización.
- `CreateConsentRequestsMenuToHtml`: Recibe un Map que contiene las instancias de procesos de solicitudes de consentimientos enviadas por un Practitioner específico y genera un String con contenido HTML que muestra estas instancias en pantalla.
- `QuestionnaireResponseToViewForm`: A partir de un String que contiene el ID de un recurso FHIR QuestionnaireResponse, genera un String con contenido HTML para mostrar en pantalla la solicitud de consentimiento enviada por un Practitioner.

3.2.3.4 Paquete com.tfg.service.impl

Dentro de este paquete se encuentran las clases de servicio que se encargarán de enviar solicitudes al servidor KIE para gestionar los procesos de negocio de jBPM previamente creados.

En este contexto, se destacan dos clases principales. La primera de ellas es la denominada `PractitionerServiceImpl`, la cual implementa la interfaz `IPractitionerService` que contiene métodos para gestionar el proceso “solicitudConsentimiento”.

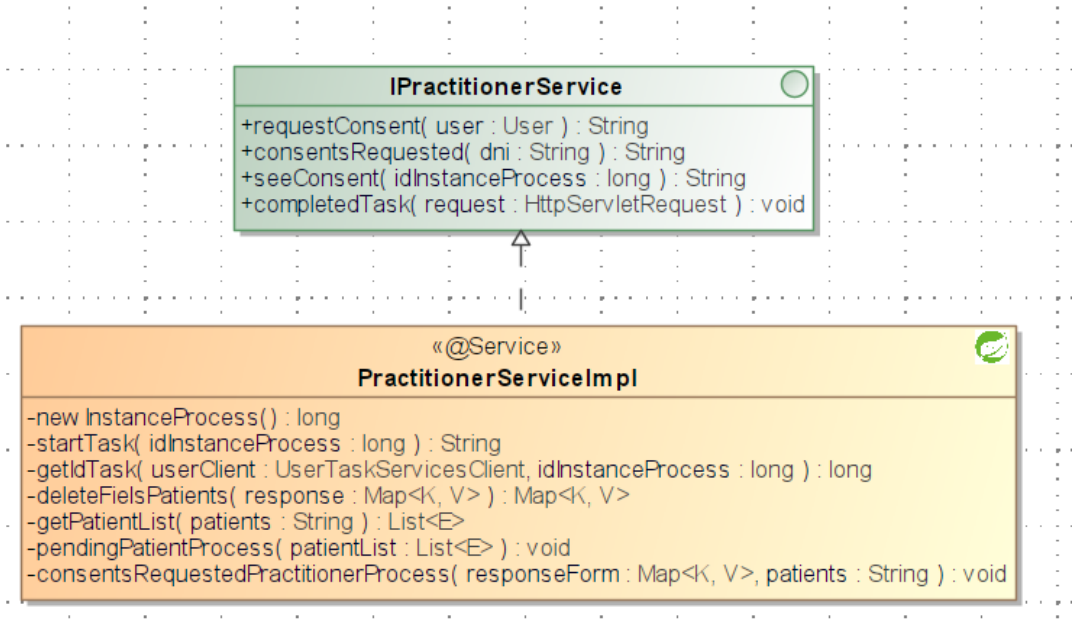

Figura 33. Clase de servicio PractitionerServiceImpl

Sus métodos realizan las siguientes funciones:

- `requestConsent`: El propósito de este método consiste en la obtención del MetaCuestionario definido en la RN-009 (Metacuestionario) en formato HTML. Este método requiere como parámetro de entrada un objeto de tipo `User` y, como resultado, retorna un valor de tipo String.

  Para ello, se lleva a cabo la creación de una nueva instancia del proceso denominado "solicitudConsentimiento". Este proceso se inicia mediante una llamada al método privado `newInstanceProcess()`, que devuelve un objeto de tipo Long. Este objeto Long representa el identificador único de la instancia de proceso recién creada. A continuación, se procede a invocar la función privada `startTask()`, pasando como parámetro de entrada el objeto Long previamente obtenido.

  Este método, en su fase inicial, se encarga de obtener el identificador de la tarea humana [4] dentro de la instancia de proceso en la que estamos trabajando. Esto se logra mediante una llamada a la función privada `getIdTask()`, que retorna un objeto de tipo Long con dicho identificador. Una vez que se ha obtenido el identificador de la tarea, se procede a iniciar la tarea y a obtener el parámetro de entrada que se configuró para ello en el apartado 3.2.2.1 Este parámetro de entrada contiene el Metacuestionario.

  El siguiente paso, en `requestConsent`, consiste en convertir este MetaCuestionario, que es un String, en un recurso FHIR Questionnaire. Una vez convertido, se invoca el método de la clase `QuestionnaireToFormPractitioner`. Este método transforma el recurso FHIR Questionnaire y devuelve un String representación en formato HTML del formulario correspondiente al Metacuestionario.

  Finalmente, este String será el valor de retorno del método `requestConsent()`.

- `completedTask`: La función cumple con la responsabilidad de ejecutar la finalización de la tarea humana [4] correspondiente. Este método acepta como parámetro de entrada un objeto `HttpServletRequest` y, al ser de tipo void, no retorna ningún valor.

  Para ello, una vez que el Practitioner ha completado el formulario de solicitud de consentimiento, los datos se recopilan en un objeto de tipo `HttpServletRequest`. Dicho objeto será convertido en un mapa de tipo "Map<String, String[]", que contendrá los valores completados por el Practitioner en el formulario.

  Posteriormente, se procede a la conversión de este mapa en un recurso FHIR de tipo QuestionnaireResponse, mediante la invocación de un método perteneciente a la clase

`MapToQuestionnaireResponse`. Una vez obtenido este recurso FHIR, se efectúa su conversión a una representación en forma de cadena de caracteres (String), la cual se incorpora en un objeto de tipo "Map<String, Object>", que albergará las variables de salida requeridas para la tarea humana, garantizando así su correcta finalización.

Una vez que la tarea humana ha sido completada, se procederá a persistir los datos necesarios para otras funcionalidades del sistema mediante las llamadas a los métodos privados `consentsRequestedPractitionerProcess` y `pendingPatientProcess`.

- `consentsRequested`: Toma como parámetro de entrada un valor de tipo String que representa el número de DNI de un profesional de la salud. Este método se utiliza para realizar una consulta de solicitudes de consentimientos. Durante esta consulta, se recuperan todas las solicitudes de consentimiento que han sido previamente enviadas por el usuario mencionado y que aún se encuentran en un estado de vigencia, es decir, cuya fecha de finalización aún no ha llegado. El resultado de la ejecución de este método es un objeto de tipo String que contiene la representación de estas solicitudes en formato HTML.

- `seeConsent`: Este método recibe como parámetro de entrada un objeto de tipo Long que representa un identificador de una instancia del proceso "solicitudConsentimiento". A través de este identificador, se accede a los datos almacenados en el sistema a través del objeto correspondiente `PractitionerInstance`. A partir de la información obtenida de esta fuente, se extrae el identificador del recurso FHIR QuestionnaireResponse creado en ese proceso. Este identificador se utiliza posteriormente como parámetro en la invocación del método de la clase "`QuestionnaireResponseToViewForm`". El resultado de esta operación es un objeto de tipo String que contiene la representación del recurso en formato HTML, el cual será el valor de retorno del método.

La segunda clase ubicada en este paquete se denomina `PatientServiceImpl` e implementa la interfaz `IPatientService`, que proporciona los métodos para gestionar las instancias del proceso "confirmacionConsentimiento".

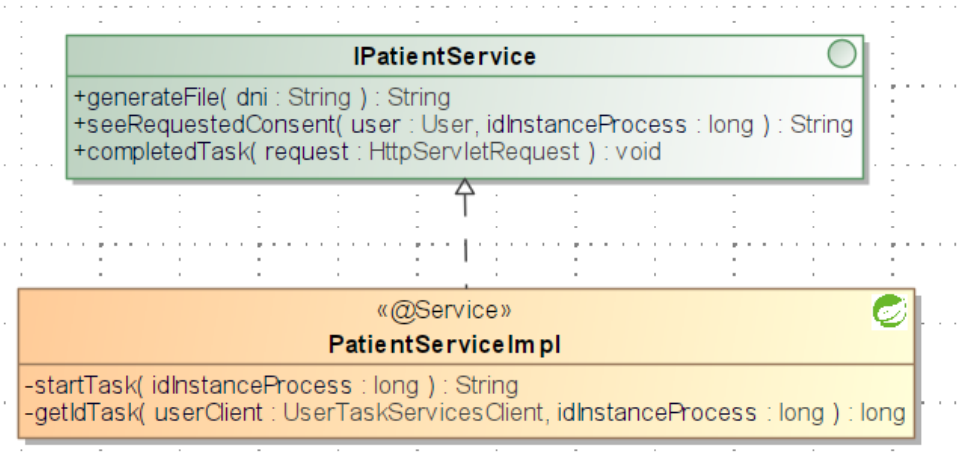

Figura 34. Clase de servicio PatientServiceImpl

Sus métodos realizan las siguientes funciones:

- `generateFile`: Este método, del parámetro de entrada de tipo String, debe devolver un String con la solicitud de consentimiento que se le solicita.

  Para ello, a partir del parámetro de entrada, que representa el número de DNI de un Patient, realiza una consulta para recuperar todas las solicitudes de consentimiento pendientes de respuesta. Estas solicitudes se almacenan en un objeto de tipo Map, que posteriormente se pasa como parámetro en la llamada al método de la clase "`CreatePatientMenu`". El resultado de esta operación es un objeto de tipo String que se devuelve como resultado del método. El contenido de este String representa las mencionadas solicitudes en formato HTML

- `seeRequestedConsent`: Este método, a partir del identificador de proceso proporcionado como

parámetro de entrada, asume la responsabilidad de obtener el identificador de la tarea humana [4] dentro de la instancia de proceso en curso. Para lograr esto, se invoca la función privada `getIdTask()`, la cual devuelve un objeto de tipo Long que contiene dicho identificador.

Una vez que se ha obtenido el identificador de la tarea, se inicia la tarea y se recupera el parámetro de entrada asociado a ella. Este parámetro contiene el cuestionario con la solicitud de consentimiento requerida.

El siguiente paso, dentro del método `seeRequestedConsent`, implica la conversión de esta solicitud, que se encuentra en formato String, en un recurso FHIR de tipo Questionnaire. Una vez realizado este proceso de conversión, se llama al método de la clase `QuestionnaireToFormPatient`. Este método transforma el recurso FHIR Questionnaire y devuelve una representación en formato HTML del formulario correspondiente a la solicitud de consentimiento.

Finalmente, el valor de retorno de este método será precisamente el String mencionado anteriormente, que representa el formulario en formato HTML.

- `completedTask`: La función cumple con la responsabilidad de ejecutar la finalización de la tarea humana [4] correspondiente. Este método acepta como parámetro de entrada un objeto `HttpServletRequest` y, al ser de tipo void, no retorna ningún valor.

  Para ello, una vez que el Patient ha completado el formulario de solicitud de consentimiento, los datos se recopilan en un objeto de tipo `HttpServletRequest`. Dicho objeto será convertido en un mapa de tipo "Map<String, String[]", que contendrá los valores completados por el Patient en el formulario.

  Luego, se procede a convertir este mapa en un recurso FHIR de tipo QuestionnaireResponse mediante la llamada a un método presente en la clase `MapToQuestionnaireResponse`. Una vez que se ha obtenido este recurso FHIR, se lleva a cabo su conversión en una representación en forma de cadena de caracteres (String). Este resultado se incorpora en un objeto de tipo "Map<String, Object>", el cual contendrá las variables de salida necesarias para la actividad humana, asegurando de esta manera su correcta finalización.

  Una vez que la tarea humana ha sido completada, se ejecuta la eliminación de los datos persistidos localmente en el sistema que corresponden a ese usuario y a ese identificador de proceso.

3.2.3.5 Paquete com.tfg.service.controller

En este paquete, se encuentran todas las clases marcadas con la anotación "`@Controller`". Esta anotación le indica a Spring que estas clases son controladoras y, por lo tanto, responsables de gestionar las solicitudes HTTP entrantes y coordinar la lógica de control. Para garantizar un manejo adecuado de esta lógica y asegurar que las solicitudes HTTP se procesen de manera correcta, también se hace uso de las anotaciones "`@RequestMapping`", "`@GetMapping`", "`@PostMapping`" y "`@ResponseBody`" [26]. La última de estas, "`@ResponseBody`", se emplea para especificar que el valor de retorno del método debe serializarse directamente en la respuesta HTTP, en lugar de considerarse como una vista que se renderizará.

Adicionalmente, se utiliza la anotación "`@RequestParam`" para asignar los parámetros de una solicitud a los parámetros del método controlador correspondiente. Asimismo, la anotación "`@ModelAttribute`" se emplea para establecer la conexión entre los datos de un objeto y el modelo, facilitando así la transferencia de datos entre el controlador y la vista.

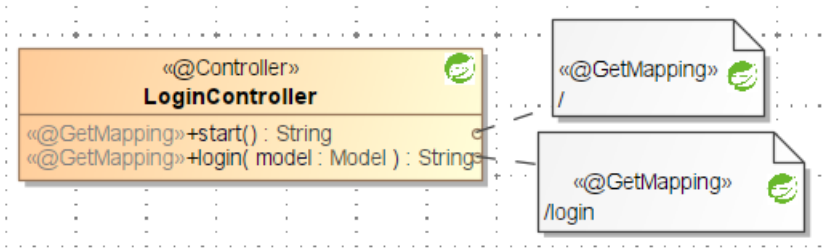

Figura 35. Clase controladora LoginController

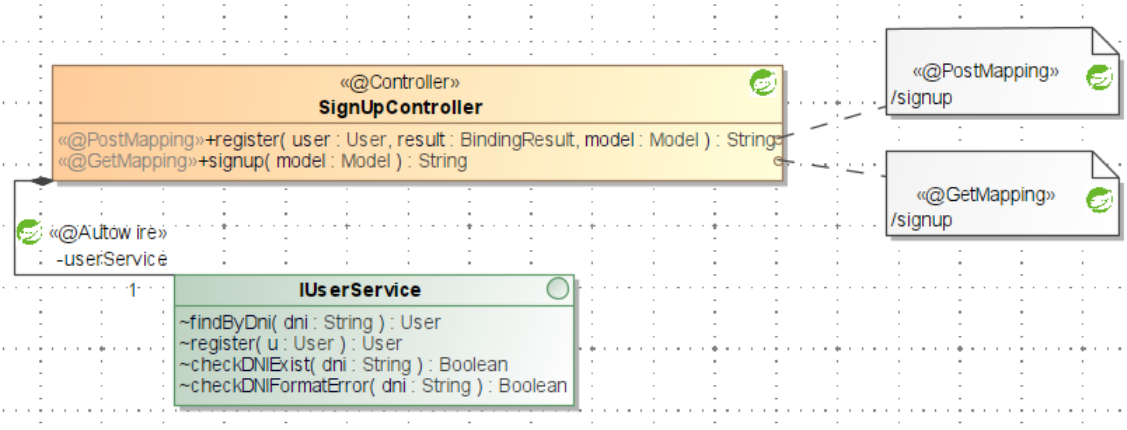

Figura 36. Clase controladora SignUpController

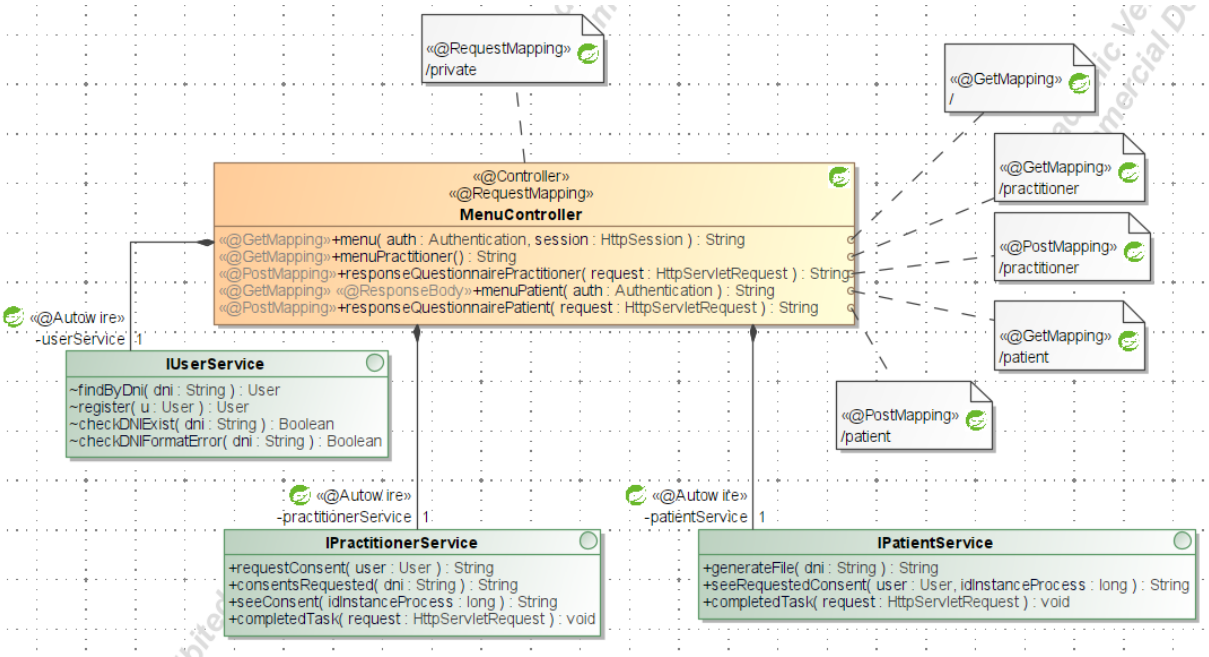

Figura 37. Clase controladora MenuController

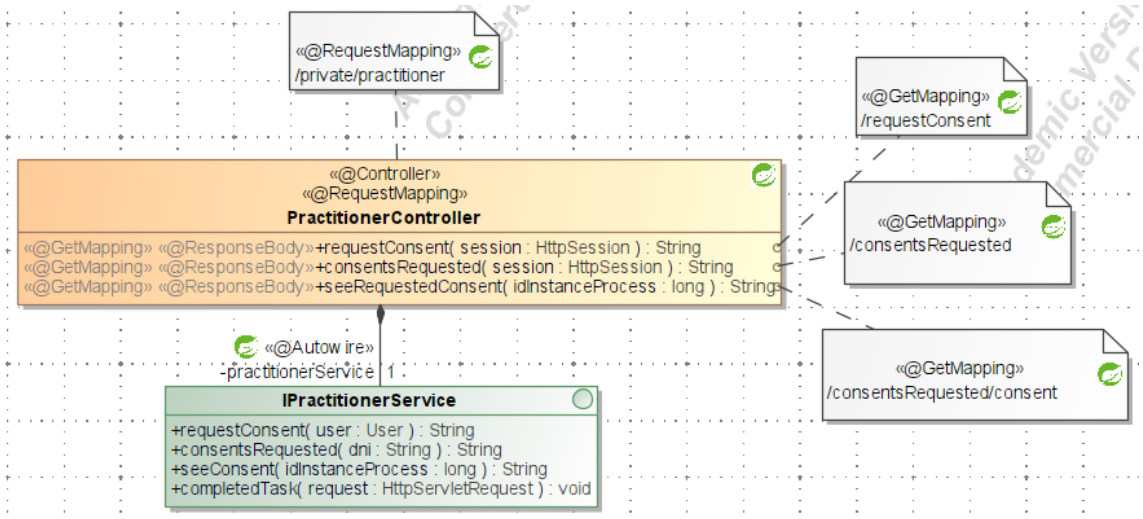

Figura 38. Clase controladora PractitionerController

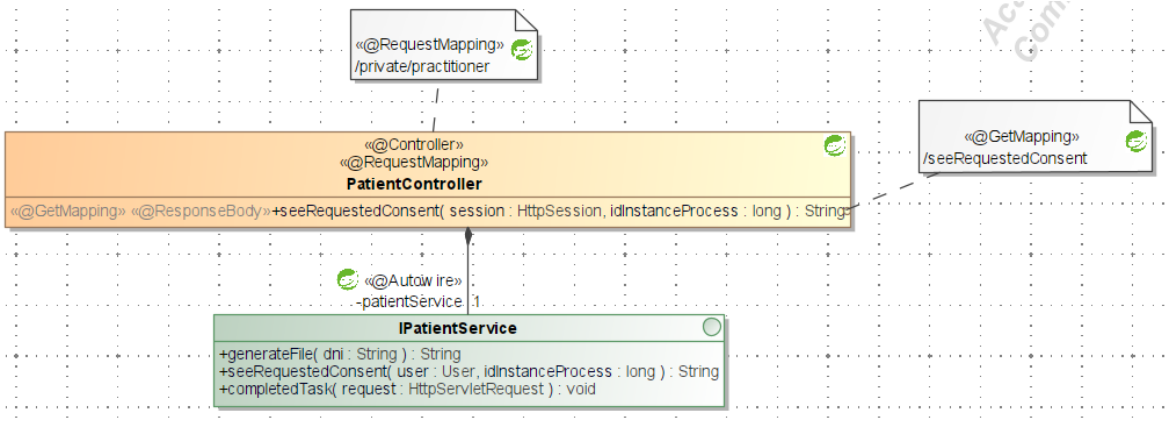

Figura 39. Clase controladora PatientController

3.2.3.6 Paquete com.tfg.service.scheduler

Dentro de este paquete se encuentran las clases necesarias para llevar a cabo una verificación diaria de las solicitudes de consentimiento enviadas por los profesionales de la salud (Practitioners) con el fin de asegurarse de que sigan siendo válidas, es decir, que aún no hayan alcanzado su fecha de finalización indicada en cada una de ellas. Para lograr este objetivo, se llevará a cabo un recorrido completo de la tabla en la base de datos denominada "practitioner_instances". Durante este proceso, se examinará el campo "`endDate`" de cada objeto `PractitionerInstance` y se comparará con la fecha actual. Únicamente en los casos en los que la fecha actual no sea posterior a la fecha analizada, se procederá a la eliminación del objeto `PractitionerInstance` correspondiente.

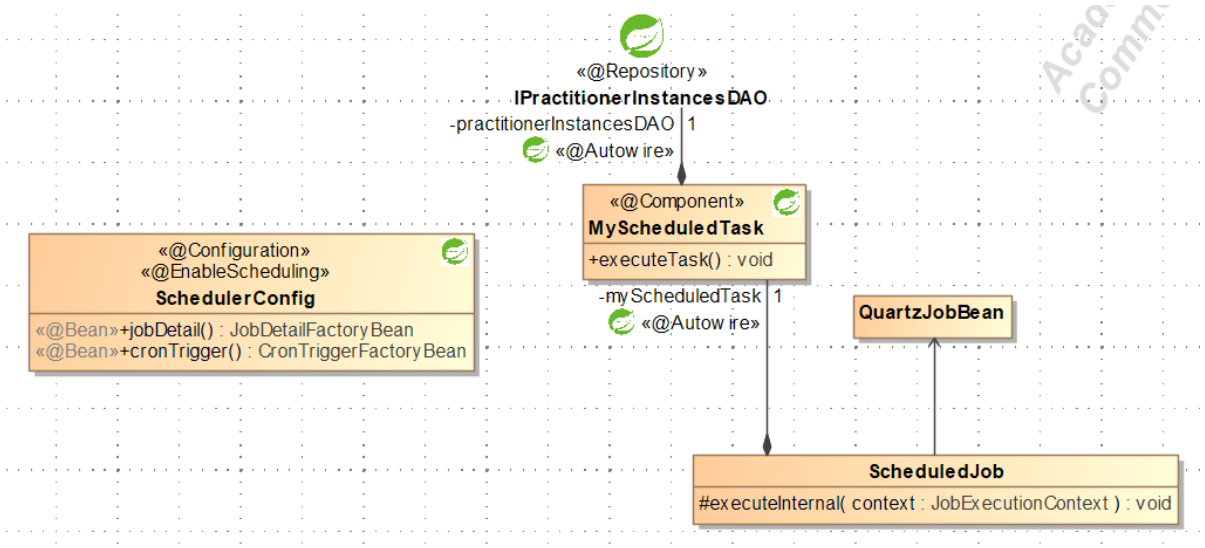

Figura 40. Paquete scheduler

3.2.3.7 Interfaz gráfica de la aplicación web

En nuestra aplicación, empleamos HTML como el lenguaje principal para crear las interfaces gráficas que hacen posible la interacción de los usuarios con nuestra plataforma. HTML, o Hypertext Markup Language, es la piedra angular de la web y nos proporciona una estructura sólida para representar y presentar información en línea de manera eficaz y coherente.

Además de HTML, aprovechamos la potencia y la versatilidad de Thymeleaf, un motor de plantillas diseñado especialmente para aplicaciones web en Java [29]. Thymeleaf actúa como un puente entre nuestros componentes Java en el servidor y la presentación de datos en las páginas web. Esto nos permite crear vistas dinámicas y altamente personalizables que se generan en el servidor antes de ser entregadas a los navegadores de los usuarios.

Es importante destacar que los archivos HTML que no requieren generación dinámica se encuentran ubicados en la carpeta "resource" de nuestro sistema, manteniendo así un orden estructurado y organizado en nuestra aplicación.

A continuación, presentamos las vistas específicas para cada uno de los casos de uso.

En primer lugar, se muestra la pantalla de inicio, donde se solicitan las credenciales para la autorización de usuario.

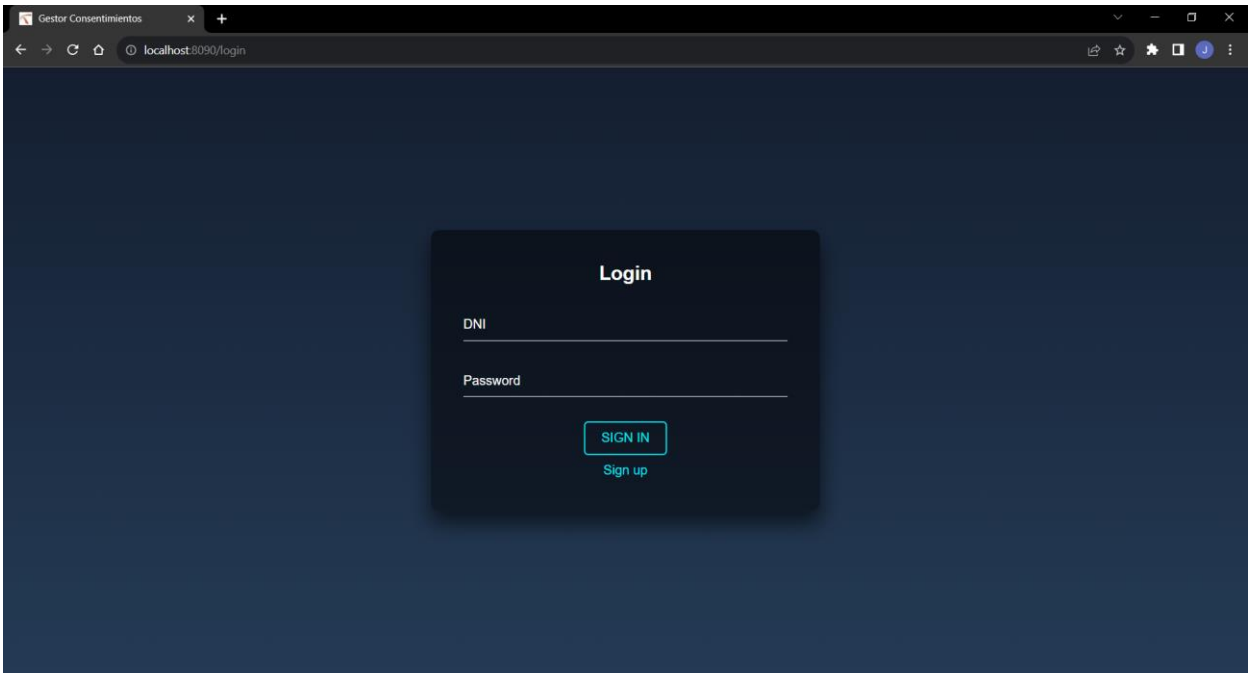

Figura 41. Vista inicial para el CU-001

En la Figura 42, se ilustra la vista de registro de los usuarios, en ella se muestran los datos que estos deben de introducir para registrarse en la aplicación.

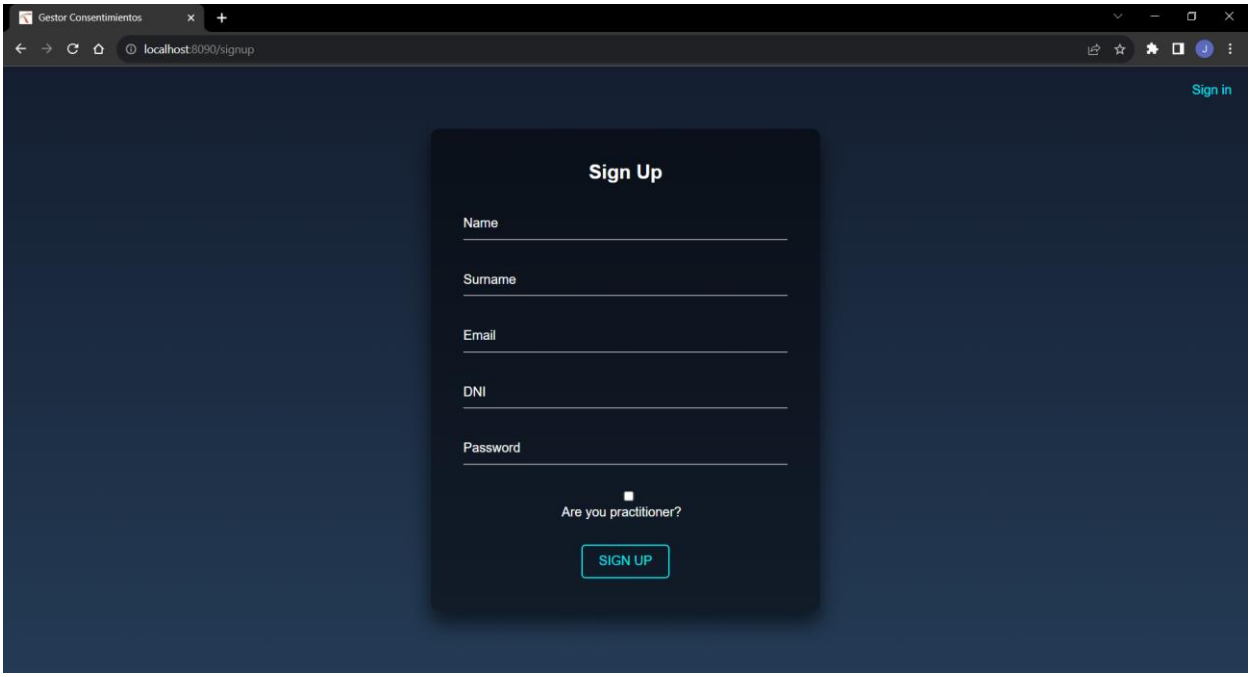

Figura 42. Vista inicial para el CU-002

En las siguientes capturas, se muestra el cuestionario genérico que el Practitioner debe rellenar para solicitar un consentimiento.

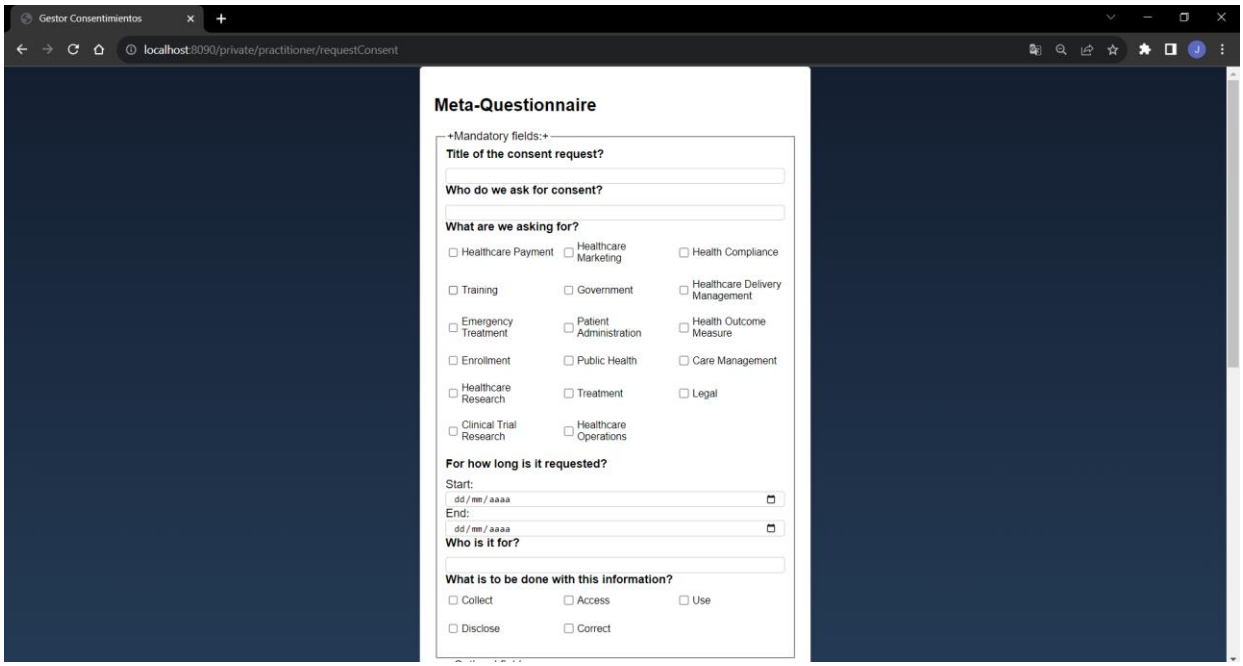

Figura 43. Vista inicial para el CU-003 (parte 1)

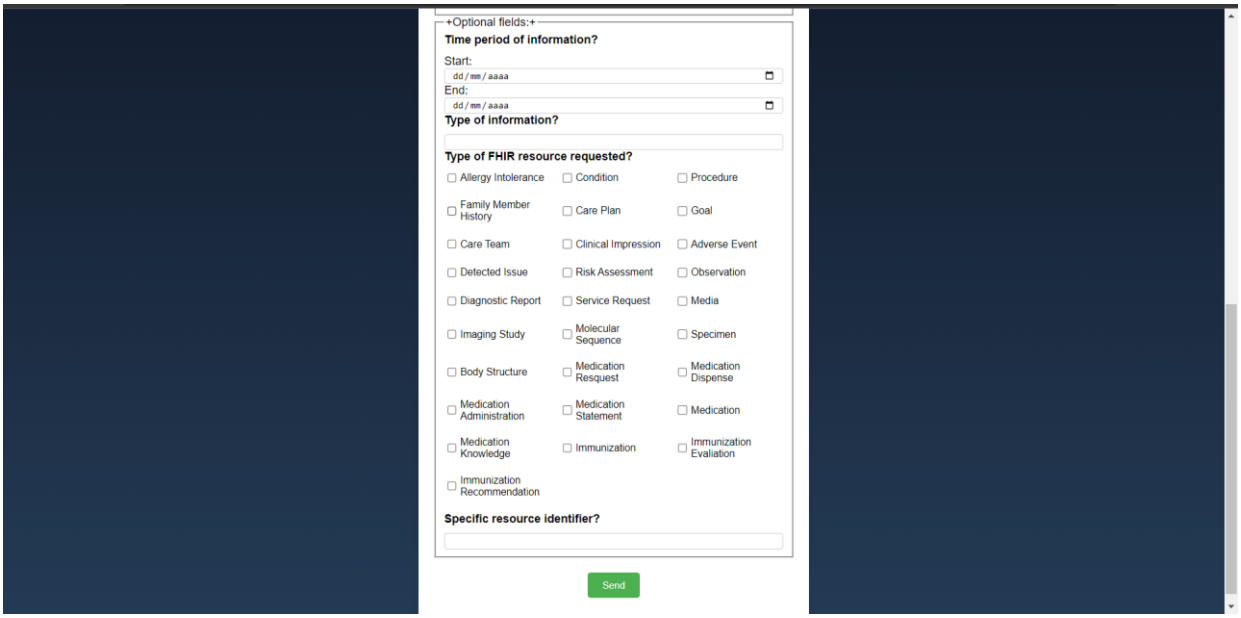

Figura 44. Vista inicial para el CU-003 (parte 2)

En la Figura 45, se muestran las solicitudes de consentimientos enviadas por el Practitioner.

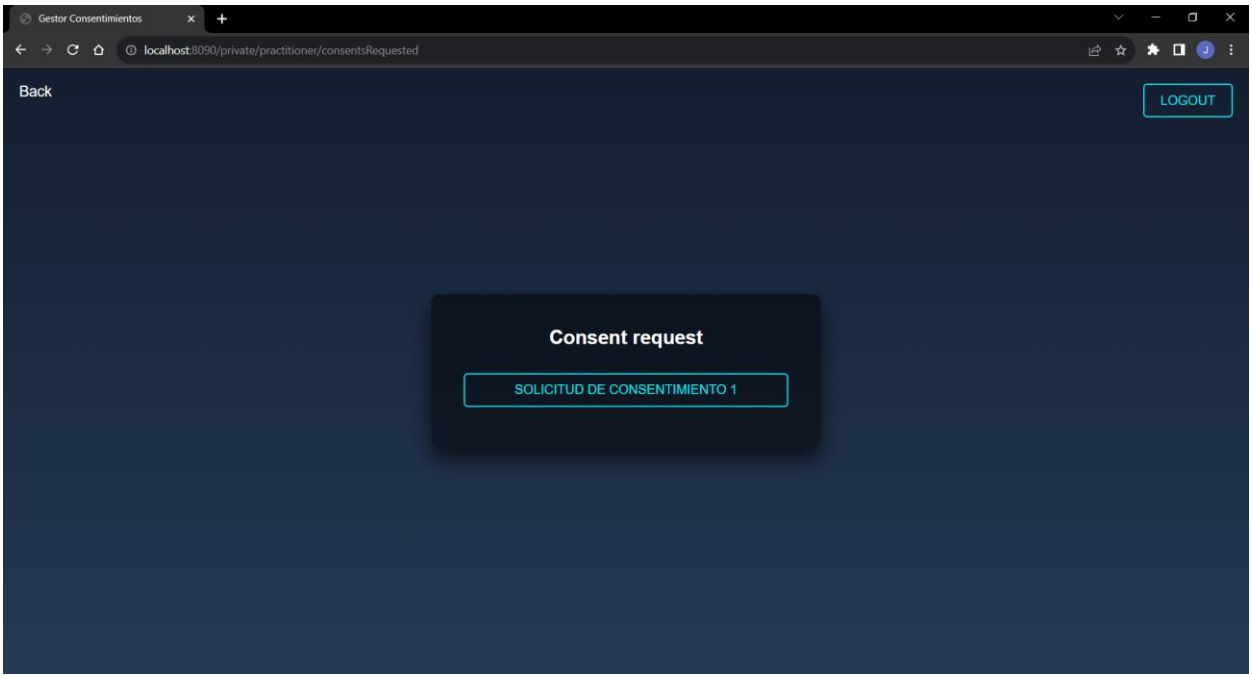

Figura 45. Vista inicial para el CU-004

En las siguientes capturas, se observa la solicitud enviada por el Practitioner, que ha debido ser seleccionada de entre todas enviadas que se mostraban en la Figura 45.

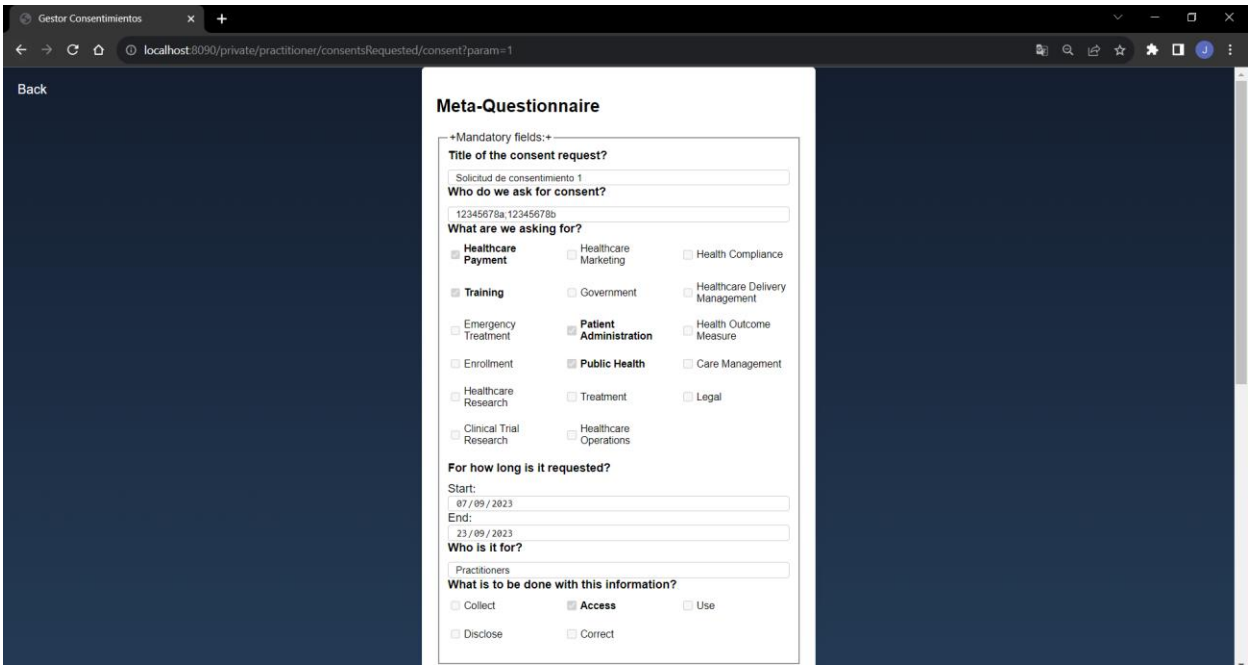

Figura 46. Vista inicial para el CU-005 (parte 1)

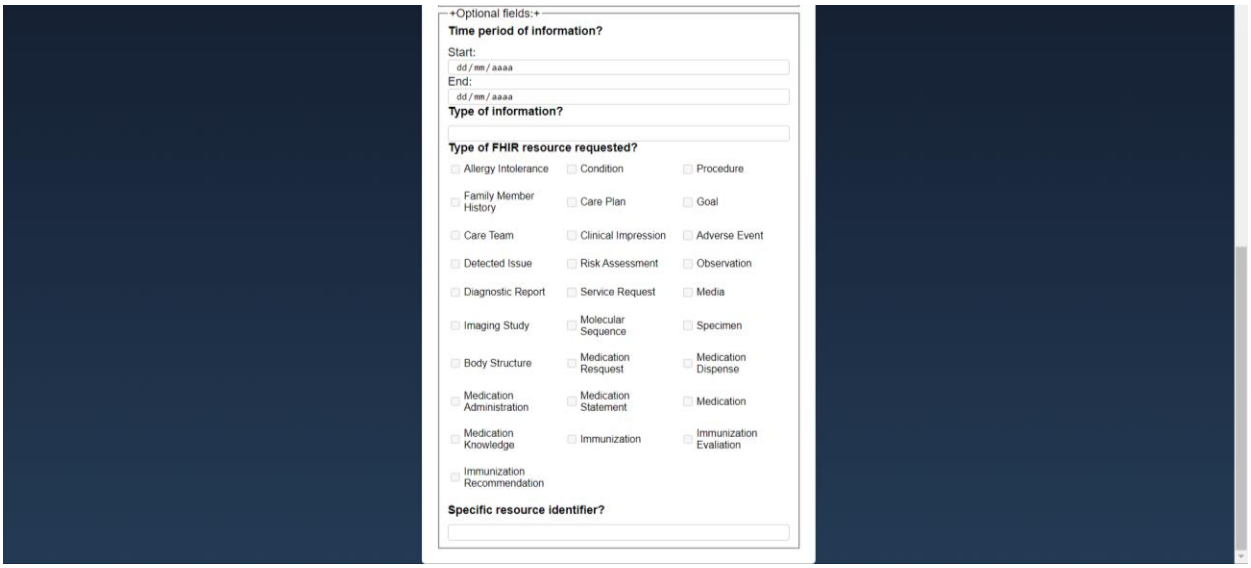

Figura 47. Vista inicial para el CU-005 (parte 2)

En la Figura 48, se muestran todas las solicitudes de consentimientos pendientes que tiene el Patient.

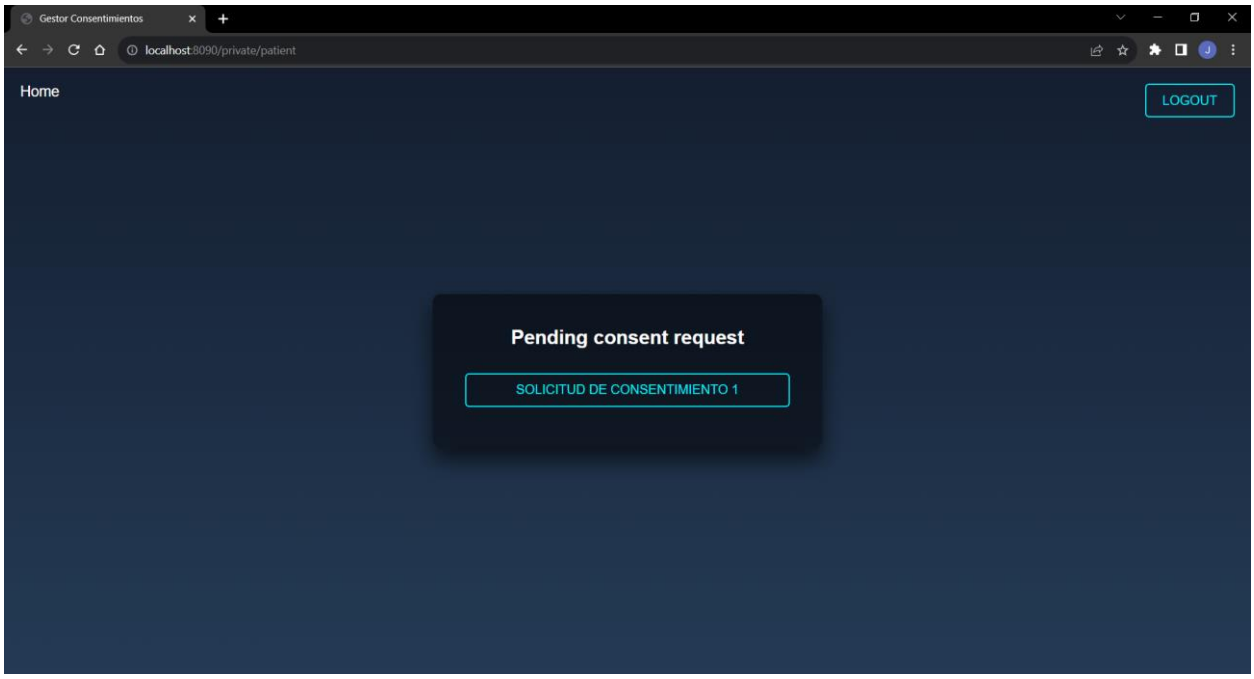

Figura 48. Vista inicial para el CU-006

Por último, en la siguiente figura, una vez seleccionada una de las solicitudes pendientes de la Figura 48, se muestra la solicitud de consentimiento, y en ella el Patient la puede aceptar o denegar.

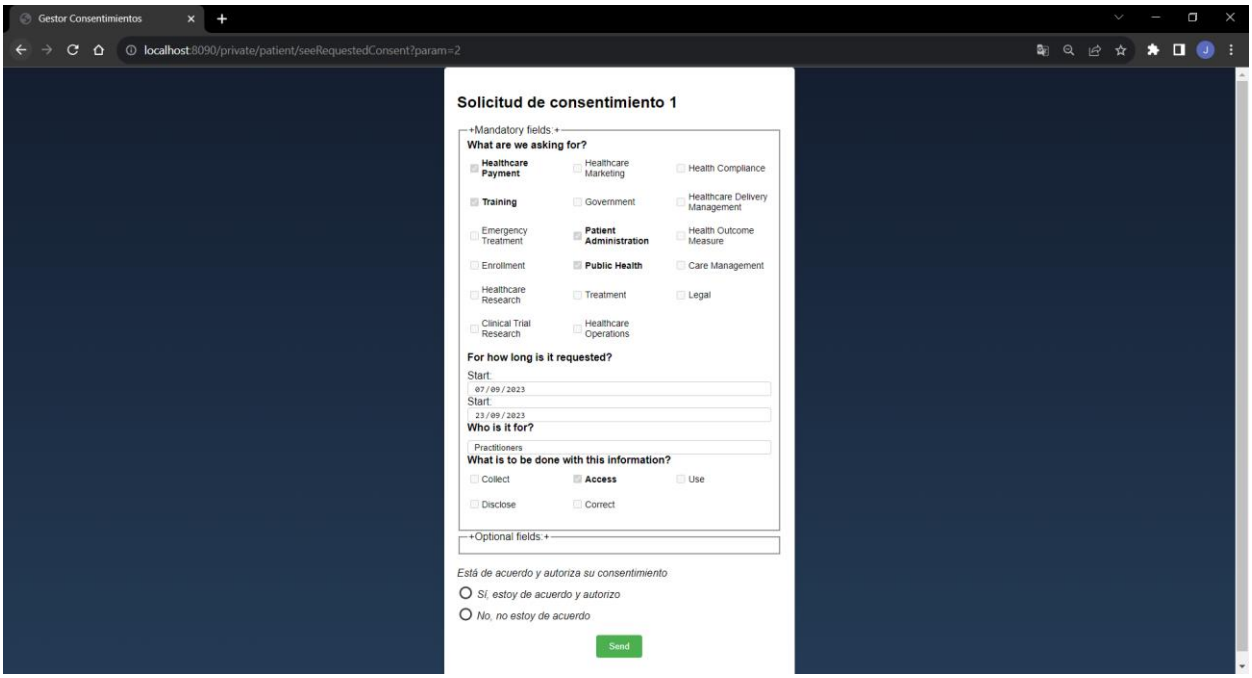

Figura 49. Vista inicial para el CU-007

4 CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo, se llevará a cabo un ejercicio de síntesis y conclusión que abarcará todos los aspectos tratados hasta el momento, brindando así una recapitulación integral de nuestra labor hasta la fecha. De igual manera, en este contexto, se indicarán una serie de consideraciones prospectivas que podrían dar lugar a la elaboración de un sistema más completo y refinado en el futuro.

4.1 Conclusiones

Se ha logrado el desarrollo de una aplicación de gestión de procesos empresariales altamente especializada en la gestión de solicitudes de consentimiento de los pacientes para el uso de su información de salud. Los procesos abordados incluyen: (1) la emisión de solicitudes de consentimiento, por parte del personal sanitario debidamente autorizado para utilizar datos sanitarios de pacientes y (2) la recepción y procesamiento de las respuestas de confirmación o denegación de consentimiento por parte de esos pacientes.

Para llevar a cabo este desarrollo, se ha utilizado un marco tecnológico compuesto por las tecnologías jBPM y Spring. Es importante destacar que la aplicación se ha diseñado considerando que los sistemas backend emplean el estándar FHIR (Fast Healthcare Interoperability Resources). Esto permite que la aplicación desarrollada se integre fácilmente en entornos ya implementados que sigan este estándar, garantizando así una interoperabilidad efectiva.

El uso de jBPM como parte de la infraestructura tecnológica brinda una ventaja significativa al permitir la optimización de los procesos desarrollados. Esto se logra mediante el análisis de los datos históricos de ejecución de instancias de procesos, lo que facilita la identificación de áreas de mejora y la toma de decisiones basada en datos con el objetivo de optimizar la eficiencia y la calidad de la gestión de consentimientos de los pacientes en el ámbito de la salud.

La realización de este proyecto ha representado una oportunidad única para enriquecer los conocimientos adquiridos a lo largo de mi carrera. Con un enfoque especializado, he realizado un análisis profundo del framework de gestión de procesos jBPM y el estándar HL7 FHIR.

jBPM es un framework que facilita la automatización de procesos empresariales y la optimización de flujos de trabajo en diversos sectores industriales. La profundidad de mi investigación y la aplicación práctica de jBPM me han permitido desarrollar una comprensión más sólida de su funcionamiento interno, sus ventajas y desafíos, y su relevancia en el panorama empresarial actual. Este conocimiento no solo me ha dotado de una habilidad valiosa, sino que también me ha permitido apreciar cómo jBPM puede contribuir significativamente a la eficiencia operativa y la toma de decisiones en las organizaciones.
Además, el proyecto me ha puesto en contacto con el estándar HL7 FHIR. Este estándar juega un papel esencial en la promoción de la interoperabilidad y el intercambio de datos de salud en el ámbito de la atención médica moderna. Mi investigación exhaustiva sobre HL7 FHIR me ha permitido explorar cómo esta tecnología está transformando la manera en que se almacenan, acceden y comparten los registros médicos y los datos de los pacientes.

Estas tecnologías son intrincadas y demandan un esfuerzo considerable debido a la complejidad de los conceptos involucrados.

Además, este proyecto me ha permitido profundizar aún más en Spring, el marco de desarrollo empleado en la creación de la aplicación web. Esta experiencia me ha brindado la oportunidad de fortalecer mis habilidades en Spring y aplicarlas de manera efectiva en un entorno práctico y real. Este perfeccionamiento en mi competencia en Spring es altamente valioso y será aplicable en futuros proyectos de desarrollo web.

En resumen, la ejecución de este proyecto no solo me ha permitido ampliar mis conocimientos académicos, sino que también enriqueció mi comprensión de tecnologías fundamentales como jBPM, HL7 FHIR y Spring.

4.2 Líneas futuras

Este proyecto es el punto de partida para desarrollar una plataforma de gestión de procesos asistenciales utilizando la herramienta jBPM. Aunque se han logrado los objetivos iniciales, aún existen muchas oportunidades de mejora tanto en la gestión de procesos como en la aplicación web, con el fin de hacerla más completa y segura.

A continuación, se presentan algunas líneas de continuación y mejoras que deben ser consideradas:

- Mejora de la autenticación: Reforzar la seguridad del sistema mediante la implementación de una autenticación más robusta. Esto puede incluir la incorporación de autenticación de dos factores o la utilización de protocolos de autenticación más seguros.
- Utilizar más recursos FHIR: Ampliar el conjunto de recursos FHIR utilizados en el proyecto para garantizar una mayor interoperabilidad y compatibilidad con otras aplicaciones de salud.
- Utilizar la persistencia ofrecida por jBPM: Aprovechar las capacidades de persistencia de jBPM para almacenar y gestionar de manera eficiente los datos relacionados con los procesos. Esto permitirá un seguimiento más completo y una gestión más efectiva.
- Añadir nuevos procesos: Identificar las necesidades del usuario y las oportunidades de mejora en el sistema para añadir procesos.
- Incorporación de herramientas de análisis de datos para la mejora continua de los procesos.
- Mejora en la selección de pacientes para las solicitudes de consentimiento: Implementar una forma más sofisticada de seleccionar a los pacientes a los que se dirigirán las solicitudes de consentimiento.

REFERENCIAS

[1] Modelo incremental. (s. f.). http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

[2] BPM: Business Process Management. (s. f.-b). Google Books. https://books.google.es/books?hl=es&lr=&id=Dm4-MGAy5vMC&oi=fnd&pg=PR1&dq=BPM&ots=zXmMIc_q5G&sig=0N-dvzUtS0H4mR3n-ksjB3yBPW4#v=onepage&q=BPM&f=false

[3] jBPM Documentation. https://docs.jbpm.org/7.73.0.Final/jbpm-docs/html_single/

[4] Definición concepto de tarea humana en jBPM. https://docs.jbpm.org/7.73.0.Final/jbpm-docs/html_single/#_jbpmtaskservice

[5] Index - FHIR v5.0.0. https://www.hl7.org/fhir/

[6] Datatypes - FHIR v5.0.0. https://www.hl7.org/fhir/datatypes.html

[7] Resource - FHIR v5.0.0. https://www.hl7.org/fhir/resource.html

[8] Questionnaire - FHIR v5.0.0. https://www.hl7.org/fhir/questionnaire.html

[9] QuestionnaireResponse - FHIR v5.0.0. https://www.hl7.org/fhir/questionnaireresponse.html

[10] JBPM Business Automation Toolkit. (s. f.). jBPM. https://www.jbpm.org/

[11] Maven – Introduction. (s. f.). https://maven.apache.org/what-is-maven.html

[12] . Maven Repository. MVNrepository. https://mvnrepository.com/

[13] Introduction - HAPI FHIR Documentation. (s. f.). https://hapifhir.io/hapi-fhir/docs/getting_started/introduction.html

[14] HAPI FHIR. (s. f.). https://hapi.fhir.org/

[15] JSON. (s. f.). https://www.json.org/json-es.html

[16] Josgarlin. (s. f.). GitHub - josgarlin/GestorConsentimientos. GitHub. https://github.com/josgarlin/GestorConsentimientos

[17] Spring boot. (s. f.). Spring Boot. https://spring.io/projects/spring-boot

[18] Spring Framework. (s. f.). Spring Framework. https://spring.io/projects/spring-framework#overview

[19] Spring Security :: Spring Security. (s. f.). https://docs.spring.io/spring-security/reference/index.html

[20] Before you continue to YouTube. (s. f.). https://www.youtube.com/@kietutorials/videos

[21] Javax.Persistence (Java EE 6). (2011, 10 febrero). https://docs.oracle.com/javaee/6/api/javax/persistence/package-summary.html

[22] Spring Data JPA. (s. f.). Spring Data JPA. https://spring.io/projects/spring-data-jpa

[23] JPARepository (Spring Data JPA Parent 3.1.3 API). (s. f.). https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html

[24] Turnquist, O. G. T. D. C. S. M. P. J. B. G. (s. f.). Spring Data JPA - reference documentation. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#dependencies

[25] Org.springframework.Stereotype (Spring Framework 6.0.11 API). (s. f.-b). https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/package-summary.html

[26] Org.springframework.web.bind.annotation (Spring Framework 6.0.11 API). (s. f.). https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/package-summary.html

[27] Org.springframework.scheduling.annotation (Spring Framework 6.0.11 API). (s. f.). https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/scheduling/annotation/package-summary.html

[28] Baeldung, & Baeldung. (2022). Scheduling in spring with Quartz | Baeldung. Baeldung. https://www.baeldung.com/spring-quartz-schedule

[29] Thymeleaf. (s. f.). https://www.thymeleaf.org/