Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación web para la gestión de tareas sanitarias en procesos BPMN

Autor: Marco Antonio Maldonado Orozco

Tutora: Dra. Isabel Román Martínez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

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

Aplicación web para la gestión de tareas sanitarias en procesos BPMN

Autor:
Marco Antonio Maldonado Orozco

Tutora:
Dra. Isabel Román Martínez
Profesora colaboradora

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Aplicación web para la gestión de tareas sanitarias en procesos BPMN

Autor: Marco Antonio Maldonado Orozco

Tutora: Dra. Isabel Román Martínez

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

En primer lugar, agradecer a mi tutora, Isabel Román Martínez, por brindarme la oportunidad de trabajar con ella, por su ayuda, consejos y el tiempo que ha dedicado a guiarme en este proyecto.

En segundo lugar, a mi familia, especialmente a mis padres, por estar siempre a mi lado, por su constante apoyo y por creer en mí incluso cuando yo mismo dudaba. Gracias por todo el sacrificio y esfuerzo que han hecho para que pudiera alcanzar mis metas

Finalmente, quiero agradecer a mis compañeros, quienes han sido parte fundamental en este camino. En especial, a aquellos que con su apoyo hicieron los momentos difíciles más llevaderos y los momentos felices aún más especiales.

Marco Antonio Maldonado Orozco.

Resumen

El objetivo principal de este proyecto es crear un servicio de gestión de tareas sanitarias para que sea conforme al paradigma de gestión de procesos de negocio (BPM, Business Process Management). Este será el servicio de partida para futuros TFGs que amplíen opciones y funcionalidades desarrolladas y por desarrollar en este TFG.

La solución ha sido diseñada en el entorno jBPM y Spring usando las librerías de ambas tecnologías, además se ha usado Thymeleaf para la gestión de las plantillas de la aplicación, así como la librería HAPI-FHIR para tratar tareas del ámbito sanitario.

Abstract

The main objective of this project is to create a healthcare task management service that adheres to the Business Process Management (BPM) paradigm. This service will serve as a foundational platform for future undergraduate projects (TFGs) that will extend the options and functionalities developed in this project and introduce new ones.

The solution has been designed using the jBPM and Spring environments, utilizing the libraries of both technologies. Additionally, Thymeleaf has been employed for template management within the application, and the FHIR library has been used to handle healthcare-related tasks.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

La motivación de este trabajo de fin de grado es crear un servicio para la gestión de las instancias de tareas humanas definidas en modelos de procesos aplicados al entorno sanitario. La solución permitirá la asignación y ejecución de las tareas a través de la presentación de cuestionarios y la gestión de sus respuestas. La aplicación es independiente de los procesos concretos, de modo que permite gestionar las tareas humanas de instancias de distintos procesos. Además, se vinculan las tareas humanas del proceso con recursos FHIR [1] (`Task`, `Questionnaire` y `QuestionnaireResponse`), facilitando la integración de la solución y el seguimiento y análisis de las tareas humanas desarrolladas en la organización sanitaria.

1.1 Motivación

El desarrollo de soluciones para la gestión de tareas humanas es esencial para mejorar la eficiencia y la calidad del sistema de información. Este proyecto tiene como objetivo principal crear un servicio de gestión de tareas del personal sanitario siguiendo el paradigma de la Gestión de Procesos de Negocio (BPM). La motivación detrás de este proyecto se puede desglosar en los siguientes puntos clave:

- **Automatización de la gestión de tareas:** Proporcionar una solución que permita a los trabajadores de un servicio clínico gestionar sus tareas, considerando tanto las necesidades del servicio clínico como los requisitos específicos de cada profesional.
- **Visualización eficiente de tareas:** Facilitar a los trabajadores una visualización clara y organizada de sus tareas, permitiéndoles priorizar y gestionar su carga de trabajo de manera eficiente y mostrándoles la información necesaria para poder hacerlo.
- **Cambios automatizados en el estado de las tareas:** Implementar un sistema en el que los cambios de estado de las tareas se realicen de forma automatizada y ofrecer una interfaz amigable, permitiendo a los profesionales reclamar tareas potenciales y también gestionar tareas asignadas de manera sencilla.
- **Representación automatizada de cuestionarios:** Desarrollar una solución que permita la representación automatizada de cuestionarios FHIR [2] ,utilizando una plantilla genérica adaptable a cualquier tipo de cuestionario, para recoger la información que se espera que el profesional introduzca al cierre de la tarea.

La solución se ha diseñado explícitamente teniendo en cuenta el paradigma de Gestión de Procesos de Negocio (BPM), que propone el uso de un marco de trabajo que facilite tanto el diseño de procesos como su ejecución y el seguimiento y análisis de dichas ejecuciones, con el objetivo de mejorar los procesos de forma continua. De este modo, podemos optimizar y adaptar los procesos a las necesidades presentes, futuras o cambiantes del cliente.

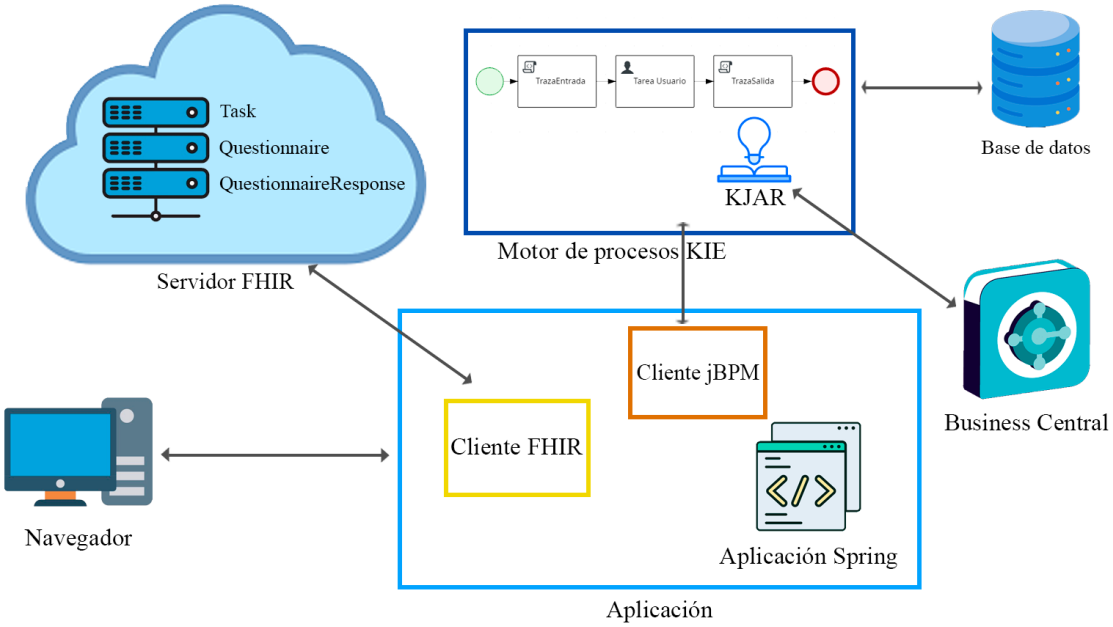

Ilustración 1. Estructura del servicio

1.2 Alcance del Proyecto

La solución implica el desarrollo de la arquitectura presentada en la ilustración 1. Los componentes principales son: una aplicación Spring, el motor de procesos KIE [3] y el servidor FHIR [1], para el manejo de recursos como tareas y cuestionarios que son necesarias para el desempeño de la aplicación.

Para lograr este objetivo se han tomado varias decisiones estratégicas:

- **Creación de una aplicación independiente de los procesos:** La aplicación se ha desarrollado de tal manera que no necesita una entrada predeterminada ni está sujeta a ningún proceso concreto, es una aplicación que sirve para cualquier proceso. Siempre que se sigan las convenciones que se especifican en el siguiente punto.
- **Conexión con los datos de la organización sanitaria:** Las instancias concretas de las tareas humanas de los procesos se vinculan a un recurso `Task` de FHIR [4], añadiendo como parámetro de entrada a la tarea el id de este recurso `Task`. A su vez cada recurso `Task` de FHIR está vinculado a un recurso `Questionnaire` de FHIR [2], que define los datos de cierre de la tarea, que deben ser introducidos por el profesional que la ejecute. Dentro del `input` del recurso `Task` hay una entrada cuyo nombre es `ClosingQuestionnaire` que tiene como valor el id del recurso `Questionnaire` correspondiente. A su vez, cuando la tarea se finalice se generará un recurso `QuestionnaireResponse` de FHIR [5] que se añadirá al recurso `Task` dentro del campo `output`. Estas relaciones facilitan la integración con la información de la organización sanitaria.
- **Uso del marco de trabajo KIE:** La suite de productos de KIE [6], que incluye jBPM, se ha seleccionado por su integración sencilla con soluciones Spring. Este marco de trabajo proporciona las herramientas necesarias para gestionar procesos de negocio y tareas de forma eficiente.
- **Foco en la gestión básica de tareas humanas:** El proyecto se ha centrado en la gestión básica de tareas humanas, esto incluye la reclamación de tareas potencialmente asignables por parte del usuario, así como las acciones de comenzar, continuar y rechazar para tareas asignadas, dejando fuera del alcance otras funcionalidades adicionales que podrán ser ampliadas en el futuro.

1.3 Descripción de los siguientes capítulos

El resto del documento estará dividido en los siguientes capítulos:

- 2. Estado de la tecnología: Explicación de las tecnologías usadas en este proyecto.
- 3. Trabajo realizado: Se describirá la solución desarrollada.
- 4. Conclusiones y líneas futuras: Se resumirán los capítulos anteriores y se dará un vistazo de los futuros pasos que se tendrán que dar para continuar con el proyecto.

2. ESTADO DE LA TECNOLOGÍA

En este capítulo comentaremos los métodos y las tecnologías usadas para poder diseñar, desarrollar e implementar este proyecto.

2.1 Java

Es un lenguaje de programación de alto nivel, orientado a objetos, que se utiliza ampliamente en el desarrollo de software gracias a su capacidad de ejecutarse en cualquier plataforma. [7]

En este proyecto usaremos la versión de Java 8, cuya documentación puede encontrarse en [8].

2.2 Paradigma BOM

Browser Object Model [9], es la forma en que los navegadores web manipulan y representan los elementos de una página web a través de un conjunto de objetos que se pueden acceder mediante JavaScript. Su enfoque radica en la interacción y manipulación de elementos en el navegador para crear aplicaciones web dinámicas e interactivas.

El proyecto se centra en priorizar la representación en el servidor usando plantillas con Thymeleaf [10], minimizando el JavaScript necesario y disminuyendo el uso del BOM lo máximo posible.

2.3 Thymeleaf

Thymeleaf [10] es un motor de plantillas moderno y flexible para Java, diseñado para trabajar tanto en entornos web como no web. Fue creado para permitir a los desarrolladores de aplicaciones Java renderizar vistas de una manera más sencilla y eficiente, con soporte para diversas tecnologías de vista y plantillas.

Las principales características de Thymeleaf son: sintaxis natural, integración con Spring, capacidad de procesamiento en cliente y servidor, motor de plantillas basado en XML/HTML, dialectos personalizados e internacionalización. Se prioriza frente a JavaScript para minimizar la ejecución en el navegador.

2.4 FHIR

FHIR [1] (Fast Healthcare Interoperability Resources) es un conjunto de reglas y especificaciones para el intercambio de información de atención médica. Está diseñado para ser flexible y adaptable, facilitando la interoperabilidad entre diferentes sistemas de salud, permitiéndoles comunicarse y compartir datos de manera segura y eficiente.

En este proyecto vamos a usar principalmente 3 recursos FHIR, estos son:

- **Task:** [4] Proporciona una forma de gestionar y rastrear la ejecución de diversas actividades o flujos de trabajo dentro del ámbito sanitario. Las tareas representan acciones individuales a realizar, como tareas administrativas, procedimientos clínicos o incluso flujos de trabajo complejos que involucran múltiples pasos.
- **Questionnaire:** [2] Se utiliza para definir un conjunto de preguntas destinadas a recopilar información introducida por individuos u organizaciones. Incluye la estructura y el contenido del cuestionario, especificando los tipos de preguntas y las posibles respuestas.
- **QuestionnaireResponse:** [5] Representa los datos recopilados al completar un cuestionario. Captura las respuestas dadas a cada pregunta del recurso `Questionnaire` correspondiente.

2.5 BPM

La gestión de procesos de negocio (Business Process Management, BPM) es un método para diseñar, ejecutar, analizar y mejorar de forma continuada cada proceso de negocio de una organización para orientarlos a objetivos concretos. [11]

Esta metodología sirve de punto de unión entre:

- Analistas de negocio: Se pueden definir de forma clara y concisa los procesos de trabajo de la organización, podrán analizar y mejorar de forma continuada los procesos definidos y desarrollados.
- Desarrolladores: Es posible desarrollar una solución a los procesos de trabajo a partir de la definición clara y concisa de los analistas.

Gracias a esta metodología se podrán diseñar mejores soluciones a los requisitos de negocio de las organizaciones. [12]

Además, la lógica de negocio no está codificada, se usa la filosofía low-code [13] para que el diseñador defina la lógica de negocio sin tener que codificar y de esta forma sea fácilmente entendible por todos los implicados.

2.5.1 Low-code

El low-code [13] es un enfoque del desarrollo de software que agiliza el desarrollo de aplicaciones usando la cantidad mínima posible de código. Las plataformas low-code proporcionan un entorno de desarrollo visual que permite generar aplicaciones de manera rápida y eficiente utilizando herramientas gráficas y configuraciones en lugar de codificación tradicional.

Esta filosofía es algo que BPM busca para poder definir toda la lógica de los procesos de forma gráfica.

2.5.2 Procesos de negocio

Un proceso de negocio [14] es el conjunto de actividades desarrolladas en el seno de una organización para alcanzar un objetivo específico. Reflejan la forma de trabajar en la organización, tareas realizadas, entradas y salidas, relaciones, dependencias, etc.

Gracias a BPM podemos evaluar los procesos existentes para encontrar formas de mejorar su eficiencia y reducir errores y costes.

2.5.3 Activos de negocio

Los activos de negocio [3] son los distintos elementos y componentes que forman parte de la definición, implementación y ejecución de los procesos de negocio. Estos activos sirven para modelar, gestionar y automatizar los procesos de negocio dentro de una organización.

Algunos ejemplos de activo de negocio son:

- **Modelos de procesos:** Representaciones gráficas de los flujos de trabajo de los procesos de negocio. Estos diagramas definen las actividades, eventos, puertas de decisión y otros elementos que componen un proceso. Estos modelos incluyen definiciones de tareas humanas, que se definen como descripciones de las tareas que requieren intervención humana, incluyendo asignaciones, plazos, formularios asociados y reglas de notificación. Estas son las tareas en las que centramos este proyecto.
- **Formularios y plantillas:** Elementos utilizados para la entrada de datos por parte de los usuarios, como formularios web, plantillas de correo electrónico, etc.
- **Variables y datos de proceso:** Información específica del proceso que se utiliza para controlar su flujo y comportamiento.

2.6 Aplicación de negocio

Una aplicación de negocio es una solución que facilita la ejecución de los procesos de interés y que da soporte tanto a la ejecución de las tareas (automáticas o manuales) como al seguimiento y análisis de dichos procesos. En lugar de codificar la lógica de los procesos en la aplicación, se expresa a través de modelos, lo que facilita su análisis, diseño y evolución. En este proyecto, hemos utilizado las herramientas proporcionadas por KIE [15] para lograr estos objetivos.

Una aplicación de negocio se divide en tres componentes fundamentales:

- **Tipos de datos:** Representa las entidades del negocio.
- **Modelo de procesos:** Representa los flujos de trabajo de los procesos de negocio.
- **Aplicación:** Se programan las tareas individuales y se gestionan las tareas humanas.

Las principales virtudes de una aplicación de negocio son:

- **Reutilización:** Se pueden integrar componentes definidos en distintas aplicaciones de negocio.
- **Evolución independiente:** Se puede modificar cada componente por separado.
- **Adaptabilidad:** Los modelos del proceso pueden cambiar y, gracias a herramientas como KIE y jBPM, se pueden realizar estos cambios de forma sencilla para adaptarse a las necesidades del negocio
- **Seguimiento del proceso:** una aplicación de negocio permite hacer un seguimiento del proceso para su mejora continua.

Los modelos de procesos por sí solos no son suficiente, necesitan un motor de procesos que interprete estos modelos, en este proyecto se ha usado la tecnología proporcionada por KIE para este propósito

2.7 KIE

KIE (Knowledge Is Everything) [6] es una plataforma de código abierto que proporciona un conjunto de herramientas y componentes para la gestión de modelos (conocimiento) y la automatización de decisiones en entornos empresariales. Sirve como ecosistema para distintos proyectos de código abierto enfocados en la automatización de procesos de negocio. [16]

Algunos proyectos dentro de este ecosistema son Kogito, Drools y jBPM.

2.7.1 jBPM

jBPM (Java Business Process Management) [12] es un conjunto de herramientas de código abierto que sirve para construir aplicaciones de negocio y facilitar el control de los flujos de procesos de negocio

jBPM se puede usar como un servicio independiente o dentro de un servicio personalizado y es posible su uso con varios frameworks, nosotros lo usaremos con el framework de SpringBoot.

Los procesos de negocio se pueden representar en modelos que muestran el detalle de las tareas a realizar para conseguir un objetivo a partir de unos datos de entrada, además de generar unos datos de salida.

Entre todas las funcionalidades que nos provee jBPM, se incluye el motor de procesos, que interpreta modelos de procesos.

Los procesos se pueden representar de forma estándar con BPMN [17], una notación normalizada. jBPM incluye estas definiciones y otros activos de empresa en un contenedor que denomina kjar.

Para implementar la lógica empresarial, jBPM facilita la definición de distintos tipos de archivos, como; procesos comerciales, reglas comerciales y restricciones de planificación, e incluye soporte de persistencia, mensajería, transacciones, etc.

2.7.1.1 Kjar

Kjar en jBPM [18] es un archivo JAR que contiene todos los activos de negocio necesarios para ejecutar procesos de negocio en la plataforma KIE. Estos activos incluyen definiciones de procesos, reglas de negocio, formularios de usuarios y otros recursos relacionados con la ejecución de procesos en el entorno jBPM.

Tener todo esto en un mismo archivo JAR facilita su distribución, implementación y gestión.

2.7.1.2 Definición de un proceso en jBPM

En jBPM, un proceso [19] se define como una serie de tareas conectadas por flujos de secuencia que representan un flujo de ejecución. Estas tareas pueden incluir actividades humanas, servicios automatizados, decisiones y subprocesos, entre otros elementos. Los procesos se modelan gráficamente y se describen utilizando lenguajes específicos como BPMN [20] (Business Process Model and Notation).

2.7.1.3 Business Central

Business Central [21] es lo que se conoce como un `WorkBench`, que nos ofrece distintas herramientas para trabajar con jBPM, por ejemplo, nos permite editar y ejecutar los procesos para poder probar si el diseño es correcto. Las funcionalidades que permite son:

- **Creación de proyectos y páginas:** Donde los usuarios podrán diseñar procesos de negocio y otros activos de negocio.
- **Control de los servidores:** Permite controlar los servidores donde se despliegan las instancias de los procesos y desplegar los procesos.
- **Gestión de acceso a los procesos:** Permite gestionar la asignación de roles y permisos a los usuarios para asegurar que solo las personas autorizadas puedan acceder a ciertos procesos y datos.
- **Seguimiento de tareas asignadas:** Nos proporciona la opción de seguir las tareas que tienen asignadas los usuarios.

Business Central soporta la filosofía low-code [13]. Permite a los usuarios modelar procesos, reglas de negocio y formularios a través de interfaces gráficas intuitivas, reduciendo la necesidad de codificación manual.

2.7.1.4 Instancia de proceso en jBPM

La instancia de proceso [19] en jBPM es la ejecución de un proceso. Cuando se crea una instancia de un proceso, se crea un token para marcar el camino de la ejecución. Este token se llama token raíz de la instancia de proceso y se posiciona en el nodo inicial de la definición del proceso. Un proceso puede instanciarse más de una vez.

2.8 SpringBoot

SpringBoot [22] es un framework de código abierto para el desarrollo de aplicaciones Java. Está diseñado para simplificar el proceso de creación y despliegue de aplicaciones basadas en Spring.

La principal ventaja de Spring es su flexibilidad, facilitando la integración con otros frameworks, como se puede observar en este proyecto en el que integramos tanto el framework de Spring, que nos ayuda con la configuración y la estructura, como el framework de jBPM. De esta forma podemos realizar una aplicación Spring que permita trabajar con la filosofía de BPM gestionando los procesos y las tareas de esos procesos.

2.8.1 Starter de Spring para jBPM

Los Starter de Spring son módulos preconfigurados que facilitan la integración de cualquier tecnología, por ejemplo, jBPM, en una aplicación basada en Spring Boot. Estos starters incluyen todas las dependencias necesarias y la configuración básica para integrarlas en la aplicación Spring.

Spring Initializr [23] genera la estructura del proyecto y añade las dependencias automáticamente, así los desarrolladores pueden comenzar rápidamente con la tecnología elegida dentro de una aplicación Spring.

La ventaja de usar el starter jBPM en Spring Boot es que podemos inyectar las dependencias de jBPM en nuestra aplicación y usar sus librerías de forma sencilla gracias a las etiquetas de Spring `@Injection` y `@Autowire`, además de poder usar todas las ventajas propias de Spring Boot como framework. [24]

2.9 Maven

Maven [25] es una herramienta de automatización de compilación que se utiliza principalmente para proyectos Java. Los objetivos de Maven son:

- Facilitar la construcción de proyectos y la gestión de dependencias.
- Proporcionar un sistema uniforme de construcción de proyecto.
- Proporcionar información de calidad del proyecto.
- Fomentar el desarrollo de buenas prácticas.

Maven es una tecnología fundamental que da soporte a toda la idea de aplicación de negocio, ya que se busca la mayor independencia posible entre las partes y la reutilización de cada parte.

Cada parte de la aplicación de negocio comentada anteriormente se maneja como un artefacto de Maven, es decir, los modelos de datos, los activos de negocio y la aplicación son artefactos independientes, que luego a la hora de compilar se componen formando una única aplicación.

Toda la información relativa a las dependencias de Maven usadas en la aplicación está dentro del fichero pom.xml.

2.10 Git

Git [26] es un sistema de control de versiones distribuido y de código abierto que permite tener un registro de los cambios realizados en el código.

Git es distribuido, esto quiere decir que los desarrolladores manejan una copia local del repositorio. Gracias a esto no se depende de un servidor centralizado para confirmar los cambios, lo que hace a Git más robusto y descentralizado.

Git permite la creación de ramas independientes en el repositorio. Cada nueva funcionalidad puede desarrollarse en su propia rama, lo que permite a los desarrolladores trabajar en paralelo sin interferir en el código base principal, las modificaciones hechas en una rama no afectan al resto del proyecto hasta que se fusionan, lo que ayuda a mantener el código base estable.

En resumen, es una herramienta esencial para el desarrollo de proyectos. El proyecto está en un repositorio remoto que se aloja en la web Github: https://github.com/tfg-projects-dit-us/Healthcare-Tasks

El Business Central también usa un proyecto git para almacenar los activos, el kjar. Hemos incluido un kjar en nuestro repositorio que nos ha servido para probar la aplicación. Para clonar ese proyecto kjar en caso de hacerle algunas modificaciones hay que seguir los siguientes pasos:

1. Se va a los ajustes de nuestro proyecto:

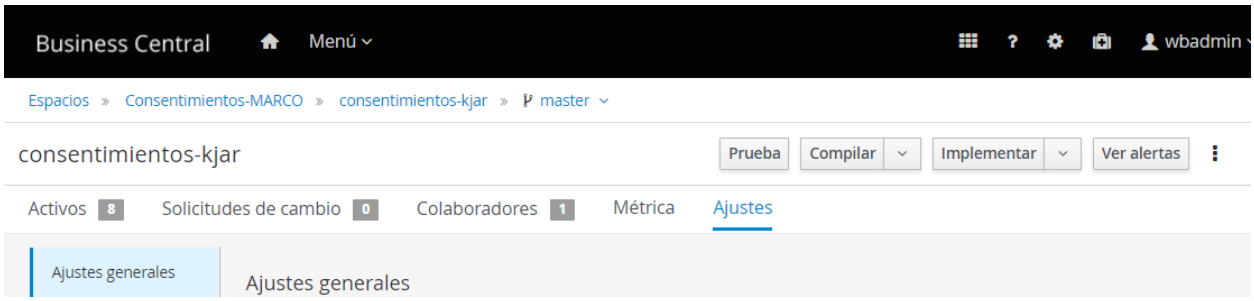

Ilustración 2. Acceso al menú ajustes de un proyecto de Business Central

2. En los ajustes generales vemos elegimos la URL de http y la copiamos:

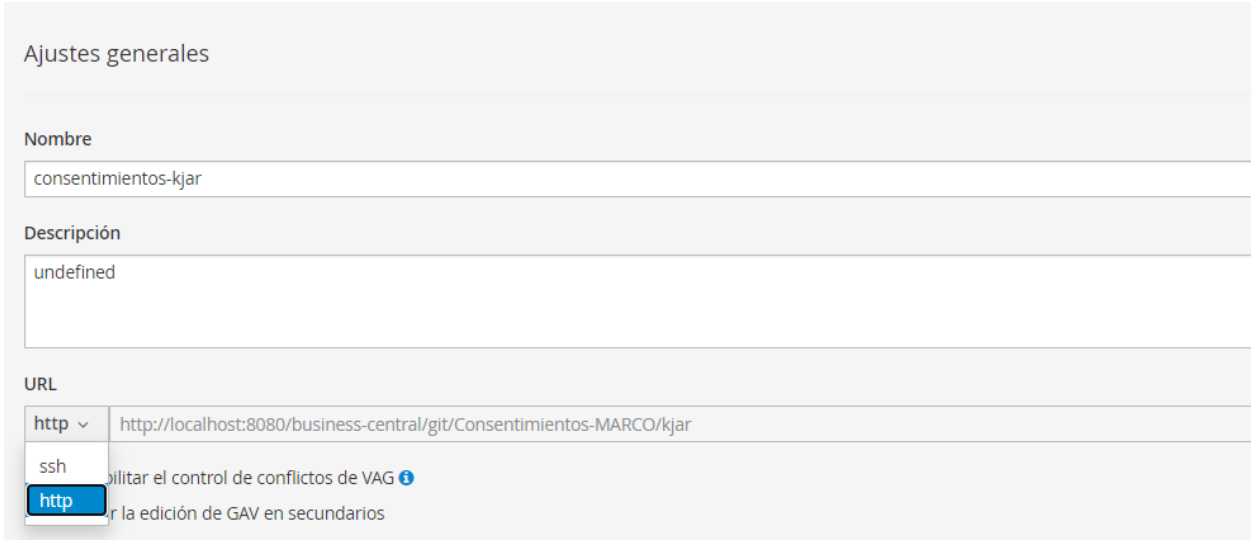

Ilustración 3. Ajustes generales de un proyecto de Business Central

3. Desde el terminal, nos vamos a la carpeta raíz del proyecto y hacemos un git clone de la URL copiada en el paso anterior:

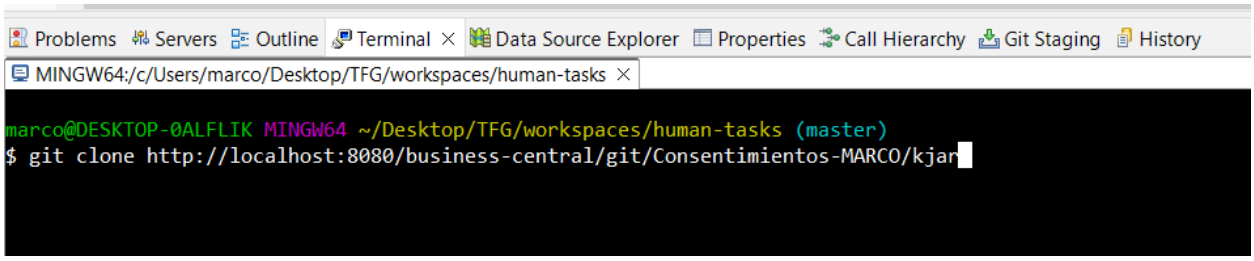

Ilustración 4. Comando git clone

Y con esto tendríamos el kjar en nuestro local.

2.10.1 Importación de un proyecto desde un repositorio remoto a Business Central

1. Seleccionamos el espacio donde queremos importar el proyecto, en caso de no tener ningún espacio lo podemos crear pulsando en “Agregar Espacio”:

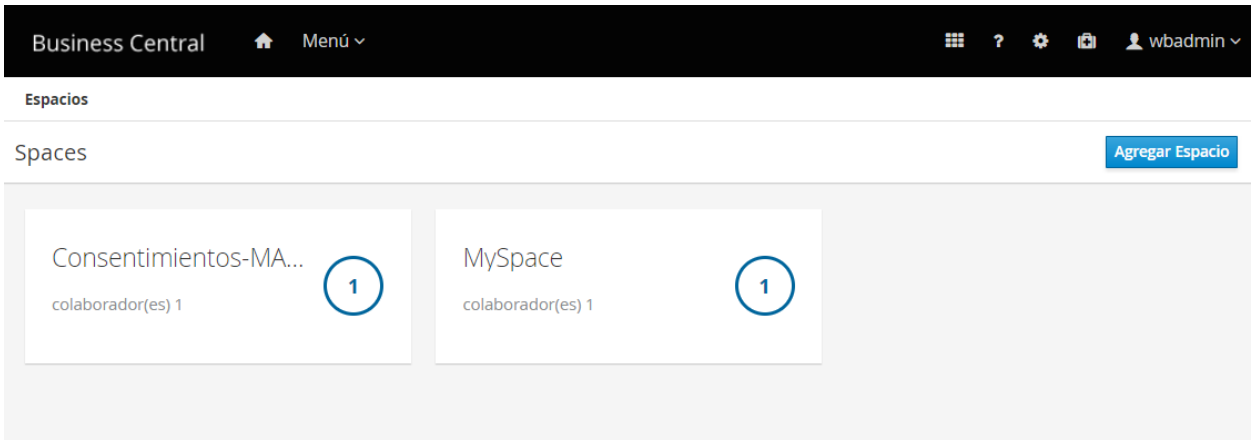

Ilustración 5. Menú selección espacios Business Central

2. Dentro del espacio seleccionado, a la derecha en el botón "Agregar proyecto" pulsamos el desplegable y clicamos en "Importar proyecto":

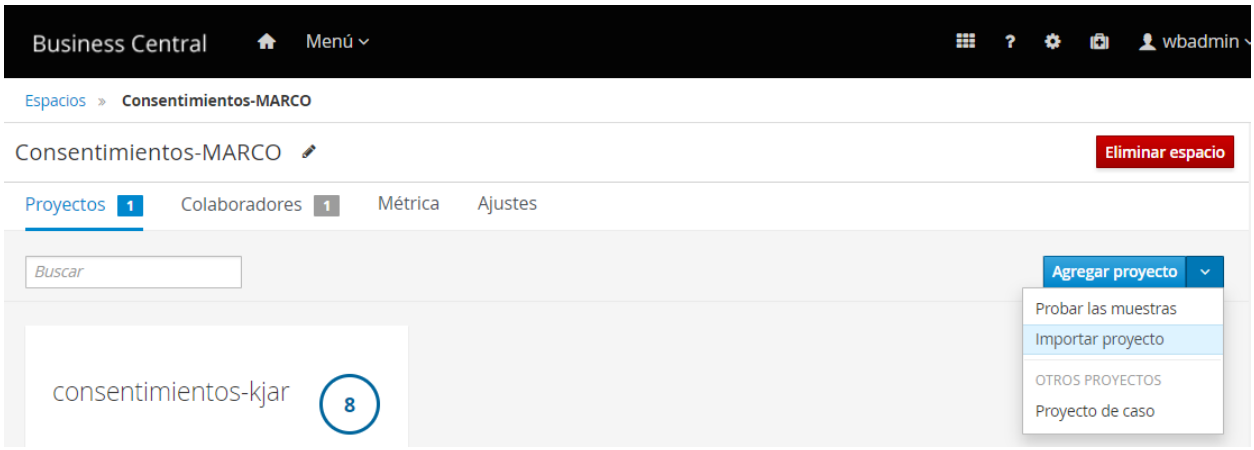

Ilustración 6. Menú proyectos Business Central

3. Añadimos la URL del proyecto que queremos importar a Business Central. Debemos saber que para que se pueda importar un proyecto desde un repositorio remoto, este repositorio **debe** tener creada la rama master, en caso contrario Business Central no podrá importar el repositorio:

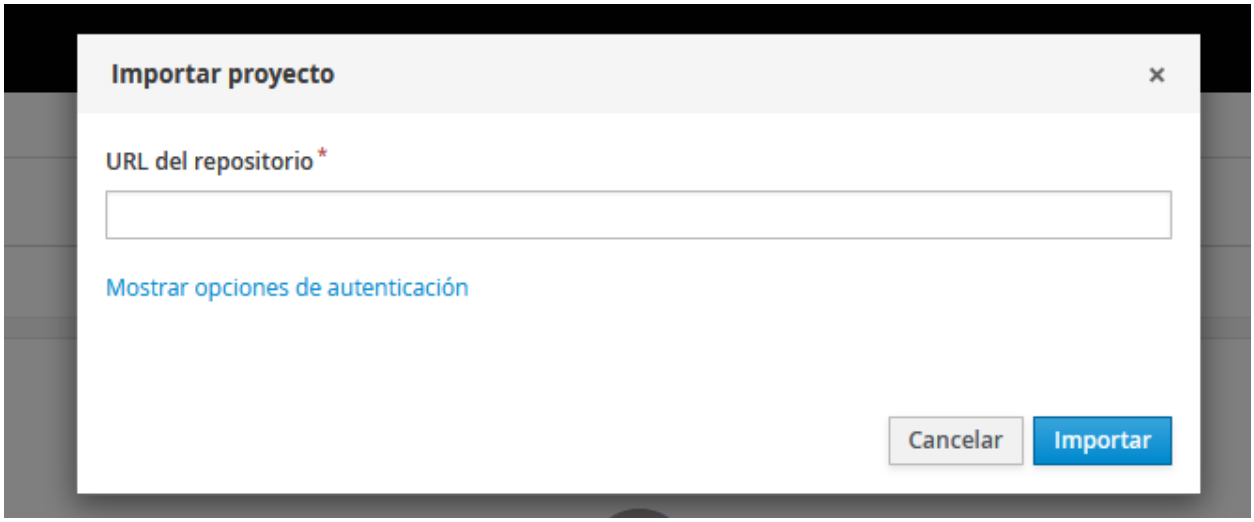

Ilustración 7. Menú introducción URL del repositorio remoto en Business Central

4. Seleccionar el kjar correspondiente y pulsar “Aceptar”:

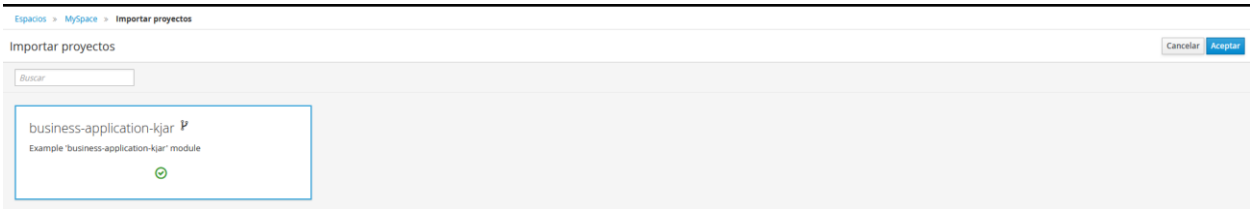

Ilustración 8. Menú importar proyectos Business Central

5. Ya estaríamos listos para trabajar con nuestros activos de negocio:

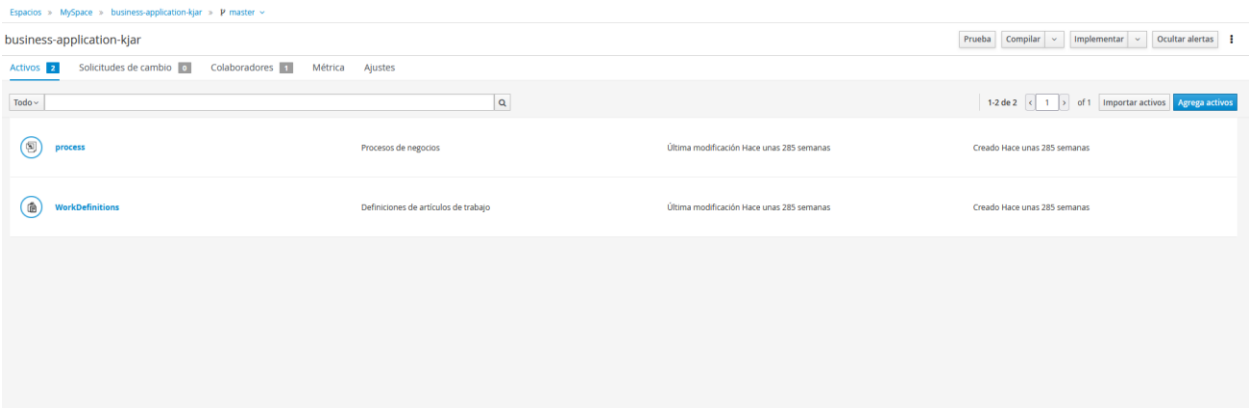

Ilustración 9. Menú principal de los activos Business Central

2.11 PostgreSQL

PostgreSQL [27] es un sistema gestor de base de datos relacional de código abierto que usa y extiende el lenguaje SQL combinado con otras características. Permite guardar de forma segura y escalar las cargas de trabajo de alto rendimiento.

Las distintas características que hacen tan atractivo a PostgreSQL son:

- **Juego de datos:** Puede guardar muchos tipos de dato distintos.
- **Integridad:** Proporciona integridad de los datos guardados.
- **Rendimiento.**
- **Fiabilidad.**
- **Seguridad.**
- **Extensibilidad.**
- **Búsquedas personalizadas:** Su búsqueda de texto permite caracteres internacionalizados.

3. TRABAJO REALIZADO

3.1 Requisitos

Desarrollo de una aplicación empresarial jBPM que permita la gestión de las tareas humanas definidas en procesos sanitarios. La información de las tareas se persistirá tanto en servidores FHIR como en el backend de persistencia del motor de procesos, según la naturaleza de la misma.

Los procesos se ejecutan en un servidor KIE embebido en la aplicación, que gestionará el estado de las tareas jBPM dentro del flujo de trabajo de una instancia de proceso y con el que se interaccionará usando la API de Java de jBPM.

La información de entrada de la tarea jBPM debe ser el id del recurso `Task` de FHIR correspondiente a la misma. Este recurso estará gestionado por un servidor FHIR externo a la aplicación.

La aplicación se responsabiliza de mantener sincronizadas las representaciones FHIR y jBPM correspondientes a la misma tarea humana del proceso y de la consistencia en los estados de ambas representaciones (ilustraciones 10 y 11).

Los datos que se espera que la persona responsable de la tarea introduzca tras la ejecución de la misma se especifican en un recurso `Questionnaire` de FHIR, cuyo id se puede encontrar en el recurso `Task` de FHIR asociado a la tarea jBPM anteriormente mencionada, dentro del campo input, de tipo `ClosingQuestionnaire`.

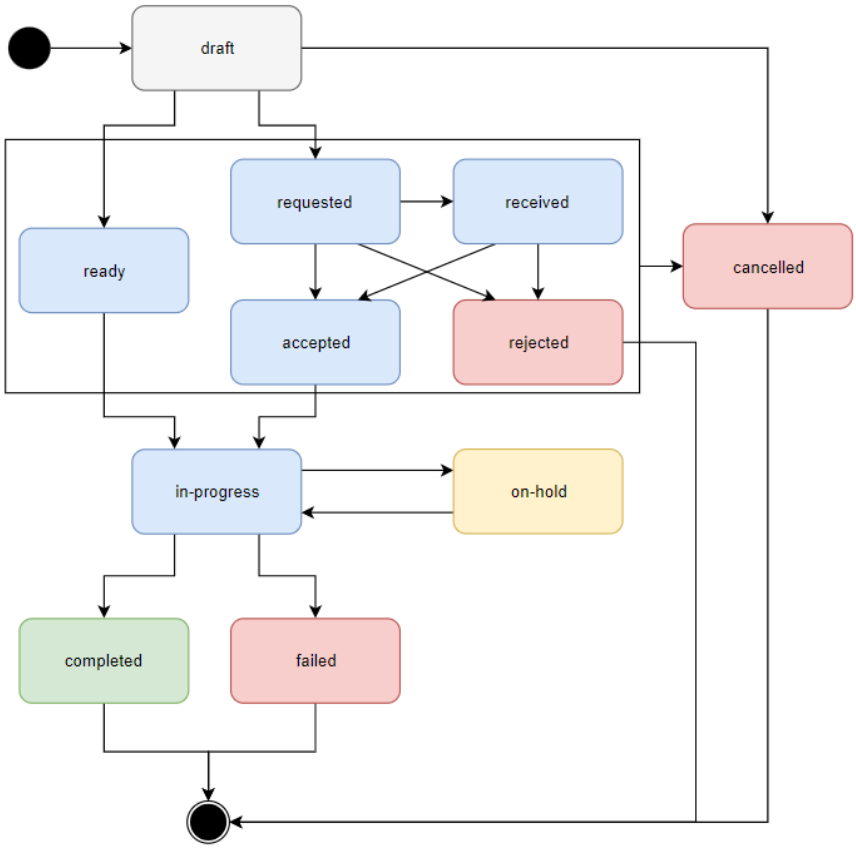

Ilustración 10. Diagrama de flujo de estados de tareas FHIR

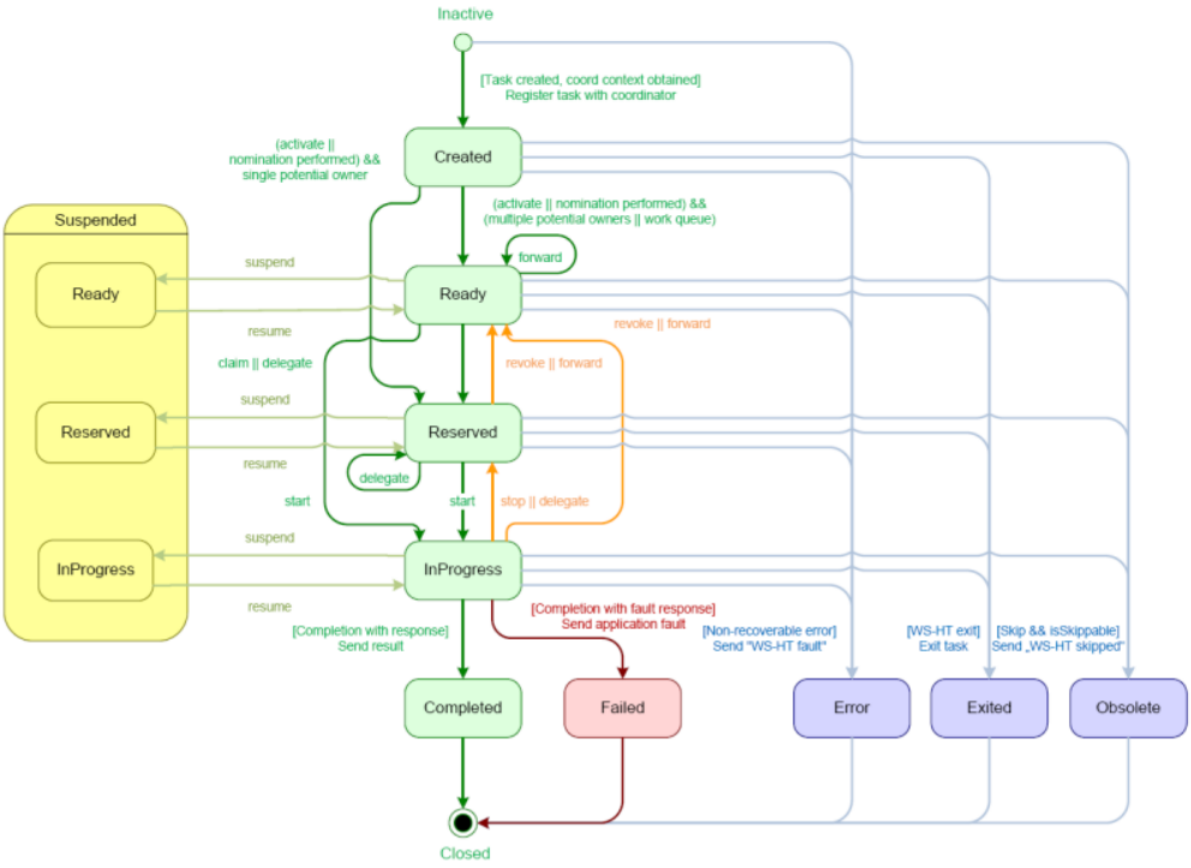

Ilustración 11. Diagrama de flujo de estados de tareas jBPM

Para explicar el alcance de la aplicación se presentan los casos de uso identificados:

CDU-01	Autenticación
Descripción	El usuario tendrá que introducir sus credenciales, usuario y contraseña para poder iniciar la aplicación.
Datos específicos	• Usuario. • Contraseña.

Tabla 1. CDU-01.

CDU-02	Consulta de tareas pendientes de asignación
Descripción	El usuario podrá ver las tareas que están pendientes de asignar y son potencialmente asignables.
Datos específicos	• Para poder ver las tareas pendientes de asignación es necesario estar autenticado.

Tabla 2. CDU-02

CDU-03	Auto asignación de una tarea
Descripción	De las tareas pendientes de asignar el usuario podrá elegir y asignarse cualquiera de las tareas presentes.
Datos específicos	• El usuario debe estar en la pantalla de consulta de tareas pendientes de asignación. • La tarea deberá estar sin usuario asignado, si aparece en la tabla implica que el usuario puede asignársela.

Tabla 3. CDU-03

CDU-04	Consulta de tareas asignadas
Descripción	El usuario podrá ver las tareas que tiene asignadas, así como información relevante, como el estado de las mismas, y realizar acciones sobre ellas.
Datos específicos	• Para poder ver las tareas asignadas el usuario debe estar autenticado.

Tabla 4. CDU-04

CDU-05	Comenzar una tarea
Descripción	El usuario podrá realizar la acción de comenzar la tarea, esto cambiará el estado de las tareas jBPM y FHIR para ponerlas en progreso, y se mostrará la pantalla del cuestionario de cierre.
Datos específicos	• El usuario debe estar en la pantalla de tareas asignadas • La tarea no se podrá comenzar si ya ha sido previamente comenzada en otro momento.

Tabla 5. CDU-05

CDU-06	Continuar una tarea
Descripción	El usuario podrá realizar la acción de continuar una tarea que previamente había comenzado. Esto mostrará la pantalla del cuestionario de cierre.
Datos específicos	• El usuario debe estar en la pantalla de tareas asignadas. • La tarea debe haber sido previamente comenzada para poder continuarla.

Tabla 6. CDU-06

CDU-07	Completar el cuestionario de cierre
Descripción	El usuario debe completar todos los campos requeridos del cuestionario de cierre, además de los opcionales en caso de que proceda, y darle a enviar. Esto finalizará las tareas jBPM y FHIR, persistiendo la respuesta en forma de recurso `QuestionnaireResponse` en el servidor FHIR y añadiendo su URI a la tarea FHIR además de finalizarla.
Datos específicos	• Se debe haber comenzado o continuado previamente una tarea • Se debe haber rellenado los campos obligatorios del cuestionario • Se debe pulsar "Enviar" una vez estén las respuestas listas.

Tabla 7. CDU-07

CDU-08	Rechazo de una tarea
Descripción	El usuario podrá rechazar una tarea que tenga asignada, esta tarea volverá al estado inicial siendo de nuevo una tarea pendiente de asignación.
Datos específicos	• El usuario debe estar en la pantalla de tareas asignadas.

Tabla 8. CDU-08

3.2 Arquitectura de la solución

En esta solución se sigue el patrón Modelo Vista Controlador (MVC) [28] y se van a presentar los componentes del mismo organizados de este modo:

3.2.1 Vista

Para la vista se utiliza un sistema de plantillas usando Thymeleaf y ciertas funcionalidades dinámicas, que explicaremos más adelante, que están realizadas en JavaScript. En Thymeleaf se pueden añadir datos al modelo para luego poder usarlos al generar la vista a partir de la plantilla. La aplicación consta de varias de estas plantillas:

- **Pantalla principal (URL: /tasks, plantilla: tasks):** Es la pantalla principal de la aplicación a la que se entra tras hacer el inicio de sesión, disponible en http://localhost:8090/tasks. Es una plantilla simple que únicamente tiene dos botones, uno para acceder a las tareas potenciales para el usuario que se autenticó y el otro para acceder a las tareas asignadas para ese usuario. En caso de haber sido redirigidos a esta pantalla tras completar una tarea, se mostrará durante 5 segundos un mensaje flotante que indica si la operación de completar la tarea se ha realizado con éxito o por el contrario ha habido algún error.

- **Pantalla de tareas potenciales (URL: /pontentialTasks, plantilla: potentialTasks):** En esta pantalla hay una tabla donde se muestran las distintas tareas jBPM [29] potencialmente asignables al usuario autenticado, es decir, las tareas jBPM que están en estado "Creada", que aún no tienen ningún usuario asignado. En esta tabla se muestra la siguiente información:
    - Proceso al que pertenece la tarea: De esta forma ayudamos al usuario a identificar la tarea a reclamar en caso de que haya dos tareas con el mismo nombre.
    - Nombre de la tarea.
    - Fecha de creación de la tarea.
    - Acciones: En esta pantalla solo podemos ejecutar la acción de reclamar la tarea. Esta acción lo único que hace es cambiar el estado de la tarea FHIR a "Solicitada" y de la tarea jBPM a "Reservada" y asignarle el usuario con el que se ha realizado la acción, por tanto, la tarea pasará de salirnos en esta pantalla a salirnos en la pantalla de tareas asignadas.
- **Pantalla de tareas asignadas (URL: /assignedTasks, plantilla: potentialTasks):** En esta pantalla hay una tabla donde se muestran las distintas tareas jBPM [29] asignadas al usuario autenticado. En esta tabla se muestra la siguiente información:
    - Proceso al que pertenece la tarea.
    - Nombre de la tarea.
    - Fecha de creación de la tarea.
    - Estado de la tarea: En esta pantalla es importante el estado en el que se encuentra la tarea ya que según el estado que sea se podrán realizar o no ciertas acciones.
    - Acciones: En esta pantalla podemos realizar 3 acciones:
        - Comenzar: Esta acción solo se podrá ejecutar si la tarea está en estado "Reservada", es decir, que el usuario se la haya asignado y aún no se haya comenzado. Al darle a comenzar se redireccionará a la URL "/tasks/start", que está manejada por el Controller `TasksController.java`, que se explicará más adelante. Esta acción cambia el estado de las tareas jBPM y FHIR a "En progreso". Además, a partir de la tarea jBPM se obtiene el id del recurso `Task` de FHIR y a partir de este, el id del recurso `Questionnaire` asociado a esa tarea. Este cuestionario se representará en la pantalla del cuestionario, donde será redirigida la aplicación, que se explica a continuación.
        - Continuar: Esta acción solo se podrá ejecutar en caso de que la tarea esté en estado comenzada. Continuar hace lo mismo que la acción de comenzar, pero sin actualizar el estado de las tareas jBPM y FHIR, ya que tienen el estado correcto.
        - Rechazar: Si se selecciona la acción de rechazar, la tarea pasará de nuevo a tarea potencial, pasando a estar en estado "Lista" y "Creada" las tareas FHIR y jBPM respectivamente y se le desasocia el usuario asignado, así otro usuario o el mismo se la podrá asignar en otro momento.
- **Pantalla del cuestionario de cierre de una tarea (URL: /questionnaire, plantilla: questionnaireForm):** Esta es la pantalla más importante de la aplicación, ya que es una plantilla genérica capaz de representar las preguntas de cualquier tipo de recurso `Questionnaire` [2] de FHIR, esta pantalla consta de un archivo principal, donde se carga el recurso `Questionnaire`. Además, para cada tipo de `Item` del `Questionnaire` hay un archivo que contiene la plantilla para representarlo. El archivo principal es el que se encarga de recorrer todos los `Item` del recurso `Questionnaire` e incluir cada plantilla de los distintos tipos cuando sea necesario.

  Respecto a los `Items` hay 3 atributos claves que debemos conocer: `LinkId`, se refiere al Id único de la pregunta dentro del cuestionario. `Required`, indica si la respuesta a esa pregunta es obligatoria para poder finalizar el cuestionario. `Repeats`: Indica si la pregunta puede tener más de una respuesta. Cada `Item` siempre va acompañado de un título donde se muestra la pregunta. Una vez conocido esto, se

describirán las plantillas correspondientes para cada tipo de `Item` del `Questionnaire`:

- **Coding:** Este es uno de los tipos más repetidos debido a su versatilidad. La plantilla de tipo `coding` permite al encuestado seleccionar una o varias opciones de una lista predefinida. La plantilla se ha creado de tal forma que, si hay 3 opciones o menos aparecerán las opciones en forma de check o radio, es decir, mostrándose las 3 opciones y el encuestado deberá seleccionar una o varias según el atributo `Repeats`. En caso de haber más de 3 opciones se mostrarán en un desplegable para la comodidad del usuario, al desplegar podrá ver todas las opciones posibles y seleccionar las que considere oportunas.
- **String:** Proporciona una línea para introducir texto donde se deberá introducir la respuesta. Si tiene atributo `Repeats` aparecerá un botón, cuya funcionalidad está hecha con JavaScript que permite añadir más cuadros de texto para que se generen las respuestas necesarias a esta pregunta.
- **Text:** Este tipo de pregunta simplemente proporciona un cuadro de texto para escribir la respuesta, es igual que el tipo `String` pero con más espacio para escribir. Si tiene atributo `Repeats` aparecerá un botón cuya funcionalidad está hecha con JavaScript que permite añadir más cuadros de texto para que se generen las respuestas necesarias a esta pregunta.
- **Boolean:** Acompañando al título aparece un check que podrá ser o no seleccionado. Este tipo de pregunta no tiene atributo `Repeats`. Respecto al enunciado que acompaña el check representa una respuesta de "Si" en caso de estar marcado o "No" en caso de no estarlo.
- **URL:** Muestra un campo de texto donde se deberá introducir una URI de una web, para que la respuesta sea válida esta debe empezar con http:// o https://. En caso de tener atributo `Repeats` aparecerá un botón cuya funcionalidad está hecha con JavaScript que permite añadir más campos para introducir URIs adicionales permitiendo generar las respuestas necesarias a esta pregunta.
- **Reference:** En este tipo de `Item` pueden venir definidas opciones para responder a esta pregunta, en caso de que vengan estas posibles respuestas se muestra un desplegable desde el que se podrá seleccionar la respuesta. En caso de que no haya opciones definidas muestra un campo de texto donde introducir una URL que deberá hacer referencia a un recurso FHIR. En caso de tener atributo `Repeats` aparecerá un botón cuya funcionalidad está hecha con JavaScript que permite añadir más campos para introducir nuevas URLs de recursos FHIR permitiendo generar las respuestas necesarias a esta pregunta
- **Integer:** Muestra un campo de tipo numérico para introducir un valor. Esta entrada tiene a la derecha unas flechas hacia arriba y hacia abajo que aumentan la cantidad introducida en 1. En caso de tener atributo `Repeats` aparecerá un botón cuya funcionalidad está hecha con JavaScript que permite añadir más campos para introducir valores numéricos adicionales permitiendo generar las respuestas necesarias a esta pregunta
- **Quantity:** Se usa la misma plantilla que para los `Items` de tipo Integer, la única diferencia es que los saltos de cantidad van de 0.1 en 0.1.
- **Decimal:** Se usa la misma plantilla que para los `Items` de tipo Integer, la única diferencia es que los saltos de cantidad van de 0.01 en 0.01.
- **Date:** Esta plantilla permite seleccionar una fecha, es un campo de texto que a la derecha tiene un icono de un calendario que al clicar se abrirá el calendario y permitirá seleccionar la fecha que se quiera responder. En caso de tener atributo `Repeats` aparecerá un botón cuya funcionalidad está hecha con JavaScript que permite añadir más campos para introducir una nueva fecha permitiendo generar las respuestas necesarias a esta pregunta
- **Datetime:** Esta plantilla es igual a la anterior, pero además de la fecha permite seleccionar la hora. Al abrir el calendario al lado de este aparecerá una lista con las distintas horas y minutos del día para seleccionar la hora deseada.

Una vez estén completas todas las preguntas obligatorias, es decir, que tengan el campo `Required`, se podrá pulsar el botón de "Enviar cuestionario". Se realiza un Post a la URL "/submit" controlada por el controlador

`FormController.java`, que es el encargado de crear el `QuestionnaireResponse` [5], y completar la tarea haciendo uso de varios servicios, cuya funcionalidad se explicará extendidamente más adelante. Cuando se realicen estas acciones se volverá a la pantalla principal mostrando el mensaje flotante correspondiente que indica si la operación se realizó con éxito (ya explicado en el apartado de la pantalla principal).

3.2.2 Controlador

Se han desarrollado los siguientes controladores:

- **`QuestionnaireController`:** Este es el controlador más sencillo, únicamente se ocupa de la URL "/questionnaire", con ayuda del servicio `QuestionnaireDAO` que se explicará más adelante, se encarga de a partir del id de la tarea FHIR [4], obtener el `Questionnaire` [2] asociado y mostrar la plantilla del cuestionario, pasando el modelo de `Questionnaire`, además del id de la tarea FHIR y el id de la tarea jBPM.
- **`FormController`:** Este controlador se ocupa solamente de la URL "/submit", como se comentaba en el apartado anterior, cuando se envía la respuesta del cuestionario. Se encarga de, a partir de las respuestas introducidas, construir correctamente el recurso `QuestionnaireResponse` [5], teniendo en cuenta el tipo de cada pregunta, si tiene una o varias respuestas, etc. Una vez tiene el `QuestionnaireResponse` construido hace uso de `FhirTasksDAO` y `TasksDAO` que son los servicios encargados de las tareas FHIR y jBPM respectivamente, para persistir el `QuestionnaireResponse` en el servidor FHIR y para completar las tareas en el servidor KIE y FHIR, además de añadirle al recurso `Task` de FHIR relacionado el URL del `QuestionnaireResponse` persistido en el campo `output`.
- **`TasksController`:** Este controlador se encarga de todo lo relacionado con tareas, tanto FHIR como jBPM, haciendo uso de los servicios `FhirTasksDAO` y `TasksDAO`. Las URLs de las que se ocupa son:
    - "/tasks": Es la URL de la página principal. No realiza ninguna acción, solo muestra la plantilla tasks, explicada en el apartado anterior.
    - "tasks/potentialTasks": El controlador se encarga de consultar todas las tareas potenciales para el usuario autenticado y las carga en el modelo para la plantilla correspondiente.
    - "tasks/assignedTasks": Al igual que el anterior, pero para las tareas asignadas para el usuario autenticado.
    - "/tasks/claim": Controla la acción de "Asignar" de la pantalla de tareas potenciales. El controlador recibe el id de la tarea jBPM que el usuario se quiere asignar, además del id del contenedor y del proceso al que pertenece la tarea y asigna al usuario la tarea jBPM correspondiente, cambiando también el estado a las tareas jBPM y FHIR a "Reservada" y "Solicitada" respectivamente. Y redirige a la URL "tasks/potentialTasks".
    - "tasks/start": Controla la acción de "Comenzar" de la pantalla de tareas asignadas. El controlador recibe el id de la tarea jBPM que el usuario quiere ejecutar, además del id del contenedor, el proceso al que pertenece la tarea y el usuario al que está asignada y lo que hace es poner las tareas jBPM y FHIR correspondientes en estado "En progreso", además, añade al modelo el id de la tarea FHIR asociada a la tarea jBPM que se ha comenzado, para que el `QuestionnaireController` pueda obtener el cuestionario asignado. Con este id de la tarea FHIR se redirecciona a la URL "/questionnaire".
    - "/tasks/continue": Controla la acción de "Continuar" de la pantalla de tareas asignadas. Hace lo mismo que la URL "/tasks/start" explicada anteriormente, pero sin modificar los estados de las tareas, ya que solo se puede realizar la acción de continuar en tareas que están ya en el estado "En progreso".
    - "/tasks/reject": Controla la acción de "Rechazar" de la pantalla de tareas asignadas. El controlador recibe el id de la tarea jBPM que el usuario quiere rechazar, además del id del contenedor, el proceso al que pertenece la tarea y el usuario al que está asignada y lo que hace es volver las tareas FHIR y jBPM a los estados "Lista" y "Creada" respectivamente, y desasociar el usuario asignado a la tarea jBPM. Y redirige a la URL "/tasks/assignedTasks".

- **`TestController`:** Este controlador es el responsable de iniciar los procesos jBPM para probar la aplicación. Las URLs de las que se ocupa son:
    - "/test/initTareaARol": Inicia el proceso tareaARol, que tiene una tarea humana asociada al rol `kie-server`. Tras iniciar el proceso redirige la aplicación a "/tasks"
    - "/test/initTareaAUsuario": Inicia el proceso tareaAUsuario, que tiene una tarea humana asociada directamente al usuario autenticado. Tras iniciar el proceso redirige la aplicación a "/tasks"

Los servicios de los que hacen uso los controladores son:

- **`QuestionnaireDAO`:** Este servicio solo contiene el método `getQuestionnaireFromTask`, que recibe el id del recurso `Task` de FHIR de la que se quiere extraer el `Questionnaire` y una cadena, el `serverBase`, que corresponde con el servidor FHIR donde se encuentran los recursos, y devuelve el recurso `Questionnaire` asociado. Este método a partir del id del recurso `Task`, hace una consulta al servidor FHIR para obtener ese recurso, luego busca dentro del campo `input` del recurso `Task` un input de tipo `ClossingQuestionnaire`, su valor será el id del recurso `Questionnaire` asociado a la tarea FHIR que a su vez está asociada a la tarea jBPM. A partir de este id del recurso `Questionnaire` se realiza otra consulta al servidor FHIR para obtener el `Questionnaire` completo y devolverlo al controlador que hizo la llamada.
- **`FhirTasksDAO`:** Este servicio es el que se encarga del resto de comunicaciones con el servidor FHIR centrado en acciones sobre las tareas y el `QuestionnaireResponse`. Está separado del `QuestionnaireDAO` para que el servicio encargado de representar el modelo, `QuestionnaireDAO`, sea independiente de este. Podemos encontrar 3 métodos:
    - `UpdateTaskStatus`: Al igual que se actualizan los estados de las tareas jBPM cuando el usuario va realizando las acciones de asignarse la tarea, comenzarla, etc. También hay que actualizar el estado de las tareas FHIR asociadas. Este método recibe el `serverBase`, que es el servidor FHIR donde se encuentran los recursos, el id de la tarea FHIR que queremos actualizar y el estado correspondiente que le queremos asignar a la tarea. Los estados que manejamos son: Creada/Lista, Reservada y En progreso. Es un método genérico que sirve para todas las acciones, simplemente se le pasa el estado correspondiente según la acción que se realice y se actualizará la tarea con ese estado.
    - `CompleteTask`: Este método es el que completa la tarea FHIR. Completar la tarea FHIR tiene varios pasos. El primero es persistir el `QuestionnaireResponse` creado al enviar las respuestas a un `Questionnaire`. El segundo paso es modificar el recurso `Task` para ponerle el estado "Completado" y a su vez añadirle en el campo `output` una entrada de tipo `ClossingQuestionnaireResponse` que tendrá como valor la URI del `QuestionnaireResponse` anteriormente persistido.
    - `SaveQuestionnaireResponse`: Este método es el único privado del servicio, recibe el `serverBase` y el recurso `QuestionnaireResponse`. Es utilizado por el método `CompleteTask` y como su nombre indica lo que hace es persistir el `QuestionnaireResponse` en el servidor FHIR y devuelve la URI correspondiente al recurso para que se añada posteriormente al recurso `Task` asociado.
- **TasksDAO:** Este servicio es el encargado de comunicarse con el servidor jBPM. En él podemos encontrar los siguientes métodos:
    - `FindAllTasks`: Este método recibe el nombre de usuario y devuelve todas las tareas jBPM potencialmente asignables y asignadas para ese usuario.
    - `FindAssignedTasks`: Recibe el nombre de usuario para el que queremos obtener las tareas jBPM asignadas, hace uso del método `findAllTasks`, filtrando las tareas por las que tienen en el campo `ActualOwner` el usuario indicado por parámetro.
    - `FindPotentialTasks`: Este método recibe el nombre de usuario y devuelve las tareas jBPM potencialmente asignables para este. Hace uso del método `findAllTasks`, pero esta vez filtrando por las tareas que no tienen usuario asignado en el campo `ActualOwner`, es decir,

devolviendo únicamente las tareas que no están asignadas a ningún usuario, pero pueden asignarse al usuario pasado por parámetro.

- `ClaimTask`: Como su nombre indica la función de este método es reclamar una tarea jBPM, le cambia el estado a "Reservada" y le asigna en el campo `ActualOwner` el usuario, recibe como parámetros el nombre del usuario al que se asignará la tarea, el contenedor jBPM al que pertenece la tarea y el id de la instancia del proceso al que pertenece la tarea. Además, devuelve el id de la tarea FHIR asociada para que el controlador que hace uso de este método pueda también actualizar el estado del recurso FHIR.
- `StartTask`: Este método recibe los mismos parámetros que el anterior, el nombre del usuario que realiza la acción, el contenedor jBPM al que pertenece la tarea y el id de la instancia del proceso al que pertenece la tarea. Este método Cambia el estado de la tarea a "En progreso", y como la anterior devuelve el id del recurso `Task` de FHIR asociado para que el controlador pueda actualizar su estado y obtener el id del recurso `Questionnaire` correspondiente a esa tarea.
- `ContinueTask`: El método `continueTask` lo que hace es devolver el id del recurso FHIR asociado a la tarea jBPM sobre la que se realiza la acción, no es necesario actualizar el estado porque solo se puede realizar esta acción sobre tareas con estado "En progreso". El método recibe los mismos parámetros que `ClaimTask` y `StartTask`, el nombre del usuario, el contenedor jBPM al que pertenece la tarea y el id de la instancia del proceso al que pertenece la tarea.
- `RejectTask`: Este método recibe como parámetros el id de la tarea jBPM sobre la que se realiza la acción, el nombre del usuario que la realiza y el id del contenedor al que pertenece la tarea. Este método se encarga devolver la tarea a su estado inicial, elimina el usuario del campo `ActualOwner` y la devuelve al estado "Lista".
- `CompleteTask`: Este método es el encargado de completar la tarea jBPM. Solo recibe el id de la tarea, a partir del cual obtiene el recurso completo para usar el id del contenedor al que pertenece y el usuario asignado que es lo necesario para completar una tarea jBPM.
- `GetTaskURIFromTaskInputContent`: Este método se utiliza en el controlador al reclamar y al rechazar una tarea para obtener el id del recurso FHIR asociado a la tarea jBPM para poder actualizar su estado. Este método recibe como parámetros el id de la tarea jBPM, el contenedor al que pertenece y el id de la instancia del proceso al que pertenece, y devuelve el id del recurso `Task` de FHIR asociado.
- `GetTaskInputContent`: Es el único método privado de este servicio, devuelve un mapa de las variables de entrada de la tarea, que incluirá el id del recurso `Task` de FHIR asociado a la tarea jBPM. Los parámetros principales son el id de la tarea jBPM, el id del contenedor que la contiene y el id de la instancia del proceso al que pertenece. A partir de ahí se obtiene el `WorkItem` de esa tarea y de ahí se sacan los parámetros de entrada. Este método se usa en los métodos `starTask`, `continueTask` y `getTaskURIFromTaskInputContent`.

3.2.3 Modelo

Para el modelo vamos utilizar las clases propias de las librerías `hapi.fhir` en su versión 6.10.0 para los recursos FHIR. Los principales recursos FHIR serán: `Task` [4], `Questionnaire` [2] y `QuestionnaireResponse` [5].

Como comentamos anteriormente en el recuso `Task` de FHIR hay un campo que nos indica el id del recurso `Questionnaire` FHIR asociado a esa tarea, este viene dentro del campo `input`, al que hay que añadirle un tipo para diferenciar en caso de que tenga otras entradas. Para ello se ha creado un modelo para definir el tipo de entrada en el que vendrá este id del `Questionnaire`. Se ha creado el archivo `TaskInputTypes` donde se ha introducido el tipo correspondiente `CLOSINGQUESTIONNAIRE`.

Para los recursos jBPM se usan las clases de la librería `org.kie.server.api` en su versión 7.74.1-Final. Los principales recursos jBPM serán: `TaskSummary` [30], `TaskInstance` [31] y `WorkItemInstance` [32]

3.3 Análisis de relaciones

En este apartado se reflejarán algunas conclusiones de la investigación que se realizó para la realización de la aplicación y que se considera de interés para trabajo futuro:

- Equivalencia de estados entre tareas jBPM y FHIR: se ha buscado cual es la equivalencia entre los principales estados que vamos a usar de los recursos `Task` de jBPM y Task de FHIR, la siguiente tabla muestra el resultado de este análisis:

Estados jBPM	**Estados FHIR**
Ready	Draft (No ha sido asignada nunca) Ready (Ha sido rechazada)
Reserved	Requested
In progress	In progress
Completed	Completed

Tabla 9. Equivalencia estados tareas jBPM y FHIR

- Equivalencia entre campos de la definición de tareas humanas en business-central y los atributos de las clases de la api `org.kie.server.api` en su versión 7.74.1-Final de los objetos que representan las tareas jBPM.

Business-central	**TaskSummary**	**TaskInstance**	**WorkItemInstance**
Prioridad	getPriority()	getPriority()	-
Asunto	getSubject()	getSubject()	-
Fecha vencimiento SLA	No se puede obtener	getSlaDueDate()	-
Documentación	No se puede obtener	No se puede obtener	-
Variables de entrada de la tarea	No se puede obtener	No se puede obtener	getParameters()

  - Los campos principales como son: `Name`, `ContainerId`, `ProcessInstanceId`, `ActualOwner`, `State`, `CreatedOn` y `Description` se pueden obtener directamente tanto del `TaskSummary` como del `TaskInstance`, además de poder verlos si se imprimen por consola estos objetos.
  - Los campos "Prioridad" y "Asunto" del Business-central, aunque se puedan obtener cómo se indica en la tabla, no se ven en vaso de imprimir el objeto, es decir, no están en el `toString()`.
  - El campo "fecha de vencimiento SLA" debe estar en formato ISO8601, y además solo se podrá obtener en caso de que la tarea se haya activado con un evento de activación. En el caso de esta

aplicación no aplicaría este campo.

- Para obtener las variables de entrada de una tarea hay que hacerlo a partir del `WorkItemInstance` asociado a esta. El `workItemInstance` se obtiene con el cliente `ProcessServicesClient` y necesitamos el `containerId`, `processInstanceId` y el `workItemId`. Los dos primeros podemos encontrarlos tanto en el objeto `TaskSummary` como `TaskInstance`, pero el `workItemId` solamente está en el `TaskInstance`. A partir del `WorkItemInstance`, usando el método `getParameters()` obtenemos un mapa con clave `String` y valor `Object` que representa las variables de entrada de la tarea.

3.4 Verificación del desarrollo

En este punto se explicarán los pasos llevados a cabo para probar la aplicación.

3.4.1 Recursos FHIR

Esta aplicación utiliza un servidor FHIR para persistir y consultar los cuestionarios y tareas FHIR, maneja los recursos `Questionnaire`, `Task` y `QuestionnaireResponse`.

El servidor FHIR de respaldo se configura en el archivo `application.properties` del servicio. Por defecto se utiliza el servidor de test, versión R5, de Hapi Fhir [33], iniciativa respaldada por Smile Digital [34].

El servidor aloja los recursos `Task` asociados a tareas humanas en el proceso. La versión actual de los procesos de prueba no genera estos recursos, por lo que deben encontrarse previamente en el servidor. Del mismo modo la versión actual de los procesos no genera los recursos `Questionnaire` que representan los datos solicitados al usuario para cerrar la tarea, por lo que también deben estar previamente alojados en el servidor.

Los json proporcionados en el paquete `resources` del repositorio (`Questionnaire.json` y `Task.json`) son ejemplos de estos recursos, y deben estar almacenados previamente en el servidor FHIR. Para ello deberá:

- Abrir en el navegador el servidor de test (R5) disponible en [33] y seleccionar el recurso `Questionnaire`. Elegir en la operación Post la opción Try y adjuntar el cuestionario proporcionado en el paquete `resources` del repositorio. La respuesta devolverá el recurso creado en el servidor, donde podemos encontrar su identificador ("id": "765063" en el ejemplo mostrado en la ilustración).

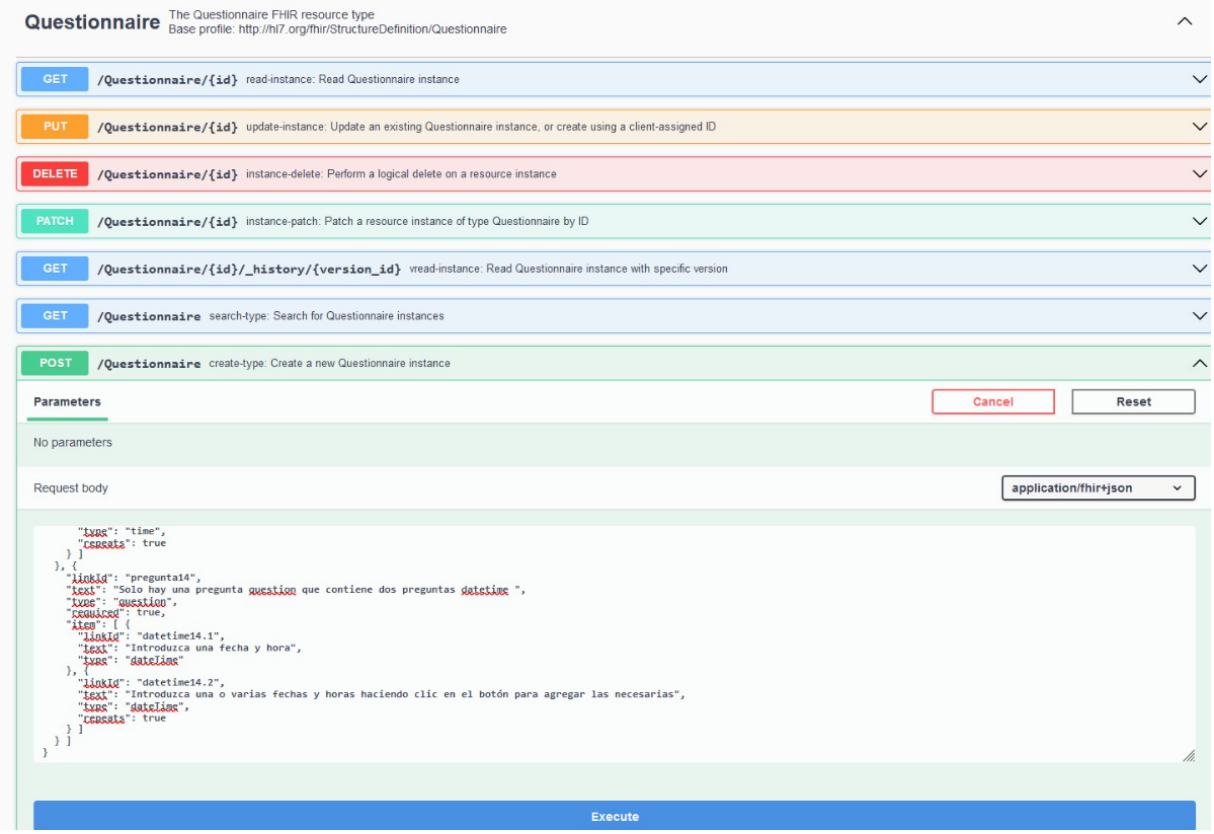

Ilustración 12. Insertar recurso Questionnaire en servidor FHIR

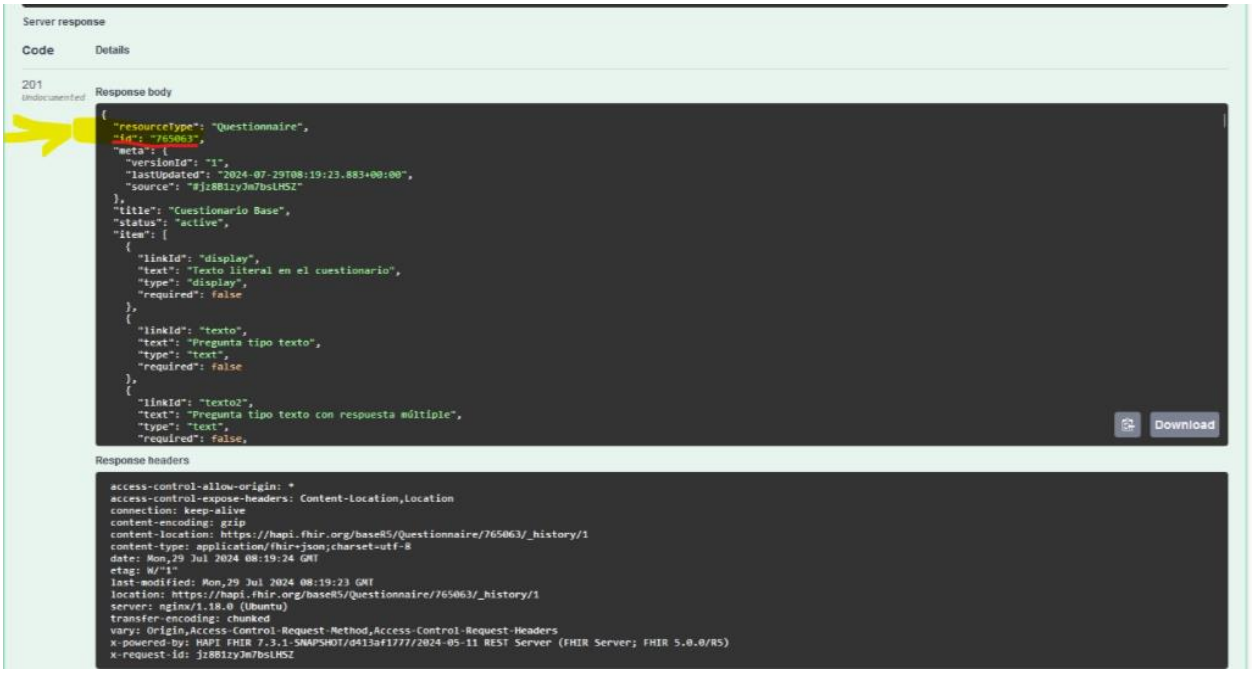

Ilustración 13. Recurso creado en el servidor FHIR

- A continuación, repetir la operación para el recurso `Task`. El fichero JSON proporcionado debe editarse para que el id del cuestionario de cierre sea el que se acaba de crear en el paso anterior. Una vez ejecutado el post el servidor devuelve el recurso creado, donde podemos encontrar su identificador ("id": "765064" en el ejemplo presentado en la siguiente ilustración)

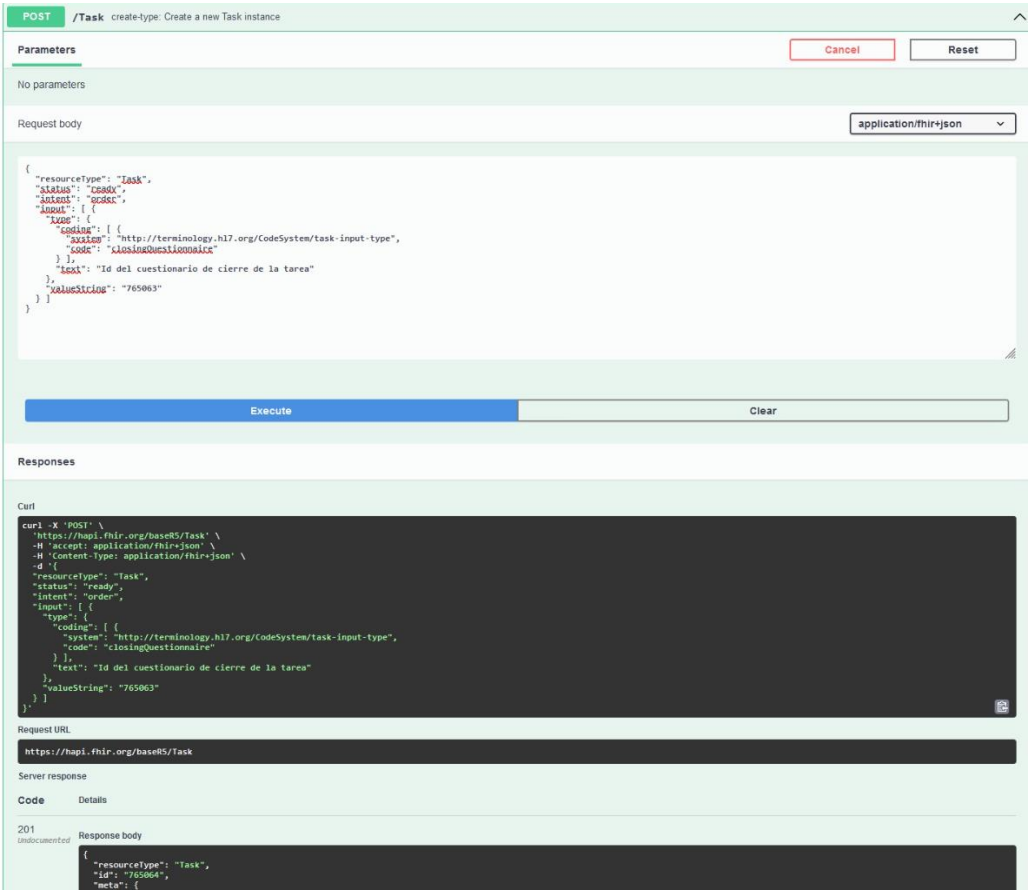

Ilustración 14. Insertar recurso Task en servidor FHIR

- Configurar el fichero `application.properties` y en la propiedad `test.taskid` poner el identificador de la tarea que acaba de crear en el paso anterior.

Tras realizar pruebas podrá recuperarse la tarea y se observará que se ha añadido la respuesta del cuestionario de cierre en las salidas. Es necesario tener en cuenta que dado que todas las tareas que se crean en las pruebas usan el mismo recurso el campo de salida de la tarea irá acumulando referencias a respuestas. Lógicamente este comportamiento cambiará en próximas versiones.

Puede usar el identificador de tarea actualmente configurado, siempre que en el backend no se haya realizado una limpieza tras la publicación de esta memoria.

3.4.2 Base de datos

La información de seguimiento de los procesos se almacena en una base de datos. La configuración de la misma se realiza también en el fichero `application.properties`.

En la configuración proporcionada se utiliza una base de datos local postgresql, de nombre `ht`.

```
#data source configuration
spring.datasource.username=jbpm
spring.datasource.password=jbpm.2.DDBB*
spring.datasource.url=jdbc:postgresql://localhost:5432/ht
spring.datasource.driver-class-name=org.postgresql.xa.PGXADataSource

#hibernate configuration
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.show_sql=false
spring.jpa.properties.hibernate.hbm2ddl.auto=update
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

Ilustración 15. Configuración BBDD

Es necesario que la base de datos esté creada, y configurar adecuadamente este fichero para incluir el usuario y contraseña de la misma.

Se presenta también la configuración necesaria para JPA, api utilizada para la persistencia, cuando se utiliza postgresql.

3.4.3 Procesos de test

En el business-central dentro de un espacio, se ha creado un proyecto titulado: “human-tasks-management”, y dentro de este se han creado dos procesos, cada uno de ellos con una tarea humana. El primer proceso llamado `tareaARol`, tiene la tarea asignada a un rol, al que pertenece nuestro usuario de pruebas, en este caso “kie-server”. El segundo proceso, llamado `tareaAUsuario`, tiene la tarea sin asignar en el business-central, porque el usuario al que se asignará se le pasará como variable. Estos procesos tienen una variable de entrada, que se asigna también como variable de entrada a las tareas humanas, el id del recurso `Task` de FHIR que se corresponde con la tarea humana.

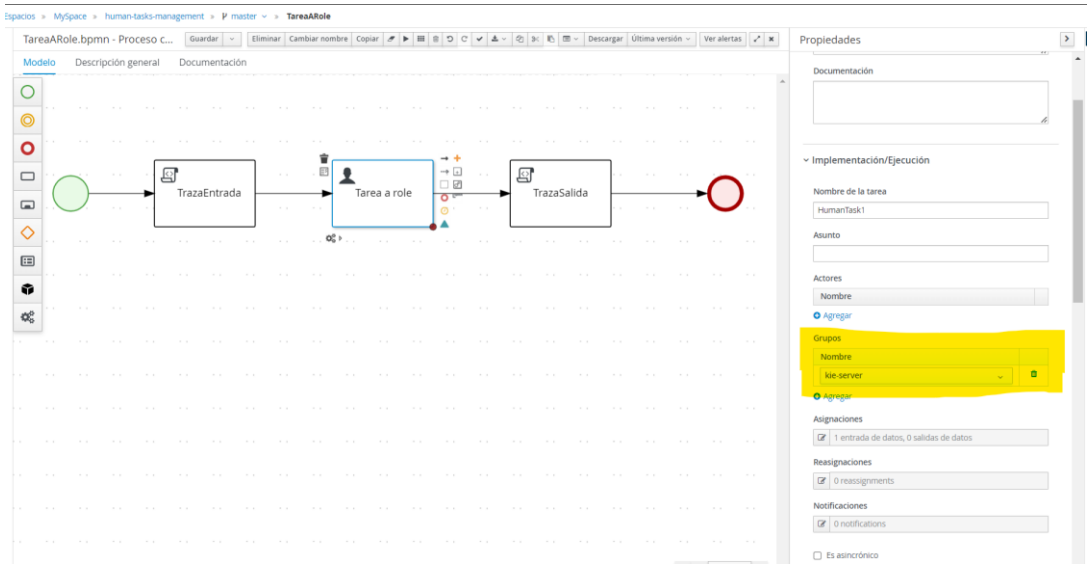

Ilustración 16. Proceso tarea a rol

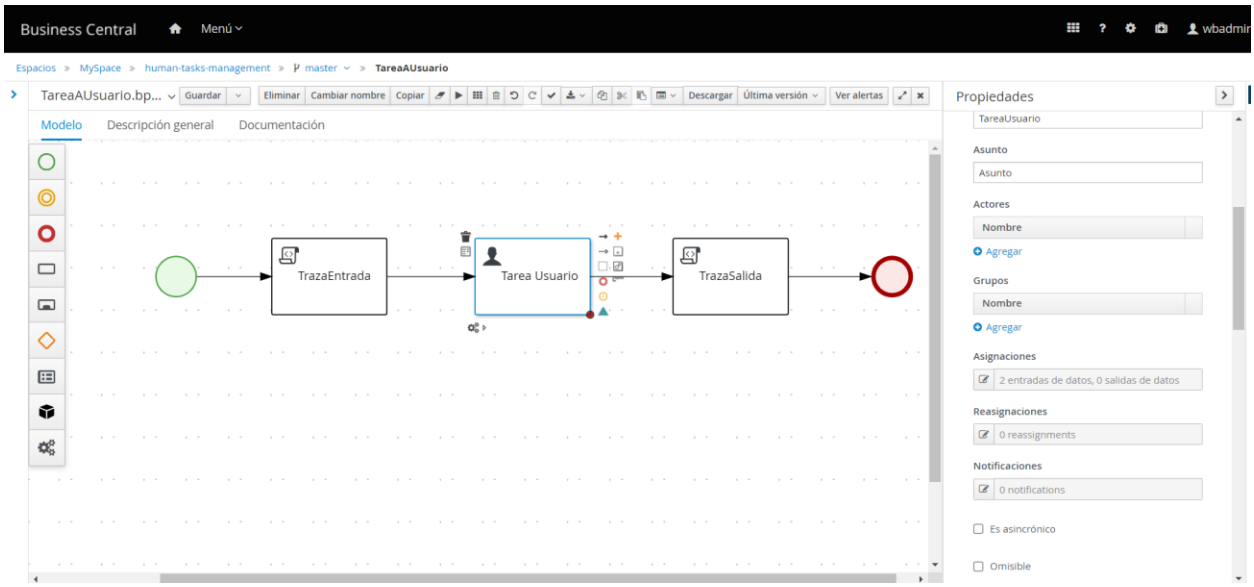

Ilustración 17. Proceso tarea a usuario

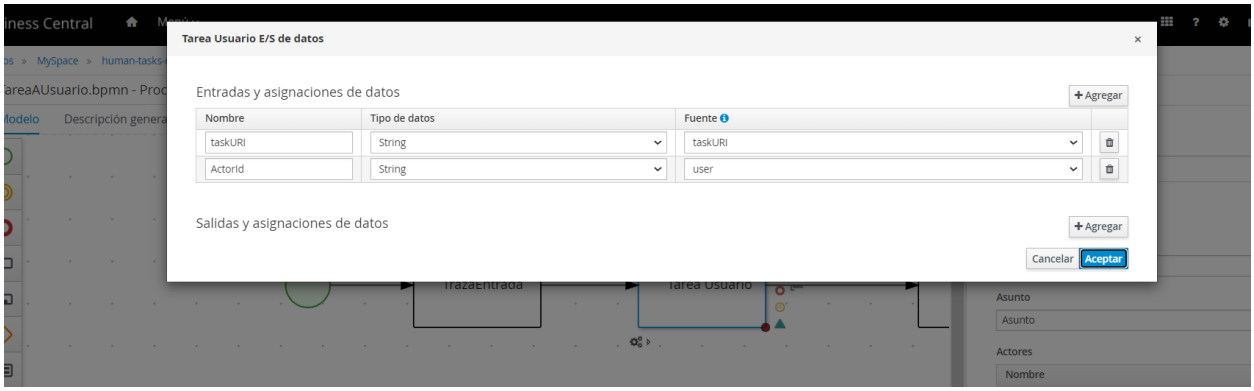

Ilustración 18. Variables de entrada proceso TareaAUsuario

El kjar que incluye estos activos está disponible en el repositorio dentro de la carpeta "human-tasks-management-kjar".

3.4.4 Ejecución

Una vez tengamos esto configurado, arrancamos la aplicación, lanzando el comando: `launch.bat clean install -Ppostgres` desde una consola ubicada en la carpeta `human-task-service` de nuestro proyecto y entramos en la url: http://localhost:8090/. Nos saltará la pantalla del login e introducimos las credenciales, en este caso usuario: "wbadmin" y contraseña: "wbadmin". Los usuarios configurados los podemos encontrar en el archivo `DefaultWebSecurityConfig.java`

Para iniciar los procesos se ha incluido la clase `TestController` con 2 métodos, el primero es `initTareaARol`, que como su nombre indica inicia el proceso llamado `tareaARol`. Para ello hay que enviar un mensaje HTTP GET a la URL `/test/initTareaARol`. Tras iniciar el proceso la aplicación se redirigirá a la página principal de las tareas. El segundo método es `initTareaAUsuario`, que hace lo mismo que el anterior pero esta vez iniciando el proceso `tareaAUsuario`, mandando un mensaje HTTP GET a `/test/initTareaAUsuario`.

Una vez hayamos iniciado cualquiera de los procesos nos saldrá la pantalla principal de las tareas http://localhost:8090/tasks

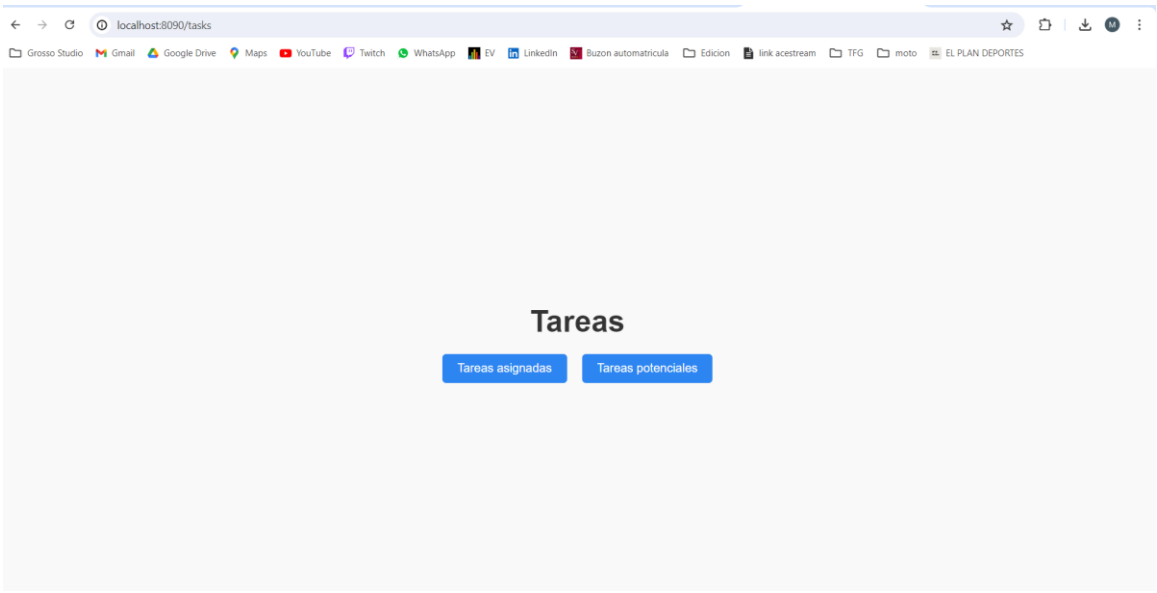

Ilustración 19. Pantalla principal tareas

Tras crear las instancias, en la URL `/tasks/potentialTasks` podemos consultar las tareas potenciales como se muestra en la siguiente ilustración. Se puede ver la tarea del proceso `tareaARol` que acabamos de instanciar. Para ello clicamos en el botón de tareas potenciales. En este caso hay más de una porque se han iniciado los procesos varias veces para distintas pruebas.

Ilustración 20. Pantalla tareas potenciales

Podemos Reclamar la tarea haciendo clic en el botón “Reclamar”. El usuario puede también consultar las tareas que tiene asignadas como muestra la ilustración 21. Para ello tendrá que acceder a la URL `/tasks/assignedTasks`, a la que puede acceder directamente pulsando el botón de “Tareas asignadas”.

Ilustración 21. Pantalla tareas asignadas

Como podemos ver aquí tenemos tanto tareas del proceso `tareaAUsuario` como tareas del proceso `tareaARol` en caso de haberlas reclamado previamente. Para continuar con el proceso podemos comenzar las nuevas tareas, clicando en "Comenzar" o continuar las que previamente habíamos comenzado, clicando en "Continuar", esto nos llevará a la pantalla del cuestionario de cierre de tarea.

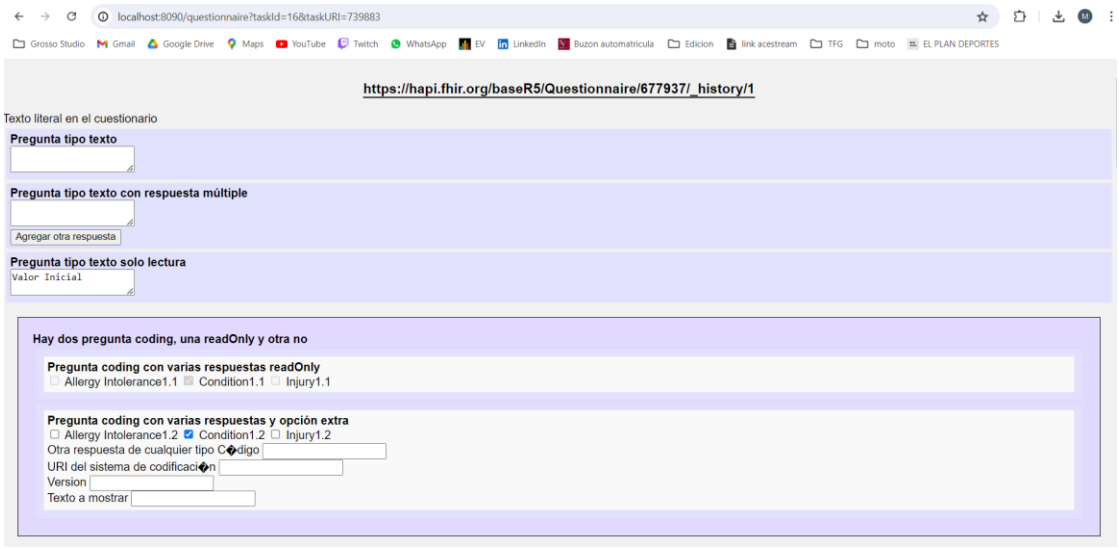

Ilustración 22. Pantalla del cuestionario.

Para poder enviar las respuestas del cuestionario habrá que responder las preguntas obligatorias, que son las que en el recurso `Questionnaire` están marcadas con el campo `Required: true`, que se podrán identificar en el cuestionario porque tienen un asterisco (*) al final del enunciado, como se muestra en la ilustración 23. En caso de darle a "enviar respuesta" si faltara alguna pregunta obligatoria por responder, se indica visualmente. Una vez se envíe el cuestionario, se lanzará el proceso de completar la tarea. En este caso como es la única tarea del proceso también finalizará el proceso y se devolverá de nuevo al usuario a la pantalla principal de las tareas, mostrando un mensaje temporal que indica si se ha completado la tarea con éxito o por el contrario ha habido un error.

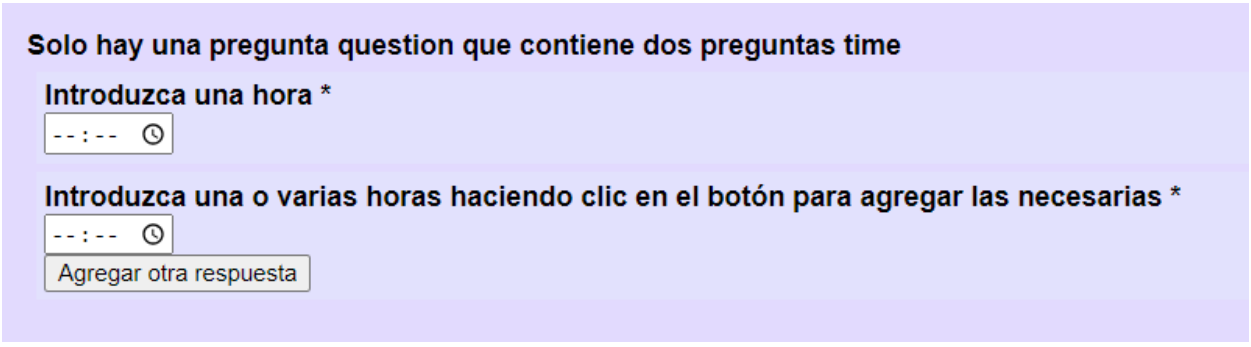

Ilustración 23. Pregunta obligatoria del cuestionario

3.4.5 Análisis del resultado de la ejecución

Para comprobar si el proceso se ha finalizado correctamente, podemos revisar en el servidor FHIR si se ha creado el campo `output` de la tarea añadiendo como valor la URI del recurso `QuestionnaireResponse` generado a partir de las respuestas del cuestionario. También podemos revisar que el recurso `QuestionnaireResponse`

está persistido en el servidor FHIR y las respuestas se corresponden con las introducidas en el cuestionario, navegando a la URI proporcionada en el recurso `Task`.

```
"resourceType": "Task",
"id": "758873",
"meta": {
  "versionId": "27",
  "lastUpdated": "2024-07-29T13:59:48.703+00:00",
  "source": "#Dh4laK6ekHhoWSz5"
},
"status": "ready",
"intent": "order",
"input": [ {
  "type": {
    "coding": [ {
      "system": "http://terminology.hl7.org/CodeSystem/task-input-type",
      "code": "closingQuestionnaire"
    } ],
    "text": "Id del cuestionario de cierre de la tarea"
  },
  "valueString": "758872"
} ],
"output": [ {
  "type": {
    "coding": [ {
      "code": "closingQuestionnaireResponse"
    } ],
    "text": "Id de la respuesta al cuestionario de cierre de la tarea"
  },
  "valueString": "https://hapi.fhir.org/baseR5/QuestionnaireResponse/758968/_history/1"
```


Ilustración 24. Campo de salida en recurso Task de FHIR

4. CONCLUSIONES Y LÍNEAS FUTURAS

Como conclusión, en este proyecto se ha desarrollado una aplicación para la gestión de tareas sanitarias en procesos BPMN. La solución integra la gestión de procesos jBPM con el estándar FHIR. Este proyecto ha logrado implementar una aplicación que facilita el manejo y seguimiento de tareas dentro de los procesos de negocio, específicamente en el entorno sanitario.

De las principales contribuciones de este proyecto es la integración de jBPM, una herramienta de gestión de procesos de negocio, con FHIR, un estándar para el intercambio electrónico de información de salud. Esta combinación permite gestionar de manera eficiente y estructurada las tareas relacionadas con la salud, proporcionando una solución flexible y adaptable a diversas necesidades clínicas.

La aplicación desarrollada se centra en la gestión de tareas humanas dentro de procesos jBPM, cada una de las cuales está vinculada a una tarea FHIR, indicada mediante un parámetro de entrada. A su vez, cada tarea FHIR

está asociada a un recurso `Questionnaire` de FHIR. Esta estructura permite que al iniciar o continuar una tarea jBPM se recupere automáticamente la tarea FHIR correspondiente y su `Questionnaire` asociado. Este cuestionario se muestra al usuario a través de un formulario generado con Thymeleaf que es capaz de adaptarse al contenido del `Questionnaire` sin importar su complejidad o número de preguntas. La respuesta a este `Questionnaire` se corresponde con los datos que se espera que la persona introduzca como salida de la tarea humana.

La ventaja fundamental de esta aplicación, es que permite gestionar tareas humanas sean cuales sean los datos que se espera que introduzca la persona que las ejecute, siempre que se especifique el cuestionario para solicitar los mismos, conforme al estándar FHIR. La aplicación es capaz de representar cualquier cuestionario FHIR y los usuarios podrán responder preguntas personalizadas sin necesidad de personalizar la plantilla de manera individual.

Además, la solución diseñada permite la creación de un recurso `QuestionnaireResponse` de FHIR al completar el cuestionario. Este recurso se persiste en el servidor FHIR y se inserta su id en la tarea FHIR original, proporcionando un registro claro y accesible de las respuestas del cuestionario.

La aplicación también aborda varios desafíos comunes en la gestión de tareas humanas. Facilita la asignación de tareas no asignadas a usuarios específicos, permitiendo a los facultativos organizar sus responsabilidades de manera eficiente. Además, la automatización de los cambios de estado de las tareas reduce la carga administrativa, permitiendo a los usuarios centrarse en sus responsabilidades clínicas. La capacidad de seguir y analizar el progreso de las tareas a través de jBPM y FHIR permite una mejora continua de los procesos, adaptándose a las necesidades cambiantes del entorno sanitario.

En resumen, este proyecto presenta un avance significativo en la gestión de tareas sanitarias, combinando la potencia de BPM con la flexibilidad del estándar FHIR. La integración de estas tecnologías proporciona una solución adaptable, eficiente y centrada en el usuario, capaz de enfrentar los desafíos de la gestión de tareas humanas en el ámbito clínico. Este enfoque no solo mejora la eficiencia operativa, sino que también sienta las bases para futuras expansiones y mejoras, allanando el camino para una gestión de procesos más inteligente y conectada en el sector sanitario.

Aunque este proyecto ha logrado cumplir los objetivos planteados, aún existen algunas áreas que podrían mejorarse en futuros desarrollos. Algunas de estas mejoras son:

- Gestión de los `ítems` de tipo `attachment` en la plantilla del `Questionnaire` de FHIR, para poder guardar esta información al construir el `QuestionnaireResponse`.
- Relacionar en el proyecto las tareas jBPM y FHIR de alguna forma más óptima para un manejo más cómodo de estas, y no tener que estar pasando los ids de cada una de ellas para cada proceso necesario.
- Añadir las acciones de cambio de estados de las tareas jBPM y FHIR en una misma transacción, para que en caso de fallo se haga un rollback de ambas y no se quede la aplicación en un estado inconsistente.
- Añadir a la tarea FHIR el usuario que la realiza, al igual que en la tarea jBPM está el campo `actualOwner`, en la tarea FHIR habría que hacer lo propio para saber qué usuario la tiene asignada, ya que ahora mismo únicamente se está cambiando el estado.
- Implementar un filtrado para la presentación de las tareas, por ejemplo, que el usuario pueda filtrar por fecha de creación o por tipo de tarea y que le aparezcan ordenadas. Sobre este punto hay algo de código desarrollado, pero no se ha llegado a la implementación final.
- Gestionar la seguridad de la aplicación, ya que ahora mismo los usuarios y contraseñas están en un archivo de configuración dentro del proyecto. Para que a la hora de lanzar esta aplicación sea una aplicación robusta y segura.
- El trabajo se ha centrado en la gestión del estado de la tarea y la interfaz para la recogida de la información requerida al profesional sanitario para el cierre de la misma, permitiendo la flexibilidad de esta acción. Para tener una solución completa sería necesario abordar de forma más profunda la ejecución completa de la tarea, incluyendo componentes de ayuda a la decisión clínica que guíen al usuario en la ejecución.
- Esta solución está implementada para que incluya un servidor KIE embebido y gestiona las tareas de

los procesos que corran en este motor de procesos. Sería interesante permitir la configuración de motores externos y que la aplicación se conecte a dichos motores para manejar las tareas de los procesos que corran en los mismos.

En resumen, este proyecto desarrolla una aplicación web en Spring para la gestión de tareas humanas en el ámbito sanitario, integrando tecnologías como jBPM, FHIR R5 y Thymeleaf. La solución permite a los usuarios gestionar tareas y cuestionarios FHIR de manera eficiente, proporcionando una interfaz adaptable y facilitando la automatización y seguimiento de los procesos. El proyecto sienta las bases para futuras mejoras y ampliaciones.

REFERENCIAS

[1] The HL7 FHIR Foundation, «HL7 FHIR Standard,» [En línea]. Available: https://www.hl7.org/fhir/overview.html. [Último acceso: 02 06 2024].

[2] The HL7 FHIR Foundation, «Questionnaire - FHIR v5.0.0,» [En línea]. Available: https://www.hl7.org/fhir/questionnaire.html. [Último acceso: 03 06 2024].

[3] jBPM, «Business Applications - Getting Started,» [En línea]. Available: https://www.jbpm.org/businessapps/gettingStarted.html. [Último acceso: 21 04 2024].

[4] The HL7 FHIR Foundation, «Task - FHIR v5.0.0,» [En línea]. Available: https://www.hl7.org/fhir/task.html. [Último acceso: 03 06 2024].

[5] The HL7 FHIR Foundation, «QuestionnaireResponse - FHIR v5.0.0,» [En línea]. Available: https://www.hl7.org/fhir/questionnaireResponse.html. [Último acceso: 03 06 2024].

[6] KIE Community, «KIE Community - About,» [En línea]. Available: https://www.kie.org/about/. [Último acceso: 21 01 2024].

[7] Oracle Corporation, «What is Java technology and why do I need it?,» [En línea]. Available: https://www.java.com/en/download/help/whatis_java.html. [Último acceso: 05 03 2024].

[8] Oracle Corporation, «Java™ Platform, Standard Edition 8,» [En línea]. Available: https://docs.oracle.com/javase/8/docs/api/. [Último acceso: 05 03 2024].

[9] D. Jones, «Understanding the JavaScript Window Object,» [En línea]. Available: https://www.sitepoint.com/javascript-window-object/#the-browser-object-model. [Último acceso: 10 04 2024].

[10] The Thymeleaf Team, «Thymeleaf,» [En línea]. Available: https://www.thymeleaf.org/documentation.html. [Último acceso: 09 02 2024].

[11] GBTEC Group, «What is BPM and what is it used for?,» [En línea]. Available: https://www.gbtec.com/resources/bpm-business-process-management/. [Último acceso: 20 01 2024].

[12] jBPM, «What is jBPM?,» [En línea]. Available: https://www.jbpm.org/. [Último acceso: 28 01 2024].

[13] IBM Corporation, «What is low-code?,» [En línea]. Available: https://www.ibm.com/topics/low-code. [Último acceso: 21 04 2024].

[14] Red HAt, «What is business process management?,» [En línea]. Available: https://www.redhat.com/en/topics/automation/what-is-business-process-management. [Último acceso: 21 04 2024].

[15] International Business Machines Corporation, «Business Applications,» [En línea]. Available: https://www.ibm.com/docs/en/taddm/7.3.0?topic=using-business-applications. [Último acceso: 10 02 2024].

[16] K. Varela, «After all, what is KIE?,» [En línea]. Available: https://karinavarela.me/2022/04/23/what-is-kie/. [Último acceso: 21 04 2024].

[17] Object Management Group, «BPMN Specification - Business Process Model and Notation,» [En línea]. Available: https://www.bpmn.org/. [Último acceso: 06 06 2024].

[18] Red Hat, «What is a KJAR? | Red Hat Developer,» [En línea]. Available: https://developers.redhat.com/blog/2018/03/14/what-is-a-kjar. [Último acceso: 10 05 2024].

[19] jBPM, «Process definitions and process instances in Business Central,» [En línea]. Available: https://docs.jbpm.org/7.74.1.Final/jbpm-docs/html_single/#process-definitions-and-instances-con-business-processes. [Último acceso: 08 05 2024].

[20] Object Management Group, «ABOUT THE BUSINESS PROCESS MODEL AND NOTATION SPECIFICATION VERSION 2.0,» [En línea]. Available: https://www.omg.org/spec/BPMN/2.0/. [Último acceso: 10 05 2024].

[21] jBPM, «Getting started - Using Single Zip Distribution,» [En línea]. Available: https://www.jbpm.org/learn/gettingStartedUsingSingleZipDistribution.html. [Último acceso: 03 03 2024].

[22] Spring Framework, «Why Spring?,» [En línea]. Available: https://spring.io/why-spring. [Último acceso: 09 03 2024].

[23] SpringBoot, «spring initializr,» VMware Tanzu, [En línea]. Available: https://start.spring.io/. [Último acceso: 23 03 2024].

[24] M. Swiderski, «jBPM Spring Boot Starter,» kiegroup, [En línea]. Available: https://github.com/kiegroup/droolsjbpm-integration/blob/main/kie-spring-boot/kie-spring-boot-starters/jbpm-spring-boot-starter-basic/README.md. [Último acceso: 23 03 2024].

[25] Apache Maven Projoect, «What is Maven?,» [En línea]. Available: https://maven.apache.org/what-is-maven.html. [Último acceso: 08 03 2024].

[26] L. Torvalds, «About - Git,» [En línea]. Available: https://git-scm.com/about. [Último acceso: 05 03 2024].

[27] PostgreSQL Global Development Group, «PostgreSQL: About,» [En línea]. Available: https://www.postgresql.org/about/. [Último acceso: 09 03 2024].

[28] C. Á. Caules, «Spring MVC Configuration,» [En línea]. Available: https://www.arquitecturajava.com/spring-mvc-configuracion/. [Último acceso: 20 02 2024].

[29] Red Hat, «jBPM Human Tasks,» [En línea]. Available: https://docs.jboss.org/jbpm/v6.0/userguide/jBPMTaskService.html. [Último acceso: 11 04 2024].

[30] Red Hat, «TaskSummary (KIE :: Execution Server :: API 7.74.1.Final),» [En línea]. Available: https://javadoc.io/doc/org.kie.server/kie-server-api/latest/org/kie/server/api/model/instance/TaskSummary.html. [Último acceso: 21 03 2024].

[31] Red Hat, «TaskInstance (KIE :: Execution Server :: API 7.74.1.Final),» [En línea]. Available: https://javadoc.io/doc/org.kie.server/kie-server-api/latest/org/kie/server/api/model/instance/TaskInstance.html. [Último acceso: 21 03 2024].

[32] Red Hat, «WorkItemInstance (KIE :: Execution Server :: API 7.74.1.Final),» [En línea]. Available: https://javadoc.io/doc/org.kie.server/kie-server-

api/latest/org/kie/server/api/model/instance/WorkItemInstance.html. [Último acceso: 26 05 2024].

[33] Apache Software License 2.0., «HAPI FHIR Test/Demo Server R5 Endpoint,» [En línea]. Available: https://hapi.fhir.org/baseR5/swagger-ui/. [Último acceso: 15 03 2024].

[34] Smile CDR Inc, «Data Fabric Solutions for the healthcare industry,» [En línea]. Available: https://www.smiledigitalhealth.com/. [Último acceso: 11 06 2024].