



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

**Desarrollo de un motor de orquestación de
flujos de trabajo**

Autor:
Adrián PALANQUES GARCÍA

Supervisor:
Antonio RIAZA CASTAÑO
Tutor académico:
Dolores María LLIDÓ ESCRIVÁ

Fecha de lectura: 11 de Octubre de 2016
Curso académico 2015/2016

Resumen

Este documento describe el proceso de desarrollo de la API de un motor de orquestación de flujos de trabajo desarrollado en Java, para permitir su gestión de forma eficiente y uniforme. Además también se ha desarrollado una aplicación web de diseño de flujos de trabajo, el cual hace uso de la API del motor mediante servicios REST.

Partiendo de la necesidad de unificar la gestión de los flujos de trabajo en varias aplicaciones desarrolladas por la empresa *Integra Consultores*, se ha realizado el diseño e implementación mediante una metodología incremental. Como resultado se ha obtenido una API funcional que cubre todas las necesidades requeridas por la empresa y que será integrado en algunos de los sistemas de información implementados por la empresa.

El desarrollo de la API del motor de flujos de trabajo y de la aplicación web de diseño de flujos de trabajo se han realizado durante la estancia en prácticas en la empresa *Integra Consultores*.

El motor, la aplicación web de diseño de flujos de trabajo y este documento conforman el Trabajo de final de grado (TFG) en el Grado de ingeniería informática en la *Universitat Jaume I*.

Palabras clave

Java, API, Web, Flujo de trabajo, Metodología ágil

Keywords

Java, API, Web, Workflow, Agile methodology

Índice general

1. Introducción	11
1.1. Contexto y motivación del proyecto	11
1.2. Objetivos del proyecto	12
2. Descripción del proyecto	15
2.1. Descripción general	15
2.2. Descripción funcional	16
2.2.1. Descripción de conceptos	16
2.2.2. Descripción funcional del motor	16
2.2.3. Descripción funcional de la aplicación web de diseño de flujos de trabajo .	18
2.3. Alcance	19
3. Planificación del proyecto	21
3.1. Metodología	21
3.2. Planificación	22
3.2.1. Planificación temporal estimada	22
3.2.2. Planificación temporal real	25
3.3. Estimación de recursos y costes del proyecto	27
4. Diseño del sistema	29

4.1.	Diseño de la arquitectura de la API del motor de flujos	29
4.1.1.	Modelo de datos	29
4.1.2.	Arquitectura de la API del motor de flujos	31
4.2.	Diseño de la arquitectura de la aplicación web de diseño de flujos de trabajo . . .	32
4.2.1.	Arquitectura de la aplicación web	32
4.2.2.	Interfaz de la aplicación web	33
5.	Detalles de la implementación	37
5.1.	Tecnologías y herramientas	37
5.2.	Control de versiones	39
5.3.	Detalles de implementación de la API del motor de flujos de trabajo	40
5.3.1.	Base de datos	40
5.3.2.	Validación de datos	40
5.3.3.	Llamadas a funciones externas	42
5.4.	Detalles de implementación de la aplicación web de diseño de flujos de trabajo .	43
5.4.1.	Implementación del método PATCH	43
5.4.2.	URLs de los servicios REST	43
5.4.3.	Implementación del patrón MVVM con AngularJS	46
5.4.4.	Implementación de la interfaz	47
6.	Desarrollo de la implementación	49
6.1.	Sprint 1	50
6.1.1.	Planificación del sprint	50
6.1.2.	Resultados obtenidos	56
6.1.3.	Pila del producto actualizada	57
6.2.	Sprint 2	58

6.2.1.	Planificación del sprint	58
6.2.2.	Resultados obtenidos	63
6.2.3.	Pila del producto actualizada	64
6.3.	Sprint 3	65
6.3.1.	Planificación del sprint	65
6.3.2.	Resultados obtenidos	68
6.3.3.	Pila del producto actualizada	69
6.4.	Sprint 4	71
6.4.1.	Planificación del sprint	71
6.4.2.	Resultados obtenidos	77
6.4.3.	Pila del producto actualizada	77
6.5.	Sprint 5	79
6.5.1.	Planificación del sprint	79
6.5.2.	Resultados obtenidos	82
6.5.3.	Pila del producto actualizada	82
7.	Conclusiones	85
7.1.	Objetivos alcanzados	85
7.2.	La importancia de documentar	85
7.3.	Valoración personal	86
7.4.	Mejoras y ampliaciones del proyecto	86
	Bibliografía	87
A.	Documentación externa de la API del motor de flujos	89
A.1.	Motivación	89

A.2. Elección de cómo y donde documentar la API	89
A.3. Fragmentos de ejemplo de la documentación de la API	90
A.3.1. Introducción	90
A.3.2. Módulo del motor de flujos	91
A.3.3. Funciones externas	94
A.3.4. Estructuras de los mensajes de éxito	96

Glosario de términos	103
-----------------------------	------------

Índice de figuras

4.1. Esquema de la arquitectura de la API del motor.	32
4.2. Esquema de la arquitectura de la aplicación web de diseño de flujos de trabajo. . .	33
4.3. Esbozo de la lista de flujos de trabajo.	34
4.4. Esbozo de la pantalla del editor de un flujo de trabajo.	35
4.5. Jerarquía de pantallas de la aplicación web.	36
5.1. Estructura final de la base de datos.	41
5.2. Adaptación del patrón MVVM a AngularJS.	46
5.3. Pantalla de la lista de flujos de trabajo.	47
5.4. Pantalla de edición de la estructura de un flujo de trabajo.	48

Índice de tablas

3.1. Distribución inicial aproximada de las historias en los sprints.	25
3.2. Distribución final de las historias en los sprints.	26
3.3. Resumen de los costes humanos del proyecto.	27
3.4. Resumen de los costes totales del proyecto.	27
5.1. URLs de los servicios REST relacionados con la estructura de un flujo de trabajo.	44
5.2. Lista de los valores que pueden ir en el parámetro <code>op</code>	45
5.3. URLs de los servicios REST relacionados con la seguridad de un flujo de trabajo y sus componentes.	45
6.1. Estado de la pila del producto tras finalizar el sprint 1.	57
6.2. Estado de la pila del producto tras finalizar el sprint 2.	64
6.3. Estado de la pila del producto tras finalizar el sprint 3.	70
6.4. Estado de la pila del producto tras finalizar el sprint 4.	78
6.5. Estado de la pila del producto tras finalizar el sprint 5.	83

Capítulo 1

Introducción

Contents

1.1. Contexto y motivación del proyecto	11
1.2. Objetivos del proyecto	12

A lo largo de este documento, se describen los detalles del proyecto informático **IdeaFlow** de la empresa Integra Consultores. En este capítulo vamos a describir los objetivos principales del proyecto junto con su contexto y motivación.

1.1. Contexto y motivación del proyecto

Integra Consultores es una empresa tecnológica situada en Castellón, que se dedica a diferentes ámbitos de la informática. Su actividad principal es la implantación de sistemas de información en los sectores de Entidades Financieras, Administración Pública e Industria [1]. Actualmente su plantilla se divide en tres equipos:

1. El equipo del SAT (Servicio de Atención Técnica) está formado por cuatro personas. Se dedica a instalar las infraestructuras oportunas para los sistemas desarrollados y ofrecer servicio técnico en caso de reparación o sustitución de algún componente.
2. El equipo de desarrollo se encarga de implementar o adaptar los sistemas de información demandados, y de su mantenimiento y actualización a lo largo del tiempo. Actualmente está formado por un programador, un diseñador-programador, un analista-programador y el jefe de proyectos. En este equipo es donde realizaré el proyecto descrito en este documento en calidad de programador en prácticas.
3. Recientemente la empresa ha desplazado a otras 5 personas (el otro equipo de desarrollo) a otra empresa que ha contratado un nuevo sistema y quiere tener a los desarrolladores trabajando en el lugar donde se va a implantar el sistema.

Durante el desarrollo de los sistemas de información a medida para algunos clientes, han observado que en muchas ocasiones existen procedimientos que se pueden gestionar como un flujo de trabajo (solicitudes de diversa índole, procedimientos de autorización, etc.). Estos flujos de trabajo son muy similares entre sí y poseen las mismas características. Un flujo de trabajo se caracteriza por estar formado por un conjunto de tareas. En todos los flujos de trabajo existen una tarea inicial y una o varias tareas finales. Además, estas tareas están relacionadas mediante transiciones y siempre tienen un responsable.

Un ejemplo de flujo de trabajo simple puede ser la solicitud de vacaciones dentro de una empresa pequeña. En este caso un trabajador rellena la solicitud con los datos oportunos (esto sería una tarea). Cuando decide presentar la solicitud (esto sería una transición) entonces esta pasa a validación (esto es otra tarea), donde el responsable oportuno (en este caso el responsable de recursos humanos) decide si aceptarla o cancelarla, llegando así a las tareas finales de aceptada o cancelada.

En la empresa, tras analizar varios motores de flujo de trabajo existentes en el mercado como jBPM [2] y Camunda [3], se decidió que la mejor solución era implementar uno propio. Se tomó esta decisión debido a que estos motores son excesivamente potentes para las funcionalidades que buscaban y requerían una curva de aprendizaje muy alta. Además, requerían poder acoplar el motor a las aplicaciones ya desarrolladas.

Este proyecto consiste en desarrollar una API en Java para gestionar de forma eficiente los flujos de trabajo. Esta API del motor será integrada en los sistemas de información implementados por la empresa.

Hemos de desarrollar una aplicación web que nos permita el diseño de flujos de trabajo con parte de las funcionalidades que ofrece la API del motor. Para poder acceder a la API desde el cliente web se desarrollarán una serie de servicios REST que permitan hacer uso de ella.

Gracias a la API del motor de flujos, la empresa Integra Consultores pretende reducir el tiempo de desarrollo de los futuros proyectos en los que necesite gestionar un flujo de trabajo. Además, espera homogeneizar completamente la forma en la cual gestiona estos flujos, al depender de la API.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es desarrollar una API en **Java** que permita gestionar cualquier flujo de trabajo simple que pueda darse en una empresa, utilizando como persistencia una base de datos MySQL. Con esta API se pretende gestionar los flujos de trabajo que la empresa tenga que gestionar en sus futuros proyectos. Además, para facilitar el diseño de flujos de trabajo, se va a implementar un cliente web. Este se comunicará con la API del motor de flujos mediante servicios REST que solo permitirán acceder a algunas funcionalidades de la API. Se ha elegido el lenguaje de programación Java para poder integrar la API en cualquier proyecto de la empresa, todos sus proyectos se desarrollan utilizando dicho lenguaje.

Este proyecto se puede dividir en dos subproyectos.

Por un lado se debe desarrollar la API del motor de flujos. Esta tiene como principales objetivos:

- Definir nuevos flujos de trabajo, ampliarlos y modificarlos.
- Permitir duplicar la definición de un flujo de trabajo.
- Permitir la cancelación de un flujo de trabajo.
- Permitir el recorrido de forma sencilla cualquier flujo de trabajo existente.
- Implementar un sistema de permisos para poder iniciar el recorrido de los flujos de trabajo tanto para usuarios como para departamentos.
- Implementar un sistema de permisos para poder realizar una tarea determinada.
- Permitir la cancelación del recorrido de un flujo de trabajo.
- Permitir realizar llamadas a funciones externas.
- Permitir definir tareas condicionales.
- Permitir definir tareas de asignación dinámica.
- Implementar un histórico que refleje cualquier cambio durante el recorrido de un flujo de trabajo y permita añadir entradas en él desde el exterior de la API.

Por otro lado se debe desarrollar la aplicación web de diseño de flujos de trabajo que hará uso de la API del motor. Sus principales objetivos son:

- Permitir hacer uso de parte de la API mediante servicios REST.
- Permitir crear y modificar flujos de trabajo.
- Permitir duplicar un flujo.
- Permitir crear y modificar tareas.
- Permitir crear y modificar transiciones.
- Permitir editar los permisos de un flujo.
- Permitir editar los permisos de una tarea.

Finalmente, el desarrollo del proyecto tiene como objetivos formativos que el alumno encargado de desarrollar la API del motor y la aplicación web de diseño de flujos de trabajo adquiera unas determinadas competencias y habilidades:

- Aprender acerca de los flujos de trabajo.
- Ampliar sus conocimientos sobre servicios REST.
- Mejorar sus habilidades sobre AngularJS.
- Conocer las tecnologías y herramientas que utiliza la empresa.
- Conocer la metodologías de trabajo de la empresa.
- Crear un subsistema que deberá poder integrarse en diversos sistemas.
- Mejorar sus habilidades en la documentación de aplicaciones.

Capítulo 2

Descripción del proyecto

Contents

2.1. Descripción general	15
2.2. Descripción funcional	16
2.2.1. Descripción de conceptos	16
2.2.2. Descripción funcional del motor	16
2.2.3. Descripción funcional de la aplicación web de diseño de flujos de trabajo	18
2.3. Alcance	19

2.1. Descripción general

El departamento de desarrollo de software de **Integra Consultores** ha decidido desarrollar un sistema informático, que consistirá en el desarrollo de **una API de un motor de flujos de trabajo**. Gracias a este sistema los nuevos proyectos que requieran gestionar flujos de trabajo podrán hacer uso de esta API para simplificar la implementación de estos. Además requiere una interfaz gráfica para hacer uso de las funcionalidades de diseño de la API del motor, por lo que se desarrollará una aplicación web de diseño de flujos de trabajo.

Concretamente se van a desarrollar dos subsistemas:

- **API del motor de flujos de trabajo:** esta API permitirá realizar todas las funcionalidades de un motor de flujos de trabajo básico. Desde la creación de flujos de trabajo sencillos hasta flujos de trabajo con tareas condicionales. Además permitirá un control de permisos sobre tareas y flujos.
- **Aplicación web de diseño de flujos de trabajo:** este permitirá a los desarrolladores crear de forma rápida y sencilla los nuevos flujos de trabajo que sean necesarios para las nuevas aplicaciones así como dar permisos a sus futuros usuarios y departamentos. Esta aplicación hará uso de la API del motor de flujos en la parte del servidor.

2.2. Descripción funcional

2.2.1. Descripción de conceptos

Un **flujo de trabajo** está formado por un conjunto de **tareas** relacionadas entre sí mediante **transiciones**. Un **flujo de trabajo** posee un **administrador**.

Una **instancia de flujo de trabajo** es el recorrido de un **flujo de trabajo** que inicia un usuario. Conforme se avanza en el flujo de trabajo se van **instanciando tareas** y finalizándolas. Cuando se instancia una tarea final entonces se finaliza la instancia del flujo de trabajo a la que pertenece.

Un flujo de trabajo posee una única **tarea inicial** y una o varias **tareas finales**. Las instancias de las tareas tienen responsable, el cual puede ser un usuario o un departamento. Las tareas de un flujo de trabajo pueden ser de distintos tipos, entre ellos condicional y dinámica. Las **tareas condicionales** son tareas cuyas instancias no requieren la interacción de un usuario y se finalizan solas tomando la transición oportuna. Las tareas de **asignación dinámica** son tareas en las que cuando se instancian, el responsable se obtiene dinámicamente (en lugar de estar predeterminado se obtiene de una lista mediante una función externa).

2.2.2. Descripción funcional del motor

La API del motor de flujos de trabajo será utilizada, **a través de otras aplicaciones**, por usuarios previamente identificados en dichas aplicaciones. Estas aplicaciones utilizarán el identificador del usuario y el identificador del departamento al que pertenece el usuario para identificarlo ante la API del motor y realizar las llamadas oportunas. A continuación se describen todas las funcionalidades que ofrece la API del motor.

Los usuarios pueden realizar dos acciones principales, instanciar (iniciar el recorrido de este) un flujo de trabajo y realizar una serie de acciones sobre instancias de tareas en un flujo de trabajo que esté siendo recorrido.

No todos los usuarios pueden realizar todas las acciones. Como hemos comentado los flujos tienen un administrador y las instancias de tareas un responsable. Existe un sistema de permisos, que se aplica a usuarios y departamentos de forma que el usuario puede tener permiso para instanciar un flujo o tener permiso para ser el responsable de una instancia de una tarea (bien directamente o bien a través de su departamento).

Cualquier usuario podrá:

- Consultar los datos referentes a la estructura de un flujo de trabajo (las tareas que contiene, las transiciones que parten de una tarea, etc).
- Consultar qué flujos de trabajo puede instanciar.
- Consultar sus instancias de tareas pendientes (aquellas que no han sido finalizadas) y las de su departamento.
- Consultar el histórico de aquellas instancias de flujo en las que haya participado.
- Gestionar (crear y eliminar) los permisos para instanciar flujos de trabajo y para realizar instancias de tareas, así como cambiar el responsable de un flujo de trabajo.
- Gestionar la estructura completa de un flujo de trabajo. Esto implica poder crear, modificar y eliminar flujos de trabajo, tareas y transiciones.

Si un **usuario** tiene **permiso** para **instanciar un flujo** de trabajo, este puede instanciarlo.

Si un **usuario** es el **responsable** de una **instancia de tarea** puede:

- Finalizar la instancia de tarea indicando la transición a tomar hacia la siguiente tarea.
- Consultar a que otros usuarios o departamentos puede asignarles la instancia de tarea.
- Asignar como responsable de la instancia de tarea a otro usuario o departamento.

Si el **departamento de un usuario** es el **responsable** de una **instancia de tarea**, este usuario puede asignarse como responsable de la instancia de tarea.

Si un **usuario** es **administrador** de un **flujo** de trabajo puede:

- Consultar el histórico de aquellas instancias de flujo que pertenezcan al flujo de trabajo que administra.
- Cancelar cualquier instancia del flujo de trabajo que administra (y por lo tanto cualquier instancia de tarea que no haya sido finalizada dentro de esa instancia de flujo).
- Finalizar cualquier instancia de tarea que pertenezca a cualquier instancia del flujo que administra indicando la transición a tomar hacia la siguiente tarea.
- Asignar como responsable de cualquier instancia de tarea no finalizada que pertenezca a cualquier instancia del flujo que administra a otro usuario o departamento.

Por último la **API del motor de flujos** debe de realizar las siguientes **funcionalidades internas**:

- Ante cualquier acción que requiera algún tipo permiso, comprobarlo y no permitir la acción en caso de no poseerlo.
- Cuando un usuario instancie un flujo de trabajo, se debe de instanciar automáticamente su tarea inicial y asignar como responsable de esta al usuario.
- Cuando un usuario finalice una instancia de tarea tomando una transición, se debe de instanciar la tarea destino de dicha transición y asignarle el responsable oportuno.
- Cuando se instancie una tarea que requiera responsable, este se le debe de asignar a partir de la lista de permisos de tareas o de forma dinámica si es el caso.
- Cuando se instancie una tarea condicional, esta debe de ser finalizada automáticamente tomando la transición oportuna.
- Cuando se instancie una tarea final, esta debe de ser finalizada automáticamente junto con la instancia del flujo de trabajo a la que pertenece.
- Cuando se tome una transición con función externa, se permitirá o no realizar la acción en función del resultado.
- Cualquier acción realizada en una instancia de un flujo de trabajo debe de ser reflejada en el histórico.
- Llamar a funciones de la aplicación que hace uso de la API del motor en las funciones condicionales, las tareas de asignación dinámica y las funciones de las transiciones.

2.2.3. Descripción funcional de la aplicación web de diseño de flujos de trabajo

Los usuarios que utilicen la aplicación web de diseño de flujos de trabajo serán considerados en esta sección como diseñadores de flujos de trabajo.

El diseñador de flujos de trabajo es el que realiza todas las tareas asociadas a la definición y modificación de la estructura de los flujos de trabajo.

El diseñador de flujos podrá realizar todas las operaciones del motor relacionadas con la definición estructural de un flujo de trabajo. Esto implica la creación y modificación de flujos, tareas y transiciones, así como desactivarlos y reactivarlos. Además podrá definir tareas condicionales y de asignación dinámica.

Por último el diseñador de flujos también podrá definir la seguridad de los flujos de trabajo, definiendo así los permisos de los flujos y las tareas y designando el administrador de cada flujo.

2.3. Alcance

La API del motor de flujos de trabajo solo abarca las funcionalidades requeridas para flujos secuenciales (flujos en los que su recorrido se realiza secuencialmente, tarea a tarea). De esta forma, la implementación de toda la funcionalidad requerida para dar soporte a posibles paralelismos queda descartada.

La API del motor de flujos de trabajo no se encargará ni de la identificación de los usuarios ni de la validez de sus credenciales. Dado que la API ha sido diseñada para ser embebida en otras aplicaciones, estas son las responsables de realizar las comprobaciones de seguridad oportunas sobre las credenciales de los usuarios.

Dado que la API del motor de flujos permite la modificación de los permisos y de la estructura de un flujo de trabajo a cualquier usuario, el hecho de solo permitir esas funcionalidades a los usuarios que lo requieran será responsabilidad de la aplicación que la contiene.

En cuanto al sistema de permisos, este ha sido diseñado para dar soporte a usuarios y departamentos, quedando totalmente descartada la posibilidad de aceptar estructuras jerárquicas de seguridad, para poder hacer uso de este tipo de estructuras existen las tareas de asignación dinámica.

La aplicación web de diseño de flujos de trabajo de momento no mostrará los diagramas de flujos, es su lugar representará la información estructural de los flujos en tablas.

La aplicación web de diseño de flujos de trabajo no se encargará de la autenticación de los usuarios ni de la validez de sus credenciales. El acceso a esta aplicación web será controlado por el sistema de autenticación general de la empresa.

Capítulo 3

Planificación del proyecto

Contents

3.1. Metodología	21
3.2. Planificación	22
3.2.1. Planificación temporal estimada	22
3.2.2. Planificación temporal real	25
3.3. Estimación de recursos y costes del proyecto	27

3.1. Metodología

Para realizar el proyecto se ha decidido utilizar una adaptación de la metodología ágil Scrum [23]. Esta metodología consiste en realizar un desarrollo incremental del proyecto en lugar de planificar y ejecutar completamente el proyecto.

Se ha decidido esta estrategia debido a que muchas de las funcionalidades que debe realizar la API aun no se sabe como realizarlas o su definición es muy escasa y general. Además, debido a que el analista normalmente solo está un día a la semana, esta estrategia es perfecta para realizar una reunión de seguimiento del sprint cada una o dos semanas y así definir las funcionalidades incrementalmente y analizar el avance del proyecto.

En esta metodología la descripción de las funcionalidades a implementar se realiza mediante historias de usuario y se estima temporalmente mediante puntos de historia.

Para organizar las historias de usuario se ha utilizado la metodología Kanban. Recientemente la empresa está empezando a utilizar el Kanban Borad de Jira, el cual facilita el trabajo.

Finalmente, dado que la API del motor de flujos que se desea implementar está formada mayormente por operaciones y funcionalidades atómicas o poco complejas, se ha decidido utilizar el desarrollo dirigido por pruebas (TDD) para implementar las partes más críticas de la API, los métodos con lógica que interaccionan con la base de datos.

El TDD consiste en escribir las pruebas antes que el código y cuando este pasa las pruebas se refactoriza. De esta forma se obtiene un código limpio y funcional, además en caso de tener que modificarlo en el futuro, si se modifica erróneamente afectando a otras funcionalidades el problema surge inmediatamente a la luz, reduciendo el tiempo de búsqueda de errores al mínimo [24].

3.2. Planificación

Dado que durante este proyecto se va a utilizar una adaptación de la metodología ágil Scrum, en este apartado se realiza una planificación temporal de la duración de cada sprint y una primera planificación de la pila del producto.

La duración de este proyecto es de aproximadamente unas 300 horas. Dado que cada jornada es de 5 horas el proyecto se realizará a lo largo de unas 12 semanas. Además se ha decidido que cada punto de historia equivale a una jornada de trabajo.

Durante esas semanas se realizarán cinco sprints, cada uno de dos semanas aproximadamente. Las otras dos semanas se utilizarán para realizar tareas que no forman parte de los sprints. Concretamente la semana inicial se utilizará para conocer la empresa, las metodologías de trabajo, las herramientas y el proyecto a realizar. La última semana se utilizará para corregir errores, realizar mejoras sobre la API y para empezar a integrar la API en otro proyecto.

3.2.1. Planificación temporal estimada

Para realizar la estimación de los puntos de historia se han tenido en cuenta varios factores. Por un lado la complejidad que el supervisor ha previsto que tendrá esa historia y por otro lado, dado que las historias de la aplicación web de diseño de flujos de trabajo dependen directamente de las de la API del motor de flujos, se ha decidido realizar solo una valoración superficial del conjunto del coste temporal que puede suponer. Además las historias que implican el desarrollo de algún apartado gráfico tienen un coste mayor, debido a que la experiencia realizando interfaces gráficas es menor y requieren de más tiempo. Por último, cabe destacar que aquellas historias que aún no se sabe con certeza como van a afectar al resto de componentes ni como se van a implementar tienen un coste estimado extra (como es el caso de la historia de usuario para implementar las tareas condicionales).

A continuación se muestra la pila del producto inicial del proyecto. En ella aparecen las historias de usuario priorizadas (según el valor para la empresa, que en nuestro caso es nuestro cliente) con una estimación en puntos de historia.¹

¹Junto al nombre de cada historia se ha indicado si pertenece a la API del motor de flujos de trabajo [M] o a la aplicación web de diseño de flujos de trabajo [W].

- HU1 Crear y modificar flujos [M]**
Como diseñador de flujos de trabajo quiero poder definir flujos de trabajo para poder definir uno de los elementos básicos de un flujo de trabajo.
Estimación: **1 PH** — Prioridad: **Muy Alta**
- HU2 Crear y modificar tareas [M]**
Como diseñador de flujos de trabajo quiero poder definir tareas para poder definir uno de los elementos básicos de un flujo de trabajo.
Estimación: **1 PH** — Prioridad: **Muy Alta**
- HU3 Crear y modificar transiciones [M]**
Como diseñador de flujos de trabajo quiero poder definir transiciones entre tareas para poder definir los diferentes caminos que puede tener un flujo de trabajo.
Estimación: **1 PH** — Prioridad: **Muy Alta**
- HU4 Consulta de flujos de trabajo [M]**
Como sistema quiero poder conocer los flujos de trabajo definidos y las tareas que lo componen para poder presentar en la aplicación las opciones necesarias.
Estimación: **1 PH** — Prioridad: **Alta**
- HU5 Instanciar un flujo de trabajo [M]**
Como usuario quiero poder instanciar un flujo de trabajo para que se realicen los trámites oportunos.
Estimación: **3 PH** — Prioridad: **Muy Alta**
- HU6 Consultar una instancia de un flujo de trabajo [M]**
Como usuario quiero poder consultar el estado de una instancia de un flujo de trabajo para ver en que estado se encuentra.
Estimación: **1 PH** — Prioridad: **Muy Alta**
- HU7 Sistema de permisos de flujos [M]**
Como diseñador de flujos de trabajo quiero poder definir la seguridad relacionada con un flujo de trabajo para asegurar que solo los usuarios autorizados (bien sean ellos mismos o bien su departamento) puedan instanciar un flujo de trabajo.
Estimación: **2 PH** — Prioridad: **Alta**
- HU8 Sistema de permisos de tareas [M]**
Como diseñador de flujos de trabajo quiero poder definir qué usuarios o departamentos son los responsables de realizar una tarea así como definir un primer responsable (persona o departamento a la que se asignará la tarea al instanciarse).
Estimación: **2 PH** — Prioridad: **Alta**
- HU9 Traspaso de tareas [M]**
Como usuario quiero poder traspasar las tareas que tenga asignada o estén asignadas a mi departamento a otros usuarios o departamentos que tengan permiso para así poder repartir mejor el trabajo.
Estimación: **2 PH** — Prioridad: **Media**
- HU10 Histórico [M]**
Como usuario quiero poder consultar todo el histórico de ejecución de una instancia de flujo de trabajo para poder conocer todas las acciones que se han llevado a cabo en una instancia de flujo de trabajo.
Estimación: **2 PH** — Prioridad: **Alta**

HU11 Cancelar una instancia de flujo de trabajo [M]

Como administrador de un flujo de trabajo quiero poder cancelar una de sus instancias y todas sus instancias de tarea para poder cancelar flujos de los que nadie se hace cargo o que se han creado erróneamente.

Estimación: **1 PH** — Prioridad: **Media**

HU12 Llamadas a métodos externos [M]

Como diseñador de flujos de trabajo quiero poder indicar que se ejecute un método de una clase externa a la API para mejorar la integración de la API con las necesidades funcionales de las aplicaciones externas.

Estimación: **2 PH** — Prioridad: **Alta**

HU13 Asignación dinámica de tareas [M]

Como diseñador de flujos de trabajo quiero poder indicar que una tarea es de asignación dinámica para así poder determinar su responsable en tiempo de ejecución y hacer mucho más flexible y potente la definición de flujos de trabajo.

Estimación: **1 PH** — Prioridad: **Media**

HU14 Tareas condicionales [M]

Como diseñador de flujos de trabajo quiero poder definir tareas condicionales para que el flujo de trabajo siga un camino u otro en función del resultado la condición.

Estimación: **5 PH** — Prioridad: **Baja**

HU15 Interfaz gráfica de edición de flujos [W]

Como diseñador de flujos de trabajo quiero poder editar la estructura básica de un flujo de trabajo desde una interfaz gráfica web para hacer mi trabajo de una forma más productiva y fácil.

Estimación: **10 PH** — Prioridad: **Alta**

HU16 Servicios REST de edición de flujos [W]

Como cliente web quiero poder comunicarme con servicios REST para que este me permita el acceso a la funcionalidad de edición de flujos de trabajo de la API.

Estimación: **10 PH** Prioridad: **Alta**

En la tabla 3.1 se ha realizado un reparto de las historias de usuario entre los sprints para tener una aproximación de las tareas a realizar en cada sprint y como se organizan en el tiempo.

Sprint	Fecha inicio	Fecha fin	Historia de usuario	Coste estimado (PH)
Sprint 1	19/02/16	07/03/16	HU01 - Crear y modificar flujos [M]	1
			HU02 - Crear y modificar tareas [M]	1
			HU03 - Crear y modificar transiciones [M]	1
			HU04 - Consulta de flujos de trabajo [M]	1
			HU05 - Instanciar un flujo de trabajo [M]	3
			HU06 - Consultar una instancia de flujo de trabajo [M]	1
			HU07 - Sistema de permisos de flujos [M]	2
Sprint 2	08/03/16	21/03/2016	HU08 - Sistema de permisos de tareas [M]	2
			HU09 - Traspaso de tareas [M]	2
			HU10 - Histórico [M]	2
			HU11 - Cancelar una instancia de flujo de trabajo [M]	1
			HU12 - Llamadas a métodos externos [M]	2
			HU13 - Asignación dinámica de tareas [M]	1
Sprint 3	22/03/16	07/04/16	HU14 - Tareas condicionales [M]	5
Sprint 4	08/04/16	21/04/16	HU15 - Interfaz gráfica web de edición de flujos [W]	10
Sprint 5	22/04/16	05/05/2016	HU16 - Servicios REST de edición de flujos [W]	10

Tabla 3.1: Distribución inicial aproximada de las historias en los sprints.

En el sprint 1 se va a implementar toda la funcionalidad básica que permita la gestión de flujos de trabajo e instancias de estos.

En el sprint 2 se va a implementar toda la funcionalidad avanzada de los flujos de trabajo y los permisos.

En el sprint 3 se va a implementar toda la funcionalidad más compleja relacionada con las tareas y las instancias de estas. Dado que se espera finalizar todas las funcionalidades de la API del motor de flujos de trabajo durante este sprint, en la estimación inicial se ha decidido dejar una holgura de 5PH. La finalidad de esta holgura es poder hacer frente a cualquier sobre coste temporal que pueda aparecer durante los sprints iniciales.

En el sprint 4 se va a implementar la interfaz de usuario de la aplicación web de diseño de flujos de trabajo que permita hacer uso de las funciones de diseño de la API del motor de flujos.

En el sprint 5 se va a implementar los servicios REST necesarios para que la aplicación web de diseño de flujos de trabajo pueda hacer uso de la API del motor de flujos a través de ellos.

3.2.2. Planificación temporal real

Durante la realización de proyecto se definieron nuevas historias de usuario y otras se modificaron. En la tabla 3.2 se puede observar la planificación real del proyecto.

Los cambios más significativos se produjeron en los sprints 4 y 5. Esto se debe a que en la planificación inicial se poseía una información muy imprecisa de los requisitos que debía de

cumplir la aplicación web de diseño de flujos de trabajo. Esta información ya se pudo concretar tras finalizar el sprint 3, en el que se pudo finalizar la API del motor de flujos de trabajo.

Sprint	Fecha inicio	Fecha fin	Historia de usuario	Coste real (PH)
Sprint 1	19/02/16	07/03/16	HU01 - Crear y modificar flujos [M] HU02 - Crear y modificar tareas [M] HU03 - Crear y modificar transiciones [M] HU04 - Consulta de flujos de trabajo [M] HU05 - Instanciar un flujo de trabajo [M] HU06 - Consultar una instancia de flujo de trabajo [M] HU07 - Sistema de permisos de flujos [M]	1 2 1 1 2 1 2
Sprint 2	08/03/16	21/03/2016	HU08 - Sistema de permisos de tareas [M] HU09 - Traspaso de tareas [M] HU10 - Histórico [M] HU11 - Cancelar una instancia de flujo de trabajo [M] HU17 - Instanciar automáticamente la tarea inicial [M] HU18 - Finalizar una tarea [M] HU19 - Designar administrador de un flujo [M]	2 1 2 1 1 2 1
Sprint 3	22/03/16	07/04/16	HU12 - Indicar funciones externas en transiciones [M] HU13 - Asignación dinámica de tareas [M] HU14 - Tareas condicionales [M] HU20 - Documentar la API [M]	2 1 3 4
Sprint 4	08/04/16	21/04/16	HU21 - Consultar los flujos de trabajo existentes [W] HU22 - Crear, modificar y dar de baja/alta un flujo de trabajo [W] HU23 - Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos [W] HU24 - Consultar la estructura de un flujo [W] HU25 - Consultar los permisos de una tarea, crearlos y eliminarlos [W] HU26 - Crear una tarea completa, modificarla y eliminarla [W]	2 1 2 3 2 -
Sprint 5	22/04/16	05/04/2016	HU26 - Crear una tarea completa, modificarla y eliminarla [W] HU27 - Crear una transición completa, modificarla y eliminarla [W] HU28 - Permitir intercambiar el orden de las funciones condicionales [W] HU29 - Permitir modificar funciones condicionales [W] HU30 - Duplicar un flujo de trabajo [M y W]	3 2 2 1 2

Tabla 3.2: Distribución final de las historias en los sprints.

3.3. Estimación de recursos y costes del proyecto

En cuanto a la estimación de recursos y costes podemos dividirlo en costes humanos, costes de software y costes hardware.

Los costes humanos son las 300 horas estimadas para realizar el proyecto. Estas horas se pueden dividir en dos partes, 260 horas de programación y 40 horas de análisis. La tabla 3.3 muestra un resumen de los costes humanos y su cálculo.

Labor	Duración (h)	Coste (€/h)	Subtotal (€)
Análisis	40	35	1400
Programación	260	20	5200
TOTAL			6600 €

Tabla 3.3: Resumen de los costes humanos del proyecto.

En cuanto al uso de software en el proyecto se ha pretendido utilizar software gratuito durante todo el proyecto. Tanto el sistema gestor de la base de datos, como el IDE de programación como las librerías utilizadas lo son. Aunque la empresa también utilice JIRA, el coste de su licencia no se tiene en cuenta debido a que son 10 € si se aloja en servidor propio [4] y otros 10 € la licencia de Bitbucket [5], además se utilizará en todos los proyectos.

Finalmente como recurso software solo se utiliza el ordenador del entorno desarrollo el cual está valorado en unos 700 €. Dado que se ha utilizado en muchos proyectos, solo se aplicará un coste proporcional de unos 60 €.

La tabla 3.4 muestra los costes totales del proyecto que ascienden a 6.660 €.

Recursos	Subtotal (€)
Recursos humanos	6.600
Recursos software	0
Recursos hardware	60
TOTAL	6.660 €

Tabla 3.4: Resumen de los costes totales del proyecto.

Capítulo 4

Diseño del sistema

Contents

4.1. Diseño de la arquitectura de la API del motor de flujos	29
4.1.1. Modelo de datos	29
4.1.2. Arquitectura de la API del motor de flujos	31
4.2. Diseño de la arquitectura de la aplicación web de diseño de flujos de trabajo	32
4.2.1. Arquitectura de la aplicación web	32
4.2.2. Interfaz de la aplicación web	33

Dado que se trata de un proyecto realizado mediante metodología ágil, el diseño inicial del proyecto software que se va a realizar es muy básico. Este diseño inicial es refinado, modificado y perfeccionado a lo largo de los sprints.

Los apartados que vienen a continuación van a mostrar el diseño final que se ha obtenido tras realizar todos los sprints.

4.1. Diseño de la arquitectura de la API del motor de flujos

En este apartado se pretende explicar el diseño de la API del motor, explicando sus elementos principales.

4.1.1. Modelo de datos

Tras las iteraciones de los sprints se modificaron, ampliaron y refinaron los requisitos de datos resumidos a continuación:

- **Flujo de trabajo:** Un flujo de trabajo debe tener una **descripción** que permita reconocer su funcionalidad y un **administrador**, el cual podrá realizar funcionalidades adicionales. Por otro lado, dado que se trata de una entidad primaria, se desea que esta implemente el borrado lógico (**baja**) y una pequeña auditoría (**fecha y usuario de alta y de modificación**).
- **Tarea:** Una tarea debe tener una **descripción** que permita reconocer su funcionalidad y estar relacionada con el **flujo de trabajo** al que pertenece. Además, una tarea puede ser **inicial**, **final**, **condicional** o de **asignación dinámica**. Si es de asignación dinámica debe de tener la **clase** y el **método** al que se llamará para obtener al responsable. Por otro lado, dado que se trata de una entidad primaria, se desea que esta implemente el borrado lógico (**baja**) y una pequeña auditoría (**fecha y usuario de alta y de modificación**).
- **Transición:** Una transición debe tener una **descripción** que permita reconocer su funcionalidad. También debe estar relacionada con su **tarea de origen y destino**. Por último debe de poder tener la **clase** y **método** que se llamarán cuando se realice la transición (en caso de tener una función externa). Además debe de contener los **mensajes de éxito y error** a indicar en el histórico si se **desea notificar** la llamada en el histórico. Por otro lado, dado que se trata de una entidad primaria, se desea que esta implemente el borrado lógico (**baja**) y una pequeña auditoría (**fecha y usuario de alta y de modificación**).
- **Función condicional:** Una función condicional debe de tener un **orden** que le otorgue mayor o menor prioridad, la **clase** y el **método** externo que contienen la función condicional. Además debe de relacionarse con la **tarea** condicional a la que pertenece y con la **transición** que permitirá tomar en caso de devolver un resultado exitoso.
- **Permiso de flujo de trabajo:** Un permiso de flujo de trabajo debe estar asociado a un **flujo de trabajo** y garantizar a un **usuario** o **departamento** instanciar dicho flujo. Por otro lado, dado que se trata de una entidad primaria, se desea que esta implemente el borrado lógico (**baja**) y una pequeña auditoría (**fecha y usuario de alta y de modificación**).
- **Permiso de tarea:** Un permiso de tarea debe estar asociado a una **tarea** y garantizar a un **usuario** o **departamento** poder realizar cualquier instancia de dicha tarea. Además, debe de tener un **primer responsable** al cual se le asignarán en primer lugar. Por otro lado, dado que se trata de una entidad primaria, se desea que esta implemente el borrado lógico (**baja**) y una pequeña auditoría (**fecha y usuario de alta y de modificación**).
- **Instancia de flujo de trabajo:** Una instancia de flujo de trabajo debe estar relacionada con el **flujo de trabajo** al que se ha instanciado. Además debe de saberse qué **usuario la instanció**. También deben de saberse la **fecha en la que se instanció** y la **fecha en la que se finalizó** así como saber si ha sido **cancelada** y por qué **motivo**.
- **Instancia de tarea:** Una instancia de tarea debe estar relacionada con la **tarea** al que se ha instanciado y con la **instancia de flujo de trabajo** a la que pertenece. Además debe saberse quién es su **responsable**. También deben de saberse la **fecha en la que se instanció**, la **fecha en la que se modificó** y la **fecha en la que se finalizó** así como saber si ha sido **cancelada**.

- **Histórico:** El histórico debe de reflejar con una **descripción** cualquier acción realizada en una **instancia de flujo de trabajo** o en una **instancia de tarea**. También debe de saberse la **fecha en la que se realizó la acción**, **quien la realizó**, quien era el **responsable** y el **usuario o departamento destino** si se está realizando un cambio de responsable.

4.1.2. Arquitectura de la API del motor de flujos

La arquitectura de la API del motor es una arquitectura de tres niveles o tres capas como se puede observar en la figura 4.1:

1. La **capa de acceso** es la encargada de recibir las credenciales de los usuarios para hacer uso de la API del motor y de ofrecer el acceso a los diferentes módulos.
2. La **capa de la lógica de negocio** es la encargada de realizar todas las operaciones necesarias para que la API del motor ofrezca la funcionalidad deseada. Esta capa está dividida en cuatro **módulos** con diferentes responsabilidades:
 - **Módulo del motor:** es el módulo encargado de permitir a los usuarios instanciar un flujo de trabajo, así como realizar las tareas oportunas. Se accede a él mediante la capa de acceso.
 - **Módulo del diseñador:** es el módulo encargado de mantener las estructuras de los flujos de trabajo. Se accede a él mediante la capa de acceso para mantener las estructuras de los flujos de trabajo así como realizar consultas relacionadas con su estructura.
 - **Módulo de seguridad:** es el módulo encargado de mantener el sistema de permisos para tareas y flujos de trabajo. Es utilizado por el módulo del motor para comprobar si los usuarios tienen los permisos correspondientes a la acción que están realizando y para obtener los responsables de las nuevas tareas instanciadas. Se accede a él mediante la capa de acceso para realizar el mantenimiento de los permisos.
 - **Módulo de auditoría:** es el módulo encargado de mantener el histórico. Es utilizado por el módulo del motor para mantener la auditoría de todas las acciones realizadas durante un recorrido de un flujo de trabajo. Se accede a él mediante la capa de acceso para consultar el historial de las instancias de los flujos de trabajo.
3. La **capa de persistencia** es la encargada de comunicarse con la base de datos. Esta debe guardar, modificar, eliminar y buscar las entidades en la base de datos. Por ello esta capa esta formada por **entidades** y de **DAOs**. Las **entidades** son objetos que representan un elemento concreto (tarea, permiso de flujo de trabajo, etc) el cual es almacenado en una tabla de la base de datos. Los **DAOs** (Data Access Object) se encargan de realizar las operaciones CRUD (Create, Read, Update and Delete) sobre las entidades, así como de realizar búsquedas y operaciones más complejas que sean requeridas por la capa de la lógica de negocio.



Figura 4.1: Esquema de la arquitectura de la API del motor.

4.2. Diseño de la arquitectura de la aplicación web de diseño de flujos de trabajo

En este apartado se pretende explicar el diseño de la aplicación web de diseño, explicando sus elementos principales.

La aplicación web de diseño debe de permitir al diseñador de flujos de trabajo gestionar los flujos de trabajo y sus elementos (crearlos y modificarlos), así como gestionar los permisos. Para ello hará uso de los módulos de diseño y seguridad de la API del motor de flujos de trabajo.

4.2.1. Arquitectura de la aplicación web

Es un sistema cliente servidor. Por un lado el cliente web, el cual es el encargado de visualizar los datos oportunos y obtenerlos/modificarlos mediante llamadas al servidor. Por otro lado, el servidor es el encargado de ofrecer diferentes puntos de acceso que permitan utilizar las funcionalidades de la API del motor mediante servicios REST. En la figura 4.2 podemos observar un esquema básico de los diferentes elementos y su interacción.

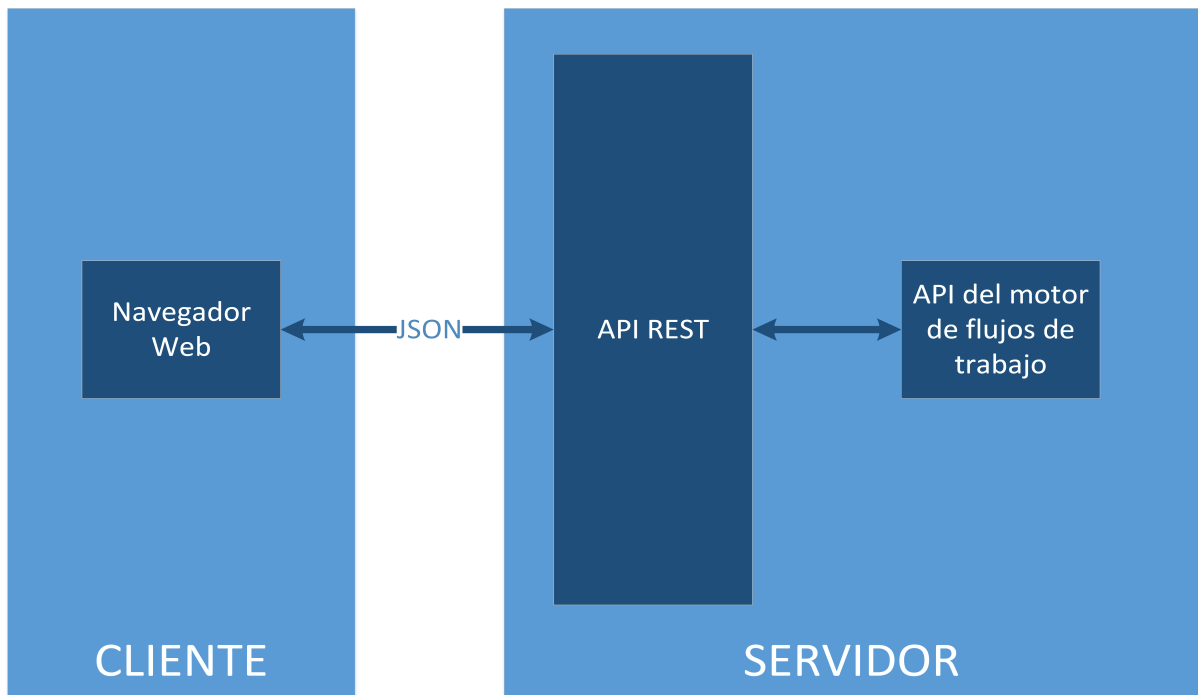


Figura 4.2: Esquema de la arquitectura de la aplicación web de diseño de flujos de trabajo.

La aplicación web de diseño de flujos de trabajo utiliza el patrón arquitectónico MVVM [7] (**Model-View-ViewModel**) en la parte del cliente. El principal motivo de usar este patrón es el hecho de utilizar AngularJS [6]. AngularJS permite implementar este patrón con gran facilidad.

En el patrón MVVM participan tres elementos:

- **Modelo:** El modelo es el encargado de contener los datos.
- **Vista:** El papel de la vista es el de representar la información gráficamente para el usuario.
- **Vista-Modelo (Modelo de vista):** El modelo de vista es el encargado de mantener la independencia entre el modelo y la vista. Este es el encargado de adaptar los datos del modelo para poder visualizarlos en la vista, además de realizar los cambios en el modelo y obtenerlos de este. Este se sirve del enlace de datos bidireccional para mantener los datos actualizados automáticamente entre él y la vista.

4.2.2. Interfaz de la aplicación web

Dado que se trata de una aplicación interna de la empresa y sus usuarios serán los propios desarrolladores, se ha priorizado en mostrar una interfaz consistente y que ofrezca las funcionalidades de forma directa y rápida, todo ello sin sobrecargar la memoria del usuario.

Interfaz gráfica

Dado que se trata de un proyecto ágil y la funcionalidad y uso de la interfaz gráfica están enfocados a ser usados por el equipo de desarrollo, se realizaron unos primeros esbozos de la interfaz gráfica durante una reunión de seguimiento con el analista. Estos esbozos se realizaron teniendo en cuenta las funcionalidades principales que debía ofrecer la aplicación web de diseño de flujos de trabajo.

En la figura 4.3 podemos observar el esbozo inicial de la que es la pantalla principal. Además en este esbozo también se puede observar la interacción que se produce al crear un nuevo flujo de trabajo.

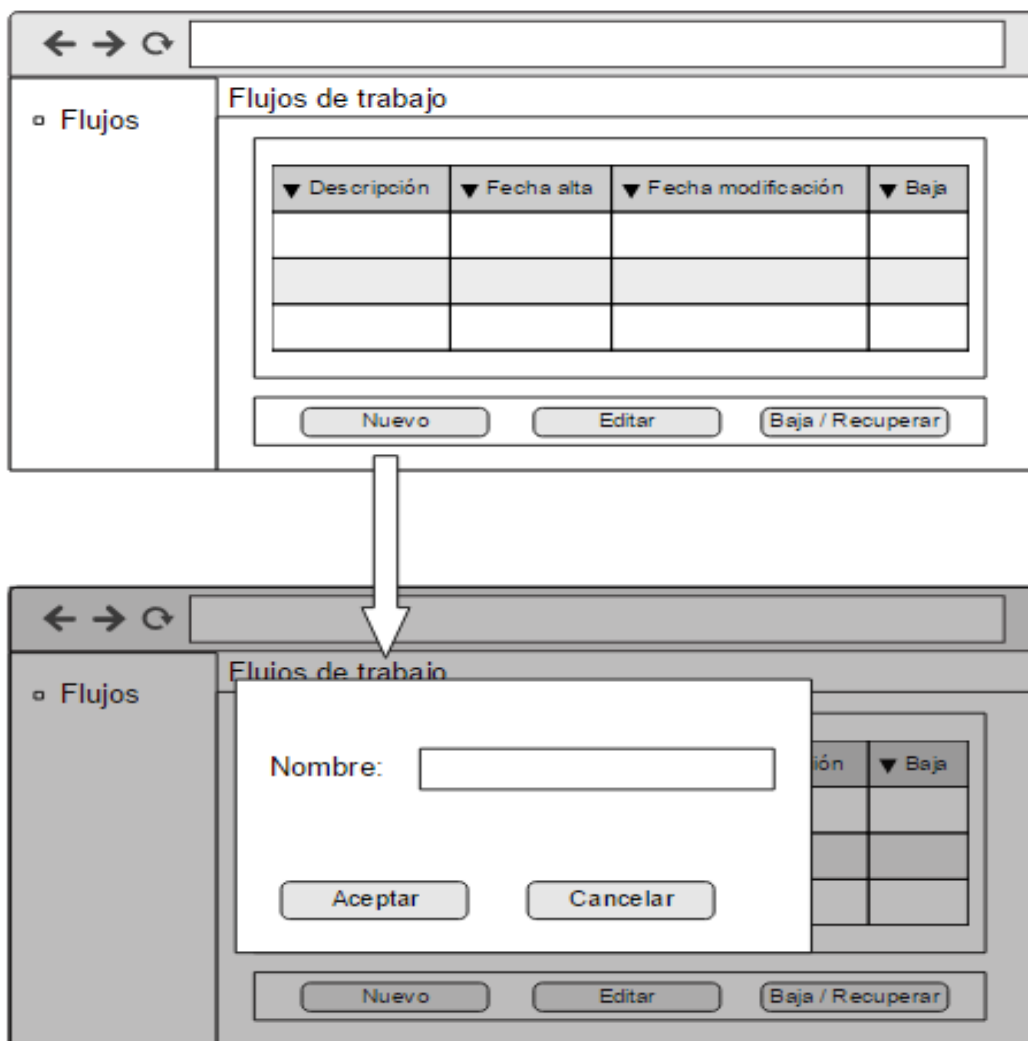


Figura 4.3: Esbozo de la lista de flujos de trabajo.

En la figura 4.4 se observa un primer esbozo de la forma en la que se representa la estructura de un flujo de trabajo en la interfaz gráfica.

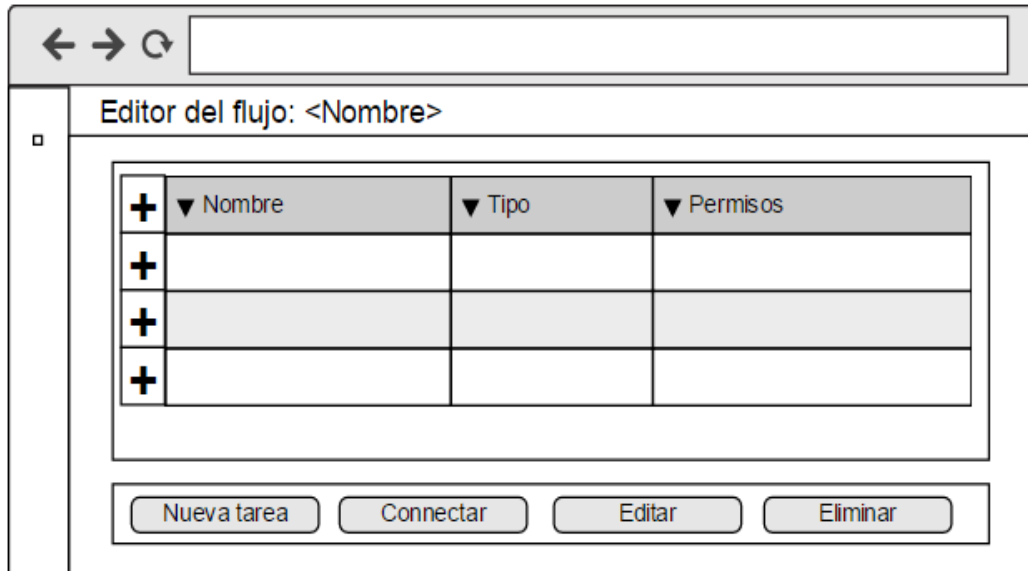


Figura 4.4: Esbozo de la pantalla del editor de un flujo de trabajo.

En cuanto a la gama de colores se ha decidido utilizar diferentes tonalidades de azul, dado que es el color ya usado en otras aplicaciones internas de la empresa.

En cuanto al modo de presentación de los datos se ha decidido utilizar tablas (y multitas) con paginación y búsqueda en todos los atributos. El objetivo de estos elementos es el de no sobrecargar la visión del usuario con demasiados datos y permitir una búsqueda rápida de los elementos.

Jerarquía de pantallas

La figura 4.5 muestra la jerarquía de pantallas de la aplicación. En ella podemos observar como se relacionan entre sí y el flujo de navegación que puede seguir el usuario.

Se ha decidido crear un menú principal para que en un futuro sea más fácil integrar nuevas funcionalidades de gestión.

Cabe mencionar que en la jerarquía de pantallas se han diferenciado las pantallas fijas de las ventanas modales utilizando diferentes representaciones. Para ello se han utilizado rectángulos y óvalos respectivamente.

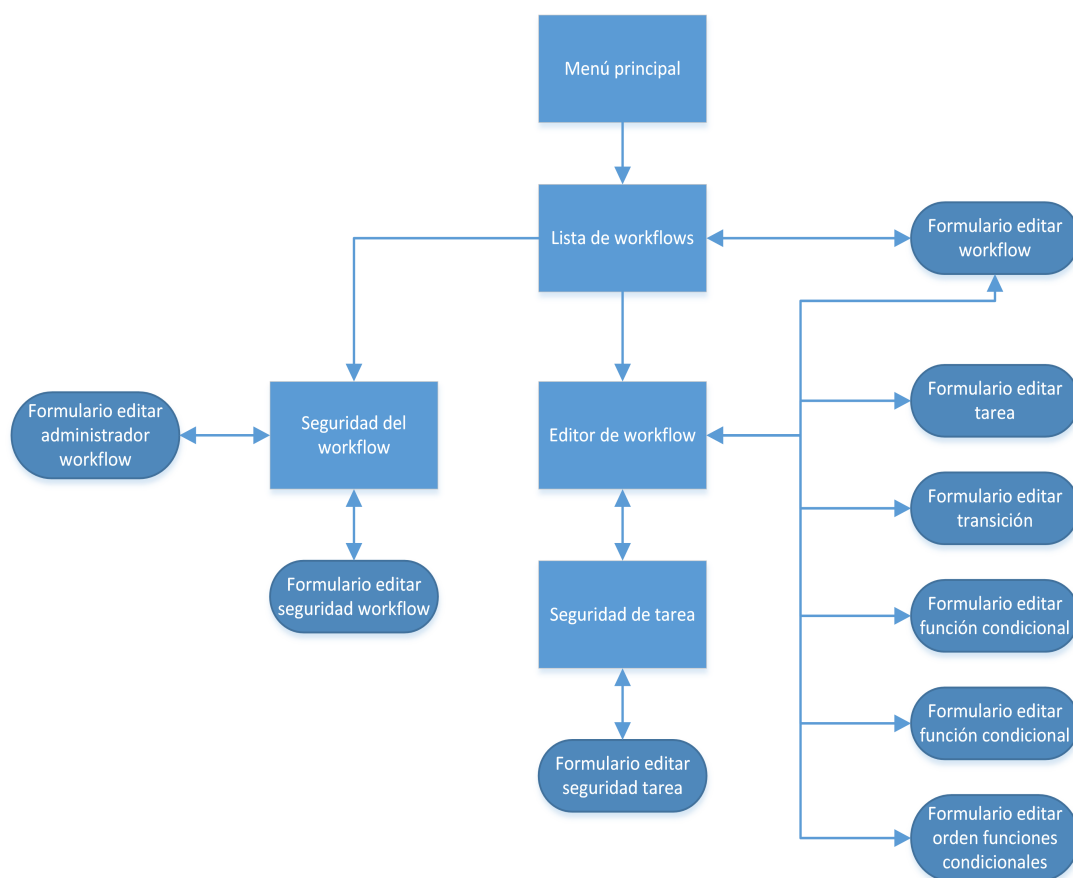


Figura 4.5: Jerarquía de pantallas de la aplicación web.

Capítulo 5

Detalles de la implementación

Contents

5.1. Tecnologías y herramientas	37
5.2. Control de versiones	39
5.3. Detalles de implementación de la API del motor de flujos de trabajo	40
5.3.1. Base de datos	40
5.3.2. Validación de datos	40
5.3.3. Llamadas a funciones externas	42
5.4. Detalles de implementación de la aplicación web de diseño de flujos de trabajo	43
5.4.1. Implementación del método PATCH	43
5.4.2. URLs de los servicios REST	43
5.4.3. Implementación del patrón MVVM con AngularJS	46
5.4.4. Implementación de la interfaz	47

A lo largo de este capítulo se van a explicar las tecnologías y herramientas utilizadas, así como algunos aspectos más técnicos relacionados con partes específicas del proyecto.

5.1. Tecnologías y herramientas

La API del motor de flujos de trabajo y la parte del servidor de la aplicación web de diseño de flujos de trabajo se han implementado usando el lenguaje de programación **Java**, concretamente con la versión 1.6. Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos [8]. Para facilitar la organización del proyecto se ha utilizado **Maven**, una herramienta de software para la gestión y construcción de proyectos [9].

Para la persistencia de los datos se ha utilizado el sistema gestor de bases de datos **MySQL**. Este es un sistema de gestión de bases de datos relacional, multihilo y multiusuario muy utilizado [10].

Para su desarrollo se han utilizado las siguientes librerías de terceros:

- **JUnit**: Librería que permite realizar pruebas unitarias de forma sencilla y agruparlas en baterías de pruebas [11].
- **Flexjson**: Librería ligera que permite serializar y deserializar de forma muy simple cualquier estructura de datos de Java en JSON. La principal diferencia de esta librería respecto a la mayoría de librerías JSON es el hecho de permitir de forma muy simple indicar el nivel de profundidad del serializado, permitiendo serializaciones tanto profundas como superficiales de los objetos, hasta incluso eligiendo el nivel de profundidad. [12].
- **Hibernate**: Librería que permite mapear objetos a una base de datos objeto relacional [13]. De esta forma la persistencia de los datos se convierte en una tarea fácil y sencilla.
- **Hibernate JPA**: Librería auxiliar de la citada anteriormente que permite utilizar las anotaciones de la API de Persistencia de Java (JPA) para ser usadas con Hibernate [13]. Gracias a esta librería los objetos que deben ser persistidos definen la estructura de la base de datos en su código sin necesidad de ningún fichero de configuración adicional.
- **MySQL Connector/J**: Driver oficial JDBC (Java Database Connectivity) para conectar con una base de datos de MySQL [14].
- **Log4j**: Librería altamente optimizada de 'logging' creada por el Apache Software Foundation Project. La principal ventaja de esta librería es el hecho de permitir redireccionar el 'log' de forma externa así como no afectar al rendimiento del código en caso de fallo [15].
- **REStEasy**: Librería que implementa la especificación JAX-RS 2.0 y que permite crear servicios RESTful en Java. Además esta librería está optimizada para funcionar en JBoss [16].

Cabe destacar que todas estas librerías poseen permisos de libre utilización y redistribución.

Para el desarrollo del cliente web se ha utilizado **HTML5**, **CSS** y **AngularJS** como controlador de la vista y para comunicarse con el servicio web. AngularJS es un potente framework de JavaScript que se utiliza para crear y mantener aplicaciones web de una sola página [17], concretamente se ha elegido la versión 1.5. Para facilitar la organización del proyecto se ha utilizado **Bower**, una herramienta de software para la gestión y construcción de proyectos JavaScript del lado del cliente [18]. Esta herramienta es muy similar a Maven, se indican los paquetes necesarios y ya se encarga de añadirlos al proyecto junto con las dependencias que necesite.

Para la parte del cliente se han utilizado los siguientes módulos de terceros, los cuales poseen permisos de libre utilización y redistribución:

- **Angular**: Módulo que contiene el framework de AngularJS.
- **AngularUI Router**: Módulo que amplía las funcionalidades de AngularJS permitiendo utilizar URLs diferentes en el cliente web sin realizar llamadas al servidor.
- **Bootstrap**: Módulo que permite realizar una interfaz de usuario que se adapte a diferentes dispositivos de forma fácil y rápida.

- **UI Bootstrap:** Módulo que amplía las funcionalidades de AngularJS permitiendo utilizar directivas para manipular diferentes elementos de Bootstrap.
- **RDash:** Módulo que permite crear fácilmente un menú de administración lateral, está basado en Bootstrap.
- **Angular UI Grid:** Módulo que permite crear y manipular tablas complejas mediante directivas de Angular y controladores.

Tanto para implementar la parte del servidor para la aplicación web de diseño como para albergar los servicios REST se ha decidido utilizar **JBoss**. JBoss es un servidor de aplicaciones Java EE de código abierto implementado en Java puro [19].

En cuanto al entorno de desarrollo cabe destacar que se ha utilizado **NetBeans** y **MySQL Workbench**. NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java [20]. MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL [21].

Para probar los servicios REST se ha utilizado **HttpRequester**. Este es un complemento para Firefox que permite realizar de forma fácil llamadas HTTP [22].

Finalmente se ha utilizado los servicios Jira y Bitbucket, ambos ofrecidos por Atlassian, los cuales están desplegados en un servidor local de la empresa. Estas herramientas han servido para organizar la planificación de la implementación y para mantener un control de versiones, respectivamente.

5.2. Control de versiones

El desarrollo del código de la API del motor de flujos y el de la aplicación web de diseño de flujos de trabajo se han realizado de forma incremental e iterativa. Debido a que durante el desarrollo se producen cambios en el código ya existente se ha utilizado una herramienta para el control de versiones. Bitbucket es un servicio de almacenamiento web, el cual permite almacenar proyectos que utilizan Mercurial y Git. Git es un software de control de versiones.

Durante el desarrollo del proyecto se han creado dos repositorios Git, uno para cada elemento principal del proyecto. En lugar de utilizar la estrategia de cuatro ramas (Master, Development, Features y Hotfix) se utilizaron solo las dos más importantes (Master y Development). Esto es debido a que por una parte el proyecto solo fue desarrollado por una persona y por otra la empresa acababa de empezar a utilizar Git y por lo tanto aún no se tenía un amplio conocimiento sobre como usar Git de forma profesional.

La estrategia realizada ha consistido en actualizar la rama Development cada vez que se finalizaba una historia de usuario o cuando se solucionaba algún conjunto de errores que se habían detectado. La rama Master solo se actualizaba cuando se terminaba la implementación de todas las historias de usuario relacionadas con cada elemento principal, la API del motor de flujos y la aplicación web de diseño de flujos de trabajo.

5.3. Detalles de implementación de la API del motor de flujos de trabajo

En este apartado se pretende explicar con más detalle algunos aspectos técnicos relacionados con la implementación de la API del motor de flujos.

5.3.1. Base de datos

Dado que la empresa ha utilizado siempre bases de datos relacionales, se ha decidido también utilizar una en este proyecto.

En la figura 5.1 podemos observar la estructura final de las tablas de la base de datos que utiliza la API del motor como persistencia de datos.

Esta estructura se obtuvo a través de varias mejoras y ampliaciones que se realizaron durante los sprints y tras analizar las entidades y como se relacionaban entre ellas.

Finalmente se ha decidido que la base de datos sea totalmente pasiva, por lo tanto todas las reglas de borrado y modificación en la base de datos son de tipo restringir. De esta forma se delega en el código la capacidad de eliminar elementos relacionados si fuera el caso.

5.3.2. Validación de datos

Dado que la API del motor puede ser utilizada por diversas aplicaciones, en todo momento se realiza una validación interna de todos los datos que recibe.

Las validaciones que realiza la API internamente son la siguientes:

- Los parámetros que acepta cualquier método de la API no pueden ser nulos.
- Los parámetros de tipo cadena de caracteres que acepta cualquier método de la API no pueden tener un tamaño superior a 300 caracteres si se trata de una cadena larga (usada para descripciones, clases, etc) o de 25 si se trata de una corta (usada para identificadores de usuario y departamento, etc).
- Los identificadores de usuario y departamento y las descripciones de los elementos que componen la estructura de un flujo de trabajo no pueden ser una cadena vacía.
- Cualquier identificador de un elemento de dentro de la API (flujo de trabajo, tarea, etc) que se pase mediante la llamada de un método, se comprobará que efectivamente ese elemento existe.
- Validar que se puede realizar la acción del método llamado dependiendo del identificador del usuario y departamento.

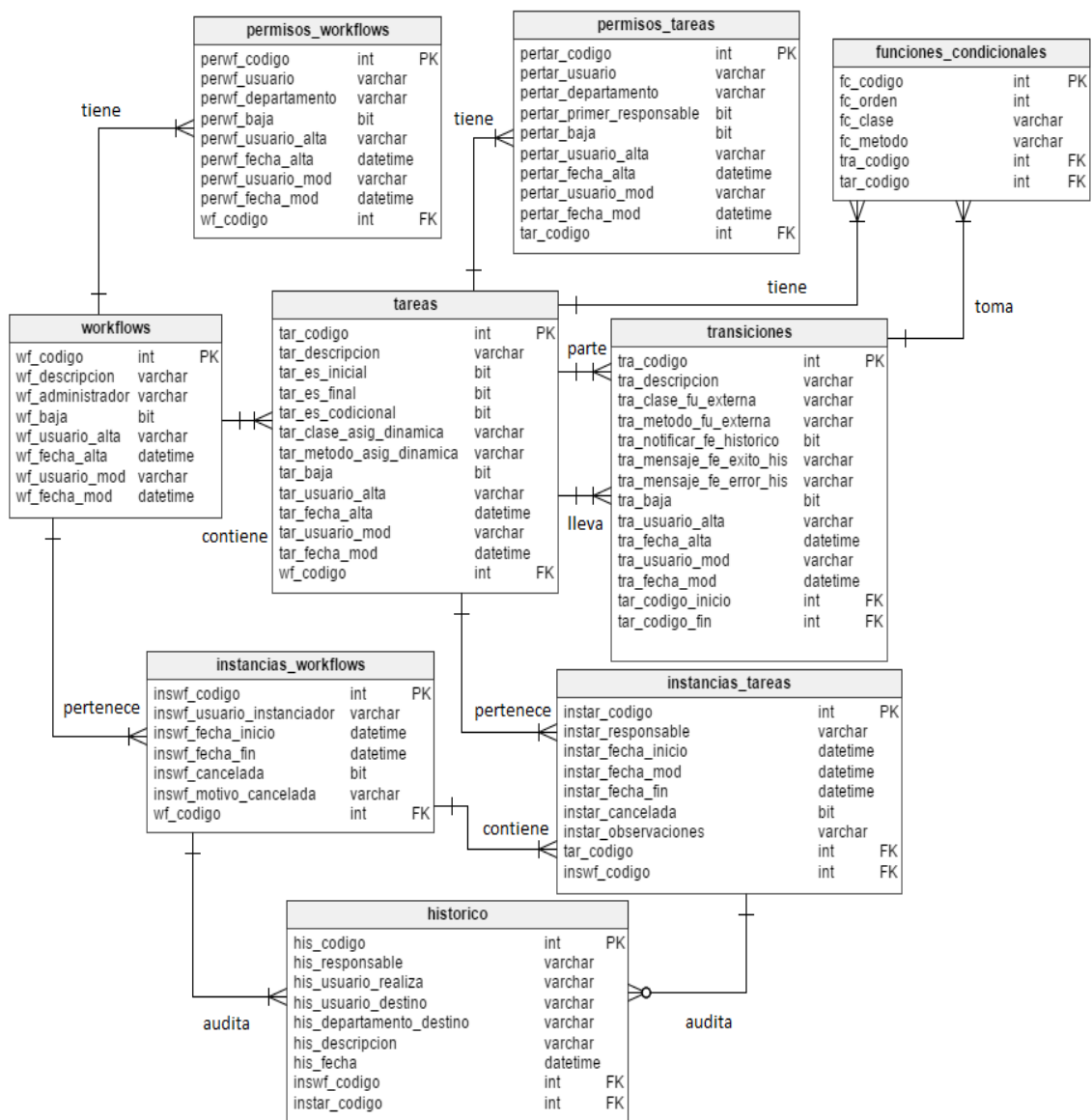


Figura 5.1: Estructura final de la base de datos.

5.3.3. Llamadas a funciones externas

Uno de los requisitos esenciales es el de poder gestionar funciones externas de diversa índole de forma uniforme sin que eso afecte a la implementación de la API del motor. Para ello se han creado tres interfaces (una para cada tipo de función externa). De esta forma la implementación quedaba uniforme desde el punto de vista de la API del motor. Esto permite implementar flujos de trabajo más complejos. Concretamente estas funciones externas pueden ser llamadas en tres ocasiones diferentes: para asignar un responsable a una instancia de tarea de forma dinámica, para comprobar si se puede realizar una transición o no y para saber que transición tomar cuando se instancia una tarea condicional (cabe recordar que estas tareas se finalizan automáticamente tomando la primera transición que puedan tomar). Para ello se han creado tres interfaces, una para cada tipo de función externa requerida para nuestra funcionalidad. En el apéndice A.3.3 se han desarrollado con detalle.

Por otro lado, la API del motor ha de ser capaz de llamar a esos métodos en tiempo de ejecución sin conocer directamente a las clases que implementaban estas interfaces. Para ello se ha decidido utilizar la reflexión que ofrece Java. La reflexión permite instanciar clases y llamar métodos entre otras funcionalidades a partir del nombre completo de la clase, el método y su firma. Para utilizar la reflexión en java se decidió crear la clase `InvocadorUtils`. En el código 5.1 ¹ se puede apreciar el método que invoca una función de una transición. Básicamente para usar la reflexión primero se obtiene la clase a partir de su nombre completo, luego se obtiene un objeto de dicha clase, a continuación se obtiene el método al que se desea llamar indicando los tipos de sus parámetros y finalmente se llama a ese método usando los propios parámetros. En el caso de las funciones que usa la API del motor, éstas siempre aceptan como parámetro de entrada un `String` con los datos pertinentes serializados en JSON.

Código 5.1: Método que invoca a cualquier clase que contiene una función externa de una transición.

```
public static boolean invocarFuncionTransicion
    (String clase, String metodo, String parametros) {

    Class[] params = new Class[1];
    params[0] = String.class;

    try {
        Class c = Class.forName(clase);
        Object ob = c.newInstance();
        Method m = c.getDeclaredMethod(metodo, params);
        return (Boolean) m.invoke(ob, parametros);
    } catch (Exception e) {
        return false;
    }
}
```

¹En el código se han omitido algunas líneas no relacionadas con la reflexión, como por ejemplo el logging en caso de excepción.

5.4. Detalles de implementación de la aplicación web de diseño de flujos de trabajo

En este apartado se pretende explicar con más detalle algunos aspectos técnicos relacionados con la implementación de la aplicación web de diseño de flujos de trabajo.

5.4.1. Implementación del método PATCH

Dado que la parte del servidor de la aplicación web de diseño de flujos de trabajo utiliza la API del motor embebida, uno de los principales problemas era adaptar las funcionalidades que ofrecía la API a las funcionalidades que se querían ofrecer en el diseñador.

Dado que la API del motor es accesible para el cliente mediante servicios REST, y ofrece bastantes métodos de modificación parcial de un recurso (por ejemplo cambiar la descripción de una tarea, indicar que es una tarea inicial, indicar que es una tarea final, indicar que es de asignación dinámica, etc) se ha decidido utilizar el método PATCH. Según el RFC-5789 (Request for Comments) [25] se debe de utilizar el método PATCH para modificaciones parciales y el método PUT para modificaciones de entidades completas. Asimismo, las llamadas con un método PATCH no solo deben de contener los datos a modificar sino también el conjunto de operaciones que deben de realizarse para modificar el recurso objetivo.

Para poder implementar el método PATCH correctamente por un lado se han definido una serie de identificadores de operación (CAMBIAR_DESCRIPCION, ESPECIFICAR_ES_INICIAL, etc) que son transmitidos desde el cliente al servicio REST objetivo dentro de un parámetro de tipo vector en el JSON. Por otro lado el servidor ya sabe como actuar ante cada uno de esas operaciones. Además, como la librería RESTEasy no ofrece una anotación directa para utilizar el método PATCH en un servidor, se ha tenido que crear (5.2). De esta forma ya se puede utilizar la anotación @PATCH de la misma forma que se utilizan las otras anotaciones.

Código 5.2: Definición de la anotación PATCH para Java.

```
@Target({ ElementType.METHOD, ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
@HttpMethod("PATCH")
@Documented
@NameBinding
public @interface PATCH {
}
```

5.4.2. URLs de los servicios REST

En las tablas 5.1 y 5.3 podemos observar todas las URLs que ofrecen los servicios REST así como los métodos que aceptan, parámetros que aceptan en el cuerpo de la petición y su finalidad.

URL	Método	Finalidad	Parámetros
/workflows	GET	Obtiene todos los flujos de trabajo (activos e inactivos)	-
/workflows	POST	Crea un nuevo flujo de trabajo	descripcion
/workflows/:id	PATCH	Modifica parcialmente un flujo de trabajo	op* descripcion
/workflows/:id/tareas	GET	Obtiene todas las tareas de un flujo (junto a las transiciones que parten de cada una de ellas)	-
/workflows/:id/tareas	POST	Crea una nueva tarea	op** descripcion claseAd
/workflows/:id/tareas/:id	GET	Obtiene los datos de una tarea	-
/workflows/:id/tareas/:id	PATCH	Modifica parcialmente una tarea	op** descripcion claseAd
/workflows/:id/tareas/:id	DELETE	Elimina una tarea	-
/workflows/:id/tareas/:id/transiciones	GET	Obtiene todas las transiciones que parten de dicha tarea	-
/workflows/:id/tareas/:id/transiciones	POST	Crea una nueva transición	op*** descripcion tarCodigoFin clase mensajeExito mensajeFracaso claseCondional
/workflows/:id/tareas/:id/transiciones	PATCH	Modifica parcialmente una transición	op*** descripcion tarCodigoFin clase mensajeExito mensajeFracaso claseCondional
/workflows/:id/tareas/:id/transiciones/:id	DELETE	Elimina una transición	-
/workflows/:id/tareas/:id/funcionesCondicionales	GET	Obtiene todas las funciones condicionales asociadas a una tarea condicional	-
/workflows/:id/tareas/:id/funcionesCondicionales	PATCH	Modifica parcialmente una función condicional	op**** orden claseCondional otraFuncion

Tabla 5.1: URLs de los servicios REST relacionados con la estructura de un flujo de trabajo.

Cabe destacar que el parámetro `op` indica las operaciones oportunas a realizar. Este parámetro permite realizar distintas llamadas a la API del motor dentro del mismo método del servicio

REST para así poder definir completamente el recurso.

En la tabla 5.2 se muestran todas las operaciones que acepta el parámetro op en cada recurso.

Parámetro	Valores
op* [workflows]	CAMBIAR_DESCRIPCION CANCELAR REACTIVAR DUPLICAR
op** [tareas]	CAMBIAR_DESCRIPCION ESPECIFICAR_ES_INICIAL ESPECIFICAR_ES_FINAL ESPECIFICAR_ES_DINAMICA CANCELAR_ES_DINAMICA ESPECIFICAR_ES_CONDICIONAL
op*** [transiciones]	CAMBIAR_DESCRIPCION ESPECIFICAR_FUNCION_EXTERNA ESPECIFICAR_FUNCION_EXTERNA_AUDITADA CANCELAR_FUNCION_EXTERNA ESPECIFICAR_FUNCION_CONDICIONAL
op**** [funciones condicionales]	MODIFICAR CAMBIAR_ORDEN

Tabla 5.2: Lista de los valores que pueden ir en el parámetro op.

URL	Método	Finalidad	Parámetros
/workflows/:id/permisos	GET	Obtiene todos los permisos para poder instanciar un flujo	-
/workflows/:id/permisos	POST	Crea un nuevo permiso para instanciar un flujo	esUsuario nombre
/workflows/:id/permisos	DELETE	Elimina un permiso para instanciar un flujo	-
/workflows/:id/permisos/administrador	POST	Indica un nuevo administrador del flujo de trabajo	admin
/tareas/:id/permisos	GET	Obtiene todos los permisos para poder realizar una tarea	-
/tareas/:id/permisos	POST	Crea un nuevo permiso para poder realizar una tarea	esUsuario nombre
/tareas/:id/permisos	DELETE	Elimina un permiso para poder realizar una tarea	-

Tabla 5.3: URLs de los servicios REST relacionados con la seguridad de un flujo de trabajo y sus componentes.

5.4.3. Implementación del patrón MVVM con AngularJS

En el capítulo anterior se ha descrito el patrón MVVM. En este apartado se va a explicar como implementarlo adaptado a AngularJS.

AngularJS permite crear aplicaciones de una página. Para ello cuenta con varios elementos principales.

Por un lado están las plantillas. Las plantillas son documentos HTML que son inyectados en la página principal de la aplicación. Estas plantillas poseen anotaciones especiales para indicar donde tiene que inyectar Angular los datos y si debe de realizar alguna operación (usar un filtro sobre los datos, llamar a una operación, etc). Por lo tanto las plantillas son las vistas del patrón MVVM.

Por otro lado, cuando se inyecta una plantilla, se indica un controlador que es inyectado junto a ella. El controlador es el encargado de preprocesar los datos (si es necesario), proporcionárselos a la plantilla y de realizar las operaciones que la plantilla le indica (por ejemplo cuando se pulsa un botón). La plantilla y el controlador se comunican a través del \$scope. Todo lo que se defina dentro de este elemento es compartido entre la plantilla y el controlador, y por lo tanto si uno de los dos realiza un cambio en ese elemento el otro lo percibe. De este modo los controladores son vista-modelo y el \$scope funciona como doble enlace de datos.

Finalmente, en Angular se pueden definir servicios. Los servicios son elementos que permiten organizar y compartir funcionalidad en varias partes de la aplicación. En este caso se ha creado un conjunto de servicios que atacan a los servicios REST del servidor. De esta forma se puede percibir a los servicios como parte del modelo del patrón MVVM, siendo la otra parte los servicios REST.

En la figura 5.2 se puede apreciar el patrón MVVM superpuesto a los elementos comentados anteriormente.

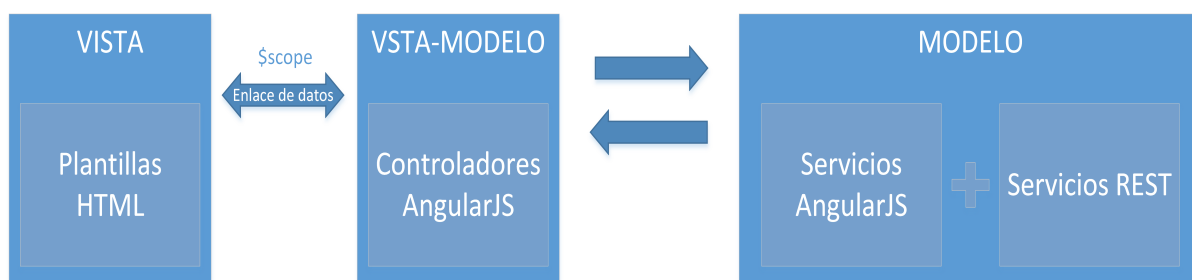
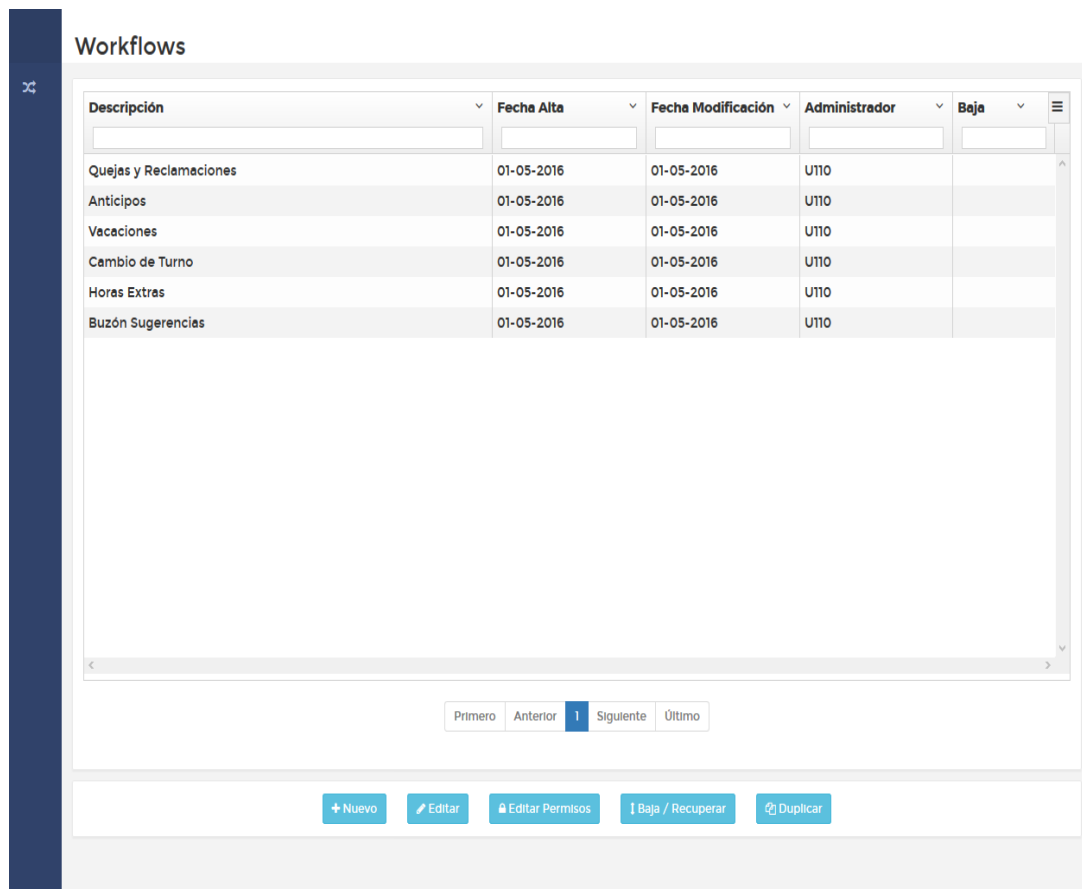


Figura 5.2: Adaptación del patrón MVVM a AngularJS.

5.4.4. Implementación de la interfaz

Tras finalizar la implementación del cliente y realizar los cambios oportunos para refinar el diseño de la aplicación, en la figura 5.3 se observa el resultado final de la lista de flujos de trabajo y en la figura 5.4 la pantalla que permite la edición de la estructura del flujo de trabajo.

Dado que la interfaz es web, se ha utilizado HTML, CSS y JavaScript para realizarla, concretamente se ha utilizado el framework AngularJS, el cual ya se ha explicado en el apartado anterior.



Descripción	Fecha Alta	Fecha Modificación	Administrador	Baja
Quejas y Reclamaciones	01-05-2016	01-05-2016	U110	
Anticipos	01-05-2016	01-05-2016	U110	
Vacaciones	01-05-2016	01-05-2016	U110	
Cambio de Turno	01-05-2016	01-05-2016	U110	
Horas Extras	01-05-2016	01-05-2016	U110	
Buzón Sugerencias	01-05-2016	01-05-2016	U110	

Primerο Anterior 1 Siguiente Último

+ Nuevo Editar Editar Permisos Baja / Recuperar Duplicar

Figura 5.3: Pantalla de la lista de flujos de trabajo.

Editor de workflow: Anticipos

Descripción	Tipo	Obligatoria	Permisos
Rellenar Solicitud	Inicial		
Tarea Destino	Transición	Función Externa	Orden
Autorizar Solicitud	Firmar		
FIN	Cancelar		
Autorizar Solicitud			
Tarea Destino	Transición	Función Externa	Orden
Rellenar Solicitud	Revisar		
Comunicar Denegación	Rechazar		
Comunicar Aprobación	Firmar		
TAREA_IGNORAR.			
Comunicar Denegación	Final		
Comunicar Aprobación	Final		
FIN	Final		

Primero Anterior 1 Siguiete Último

[Renombrar Workflow](#)
[+ Nueva Tarea](#)
[✎ Editar Tarea](#)
[🔒 Editar Permisos](#)
[🔗 Conectar Destino](#)
[✎ Editar Transición](#)
[🗑 Eliminar Tarea](#)

Figura 5.4: Pantalla de edición de la estructura de un flujo de trabajo.

Capítulo 6

Desarrollo de la implementación

Contents

6.1. Sprint 1	50
6.1.1. Planificación del sprint	50
6.1.2. Resultados obtenidos	56
6.1.3. Pila del producto actualizada	57
6.2. Sprint 2	58
6.2.1. Planificación del sprint	58
6.2.2. Resultados obtenidos	63
6.2.3. Pila del producto actualizada	64
6.3. Sprint 3	65
6.3.1. Planificación del sprint	65
6.3.2. Resultados obtenidos	68
6.3.3. Pila del producto actualizada	69
6.4. Sprint 4	71
6.4.1. Planificación del sprint	71
6.4.2. Resultados obtenidos	77
6.4.3. Pila del producto actualizada	77
6.5. Sprint 5	79
6.5.1. Planificación del sprint	79
6.5.2. Resultados obtenidos	82
6.5.3. Pila del producto actualizada	82

A lo largo de este capítulo se va a realizar un seguimiento de los sprints realizados para implementar el proyecto. En ellos se describirá el estado de la pila del producto al iniciar el sprint, las historias de usuario elegidas para realizarlas en ese sprint y los resultados obtenidos al finalizar el sprint.

Al inicio del sprint se definen las tareas a realizar y los criterios de aceptación de las historias elegidas.

6.1. Sprint 1

6.1.1. Planificación del sprint

A continuación se muestran las historias de usuario elegidas para elaborar la pila del sprint 1. Junto a las historias se han definido una serie de tareas a realizar. Además se han especificado los criterios de aceptación que se deben cumplir para poder considerar que la historia de usuario ha sido finalizada.

HU01 **Crear y modificar flujos** [M]

Como diseñador de flujos de trabajo quiero poder definir flujos de trabajo para poder definir uno de los elementos básicos de un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar los flujos de datos.
- T2 Crear el DAO para manipular flujos de trabajo.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder manipular flujos de trabajo mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se llama a la API para crear un nuevo flujo de trabajo con una descripción válida ENTONCES este es creado y se devuelve un mensaje de éxito con los datos del flujo creado.
- C2 CUANDO se llama a la API para crear un nuevo flujo de trabajo con una descripción inválida ENTONCES este no es creado y se devuelve un mensaje de error indicando que la descripción es inválida.
- C3 DADO que existe un flujo de trabajo CUANDO se llama a la API para modificar la descripción de dicho flujo con una descripción válida ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicho flujo.
- C4 DADO que existe un flujo de trabajo CUANDO se llama a la API para modificar la descripción de dicho flujo con una descripción inválida ENTONCES este no es modificado y se devuelve un mensaje de error indicando que la descripción es inválida.
- C5 DADO que existe un flujo de trabajo CUANDO se llama a la API para indicar la baja de dicho flujo (que ya no es operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con los datos de dicho flujo.
- C6 DADO que existe un flujo de trabajo CUANDO se llama a la API para indicar la alta de dicho flujo (que ya vuelve a ser operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con los datos de dicho flujo.
- C7 DADO que no existe un flujo de trabajo CUANDO se llama a la API para realizar alguna acción sobre este ENTONCES se devuelve un mensaje de error indicando que no existe.

HU02 **Crear y modificar tareas** [M]

Como diseñador de flujos de trabajo quiero poder definir tareas para poder definir uno de los elementos básicos de un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar las tareas.
- T2 Crear el DAO para manipular tareas.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder manipular tareas mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que no existe un flujo de trabajo CUANDO se llama a la API para crear una tarea asociada a dicho flujo ENTONCES esta no es creada y se devuelve un mensaje de error indicando que el flujo no existe.
- C2 DADO que existe un flujo de trabajo CUANDO se llama a la API para crear una tarea asociada a dicho flujo con una descripción inválida ENTONCES esta no es creada y se devuelve un mensaje de error indicando que la descripción es inválida.
- C3 DADO que existe un flujo de trabajo CUANDO se llama a la API para crear una tarea asociada a dicho flujo con una descripción válida ENTONCES esta es creada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C4 DADO que existe una tarea CUANDO se llama a la API para modificar la descripción de dicha tarea con una descripción inválida ENTONCES esta no es modificada y se devuelve un mensaje de error indicando que la descripción es inválida.
- C5 DADO que existe una tarea CUANDO se llama a la API para modificar su la descripción de dicha tarea con una descripción válida ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C6 DADO que existe una tarea CUANDO se llama a la API para indicar que dicha tarea es una tarea inicial ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C7 DADO que existe una tarea CUANDO se llama a la API para indicar que es una tarea final ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C8 DADO que existe una tarea CUANDO se llama a la API para indicar la baja de dicha tarea (que ya no es operativa) ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C9 DADO que existe una tarea CUANDO se llama a la API para indicar la alta de dicha tarea (que ya vuelve a ser operativa) ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C10 DADO que no existe una tarea CUANDO se llama a la API para realizar alguna acción sobre dicha tarea ENTONCES se devuelve un mensaje de error indicando que no existe.

HU03 **Crear y modificar transiciones** [M]

Como diseñador de flujos de trabajo quiero poder definir transiciones entre tareas para poder definir los diferentes caminos que puede tener un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar las transiciones.
- T2 Crear el DAO para manipular transiciones.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder manipular transiciones mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que no existe una tarea origen o una destino CUANDO se llama a la API para crear una transición asociada a dichas tareas ENTONCES esta no es creada y se devuelve un mensaje de error indicando que una de las tareas no existe.
- C2 DADO que existe una tarea origen y una destino CUANDO se llama a la API para crear una transición asociada a dichas tareas con una descripción inválida ENTONCES esta no es creada y se devuelve un mensaje de error indicando que la descripción es inválida.
- C3 DADO que existe una tarea origen y una destino CUANDO se llama a la API para crear una transición asociada a dichas tareas con una descripción inválida ENTONCES esta es creada y se devuelve un mensaje de éxito con los datos de dicha transición.
- C4 DADO que existe una transición CUANDO se llama a la API para modificar la descripción de esta con una descripción inválida ENTONCES esta no es modificada y se devuelve un mensaje de error indicando que la descripción es inválida.
- C5 DADO que existe una transición CUANDO se llama a la API para modificar la descripción de esta con una descripción válida ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha transición.
- C6 DADO que existe una transición CUANDO se llama a la API para indicar la baja de dicha transición (que ya no es operativa) ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha transición.
- C7 DADO que existe una transición CUANDO se llama a la API para indicar la alta de dicha transición (que ya vuelve a ser operativa) ENTONCES esta es modificada y se devuelve un mensaje de éxito con los datos de dicha transición.
- C8 DADO que no existe una transición CUANDO se llama a la API para realizar alguna acción sobre esta ENTONCES se devuelve un mensaje de error indicando que no existe.

HU04 **Consulta de flujos de trabajo** [M]

Como sistema quiero poder conocer los flujos de trabajo definidos y las tareas que lo componen para poder presentar en la aplicación las opciones necesarias.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar los DAOs de flujos de trabajo, tareas y transiciones para poder realizar dichas consultas.
- T2 Ampliar las pruebas de los DAOs para comprobar su correcto funcionamiento.
- T3 Implementar la lógica necesaria para poder obtener los datos de interés mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que no existe un flujo de trabajo CUANDO se llama a la API para consultar los datos de este ENTONCES se devuelve un mensaje de error indicando que no existe.
- C2 DADO que existe un flujo de trabajo CUANDO se llama a la API para consultar los datos de este ENTONCES se devuelve un mensaje de éxito con los datos del flujo de trabajo consultado.
- C3 DADO que existe un flujo de trabajo CUANDO se llama a la API para consultar las tareas de dicho flujo ENTONCES se devuelve un mensaje de éxito con los datos de sus tareas.
- C4 DADO que no existe una tarea CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de error indicando que no existe.
- C5 DADO que existe una tarea CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de éxito con sus datos.
- C6 DADO que existe una tarea CUANDO se llama a la API para consultar las transiciones que parten de ella ENTONCES se devuelve un mensaje de éxito con los datos de las transiciones operativas.
- C7 DADO que existe una tarea CUANDO se llama a la API para consultar todas las transiciones que parten de ella ENTONCES se devuelve un mensaje de éxito con los datos de todas las transiciones (operativas y no operativas).
- C8 DADO que no existe una transición CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de error indicando que no existe.
- C9 DADO que existe una transición CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de éxito con sus datos.

HU05 **Instanciar un flujo de trabajo** [M]

Como usuario quiero poder instanciar un flujo de trabajo para que se realicen los tramites oportunos.

A continuación se describen las **tareas** de esta historia:

T1 Crear la tabla en la base de datos para almacenar las instancias de flujos de trabajo.

T2 Crear el DAO para manipular instancias de flujos de trabajo.

T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.

T4 Implementar la lógica necesaria para poder instanciar flujos de trabajo mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

C1 DADO que no existe un flujo de trabajo CUANDO se llama a la API para instanciarlo ENTONCES la instancia del flujo de trabajo no es creada y se devuelve un mensaje de error indicando que no existe dicho flujo de trabajo.

C2 DADO que existe un flujo de trabajo CUANDO se llama a la API para instanciarlo ENTONCES la instancia es creada, se le asigna como propietario de esta al usuario instanciador y se devuelve un mensaje de éxito con los datos de la instancia.

HU06 **Consultar una instancia de un flujo de trabajo** [M]

Como usuario quiero poder consultar el estado de una instancia de un flujo de trabajo para ver en que estado se encuentra.

A continuación se describen las **tareas** de esta historia:

T1 Ampliar el DAO de instancias de flujos de trabajo para poder realizar las consultas oportunas.

T2 Ampliar las pruebas del DAO para comprobar su correcto funcionamiento.

T3 Implementar la lógica necesaria para poder consultar las instancias de flujos de trabajo mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

C1 DADO que no existe una instancia de flujo de trabajo CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de error indicando que no existe dicha instancia.

C2 DADO que existe una instancia de flujo de trabajo CUANDO se llama a la API para consultar sus datos ENTONCES se devuelve un mensaje de éxito con los datos de dicha instancia.

HU07 Sistema de permisos de flujos [M]

Como diseñador de flujos de trabajo quiero poder definir la seguridad relacionada con un flujo de trabajo para asegurar que solo los usuarios autorizados (bien sean ellos mismos o bien su departamento) puedan instanciar un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar los permisos para instanciar flujos de trabajo.
- T2 Crear el DAO para manipular permisos para instanciar flujos de trabajo.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder manipular permisos para instanciar flujos de trabajo mediante llamadas a la API.
- T5 Ampliar la lógica interna de la API para consultar dichos permisos a la hora de instanciar un flujo de trabajo.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe un flujo de trabajo CUANDO se llama a la API para crear un permiso para un usuario sobre dicho flujo ENTONCES se devuelve un mensaje de éxito con los datos del permiso creado.
- C2 DADO que existe un flujo de trabajo CUANDO se llama a la API para crear un permiso para un departamento sobre dicho flujo ENTONCES se devuelve un mensaje de éxito con los datos del permiso creado.
- C3 DADO que no existe un flujo de trabajo CUANDO se llama a la API para crear un permiso sobre dicho flujo ENTONCES se devuelve un mensaje de error indicando que el flujo no existe.
- C4 DADO que existen permisos para instanciar un flujo de trabajo CUANDO se llama a la API para obtener los permisos de un usuario ENTONCES se devuelve un mensaje de éxito con los datos de los permisos.
- C5 DADO que existen permisos para instanciar un flujo de trabajo CUANDO se llama a la API para obtener los permisos de un departamento ENTONCES se devuelve un mensaje de éxito con los datos de los permisos.
- C6 DADO que existen permisos para instanciar un flujo de trabajo CUANDO se llama a la API para obtener los permisos de un flujo ENTONCES se devuelve un mensaje de éxito con los datos de los permisos.
- C7 DADO que existe un permiso para instanciar un flujo de trabajo CUANDO se llama a la API para indicar la baja de dicho permiso (que ya no es operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con sus datos.

- C8 DADO que existe un permiso para instanciar un flujo de trabajo CUANDO se llama a la API para indicar la alta de dicho permiso (que ya vuelve a ser operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con sus datos.
- C9 DADO que existe un flujo de trabajo y un permiso para instanciar un flujo de trabajo para el usuario (operativo) CUANDO se llama a la API para instanciar un flujo de trabajo usando las credenciales oportunas ENTONCES se devuelve un mensaje de éxito con los datos de la instancia del flujo de trabajo creada.
- C10 DADO que existe un flujo de trabajo y un permiso para instanciar un flujo de trabajo para el departamento (operativo) CUANDO se llama a la API para instanciar un flujo de trabajo usando las credenciales oportunas ENTONCES se devuelve un mensaje de éxito con los datos de la instancia del flujo de trabajo creada.
- C11 DADO que existe un flujo de trabajo sin ningún permiso para instanciar un flujo de trabajo para el usuario o departamento (operativo) CUANDO se llama a la API para instanciar un flujo de trabajo usando las credenciales oportunas ENTONCES se devuelve un mensaje de error indicando que no tiene permisos.

6.1.2. Resultados obtenidos

La realización del sprint se ha llevado a cabo durante el tiempo estimado inicialmente (del 19/02/2016 al 07/03/2016).

Durante este sprint se han realizado todas las historias de usuario elegidas con éxito (pasando todos los criterios de aceptación).

La estimación temporal ha sido sido correcta a excepción de dos historias de usuario.

Por un lado, realizar la historia de usuario HU05 (Instanciar un flujo de trabajo) ha resultado ser más fácil de realizar que lo que se estimó inicialmente. Esto se debe principalmente a que cuando se definió, el concepto de instancia de flujo de trabajo aún no estaba claro del todo.

Por otro lado, realizar la historia de usuario HU02 (Crear y modificar tareas) ha resultado ser más costosa de lo estimado inicialmente. Este sobrecoste temporal ha sido causado por el hecho de tener diferentes tipos de tareas, concepto que también estaba un poco difuso cuando se definió la historia de usuario.

Estas desviaciones no han afectado a la estimación temporal del sprint debido a que se anulan entre ellas.

6.1.3. Pila del producto actualizada

En la tabla 6.1 podemos observar el estado de la pila del producto tras finalizar el sprint 1. Como podemos ver, en ella aparecen todas las historias de usuario elegidas en el sprint 1 como finalizadas debido a que se acertó en la estimación temporal conjunta de estas.

Además se han realizado varias modificaciones a la pila del producto. Se han añadido las historias HU17 (Instanciar automáticamente la tarea inicial) y la HU18 (Finalizar una tarea).

Estas modificaciones han sido realizadas debido a la necesidad de utilizar instancias de tarea en las instancias de flujo en lugar de usar tareas directamente, ya que esto reducía la flexibilidad del flujo de trabajo.

Por último se ha añadido la HU19 (Designar administrador de un flujo) debido a que ahora los flujos necesitan un administrador para poder supervisar sus instancias.

ID	Historia de usuario	Coste estimado (PH)	Coste real (PH)	Sprints	Estado
HU01	Crear y modificar flujos [M]	1	1	SP1	FINALIZADA
HU02	Crear y modificar tareas [M]	1	2	SP1	FINALIZADA
HU03	Crear y modificar transiciones [M]	1	1	SP1	FINALIZADA
HU04	Consulta de flujos de trabajo [M]	1	1	SP1	FINALIZADA
HU05	Instanciar un flujo de trabajo [M]	3	2	SP1	FINALIZADA
HU06	Consultar una instancia de flujo de trabajo [M]	1	1	SP1	FINALIZADA
HU07	Sistema de permisos de flujos [M]	2	2	SP1	FINALIZADA
HU08	Sistema de permisos de tareas [M]	2			
HU09	Traspaso de tareas [M]	2			
HU10	Histórico [M]	2			
HU11	Cancelar una instancia de flujo de trabajo [M]	1			
HU12	Llamadas a métodos externos [M]	2			
HU13	Asignación dinámica de tareas [M]	1			
HU14	Tareas condicionales [M]	5			
HU15	Interfaz gráfica web de edición de flujos [W]	10			
HU16	Servicios REST de edición de flujos [W]	10			
HU17	Instanciar automáticamente la tarea inicial [M]	1			
HU18	Finalizar una tarea [M]	2			
HU19	Designar administrador de un flujo [M]	1			

Tabla 6.1: Estado de la pila del producto tras finalizar el sprint 1.

6.2. Sprint 2

6.2.1. Planificación del sprint

A continuación se muestran las historias de usuario elegidas para elaborar la pila del sprint 2. Junto a las historias se han definido una serie de tareas a realizar. Además se han especificado los criterios de aceptación que se deben cumplir para poder considerar que la historia de usuario ha sido finalizada.

HU17 **Instanciar automáticamente la tarea inicial** [M]

Como usuario quiero que se instancie automáticamente la tarea inicial del flujo de trabajo que he instanciado para poder realizar acciones sobre él.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar las instancias de tareas.
- T2 Crear el DAO para manipular instancias de tarea.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Ampliar la lógica interna de la API para que se instancie automáticamente la tarea inicial de un flujo de trabajo cuando este es instanciado.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe un flujo de trabajo con una tarea inicial y se poseen permisos para poder instanciarlo CUANDO se llama a la API para instanciar dicho flujo ENTONCES este es instanciado, se instancia la tarea inicial y se asigna como responsable de la tarea al usuario instanciador del flujo.
- C2 DADO que existe un flujo de trabajo sin tarea inicial y se poseen permisos CUANDO se llama a la API para instanciar dicho flujo ENTONCES se devuelve un mensaje de error indicando que la estructura del flujo de trabajo es incorrecta.

HU08 **Sistema de permisos de tareas** [M]

Como diseñador de flujos de trabajo quiero poder definir qué usuarios o departamentos son los responsables de realizar una tarea, así como definir un primer responsable (persona o departamento a la que se le asignará la tarea al instanciarse).

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar los permisos para realizar una instancia de tarea.
- T2 Crear el DAO para manipular permisos para realizar una instancia de tarea.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder manipular permisos para realizar instancias de tarea mediante llamadas a la API.
- T5 Ampliar la lógica interna de la API para consultar dichos permisos a la hora de instanciar una tarea y asignarle responsable.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una tarea CUANDO se llama a la API para crear un permiso para realizar las instancias de dicha tarea para un usuario ENTONCES este es creado y se devuelve un mensaje de éxito con los datos del permiso.
- C2 DADO que existe una tarea CUANDO se llama a la API para crear un permiso para realizar las instancias de dicha tarea para un departamento ENTONCES este es creado y se devuelve un mensaje de éxito con los datos del permiso.
- C3 DADO que existe una tarea CUANDO se llama a la API para crear un permiso de primer responsable de tarea para las instancias de dicha tarea para un usuario ENTONCES este es creado, se degrada el anterior permiso de primer responsable a permiso normal y se devuelve un mensaje de éxito con los datos del permiso creado.
- C4 DADO que existe una tarea CUANDO se llama a la API para crear un permiso de primer responsable de tarea para las instancias de dicha tarea para un departamento ENTONCES este es creado, se degrada el anterior permiso de primer responsable a permiso normal y se devuelve un mensaje de éxito con los datos del permiso creado.
- C5 DADO que no existe una tarea CUANDO se llama a la API para crear un permiso relacionado con dicha tarea ENTONCES se devuelve un mensaje de error indicando que la tarea no existe.
- C6 DADO que existe un permiso para realizar instancias de una tarea CUANDO se llama a la API para indicar la baja de dicho permiso (que ya no es operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con sus datos.
- C7 DADO que existe un permiso para realizar instancias de una tarea CUANDO se llama a la API para indicar la alta de dicho permiso (que ya vuelve a ser operativo) ENTONCES este es modificado y se devuelve un mensaje de éxito con sus datos.
- C8 DADO que existen permisos para realizar instancias de una tarea CUANDO se realiza una instancia de esa tarea ENTONCES se asigna como responsable de la nueva instancia de tarea al usuario o departamento indicado como primer responsable en dichos permisos.
- C9 DADO que existen permisos para realizar instancias de una tarea y ninguno de ellos es de primer responsable CUANDO se realiza una instancia de esa tarea ENTONCES se asigna como responsable de la nueva instancia de tarea a algún departamento con autorización de forma aleatoria.
- C10 DADO que existen permisos para realizar instancias de una tarea, ninguno de ellos es de primer responsable y todos son de usuarios CUANDO se realiza una instancia de esa tarea ENTONCES se asigna como responsable a algún usuario con autorización de forma aleatoria.
- C11 DADO que no existen permisos para realizar instancias de una tarea CUANDO se realiza una instancia de esa tarea ENTONCES se asigna como responsable al usuario instanciador del flujo al que pertenece dicha instancia de tarea.

HU18 **Finalizar una tarea** [M]

Como usuario responsable de una instancia de tarea quiero poder finalizarla para así continuar con el flujo.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar el DAO de instancias de tarea para poder finalizar instancias de tarea.
- T2 Ampliar las pruebas del DAO para comprobar su correcto funcionamiento.
- T3 Implementar la lógica necesaria para poder finalizar instancias de tarea mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una instancia de tarea no finalizada CUANDO se llama a la API para finalizar dicha instancia indicando la transición a tomar siendo el responsable de esta ENTONCES se finaliza la instancia de tarea actual y se instancia la tarea destino de la transición.
- C2 DADO que existe una instancia de tarea no finalizada CUANDO se llama a la API para finalizar dicha instancia indicando la transición a tomar no siendo el responsable de esta ENTONCES se devuelve un mensaje de error indicando que no se poseen permisos suficientes.
- C3 DADO que existe una instancia de tarea no finalizada CUANDO se llama a la API para finalizar dicha instancia indicando una transición inválida (no existe o no está operativa) ENTONCES se devuelve un mensaje de error indicando que no se puede tomar dicha transición.
- C4 DADO que existe una instancia de tarea finalizada CUANDO se llama a la API para finalizar dicha instancia ENTONCES se devuelve un mensaje de error indicando que no se puede realizar dicha acción.

HU19 **Designar administrador de un flujo** [M]

Como diseñador de flujos de trabajo quiero poder designar un administrador de flujo de trabajo para que sea el responsable de este.

A continuación se describen las **tareas** de esta historia:

- T1 Modificar la tabla de flujos de trabajo para poder almacenar el administrador.
- T2 Implementar la lógica necesaria para poder gestionar el administrador de un flujo de trabajo mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se llama a la API para crear un flujo de trabajo ENTONCES se pone como administrador al usuario creador de dicho flujo y se envía un mensaje de éxito.
- C2 DADO que existe un flujo de trabajo CUANDO se llama a la API para indicar un nuevo administrador de dicho flujo ENTONCES se reemplaza el antiguo administrador por el nuevo y se envía un mensaje de éxito.
- C3 DADO que no existe un flujo de trabajo CUANDO se llama a la API para indicar un nuevo administrador de dicho flujo ENTONCES se envía un mensaje de error indicando que el flujo no existe.

HU09 Traspaso de tareas [M]

Como usuario quiero poder traspasar las tareas que tenga asignadas o estén asignadas a mi departamento a otros usuarios o departamentos que tengan permiso para así poder repartir mejor el trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar el DAO de permisos para realizar instancias de tareas.
- T2 Implementar la lógica necesaria para poder consultar los departamentos y usuarios a los que se les puede traspasar una instancia de tarea mediante llamadas a la API.
- T3 Implementar la lógica necesaria para poder traspasar una instancia de tarea mediante llamadas a la API.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una instancia de tarea CUANDO se llama a la API para obtener los usuarios a los que se puede asignar dicha instancia de tarea ENTONCES se devuelve una lista de los usuarios autorizados a excepción del propio responsable si se trata de un usuario.
- C2 DADO que existe una instancia de tarea CUANDO se llama a la API para obtener los departamentos a los que se puede asignar dicha instancia de tarea ENTONCES se devuelve una lista de los departamentos autorizados a excepción del propio responsable si se trata de un usuario.
- C3 DADO que no existe una instancia de tarea CUANDO se llama a la API para obtener los departamentos o usuarios a los que se puede asignar dicha instancia de tarea ENTONCES se devuelve un mensaje de error indicando que la instancia de tarea no existe.
- C4 DADO que existe una instancia de tarea asignada a un usuario CUANDO el usuario llama a la API para reasignar la instancia de tarea a otro usuario con permisos ENTONCES se reasigna la instancia de tarea y se envía un mensaje de éxito.
- C5 DADO que existe una instancia de tarea asignada a un usuario CUANDO el usuario llama a la API para reasignar la instancia de tarea a otro usuario sin permisos ENTONCES se envía un mensaje de error indicando que el usuario objetivo no tiene permiso.
- C6 DADO que existe una instancia de tarea asignada a un usuario CUANDO el usuario llama a la API para reasignar la instancia de tarea a un departamento con permisos ENTONCES se reasigna la instancia de tarea y se envía un mensaje de éxito.
- C7 DADO que existe una instancia de tarea asignada a un usuario CUANDO el usuario llama a la API para reasignar la instancia de tarea a un departamento sin permisos ENTONCES se envía un mensaje de error indicando que el departamento objetivo no tiene permiso.
- C8 DADO que existe una instancia de tarea asignada a un departamento CUANDO un usuario de este departamento llama a la API para reasignar la instancia de tarea a otro usuario con permisos ENTONCES se reasigna la instancia de tarea y se envía un mensaje de éxito.

- C9 DADO que existe una instancia de tarea asignada a un departamento CUANDO un usuario de este departamento llama a la API para reasignar la instancia de tarea a otro usuario sin permisos ENTONCES se envía un mensaje de error indicando que el usuario objetivo no tiene permiso.
- C10 DADO que existe una instancia de tarea asignada a un departamento CUANDO un usuario de este departamento llama a la API para reasignar la instancia de tarea a otro departamento con permisos ENTONCES se reasigna la instancia de tarea y se envía un mensaje de éxito.
- C11 DADO que existe una instancia de tarea asignada a un departamento CUANDO un usuario de este departamento llama a la API para reasignar la instancia de tarea a otro departamento sin permisos ENTONCES se envía un mensaje de error indicando que el departamento objetivo no tiene permiso.
- C12 DADO que existe una instancia de tarea asignada a un departamento CUANDO un usuario de otro departamento llama a la API para reasignar la instancia de tarea a otro departamento o usuario ENTONCES se envía un mensaje de error indicando que no tiene permiso.
- C13 DADO que existe una instancia de tarea asignada a un usuario CUANDO otro usuario llama a la API para reasignar la instancia de tarea a otro departamento o usuario ENTONCES se envía un mensaje de error indicando que no tiene permiso.

HU11 **Cancelar una instancia de flujo de trabajo [M]**

Como administrador de un flujo de trabajo quiero poder cancelar una de sus instancias y todas sus instancias de tarea para poder cancelar flujos de los que nadie se hace cargo o que se han creado erróneamente.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar la tabla de la base de datos que representa las instancias de flujos de trabajo para que quede registrada su cancelación.
- T2 Ampliar el DAO de instancias de flujos de trabajo para poder cancelarlos.
- T3 Ampliar el DAO de flujos de trabajo para consultar quien es su administrador.
- T4 Implementar la lógica necesaria para poder cancelar una instancia de flujo de trabajo mediante llamadas a la API.
- T5 Ampliar la lógica interna de la API para que compruebe que solo el administrador puede cancelar instancias de flujo de trabajo no finalizadas.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una instancia de flujo de trabajo CUANDO el administrador del flujo llama a la API para cancelarlo ENTONCES este es cancelado junto con todas sus instancias de tarea activas y se envía un mensaje de éxito.
- C2 DADO que existe una instancia de flujo de trabajo CUANDO otro usuario que no es el administrador del flujo llama a la API para cancelarlo ENTONCES se envía un mensaje de error indicando que no tiene permisos de administrador.
- C3 DADO que no existe una instancia de flujo de trabajo CUANDO se llama a la API para cancelarlo ENTONCES se envía un mensaje de error indicando que dicha instancia no existe.

HU10 **Histórico** [M]

Como usuario quiero poder consultar todo el histórico de ejecución de una instancia de flujo de trabajo para poder conocer todas las acciones que se han llevado a cabo en una instancia de flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear la tabla en la base de datos para almacenar las entradas del histórico.
- T2 Crear el DAO para manipular entradas del histórico.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder consultar y añadir entradas al histórico mediante llamadas a la API.
- T5 Ampliar la lógica interna de la API para que cualquier acción en relacionada con un flujo de trabajo quede reflejada en el histórico.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se instancia un flujo de trabajo ENTONCES se añade una entrada al histórico.
- C2 CUANDO se finaliza un flujo de trabajo ENTONCES se añade una entrada al histórico.
- C3 CUANDO se instancia una tarea ENTONCES se añade una entrada al histórico.
- C4 CUANDO se reasigna una instancia de tarea ENTONCES se añade una entrada al histórico.
- C5 CUANDO se finaliza una instancia de tarea ENTONCES se añade una entrada al histórico.
- C6 DADO que existe una instancia de flujo de trabajo CUANDO un usuario asignado a una instancia de tarea no finalizada llama a la API para añadir una entrada al histórico ENTONCES se añade una entrada al histórico y se devuelve un mensaje de éxito.
- C7 DADO que existe una instancia de flujo de trabajo CUANDO un usuario llama a la API para consultar el histórico ENTONCES se devuelve un mensaje de éxito y los datos del histórico de dicha instancia.

6.2.2. Resultados obtenidos

La realización del sprint se ha llevado a cabo durante el tiempo estimado inicialmente (del 08/03/2016 al 21/03/2016).

Durante este sprint se han realizado todas las historias de usuario elegidas con éxito (pasando todos los criterios de aceptación). La causa fundamental de haber podido realizar todas las historias durante el sprint ha sido que el tiempo estimado para la HU09 (Traspaso de tareas) ha tenido un coste real de 1 PH.

6.2.3. Pila del producto actualizada

En la tabla 6.2 podemos observar el estado de la pila del producto tras finalizar el sprint 2. Como podemos ver, en ella aparecen todas las historias de usuario elegidas en el sprint 2.

Por último se ha decidido añadir la historia de usuario HU20 (Documentar la API). Este requisito ha sido demandado por la empresa debido a que se pretende integrar este motor en futuros proyectos y necesitan una documentación que sirva de ayuda a los programadores.

Además se ha renombrado y redefinido la HU12. Esto se debe a que el antiguo nombre y descripción no eran del todo descriptivos. Ahora se llama “Indicar funciones externas en transiciones”.

ID	Historia de usuario	Coste estimado (PH)	Coste real (PH)	Sprints	Estado
HU01	Crear y modificar flujos [M]	1	1	SP1	FINALIZADA
HU02	Crear y modificar tareas [M]	1	2	SP1	FINALIZADA
HU03	Crear y modificar transiciones [M]	1	1	SP1	FINALIZADA
HU04	Consulta de flujos de trabajo [M]	1	1	SP1	FINALIZADA
HU05	Instanciar un flujo de trabajo [M]	3	2	SP1	FINALIZADA
HU06	Consultar una instancia de flujo de trabajo [M]	1	1	SP1	FINALIZADA
HU07	Sistema de permisos de flujos [M]	2	2	SP1	FINALIZADA
HU08	Sistema de permisos de tareas [M]	2	2	SP2	FINALIZADA
HU09	Traspaso de tareas [M]	2	1	SP2	FINALIZADA
HU10	Histórico [M]	2	2	SP2	FINALIZADA
HU11	Cancelar una instancia de flujo de trabajo [M]	1	1	SP2	FINALIZADA
HU12	Indicar funciones externas en transiciones [M]	2			
HU13	Asignación dinámica de tareas [M]	1			
HU14	Tareas condicionales [M]	5			
HU15	Interfaz gráfica web de edición de flujos [W]	10			
HU16	Servicios REST de edición de flujos [W]	10			
HU17	Instanciar automáticamente la tarea inicial [M]	1	1	SP2	FINALIZADA
HU18	Finalizar una tarea [M]	2	2	SP2	FINALIZADA
HU19	Designar administrador de un flujo [M]	1	1	SP2	FINALIZADA
HU20	Documentar la API [M]	2			

Tabla 6.2: Estado de la pila del producto tras finalizar el sprint 2.

6.3. Sprint 3

6.3.1. Planificación del sprint

A continuación se muestran las historias de usuario elegidas para elaborar la pila del sprint 3. Durante este sprint se pretende terminar todas las historias de usuario relacionadas con la API del motor. Junto a las historias se han definido una serie de tareas a realizar. Además se han especificado los criterios de aceptación que se deben cumplir para poder considerar que la historia de usuario ha sido finalizada.

HU12 **Indicar funciones externas en transiciones** [M]

Como diseñador de flujos de trabajo quiero poder indicar que una instancia de tarea solo podrá ser finalizada si la transición indicada determina si se puede finalizar o no en tiempo de ejecución (en caso de tener una función externa asociada).

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar la tabla de la base de datos que almacena las transiciones para almacenar la clase y el método al que llamar en caso de tener función externa.
- T2 Ampliar el DAO de transiciones para poder gestionar una función externa relacionada con la transición.
- T3 Ampliar las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder definir y cancelar funciones externas de transiciones mediante llamadas a la API.
- T5 Modificar la lógica interna de la API para que cuando se llame a la API para finalizar una instancia de tarea, se llame a la función externa de la transición (si existe).

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una transición CUANDO se llama a la API para indicar una función externa válida asignada a dicha transición ENTONCES se modifica la transición para añadirle la función externa y se devuelve un mensaje de éxito con los datos de la transición.
- C2 DADO que existe una transición CUANDO se llama a la API para indicar una función externa inválida ENTONCES se devolverá un mensaje de error indicando que la función externa no existe o no implementa la interfaz requerida.
- C3 DADO que existe una transición con una función externa CUANDO se llama a la API para indicar que ya no posee función externa ENTONCES se modifica y se devuelve un mensaje de éxito con los datos de la transición.
- C4 DADO que no existe una transición CUANDO se llama a la API para indicar una función externa ENTONCES se devuelve un mensaje de error indicando que la transición no existe.
- C5 DADO que existe una transición con una función externa CUANDO se llama a la API para finalizar una instancia de tarea tomando dicha transición ENTONCES se llamara a dicha función y dependiendo de la respuesta de dicha función se finalizará o no la instancia de tarea.

HU13 **Asignación dinámica de tareas** [M]

Como diseñador de flujos de trabajo quiero poder indicar que una tarea es de asignación dinámica para así poder determinar su responsable en tiempo de ejecución y hacer mucho más flexible y potente la definición de flujos de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar la tabla de la base de datos que almacena las tareas para almacenar la clase y el método al que llamar en caso de ser de asignación dinámica.
- T2 Ampliar el DAO de tareas para poder gestionar una función de asignación dinámica relacionada con la tarea.
- T3 Ampliar las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder definir y cancelar funciones de asignación dinámica de tareas mediante llamadas a la API.
- T5 Modificar la lógica interna de la API para que cuando se llame a la API para finalizar una instancia de tarea, se compruebe que la nueva tarea instanciada sea de asignación dinámica, y si es así entonces obtener su responsable llamando a dicha función.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una tarea CUANDO se llama a la API para indicar que es de asignación dinámica indicándole una función válida ENTONCES se actualiza y se devuelve un mensaje de éxito con los datos de dicha tarea.
- C2 DADO que existe una tarea CUANDO se llama a la API para indicar que es de asignación dinámica indicándole una función inválida ENTONCES se devuelve un mensaje de error indicando que la función objetivo no existe o no implementa la interfaz requerida.
- C3 DADO que no existe una tarea CUANDO se llama a la API para indicar que es de asignación dinámica ENTONCES se devuelve un mensaje de error indicando que no existe dicha tarea.
- C4 DADO que existe una tarea de asignación dinámica CUANDO se llama a la API para finalizar una instancia de tarea y esta se finaliza correctamente instanciando la siguiente tarea ENTONCES se llama a la función externa de asignación para conocer su responsable.
- C5 DADO que existe una tarea de asignación dinámica CUANDO se llama a la API para finalizar una instancia de tarea y esta se finaliza correctamente instanciando la siguiente tarea y esta devuelve una respuesta inválida ENTONCES se asigna un responsable mediante los permisos para realizar instancias de tareas.

HU14 **Tareas condicionales** [M]

Como diseñador de flujos de trabajo quiero poder definir tareas condicionales para que el flujo de trabajo siga un camino u otro en función del resultado de las condiciones.

A continuación se describen las **tareas** de esta historia:

- T1 Crear en la base de datos la tabla que almacenará las funciones condicionales.
- T2 Crear el DAO encargado de gestionar las funciones condicionales.
- T3 Crear las pruebas del DAO para comprobar su correcto funcionamiento.
- T4 Implementar la lógica necesaria para poder definir y cancelar funciones condicionales mediante llamadas a la API.
- T5 Implementar la lógica necesaria para poder consultar funciones condicionales de una tarea mediante llamadas a la API.
- T6 Modificar la lógica interna de la API para que cuando se llame a la API para finalizar una instancia de tarea, se compruebe que la nueva tarea instanciada sea condicional, y si es así finalizarla automáticamente tomando la transición indicada por las funciones condicionales (y así sucesivamente si hay más de una condicional enlazada).

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que existe una tarea CUANDO se llama a la API para indicar que es condicional ENTONCES se modifica y se devuelve un mensaje de éxito con los datos de la tarea.
- C2 DADO que existe una tarea condicional con transiciones CUANDO se llama a la API para indicar una función condicional válida relacionada con una transición ENTONCES se modifica y se devuelve un mensaje de éxito con los datos de la función condicional.
- C3 DADO que existe una tarea condicional con transiciones CUANDO se llama a la API para indicar una función condicional inválida relacionada con una transición ENTONCES se devuelve un mensaje de error indicando que la función no existe o no implementa la interfaz requerida.
- C4 DADO que existe una tarea condicional con transiciones con funciones condicionales CUANDO se llama a la API para consultar sus funciones condicionales ENTONCES se devuelve un mensaje de éxito con los datos de estas.
- C5 DADO que existe una tarea condicional con transiciones con funciones condicionales CUANDO se llama a la API para intercambiar la prioridad de dos de sus funciones condicionales ENTONCES se intercambian la prioridad y se devuelve un mensaje de éxito.
- C6 DADO que existe una tarea condicional con transiciones con funciones condicionales CUANDO se llama a la API para indicar que una transición ya no es condicional ENTONCES se modifica y se devuelve un mensaje de éxito.
- C7 DADO que existe una tarea condicional con transiciones con funciones condicionales CUANDO se llama a la API para finalizar una instancia de tarea y esta se finaliza correctamente instanciando la siguiente tarea que es de tipo condicional ENTONCES esta es finalizada automáticamente tomando la primera transición válida entre las transiciones con funciones condicionales ordenadas por prioridad.

HU20 Documentar la API [M]

Como programador que va a integrar el motor en un nuevo proyecto quiero poder entender fácilmente como comunicarme con la API del motor y saber que funcionalidades me ofrece para así poder desarrollar mi trabajo de forma más fácil.

A continuación se describen las **tareas** de esta historia:

- T1 Crear y/o ampliar el Javadoc de todos los métodos que son accesibles desde fuera del motor.
- T2 Crear una documentación complementaria en el README del proyecto donde se expliquen los métodos y las estructuras JSON que devuelven.

6.3.2. Resultados obtenidos

La realización del sprint se ha llevado a cabo durante el tiempo estimado inicialmente (del 22/03/2016 al 07/04/2016).

Durante este sprint se han realizado todas las historias de usuario elegidas con éxito (pasando todos los criterios de aceptación).

Cabe destacar que la planificación temporal de dos historias de usuario fue errónea. Por un lado la historia de usuario HU20 (Documentar la API) resulto ser más costosa de lo estimado previamente necesitando 4PH para poder realizarla. Esto ha sido causado principalmente al hecho de tener que revisar la mayor parte del código para poder hacer una documentación correcta.

Por otro lado la implementación de la historia de usuario HU14 (Tareas condicionales) resulto costar menos de lo estimado inicialmente, quedando terminada con 3PH. Inicialmente se estimó en 5PH debido a que era una de las funcionalidades de la API menos clara al principio del proyecto.

Al terminar este sprint se ha terminado con éxito la implementación del motor.

6.3.3. Pila del producto actualizada

En la tabla 6.3 podemos observar el estado de la pila del producto tras finalizar el sprint 3. Como podemos ver, en ella aparecen todas las historias de usuario relacionadas con el motor de flujo de trabajo finalizadas.

Se ha decidido eliminar las historias de usuario HU15 y HU16 y en su lugar definir las necesarias para desarrollar la aplicación web de diseño de flujos de trabajo. Además se ha tomado la decisión de desarrollar los servicios REST al mismo tiempo que la interfaz de la aplicación para así ofrecer exactamente los recursos requeridos por este.

Concretamente se han definido las siguientes historias de usuario, las cuales definen toda la funcionalidad que la **aplicación web para el diseño de flujos de trabajo** debe ofrecer:

- HU21: Consultar los flujos de trabajo existentes.
- HU22: Crear, modificar y dar de baja/alta un flujo de trabajo.
- HU23: Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos.
- HU24: Consultar la estructura de un flujo.
- HU25: Consultar los permisos de una tarea, crearlos y eliminarlos.
- HU26: Crear una tarea completa, modificarla y eliminarla.
- HU27: Crear una transición completa, modificarla y eliminarla.
- HU28: Permitir intercambiar el orden de las funciones condicionales.
- HU29: Permitir modificar funciones condicionales.

ID	Historia de usuario	Coste estimado (PH)	Coste real (PH)	Sprints	Estado
HU01	Crear y modificar flujos [M]	1	1	SP1	FINALIZADA
HU02	Crear y modificar tareas [M]	1	2	SP1	FINALIZADA
HU03	Crear y modificar transiciones [M]	1	1	SP1	FINALIZADA
HU04	Consulta de flujos de trabajo [M]	1	1	SP1	FINALIZADA
HU05	Instanciar un flujo de trabajo [M]	3	2	SP1	FINALIZADA
HU06	Consultar una instancia de flujo de trabajo [M]	1	1	SP1	FINALIZADA
HU07	Sistema de permisos de flujos [M]	2	2	SP1	FINALIZADA
HU08	Sistema de permisos de tareas [M]	2	2	SP2	FINALIZADA
HU09	Traspaso de tareas [M]	2	1	SP2	FINALIZADA
HU10	Histórico [M]	2	2	SP2	FINALIZADA
HU11	Cancelar una instancia de flujo de trabajo [M]	1	1	SP2	FINALIZADA
HU12	Indicar funciones externas en transiciones [M]	2	2	SP3	FINALIZADA
HU13	Asignación dinámica de tareas [M]	1	1	SP3	FINALIZADA
HU14	Tareas condicionales [M]	5	3	SP3	FINALIZADA
HU15	Interfaz gráfica web de edición de flujos [W]	10			
HU16	Servicios REST de edición de flujos [W]	10			
HU17	Instanciar automáticamente la tarea inicial [M]	1	1	SP2	FINALIZADA
HU18	Finalizar una tarea [M]	2	2	SP2	FINALIZADA
HU19	Designar administrador de un flujo [M]	1	1	SP2	FINALIZADA
HU20	Documentar la API [M]	2	4	SP3	FINALIZADA
HU21	Consultar los flujos de trabajo existentes [W]	1			
HU22	Crear, modificar y dar de baja/alta un flujo de trabajo [W]	1			
HU23	Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos [W]	2			
HU24	Consultar la estructura de un flujo [W]	2			
HU25	Consultar los permisos de una tarea, crearlos y eliminarlos [W]	2			
HU26	Crear una tarea completa, modificarla y eliminarla [W]	3			
HU27	Crear una transición completa, modificarla y eliminarla [W]	2			
HU28	Permitir intercambiar el orden de las funciones condicionales [W]	1			
HU29	Permitir modificar funciones condicionales [W]	1			

Tabla 6.3: Estado de la pila del producto tras finalizar el sprint 3.

6.4. Sprint 4

6.4.1. Planificación del sprint

A continuación se muestran las historias de usuario elegidas para elaborar la pila del sprint 4. Junto a las historias se han definido una serie de tareas a realizar. Además se han especificado los criterios de aceptación que se deben cumplir para poder considerar que la historia de usuario ha sido finalizada.

HU21 Consultar los flujos de trabajo existentes [W]

Como diseñador de flujos de trabajo quiero poder ver todos los datos importantes de los flujos de trabajo existentes (nombre, fechas, administrador y si están activos), así como poder buscar por uno o varios de esos datos para así facilitar el mantenimiento de estos.

A continuación se describen las **tareas** de esta historia:

- T1 Crear el servicio REST en el servidor que responda a las peticiones relacionadas con flujos de trabajo.
- T2 Crear el método en el servicio REST que responda a la petición de obtener todos los flujos de trabajo haciendo uso de la API del motor.
- T3 Crear el servicio en el cliente que se encargue de realizar las llamadas al servicio REST de flujos de trabajo.
- T4 Crear la pantalla de presentación con una tabla que permita filtrar por todos los campos.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se consulta la lista de flujos de trabajo ENTONCES debe aparecer una lista de todos los flujos de trabajo con sus características (descripción, si está activo o no, su fecha de creación y modificación y quién es su administrador).
- C2 DADO que se consulta la lista de flujos de trabajo CUANDO se filtran por cualquiera de sus campos ENTONCES solo deben aparecer aquellos que concuerden con los datos filtrados.

HU22 **Crear, modificar y dar de baja/alta un flujo de trabajo [W]**

Como diseñador de flujos de trabajo quiero poder crear flujos de trabajo, modificar su descripción y activarlos y desactivarlos para así poder tener control absoluto sobre estos.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar el servicio REST de flujos de trabajo con un método para poder crearlos haciendo uso de la API.
- T2 Ampliar el servicio REST de flujos de trabajo con un método para poder actualizarlos haciendo uso de la API.
- T3 Ampliar el servicio en el cliente para realizar las llamadas oportunas a los nuevos métodos del servicio REST de flujos de trabajo.
- T4 Ampliar la pantalla de presentación con las nuevas opciones.
- T5 Crear el formulario para crear/editar un flujo de trabajo.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que se está creando un nuevo flujo de trabajo CUANDO se elige la opción de cancelar ENTONCES este no debe de crearse.
- C2 DADO que se está creando un nuevo flujo de trabajo CUANDO se indica una descripción válida ENTONCES debe habilitarse la opción para guardarlo.
- C3 DADO que se está creando un nuevo flujo de trabajo CUANDO se indica una descripción inválida ENTONCES debe deshabilitarse la opción para guardarlo.
- C4 DADO que se ha seleccionado un flujo de trabajo CUANDO se elige la opción para modificarlo y se indica una descripción inválida ENTONCES este no se modifica.
- C5 DADO que se ha seleccionado un flujo de trabajo CUANDO se elige la opción para modificarlo y se indica una descripción válida ENTONCES este se modifica.
- C6 DADO que no se ha seleccionado un flujo de trabajo CUANDO se elige la opción para modificarlo ENTONCES se muestra un mensaje de error indicando que se debe seleccionar un flujo de trabajo.
- C7 DADO que se ha seleccionado un flujo de trabajo que estaba de baja CUANDO se elige la opción para darlo de baja/alta ENTONCES este es dado de alta (reactivado).
- C8 DADO que se ha seleccionado un flujo de trabajo que estaba de alta (activo) CUANDO se elige la opción para darlo de baja/alta ENTONCES este es dado de baja.
- C9 DADO que no se ha seleccionado un flujo de trabajo que estaba de alta (activo) CUANDO se elige la opción para darlo de baja/alta ENTONCES se muestra un mensaje de error indicando que se debe seleccionar un flujo de trabajo.

HU23 Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos [W]

Como diseñador de flujos de trabajo quiero poder crear o revocar permisos a usuarios y departamentos para instanciar un flujo de trabajo, así como poder elegir el administrador de un flujo poder gestionar la seguridad de un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear el servicio REST en el servidor que responda a las peticiones relacionadas con los permisos de un flujo de trabajo.
- T2 Ampliar el servicio REST de permisos de flujos de trabajo con un método para poder obtener todos los permisos de un flujo haciendo uso de la API.
- T3 Ampliar el servicio REST de permisos de flujos de trabajo con un método para poder crear un nuevo permiso haciendo uso de la API.
- T4 Ampliar el servicio REST de permisos de flujos de trabajo con un método para poder eliminar un permiso haciendo uso de la API.
- T5 Ampliar el método de actualización del servicio REST de flujos de trabajo para poder actualizar el administrador haciendo uso de la API.
- T6 Crear el servicio en el cliente que se encargue de realizar las llamadas al servicio REST de permisos de flujo de trabajo.
- T7 Crear la pantalla de permisos para instanciar flujos de trabajo con una tabla que permita filtrar por todos los campos y eliminarlos e indique el administrador.
- T8 Crear el formulario para crear un nuevo permiso y enlazarlo con la pantalla de permisos para instanciar flujos de trabajo.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se consulta la seguridad de un flujos de trabajo ENTONCES debe aparecer una lista de todos los permisos (descripción del autorizado y su tipo [usuario o departamento]) y el actual administrador del flujo de trabajo.
- C2 DADO que se consulta la lista de todos los permisos CUANDO se filtran por cualquiera de sus campos ENTONCES solo deben aparecer aquellos que concuerden con los datos filtrados.
- C3 DADO que se consulta la seguridad de un flujo de trabajo CUANDO se elige la opción para crear un nuevo permiso ENTONCES debe poder darse la opción para crear un nuevo permiso.
- C4 DADO que se está creando un nuevo permiso CUANDO se elige la opción de cancelar ENTONCES este no se debe crear.
- C5 DADO que se está creando un nuevo permiso CUANDO se indica el identificador del autorizado y si es usuario o departamento ENTONCES debe de habilitarse la opción para guardarla.
- C6 DADO que se está creando un nuevo permiso CUANDO no se indica el identificador del autorizado o si es usuario o departamento ENTONCES debe de deshabilitarse la opción para guardarla.
- C7 DADO que se ha seleccionado un permiso CUANDO se elige la opción para eliminarlo ENTONCES este es eliminado.

- C8 DADO que no se ha seleccionado un permiso CUANDO se elige la opción para eliminarlo ENTONCES se muestra un mensaje de error indicando que debe seleccionar un permiso.
- C9 DADO que se consulta la seguridad de un flujo de trabajo CUANDO se elige la opción para indicar un nuevo administrador ENTONCES debe poder darse la opción para nombrarlo.
- C10 DADO que se está indicando un nuevo administrador CUANDO se elige la opción de cancelar ENTONCES debe mantenerse el antiguo administrador.
- C11 DADO que se está indicando un nuevo administrador CUANDO se indica un identificador de administrador inválido ENTONCES debe de deshabilitarse la opción para guardarlo.
- C12 DADO que se está indicando un nuevo administrador CUANDO se indica un identificador de administrador inválido ENTONCES debe de habilitarse la opción para guardarlo.

HU24 Consultar la estructura de un flujo [W]

Como diseñador de flujos de trabajo quiero poder consultar la estructura de un flujo de trabajo determinado para así poder ver todas sus tareas y como se relacionan entre ellas.

A continuación se describen las **tareas** de esta historia:

- T1 Crear el servicio REST en el servidor que responda a las peticiones relacionadas con las tareas.
- T2 Ampliar el servicio REST de tareas con un método para obtener todas las tareas y las transiciones que forman parte de un flujo haciendo uso de la API.
- T3 Crear el servicio en el cliente que se encargue de realizar las llamadas al servicio REST de tareas.
- T4 Crear la pantalla que muestra la estructura de un flujo de trabajo mediante tablas anidadas en las que cada tabla interior son las transiciones de la tarea elegida.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se consulta la estructura de un flujo de trabajo ENTONCES se debe de mostrar una lista de las tareas que lo forman junto a sus atributos (descripción y tipo).
- C2 CUANDO se expande una tarea ENTONCES se debe de mostrar una lista de todas las transiciones que parten de ella junto a sus atributos (descripción y si posee función externa) así como indicar la tarea destino a la que enlaza.

HU25 Consultar los permisos de una tarea, crearlos y eliminarlos [W]

Como diseñador de flujos de trabajo quiero poder crear o revocar permisos a usuarios y departamentos para realizar una tarea, así como poder indicar si es un permiso de primer responsable.

A continuación se describen las **tareas** de esta historia:

- T1 Crear el servicio REST en el servidor que responda a las peticiones relacionadas con los permisos de una tarea.
- T2 Ampliar el servicio REST de permisos de tareas con un método para poder obtener todos los permisos para realizar una tarea haciendo uso de la API.

- T3 Ampliar el servicio REST de permisos de tareas con un método para poder crear un nuevo permiso haciendo uso de la API.
- T4 Ampliar el servicio REST de permisos de tareas con un método para poder eliminar un permiso haciendo uso de la API.
- T5 Crear el servicio en el cliente que se encargue de realizar las llamadas al servicio REST de permisos de tareas.
- T6 Crear la pantalla de permisos para realizar tareas con una tabla que permita filtrar por todos los campos y eliminarlos e indique si es un permiso de primer responsable.
- T7 Crear el formulario para crear un nuevo permiso y enlazarlo con la pantalla de permisos para realizar tareas.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 CUANDO se consulta la seguridad de una tarea ENTONCES debe aparecer una lista de todos los permisos (descripción del autorizado y su tipo [usuario o departamento]).
- C2 DADO que se consulta la lista de todos los permisos CUANDO se filtran por cualquiera de sus campos ENTONCES solo deben aparecer aquellos que concuerden con los datos filtrados.
- C3 DADO que se consulta la seguridad de una tarea CUANDO se elige la opción para crear un nuevo permiso ENTONCES debe poder darse la opción para crear un nuevo permiso.
- C4 DADO que se está creando un nuevo permiso CUANDO se elige la opción de cancelar ENTONCES este no se debe crear.
- C5 DADO que se está creando un nuevo permiso CUANDO se indica el identificador del autorizado y si es usuario o departamento ENTONCES debe de habilitarse la opción para guardarla.
- C6 DADO que se está creando un nuevo permiso CUANDO no se indica el identificador del autorizado o si es usuario o departamento ENTONCES debe de deshabilitarse la opción para guardarla.
- C7 DADO que se consulta la seguridad de una tarea CUANDO se elige la opción para crear un nuevo permiso de primer responsable ENTONCES debe poder darse la opción para crear un nuevo permiso de primer responsable.
- C8 DADO que se está creando un nuevo permiso de primer responsable CUANDO se elige la opción de cancelar ENTONCES este no se debe crear.
- C9 DADO que se está creando un nuevo permiso de primer responsable CUANDO se indica el identificador del autorizado y si es usuario o departamento ENTONCES debe de habilitarse la opción para guardarla.
- C10 DADO que se está creando un nuevo permiso de primer responsable CUANDO no se indica el identificador del autorizado o si es usuario o departamento ENTONCES debe de deshabilitarse la opción para guardarla.
- C11 DADO que se ha seleccionado un permiso CUANDO se elige la opción para eliminarlo ENTONCES este es eliminado.
- C12 DADO que no se ha seleccionado un permiso CUANDO se elige la opción para eliminarlo ENTONCES se muestra un mensaje de error indicando que debe seleccionar un permiso.

HU26 **Crear una tarea completa, modificarla y eliminarla [W]**

Como diseñador de flujos de trabajo quiero poder crear, modificar o eliminar una tarea completa (tarea completamente definida, indicando si es inicial, final, de asignación dinámica, etc) para así poder gestionar uno de los elementos fundamentales de los flujos de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar el servicio REST de tareas con un método para poder obtener todos los datos de una tarea haciendo uso de la API.
- T2 Ampliar el servicio REST de tareas con un método para poder crear una tarea completa haciendo uso de la API.
- T3 Ampliar el servicio REST de tareas con un método para poder modificar una tarea haciendo uso de la API.
- T4 Ampliar el servicio REST de tareas con un método para poder eliminar una tarea haciendo uso de la API.
- T5 Ampliar el servicio en el cliente que se encarga de realizar llamadas al servicio REST de tareas.
- T6 Ampliar la pantalla de consulta de la estructura de un flujo para poder eliminar tareas.
- T7 Crear el formulario para crear/editar tareas y enlazarlo con la pantalla de consulta de la estructura de un flujo.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que se está creando una tarea CUANDO se elige la opción de cancelar ENTONCES debe de cancelarse su creación.
- C2 DADO que se está creando una tarea CUANDO se indica una descripción válida ENTONCES debe habilitarse la opción para guardarla.
- C3 DADO que se está creando una tarea CUANDO se indica una descripción inválida ENTONCES debe deshabilitarse la opción para guardarla.
- C4 DADO que se está creando una tarea CUANDO se indica que es una tarea de asignación dinámica y se indica la clase que contiene la función de asignación dinámica ENTONCES debe habilitarse la opción para guardarla.
- C5 DADO que se está creando una tarea CUANDO se indica que es una tarea de asignación dinámica y no se indica la clase que contiene la función de asignación dinámica ENTONCES debe deshabilitarse la opción para guardarla.
- C6 DADO que se ha seleccionado una tarea CUANDO se selecciona la opción para editarla ENTONCES debe darse la opción para editarla.
- C7 DADO que se está editando una tarea CUANDO se modifica su descripción por una descripción inválida ENTONCES debe deshabilitarse la opción para guardarla.
- C8 DADO que se está editando una tarea CUANDO se modifica su descripción por una descripción válida ENTONCES debe habilitarse la opción para guardarla.
- C9 DADO que se está editando una tarea CUANDO se modifican sus atributos ENTONCES no debe permitirse que se pueda cambiar su tipo (inicial, final y condicional).
- C10 DADO que no se ha seleccionado una tarea CUANDO se selecciona la opción para editarla ENTONCES se muestra un mensaje de error indicando que se debe seleccionar una tarea.

C11 DADO que se ha seleccionado una tarea CUANDO se selecciona la opción para eliminarla ENTONCES esta es eliminada.

C12 DADO que no se ha seleccionado una tarea CUANDO se selecciona la opción para eliminarla ENTONCES se muestra un mensaje de error indicando que se debe seleccionar una tarea.

6.4.2. Resultados obtenidos

La realización del sprint se ha llevado a cabo durante el tiempo estimado inicialmente (del 08/04/2016 al 21/04/2016).

En cambio, durante este sprint no se han podido realizar todas las historias de usuario elegidas. El principal motivo del retraso ha sido el hecho que realizar la parte gráfica ha requerido más tiempo del que se había previsto en un principio.

La historia HU21 (Consultar los flujos de trabajo existentes) ha requerido 2PH en lugar de 1PH. El principal motivo del sobre-coste temporal ha sido causado por tener que invertir tiempo extra en buscar y probar un módulo para Angular que ofreciese la posibilidad de mostrar tablas complejas y permitiera filtrar cualquier campo de forma automática. Concretamente se probaron tres módulos.

La historia HU24 (Consultar la estructura de un flujo) ha requerido 3PH en lugar de 1PH. Ha habido dos motivos principales que han causado este retraso. Por un lado ha resultado más complejo de lo esperado utilizar la funcionalidad de tablas anidadas que ofrecida la aplicación y moldearla a la funcionalidad deseada. Por otro lado se ha requerido realizar un procesado de los datos ofrecidos por la API para adaptarlos a las estructuras que aceta la funcionalidad de tablas anidadas.

A causa del incremento temporal de las historias anteriores, la historia HU26 solo se ha podido iniciar, implementando una escasa parte de esta, por lo que se espera finalizarla en el sprint 5.

Las otras historias elegidas para este sprint se han realizado en el tiempo estimado y han pasado todos los criterios de aceptación.

6.4.3. Pila del producto actualizada

En la tabla 6.4 podemos observar el estado de la pila del producto tras finalizar el sprint 4. Como podemos ver, en ella aparecen todas las historias de usuario elegidas en el sprint 4 como finalizadas a excepción de la HU26 la cual no se ha podido completar.

Por último se ha añadido la HU30 (Duplicar un flujo de trabajo) debido a que ha surgido la necesidad de poder replicar la estructura de un flujo ya existente debido a que existe flujos de estructura muy similar. Dado que inicialmente no se tuvo en cuenta esta necesidad durante el desarrollo de la API, por lo tanto esta historia engloba tanto la ampliación de la API como del

diseñador gráfico.

ID	Historia de usuario	Coste estimado (PH)	Coste real (PH)	Sprints	Estado
HU01	Crear y modificar flujos [M]	1	1	SP1	FINALIZADA
HU02	Crear y modificar tareas [M]	1	2	SP1	FINALIZADA
HU03	Crear y modificar transiciones [M]	1	1	SP1	FINALIZADA
HU04	Consulta de flujos de trabajo [M]	1	1	SP1	FINALIZADA
HU05	Instanciar un flujo de trabajo [M]	3	2	SP1	FINALIZADA
HU06	Consultar una instancia de flujo de trabajo [M]	1	1	SP1	FINALIZADA
HU07	Sistema de permisos de flujos [M]	2	2	SP1	FINALIZADA
HU08	Sistema de permisos de tareas [M]	2	2	SP2	FINALIZADA
HU09	Traspaso de tareas [M]	2	1	SP2	FINALIZADA
HU10	Histórico [M]	2	2	SP2	FINALIZADA
HU11	Cancelar una instancia de flujo de trabajo [M]	1	1	SP2	FINALIZADA
HU12	Indicar funciones externas en transiciones [M]	2	2	SP3	FINALIZADA
HU13	Asignación dinámica de tareas [M]	1	1	SP3	FINALIZADA
HU14	Tareas condicionales [M]	5	3	SP3	FINALIZADA
HU17	Instanciar automáticamente la tarea inicial [M]	1	1	SP2	FINALIZADA
HU18	Finalizar una tarea [M]	2	2	SP2	FINALIZADA
HU19	Designar administrador de un flujo [M]	1	1	SP2	FINALIZADA
HU20	Documentar la API [M]	2	4	SP3	FINALIZADA
HU21	Consultar los flujos de trabajo existentes	1	2	SP4	FINALIZADA
HU22	Crear, modificar y dar de baja/alta un flujo de trabajo [W]	1	1	SP4	FINALIZADA
HU23	Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos [W]	2	2	SP4	FINALIZADA
HU24	Consultar la estructura de un flujo [W]	2	3	SP4	FINALIZADA
HU25	Consultar los permisos de una tarea, crearlos y eliminarlos [W]	2	2	SP4	FINALIZADA
HU26	Crear una tarea completa, modificarla y eliminarla [W]	3		SP4	INACABADA
HU27	Crear una transición completa, modificarla y eliminarla [W]	2			
HU28	Permitir intercambiar el orden de las funciones condicionales [W]	1			
HU29	Permitir modificar funciones condicionales [W]	1			
HU30	Duplicar un flujo de trabajo [M y W]	3			

Tabla 6.4: Estado de la pila del producto tras finalizar el sprint 4.

6.5. Sprint 5

6.5.1. Planificación del sprint

A continuación se muestran las historias de usuario elegidas para elaborar la pila del sprint 5. Junto a las historias se han definido una serie de tareas a realizar. Además se han especificado los criterios de aceptación que se deben cumplir para poder considerar que la historia de usuario ha sido finalizada. Cabe destacar que la historia HU26 (Crear una tarea completa, modificarla y eliminarla) se ha añadido a la pila del sprint 5 debido a que no se terminó en el sprint 4 (donde está especificada) y por lo tanto no ha vuelto a ser especificada en esta sección.

HU27 **Crear una transición completa, modificarla y eliminarla.** [W]

Como diseñador de flujos de trabajo quiero poder crear, modificar o eliminar una transición completa(transición completamente definida, indicando si debe llamar a una función externa, etc) para así poder gestionar uno de los elementos fundamentales de los flujos de trabajo.

A continuación se describen las **tareas** de esta historia:

- T1 Crear el servicio REST en el servidor que responda a las peticiones relacionadas con las transiciones.
- T2 Ampliar el servicio REST de transiciones con un método para poder crear una transición haciendo uso de la API.
- T3 Ampliar el servicio REST de transiciones con un método para poder modificar una transición haciendo uso de la API.
- T4 Ampliar el servicio REST de transiciones con un método para poder eliminar una transición haciendo uso de la API.
- T5 Crear el servicio en el cliente para realizar las llamadas oportunas a los métodos del nuevo servicio REST.
- T6 Ampliar la pantalla de consulta de la estructura de un flujo con las nuevas opciones.
- T7 Crear el formulario para crear/editar una transición.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que no se ha seleccionado ninguna tarea CUANDO se elige la opción para conectarla con otra ENTONCES se muestra un mensaje de error indicando que se debe seleccionar una tarea.
- C2 DADO que existe más de una tarea y se ha seleccionado CUANDO se elige la opción para conectarla con otra ENTONCES debe poder darse la opción para crear una nueva transición.
- C3 DADO que se está creando una transición CUANDO se elige la opción de cancelar ENTONCES debe de cancelarse su creación.
- C4 DADO que se está creando una transición CUANDO se indica su descripción y la tarea destino ENTONCES debe de habilitarse la opción para guardarla.
- C5 DADO que se está creando una transición CUANDO no se indica su descripción o la tarea destino ENTONCES debe de deshabilitarse la opción para guardarla.

- C6 DADO que se está creando una transición CUANDO se indica que debe llamar a una función externa y no se indica la clase que contiene dicha función ENTONCES debe de deshabilitarse la opción para guardarla.
- C7 DADO que se está creando una transición CUANDO se indica que debe llamar a una función externa y se indica la clase que contiene dicha función ENTONCES debe de habilitarse la opción para guardarla.
- C8 DADO que se está creando una transición CUANDO se indica que debe llamar a una función externa auditada y no se indica la clase que contiene dicha función o los mensajes de éxito y fracaso que quedaran reflejados en la auditoría ENTONCES debe de deshabilitarse la opción para guardarla.
- C9 DADO que se está creando una transición CUANDO se indica que debe llamar a una función externa auditada y se indica la clase que contiene dicha función y los mensajes de éxito y fracaso que quedaran reflejados en la auditoría ENTONCES debe de habilitarse la opción para guardarla.
- C10 DADO que se está creando una transición que parte de una tarea condicional CUANDO no se indica la clase que contiene la función condicional ENTONCES debe de deshabilitarse la opción para guardarla.
- C11 DADO que se está creando una transición que parte de una tarea condicional CUANDO se indica la clase que contiene la función condicional ENTONCES debe de habilitarse la opción para guardarla.
- C12 DADO que se ha seleccionado una transición CUANDO se selecciona la opción para editarla ENTONCES debe de poder darse la opción para editarla.
- C13 DADO que se está editando una transición CUANDO se modifica su descripción por una incorrecta ENTONCES debe de habilitarse la opción para guardar su edición.
- C14 DADO que se está editando una transición CUANDO se modifica su descripción por una correcta ENTONCES debe de deshabilitarse la opción para guardar su edición.
- C15 DADO que se está editando una transición CUANDO se indica que usa una función externa y no se indica la clase que la contiene ENTONCES debe de deshabilitarse la opción para guardar su edición.
- C16 DADO que se está editando una transición CUANDO se indica que usa una función externa y se indica la clase que la contiene ENTONCES debe de habilitarse la opción para guardar su edición.
- C17 DADO que se está editando una transición CUANDO se cancela su edición ENTONCES debe cancelarse su edición.
- C18 DADO que no se ha seleccionado una transición CUANDO se selecciona la opción para editarla ENTONCES se muestra un mensaje de error indicando que se debe seleccionar una transición.
- C19 DADO que se ha seleccionado una transición CUANDO se selecciona la opción para eliminarla ENTONCES esta es eliminada.
- C20 DADO que no se ha seleccionado una transición CUANDO se selecciona la opción para eliminarla ENTONCES se muestra un mensaje de error indicando que se debe seleccionar una transición.

HU28 Permitir intercambiar el orden de las funciones condicionales. [W]

Como diseñador de flujos de trabajo quiero poder modificar el orden en el que se analizan las transiciones de una función condicional para así poder modificar el recorrido de un flujo de trabajo.

A continuación se describen las **tareas** de esta historia:

T1 Crear el servicio REST en el servidor que responda a peticiones relacionadas con funciones condicionales.

T2 Ampliar el servicio REST de funciones condicionales con un método para poder modificar funciones condicionales haciendo uso de la API.

T3 Ampliar el servicio REST de funciones condicionales con un método para poder obtener las funciones condicionales de una tarea junto a su transición relacionada haciendo uso de la API.

T4 Ampliar la pantalla de consulta de la estructura de un flujo con la opción para modificar el orden de una función condicional.

T5 Crear el formulario para intercambiar el orden entre dos funciones condicionales.

A continuación se describen los **criterios de aceptación** de esta historia:

C1 DADO que existe una transición que tiene una función condicional CUANDO esta se selecciona ENTONCES aparece la opción para intercambiar su orden con otra.

C2 DADO que se ha seleccionado una transición que tiene una función condicional CUANDO se elige la opción para intercambiar su orden ENTONCES debe permitirse seleccionar la otra función externa con la que se quiere cambiar el orden.

HU29 Permitir modificar funciones condicionales. [W]

Como diseñador de flujos de trabajo quiero poder modificar funciones condicionales para poder adaptar las tareas condicionales ante nuevos requerimientos.

A continuación se describen las **tareas** de esta historia:

T1 Modificar el servicio REST de funciones condicionales para que el método de modificación permita cambiar la función externa y el orden de una función condicional haciendo uso de la API.

T2 Ampliar la pantalla de consulta de la estructura de un flujo con la nueva opción.

T3 Crear el formulario para editar una función condicional.

A continuación se describen los **criterios de aceptación** de esta historia:

C1 DADO que existe una transición que tiene una función condicional CUANDO esta se selecciona ENTONCES aparece la opción para modificarla.

C2 DADO que se ha seleccionado una transición que tiene una función condicional CUANDO se elige la opción para modificarla ENTONCES debe permitirse modificar la clase que contiene dicha función condicional.

HU30 Duplicar un flujo de trabajo. [M y W]

Como diseñador de flujos de trabajo quiero poder duplicar toda la estructura básica de un flujo de trabajo (tareas y transiciones) para así poder crear nuevos flujos que se parezcan a otros ya existentes de forma rápida y eficiente.

A continuación se describen las **tareas** de esta historia:

- T1 Ampliar la API para ofrecer la nueva funcionalidad.
- T2 Ampliar el servicio REST de flujos con un nuevo método que permita duplicar toda la estructura de un flujo haciendo uso de la API.
- T3 Ampliar el servicio del cliente de flujos de trabajo para llamar al nuevo método.
- T4 Ampliar la pantalla de consulta de la estructura de un flujo con la nueva opción.

A continuación se describen los **criterios de aceptación** de esta historia:

- C1 DADO que se ha seleccionado un flujo de trabajo CUANDO se elige la opción para duplicarlo ENTONCES se duplica toda su estructura básica (tareas y transiciones) y el nuevo flujo se llama igual que el anterior más _1.
- C2 DADO que no se ha seleccionado un flujo de trabajo CUANDO se elige la opción para duplicarlo ENTONCES se muestra un mensaje de error indicando que se debe seleccionar un flujo.

6.5.2. Resultados obtenidos

La realización del sprint se ha llevado a cabo durante el tiempo estimado inicialmente (del 22/04/2016 al 05/05/2016).

La planificación temporal en el conjunto de historias se ha cumplido aunque ha habido variaciones respecto a la estimación y el coste real de dos historias. En concreto la historia HU28 (Permitir intercambiar el orden de las funciones condicionales) ha resultado ser más compleja de lo previsto debido a problemas con la parte gráfica. Por otro lado la historia HU30 (Duplicar un flujo de trabajo) ha requerido menos tiempo del estimado inicialmente.

Durante el sprint se ha terminado de implementar toda la funcionalidad básica requerida en la interfaz gráfica para el diseño de flujos de trabajo.

6.5.3. Pila del producto actualizada

En la tabla 6.5 podemos observar el estado de la pila del producto tras finalizar el sprint 5. Como podemos ver, en ella aparecen todas las historias de usuario elegidas en el sprint 5 como finalizadas.

Dado que el sprint 5 era el último sprint, la pila del producto de este es la pila del producto final.

ID	Historia de usuario	Coste estimado (PH)	Coste real (PH)	Sprints	Estado
HU01	Crear y modificar flujos [M]	1	1	SP1	FINALIZADA
HU02	Crear y modificar tareas [M]	1	2	SP1	FINALIZADA
HU03	Crear y modificar transiciones [M]	1	1	SP1	FINALIZADA
HU04	Consulta de flujos de trabajo [M]	1	1	SP1	FINALIZADA
HU05	Instanciar un flujo de trabajo [M]	3	2	SP1	FINALIZADA
HU06	Consultar una instancia de flujo de trabajo [M]	1	1	SP1	FINALIZADA
HU07	Sistema de permisos de flujos [M]	2	2	SP1	FINALIZADA
HU08	Sistema de permisos de tareas [M]	2	2	SP2	FINALIZADA
HU09	Traspaso de tareas [M]	2	1	SP2	FINALIZADA
HU10	Histórico [M]	2	2	SP2	FINALIZADA
HU11	Cancelar una instancia de flujo de trabajo [M]	1	1	SP2	FINALIZADA
HU12	Indicar funciones externas en transiciones [M]	2	2	SP3	FINALIZADA
HU13	Asignación dinámica de tareas [M]	1	1	SP3	FINALIZADA
HU14	Tareas condicionales [M]	5	3	SP3	FINALIZADA
HU17	Instanciar automáticamente la tarea inicial [M]	1	1	SP2	FINALIZADA
HU18	Finalizar una tarea [M]	2	2	SP2	FINALIZADA
HU19	Designar administrador de un flujo [M]	1	1	SP2	FINALIZADA
HU20	Documentar la API [M]	2	4	SP3	FINALIZADA
HU21	Consultar los flujos de trabajo existentes [W]	1	2	SP4	FINALIZADA
HU22	Crear, modificar y dar de baja/alta un flujo de trabajo [W]	1	1	SP4	FINALIZADA
HU23	Consultar los permisos y el administrador de un flujo, crearlos y eliminarlos [W]	2	2	SP4	FINALIZADA
HU24	Consultar la estructura de un flujo [W]	2	3	SP4	FINALIZADA
HU25	Consultar los permisos de una tarea, crearlos y eliminarlos [W]	2	2	SP4	FINALIZADA
HU26	Crear una tarea completa, modificarla y eliminarla [W]	3	3	SP4 SP5	FINALIZADA
HU27	Crear una transición completa, modificarla y eliminarla [W]	2	2	SP5	FINALIZADA
HU28	Permitir intercambiar el orden de las funciones condicionales [W]	1	2	SP5	FINALIZADA
HU29	Permitir modificar funciones condicionales [W]	1	1	SP5	FINALIZADA
HU30	Duplicar un flujo de trabajo [M y W]	3	2	SP5	FINALIZADA

Tabla 6.5: Estado de la pila del producto tras finalizar el sprint 5.

Capítulo 7

Conclusiones

Contents

7.1. Objetivos alcanzados	85
7.2. La importancia de documentar	85
7.3. Valoración personal	86
7.4. Mejoras y ampliaciones del proyecto	86

7.1. Objetivos alcanzados

Una vez finalizado el proyecto y tras analizar toda la funcionalidad obtenida, el supervisor del proyecto ha concluido que se han logrado todos los objetivos planteados antes y durante el desarrollo de este. Con la finalización del proyecto se ha obtenido una versión funcional capaz de permitir recorrer un flujo de trabajo sencillo, así como un editor gráfico para poder definirlos. Por estos motivos la empresa ha decidido utilizar el motor de flujos de trabajo en uno de sus próximos proyectos.

7.2. La importancia de documentar

Una de las últimas tareas realizadas respecto al motor fue la de documentar la API del motor de flujos en el README del repositorio de Bitbucket. Durante los últimos días del proyecto, en la empresa se empezó a desarrollar una nueva aplicación para la gestión de recursos humanos. El hecho de disponer una documentación de cómo usar la API del motor y las estructuras de los mensajes que devolvía ayudó a agilizar la comprensión de cómo utilizarla y el desarrollo del módulo para comunicarse con ella.

7.3. Valoración personal

A nivel personal, este proyecto me ha resultado muy interesante debido a varias razones. Por un lado, el hecho de ser el primer proyecto profesional que realicé ha resultado ser una experiencia muy positiva. El hecho de haber podido realizar un proyecto cuyo desarrollo principal se centra en el desarrollo de “back-end” me ha servido de motivación debido a que es una de las labores que más me gusta.

Por otro lado, el hecho de haber podido realizar este proyecto mediante una metodología ágil, me ha permitido poder afrontar los cambios de especificaciones de forma rápida y fácil, lo cual ha afectado positivamente a mi habilidad para reaccionar a estos y saber como afrontarlos.

A nivel formativo, el desarrollo de este proyecto me ha permitido profundizar en mis conocimientos sobre AngularJS, el cual empecé a aprender un mes antes de empezar el proyecto. También he ampliado mis conocimientos en Java aprendiendo a utilizar la reflexión y ampliando mis conocimientos generales.

Por último, el hecho de utilizar el servidor de aplicaciones JBoss me ha servido para aprender algunos conceptos básicos sobre su funcionamiento y configuración (estructura de directorios, configuración de “datasources”, etc).

7.4. Mejoras y ampliaciones del proyecto

En esta sección se plantean algunas mejoras y ampliaciones sobre la API del motor de flujos de trabajo. Estas se han planteado sobre el final del proyecto cuando se está empezando a integrar en una nueva aplicación de la empresa.

Una de las mejoras planteadas durante el proyecto es la de añadir metainformación a las tareas y a las transiciones. Esta podría proporcionar más flexibilidad a las aplicaciones que utilizan la API del motor de flujos.

Otra mejora que se plantea realizar en breve es la de asignar el último responsable de una instancia de tarea en el caso de vuelta hacia atrás durante el recorrido de un flujo.

También se plantea ampliar la funcionalidad que permite consultar al histórico dotándola de la posibilidad de utilizar filtros.

Por último, otra posible mejora consistiría en poder realizar una llamada a una función externa cuando se instancia una tarea.

Bibliografía

- [1] INTEGRA Consultores. [Consulta: 15 sep. 2016].
<http://integraconsultores.es/>
- [2] jBPM Documentation. [Consulta: 15 sep. 2016].
http://docs.jboss.org/jbpm/release/6.4.0.Final/jbpm-docs/html_single/index.html
- [3] The Camunda BPM Manual. [Consulta: 15 sep. 2016].
<https://docs.camunda.org/manual/7.5/>
- [4] JIRA Software, Precios. [Consulta: 15 sep. 2016].
<https://es.atlassian.com/software/jira/pricing?tab=host-on-your-server>
- [5] Bitbucket Server Licenses. [Consulta: 15 sep. 2016].
<https://es.atlassian.com/licensing/bitbucket-server/>
- [6] AngularJS 1.5.0 API Docs. [Consulta: 15 sep. 2016].
<https://code.angularjs.org/1.5.0/docs/api>
- [7] Model-view-viewmodel. Wikipedia. [Consulta: 15 sep. 2016].
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>
- [8] Java. Wikipedia. [Consulta: 15 sep. 2016].
[https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))
- [9] Maven. Wikipedia. [Consulta: 15 sep. 2016].
<https://es.wikipedia.org/wiki/Maven>
- [10] MySQL. Wikipedia. [Consulta: 15 sep. 2016].
<https://es.wikipedia.org/wiki/MySQL>
- [11] JUnit. [Consulta: 15 sep. 2016].
<http://junit.org/>
- [12] Charlie Hubbard. Flexjson. [Consulta: 15 sep. 2016].
<http://flexjson.sourceforge.net/>
- [13] Hibernate ORM. [Consulta: 15 sep. 2016].
<http://hibernate.org/orm/>
- [14] MySQL Connector/J. [Consulta: 15 sep. 2016].
<https://dev.mysql.com/downloads/connector/j/>

- [15] Log4j Frequently Asked Technical Questions. [Consulta: 15 sep. 2016].
<https://logging.apache.org/log4j/1.2/faq.html>
- [16] REStEasy JAX-RS, REStFul Web Services for Java. [Consulta: 15 sep. 2016].
http://docs.jboss.org/resteasy/docs/2.0.0.GA/userguide/html_single/index.html
- [17] AngularJS. Wikipedia. [Consulta: 15 sep. 2016].
<https://es.wikipedia.org/wiki/AngularJS>
- [18] Bower, A package manager for the web. [Consulta: 15 sep. 2016].
<http://bower.io/>
- [19] Miguel Durán Uña. WildFly (Anteriormente JBoss). Wikipedia. [Consulta: 15 sep. 2016].
<https://es.wikipedia.org/wiki/WildFly>
- [20] NetBeans. Wikipedia. [Consulta: 15 sep. 2016].
<https://es.wikipedia.org/wiki/NetBeans>
- [21] MySQL Workbench. Wikipedia. [Consulta: 15 sep. 2016].
https://es.wikipedia.org/wiki/MySQL_Workbench
- [22] Tom Mutdosch. HttpRequester README. [Consulta: 15 sep. 2016].
<https://github.com/tommut/HttpRequester>
- [23] Dan Radigan. A brief introduction to scrum. [Consulta: 15 sep. 2016].
<https://es.atlassian.com/agile/scrum>
- [24] Martin Fowler. TestDrivenDevelopment. [Consulta: 15 sep. 2016].
<http://martinfowler.com/bliki/TestDrivenDevelopment.html>
- [25] L. Dusseault, Linden Lab y J. Snell. PATCH Method for HTTP. [Consulta: 15 sep. 2016].
<https://tools.ietf.org/html/rfc5789>

Anexo A

Documentación externa de la API del motor de flujos

A.1. Motivación

Dado que parte de este proyecto ha consistido en realizar una API de un motor de flujos que será utilizada por otras aplicaciones de la empresa, la buena documentación de esta es un requisito indispensable.

Es un hecho que una API o librería si dispone de una documentación pésima o simplemente carece de ella va a generar inmediatamente un sentimiento de rechazo o desconfianza de los desarrolladores que tengan que usarla. Además, si estos disponen de más de una opción, seguramente las opciones sin documentación o de baja calidad serán más propensas a ser descartadas.

Por último, aunque el código disponga de Javadoc, se decidió realizar una documentación externa que ofreciese una imagen más clara de la API y de cómo comunicarse con ella.

A.2. Elección de cómo y donde documentar la API

Tras analizar la API desarrollada, se decidió estructurar la documentación en seis apartados diferentes. Por un lado en el primer apartado se explicaría como utilizar la capa de acceso de la API. Por otro lado se explicarían los cuatro módulos que son accesibles desde la capa de acceso (motor, diseñador, seguridad y auditoría) y las funcionalidades que ofrecen. Finalmente el último apartado sería un anexo con las estructuras JSON que devuelve la API.

Para documentar la API se analizaron dos formas diferentes, crear un documento de texto o utilizar el propio README del repositorio de Bitbucket. Finalmente se decidió optar por el README debido a que ofrecía varias ventajas respecto al documento de texto. Su mantenimiento es más sencillo al estar situado dentro del repositorio proyecto. Además, al estar escrito en Markdown (lenguaje de marcado ligero) permite obtener un aspecto visual limpio y de cali-

dad sin requerir de mucho esfuerzo ni de la ayuda de un editor específico. Finalmente, al ser el documento README del proyecto, este es presentado automáticamente en la página principal del repositorio, facilitando su consulta.

A.3. Fragmentos de ejemplo de la documentación de la API

A lo largo de este apartado se van a mostrar diferentes fragmentos de la documentación realizada sobre la API, concretamente sus métodos y las estructuras de datos que devuelve. Dado que la documentación es bastante extensa, en este apartado se ha optado por solo mostrar algunas secciones (la introducción, la explicación de uno de sus módulos, las funciones externas y parte de las estructuras JSON devueltas por la API).

A.3.1. Introducción

La API de IdeaFlow permite definir, gestionar y recorrer flujos de trabajo. Para hacer uso de sus funcionalidades se debe de obtener una instancia de uno de sus cuatro módulos a través de la clase de acceso `IdeaflowFactory`.

Todas las respuestas de la API tienen la misma estructura descrita a continuación:

```
{
  "exito": Boolean,
  "datos": Object
}
```

Concretamente, si la llamada no tiene éxito, este siempre tendrá la siguiente forma donde `datos` contiene el mensaje del error:

```
{
  "exito": False,
  "datos":
  {
    "codigo": Number,
    "mensaje": String
  }
}
```

A.3.2. Módulo del motor de flujos

El motor sirve para instanciar, recorrer y gestionar flujos de trabajo.

El motor ya se encarga internamente de realizar las comprobaciones de seguridad oportunas en cada método y en realizar la auditoría necesaria si lo requiere la acción.

Para poder acceder a la API del motor primero se debe instanciar.

```
IMotorWorkflow motor = IdeaflowFactory
    .getMotorWorkflow(idUsuario, idDepartamento, entityManager);
```

Tanto el `idUsuario` como el `idDepartamento` son cadenas que representan el identificador del usuario que está haciendo uso del motor y el identificador de su departamento. Estos identificadores son los que utiliza internamente la API. El `EntityManager` es inyectado en la API y es responsabilidad de la aplicación exterior. Concretamente esta deberá iniciar la transacción y finalizarla. Esto permite a la aplicación exterior cancelar una transacción que afecta al motor de flujos en el caso de que se produzca un error exterior para no causar una inconsistencia en los datos. Así pues la API indicará en la transacción si se debe de hacer rollback en caso de producirse algún error interno.

A continuación se describen todas las funcionalidades que ofrece este módulo.

- **Instanciar un flujo de trabajo**

- Se realiza mediante el método `instanciarWorkflow`.
- Todos los usuarios pueden realizar esta acción siempre y cuando tengan permiso para instanciar el flujo de trabajo indicado.
- Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de la instancia del flujo de trabajo creada.

- **Cancelar una instancia de flujo de trabajo**

- Se realiza mediante el método `cancelarInstanciaWorkflow`.
- Solo el administrador del propio flujo de trabajo tiene permiso para cancelar una de sus instancias.
- Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de la instancia del flujo de trabajo cancelada.

- **Obtener las tareas activas del usuario**

- Se realiza mediante el método `obtenerMisTareasActivas`.
- Todos los usuarios pueden realizar esta acción.
- Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas y cuyo responsable es el propio usuario.

- **Obtener las tareas activas del departamento del usuario**

- Se realiza mediante el método `obtenerTareasActivasMiDepartamento`.
- Todos los usuarios pueden realizar esta acción.
- Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas y cuyo responsable es el departamento del usuario.

- **Obtener las tareas activas del usuario sobre un tipo de flujo de trabajo**
 - Se realiza mediante el método `obtenerMisTareasActivasWorkflow`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas que pertenezcan al tipo de flujo de trabajo indicado y cuyo responsable es el propio usuario.

- **Obtener las tareas activas del departamento del usuario sobre un tipo de flujo de trabajo**
 - Se realiza mediante el método `obtenerTareasActivasMiDepartamentoWorkflow`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas que pertenezcan al tipo de flujo de trabajo indicado y cuyo responsable es el departamento del usuario.

- **Obtener las tareas activas del usuario sobre una instancia de flujo de trabajo**
 - Se realiza mediante el método `obtenerMisTareasActivasWorkflow`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas que pertenezcan a la instancia de flujo de trabajo indicada y cuyo responsable es el propio usuario.

- **Obtener las tareas activas del departamento del usuario sobre una instancia de flujo de trabajo**
 - Se realiza mediante el método `obtenerTareasActivasMiDepartamentoWorkflow`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de tarea que no han sido finalizadas que pertenezcan a la instancia de flujo de trabajo indicada y cuyo responsable es el departamento del usuario.

- **Obtener a que otros usuarios se puede reasignar una instancia de tarea**
 - Se realiza mediante el método `obtenerUsuariosReasignarTarea`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá una lista con los identificadores de los usuarios con permisos para poder ser asignados.

- **Obtener a que otros departamentos se puede reasignar una instancia de tarea**
 - Se realiza mediante el método `obtenerDepartamentosReasignarTarea`.
 - Todos los usuarios pueden realizar esta acción.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá una lista con los identificadores de los departamentos con permisos para poder ser asignados.

- **Reasignar una instancia de tarea a un usuario**
 - Se realiza mediante el método `reasignarResponsabilidadInstanciaTareaUsuario`.
 - Solo el usuario que sea responsable de la instancia de tarea, el usuario cuyo departamento sea responsable de la instancia de tarea o el administrador del flujo de trabajo al que pertenece la instancia de tarea pueden realizar esta acción. Además, el usuario destino debe poseer permisos para poder ser el responsable de la instancia de tarea.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de la instancia de tarea actualizados.

- **Reasignar una instancia de tarea a un departamento**
 - Se realiza mediante el método `reasignarResponsabilidadInstanciaTareaDepartamento`.
 - Solo el usuario que sea responsable de la instancia de tarea, el usuario cuyo departamento sea responsable de la instancia de tarea o el administrador del flujo de trabajo al que pertenece la instancia de tarea pueden realizar esta acción. Además, el departamento destino debe poseer permisos para poder ser el responsable de la instancia de tarea.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de la instancia de tarea actualizados.

- **Obtener las instancias de tarea activas de una instancia de flujo de trabajo**
 - Se realiza mediante el método `obtenerTareasActivasEjecucionWorkflow`.
 - Solo el usuario que instanció el flujo de trabajo y el administrador pueden realizar esta operación.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de las tareas que no hayan sido finalizadas.

- **Obtener las instancias no finalizadas de un flujo de trabajo**
 - Se realiza mediante el método `obtenerInstanciasActivasWorkflow`.
 - Solo el administrador pueden realizar esta operación.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de las instancias de los flujos de trabajo que no hayan sido finalizados.

- **Finalizar una instancia de tarea activa de una instancia de flujo de trabajo**
 - Se realiza mediante el método `finalizarTareaEnEjecucionWorkflow`.
 - Solo el responsable de la tarea (el usuario al que se le ha asignado la realización de dicha instancia de tarea) o el administrador pueden realizar esta operación.
 - Si tiene éxito el parámetro `datos` de la respuesta contendrá los datos de la nueva instancia de la tarea destino.

A.3.3. Funciones externas

Las funciones externas son métodos llamados por la API para obtener ciertos datos o realizar ciertas tareas que no están a su alcance.

Las funciones externas asociadas a una transición se pueden utilizar para realizar algún tipo de auditoría externa (para tener constancia de que se ha realizado dicha transición) o realizar alguna operación más compleja. Estas funciones deben devolver un valor booleano para indicarle a la API si la operación exterior se ha realizado correctamente para poder continuar o si debe cancelar el paso de una tarea a otra.

Aquellas clases que alberguen una función externa de una transición deben implementar la interfaz `IFuncionTransicion` proporcionada por la API. Esta interfaz obliga a implementar el método `puedeRealizarTransicion` el cual recibe un `String` como parámetro. Este es un JSON con la siguiente estructura:

```
{
  "wflCodigo": Number ,
  "insWflCodigo": Number ,
  "tarOrigenCodigo": Number ,
  "insTarOrigenCodigo": Number ,
  "tarDestinoCodigo": Number
}
```

Las funciones externas de asignación dinámica permiten designar el responsable de una instancia de tarea desde el exterior de la API. Esta funcionalidad permite crear flujos de trabajo en los cuales algunas tareas requieran un responsable específico que sea elegido en tiempo de ejecución dependiendo de alguna lógica de negocio de la aplicación que utiliza la API.

Aquellas clases que alberguen una función externa de asignación dinámica deben implementar la interfaz `IFuncionAsignacionDinamica` proporcionada por la API. Esta interfaz obliga a implementar el método `obtenerResponsable` el cual recibe un `String` como parámetro. Este es un JSON con la siguiente estructura:

```
{
  "wflCodigo": Number ,
  "insWflCodigo": Number ,
  "tarCodigo": Number ,
  "insTarCodigo": Number
}
```

Este método debe de devolver un JSON en formato de String con la siguiente estructura:

```
{
  "esUsuario": Boolean ,
  "responsable": String
}
```

Por último, las funciones condicionales permiten definir la lógica de flujos más complejos. Las funciones condicionales actúan a modo de semáforos (devolviendo un valor booleano) en las tareas condicionales y cada una de ellas está relacionada con una transición que parte de dicha tarea. Uno de sus usos es el de poder definir un flujo en el que se requiera la realización de más tareas por parte de superiores si se dan ciertas características en el exterior de la API.

Aquellas clases que alberguen una función condicional deben implementar la interfaz `IFuncionCondicional` proporcionada por la API. Esta interfaz obliga a implementar el método `cumpleCondicion` el cual recibe un `String` como parámetro. Este es un JSON con la siguiente estructura:

```
{
  "wflCodigo": Number ,
  "insWflCodigo": Number ,
  "tarCodigo": Number ,
  "insTarCodigo": Number
}
```

A.3.4. Estructuras de los mensajes de éxito

Todos los métodos que proporciona la API devuelven estructuras JSON en forma de `String`. A lo largo de este apartado se describen algunas de las estructuras que pueden aparecer en el parámetro datos.

Flujo de trabajo (workflow)

```
{
  "administrador":String,
  "baja":Boolean,
  "codigo":Number,
  "descripcion":String,
  "fechaAlta":Number,
  "fechaMod":Number,
  "usuarioAlta":String,
  "usuarioMod":String
}
```

El parámetro `fechaMod` tendrá el valor `null` si el flujo de trabajo nunca ha sido modificado, de lo contrario tendrá una fecha en milisegundos.

El parámetro `usuarioMod` tendrá el valor `null` si el flujo de trabajo nunca ha sido modificado.

Tarea

```
{
  "baja":Boolean,
  "claseAd":String,
  "codigo":Number,
  "descripcion":String,
  "esCondicional":Boolean,
  "esFinal":Boolean,
  "esInicial":Boolean,
  "fechaAlta":Number,
  "fechaMod":Number,
  "usuarioAlta":String,
  "usuarioMod":String,
  "wflCodigo":{
    "codigo":Number,
    "descripcion":String
  }
}
```


El parámetro `fechaMod` tendrá el valor `null` si la tarea nunca ha sido modificada, de lo contrario tendrá una fecha en milisegundos.

El parámetro `usuarioMod` tendrá el valor `null` si la tarea nunca ha sido modificada.

El parámetro `claseAd` tendrán el valor `null` si la tarea no es de asignación dinámica, de lo contrario tendrá el nombre completo de la clase.

Transición

```
{
  "baja": Boolean ,
  "claseFe": String ,
  "codigo": Number ,
  "descripcion": String ,
  "fechaAlta": Number ,
  "fechaMod": Number ,
  "mensajeFeNoHis": String ,
  "mensajeFeSiHis": String ,
  "notificarFeHis": Boolean ,
  "tarCodigoFin": {
    "codigo": Number ,
    "descripcion": String
  },
  "tarCodigoInicio": {
    "codigo": Number ,
    "descripcion": String
  },
  "usuarioAlta": String ,
  "usuarioMod": String
}
```

El parámetro `fechaMod` tendrá el valor `null` si la transición nunca ha sido modificada, de lo contrario tendrá una fecha en milisegundos.

El parámetro `usuarioMod` tendrá el valor `null` si la transición nunca ha sido modificada.

El parámetro `claseFe` tendrán el valor `null` si la transición no tiene asociada ninguna función externa, de lo contrario tendrá el nombre completo de la clase.

Los parámetros `mensajeFeSiHis` y `mensajeFeNoHis` tendrán el valor `null` si no hay función o si no se debe notificar en el histórico.

Permiso para instanciar un flujo

```
{
  "baja": Boolean ,
  "codigo": Number ,
  "departamento": String ,
  "fechaAlta": Number ,
  "fechaMod": Number ,
  "usuario": String ,
  "usuarioAlta": String ,
  "usuarioMod": String ,
  "wflCodigo":{
    "codigo": Number ,
    "descripcion": String
  }
}
```

El parámetro `departamento` tendrá el valor `null` si el permiso es de un usuario.

El parámetro `usuario` tendrá el valor `null` si el permiso es de un departamento.

Los parámetros `fechaMod` y `usuarioMod` tendrán el valor `null` si el permiso nunca ha sido modificado, de lo contrario `fechaMod` tendrá una fecha en milisegundos.

Permiso para realizar una tarea

```
{
  "baja": Boolean ,
  "codigo": Number ,
  "departamento": String ,
  "fechaAlta": Number ,
  "fechaMod": Number ,
  "primerResponsable": Boolean ,
  "tarCodigo":{
    "codigo": Number ,
    "descripcion": String
  },
  "usuario": String ,
  "usuarioAlta": String ,
  "usuarioMod": String
}
```

El parámetro `primerResponsable` tendrá el valor `true` si se trata del primer responsable al que se le asignarán las tareas.

El parámetro `departamento` tendrá el valor `null` si el permiso es de un usuario.

El parámetro `usuario` tendrá el valor `null` si el permiso es de un departamento.

Los parámetros `fechaMod` y `usuarioMod` tendrán el valor `null` si el permiso nunca ha sido modificado, de lo contrario `fechaMod` tendrá una fecha en milisegundos.

Instancia de flujo de trabajo

```
{
  "baja":String,
  "cancelado":Boolean,
  "codigo":Number,
  "fechaFin":Number,
  "fechaInicio":Number,
  "motivoCancelada":String,
  "usuarioInstanciador":String,
  "wflCodigo":{
    "codigo":Number,
    "descripcion":String
  }
}
```

El parámetro `motivoCancelada` tendrá el valor `null` siempre excepto cuando la instancia se ha cancelado.

El parámetro `fechaFin` tendrá el valor `null` si la instancia no ha terminado, de lo contrario tendrá una fecha en milisegundos. Si se cancela entonces tendrá la fecha en la que se canceló.

Instancia de tarea

```
{
  "baja": Boolean ,
  "cancelada": Boolean ,
  "codigo": Number ,
  "fechaFin": Number ,
  "fechaInicio": Number ,
  "fechaMod": Number ,
  "insWfCodigo":{
    "codigo": Number
  },
  "observaciones": String ,
  "responsable": String ,
  "tarCodigo":{
    "codigo": Number ,
    "descripcion": String
  }
}
```

El parámetro `fechaMod` tendrá el valor `null` si la instancia no se ha modificado.

El parámetro `fechaFin` tendrá el valor `null` si la instancia no ha terminado, de lo contrario tendrá una fecha en milisegundos.

El parámetro `observaciones` tendrá el valor `null` si no hay observaciones.

El parámetro `responsable` tendrá el valor `null` si no hay responsable.

Entrada del histórico

```
{
  "codigo": Number ,
  "descripcion": String ,
  "fecha": Number ,
  "insTarCodigo":{
    "codigo": Number
  },
  "insWfCodigo":{
    "codigo": Number
  },
  "responsable": String ,
  "usuarioRealiza": String ,
  "usuarioDestino": String ,
  "departamentoDestino": String
}
```

El parámetro `insTarCodigo` tendrá el valor `null` si el evento no está relacionado con una instancia de tarea.

El parámetro `responsable` tendrá el valor `null` si se trata de una instancia de tarea final.

El parámetro `usuarioDestino` tendrá el valor `null` si el evento no es de reasignación a un usuario.

El parámetro `departamentoDestino` tendrá el valor `null` si el evento no es de reasignación a un departamento.

Función condicional

```
{
  "clase":String,
  "codigo":Number,
  "orden":Number,
  "tarCodigo":{
    "codigo":Number,
    "descripcion":String"
  },
  "tranCodigo":{
    "codigo":Number,
    "descripcion":String
  }
}
```


Glosario de términos

- **API (Application Programming Interface):** Es el conjunto de métodos ofrecidos por una biblioteca para ser utilizados por otra aplicación ofreciendo una capa de abstracción.
- **REST (Representational State Transfer):** Término que se utiliza para describir cualquier interfaz que utiliza directamente HTTP para comunicar datos.
- **TDD (Test-driven development):** El desarrollo guiado por pruebas es una metodología de la ingeniería del software que consiste en escribir de forma incremental las pruebas que debe de pasar un código, escribiendo siempre antes la prueba que el código que debe de pasarla. Durante las iteraciones el código se va refactorizando si es necesario.
- **PH (Punto de Historia):** En la metodología Scrum se utilizan para determinar el "coste" de realizar una historia de usuario dependiendo (principalmente) de su complejidad y riesgo.
- **DAO (Data Access Object):** Componente que suministra una interfaz común entre la aplicación que lo usa y la base de datos a la que se conecta. El uso de estos componentes permite desacoplar los objetos de negocio y la base de datos.
- **CRUD (Create, Read, Update and Delete):** Son las funciones básicas de cualquier capa de persistencia de un software.
- **JSON (JavaScript Object Notation):** Es un estándar que utiliza texto plano para el intercambio de datos. Es independiente de cualquier lenguaje de programación.
- **MVVM (Model View ViewModel):** Patrón arquitectónico del software que permite separar la capa de presentación de la capa de negocio.
- **README:** Documento de texto que normalmente se incluye junto a los programas. En el se incluye información respecto a su uso, cambios en las diferentes versiones, errores, etc.