



Universidad
Zaragoza

Proyecto Fin de Grado

Desarrollo de una herramienta para la creación de
Workflows. Workflow Designer.

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

RESUMEN

Este proyecto consiste en el proceso de análisis, diseño y desarrollo de un módulo para la creación de flujos de trabajo, en adelante “*Workflow*”. El módulo está integrado dentro del producto *EndaliaHR* que oferta la empresa donde se ha realizado el proyecto.

El producto de la empresa soporta un módulo de ejecución de Workflows, el núcleo de este módulo es el motor de Workflows. El motor se encarga de interpretar el modelo del Workflow y controlar los estados, participantes y la información de cada fase y de producir una interfaz para trabajar con él.

El motor es capaz de interpretar el modelo del Workflow, que no es más que información almacenada en una base de datos. El proyecto surge para dar solución a la problemática de modelar estos Workflows, puesto que no hay una herramienta que cubra dicha necesidad. Se intenta también solventar el problema de comunicación del modelo del Workflow, es decir, de las etapas y las acciones que se ejecutan durante el avance del flujo que quiere el cliente que utiliza el producto y que desea diseñar un Workflow para su uso.

A nivel de desarrollo se necesita una mejora para la mantenibilidad del código y su reutilización entre los diferentes Workflows puesto que algunos comparten operaciones comunes que pueden ser reutilizadas.

En cuanto a la arquitectura se busca crear un proyecto que cumpla con el uso del modelo de datos y que esté sujeto a las tecnologías del producto además de permitir crear Workflows que estén soportados por el motor de Workflows de la empresa.

El desarrollo del proyecto se planificó llevando a cabo ciclos con metodología Ágil, con revisiones en cada apartado que llevaban a realizar mejoras en algunos aspectos que ya se habían analizado. Siguiendo las pautas del diseño centrado en el usuario en un principio se realizaron diferentes reuniones con el equipo interno que haría uso del módulo. A este grupo se le consultaron funcionalidades y problemas a la hora de la creación del Workflow. En cada nueva reunión se defendía un prototipo que incorporaba las mejoras de funcionalidad, siempre teniendo en cuenta el alcance del proyecto. Finalmente se diseñaron unos prototipos que consideran las principales funcionalidades y cumplían los requisitos.

Durante las iteraciones y revisiones de los prototipos se acordaron un conjunto de casos de uso que llevaría a cabo el usuario, dichos casos responden a todos los requisitos y atienden a los prototipos de forma que se puede obtener un flujo de funcionalidad que permite modelar los Workflows de forma intuitiva.

Una vez conseguida la funcionalidad a cumplir se revisó y modeló el conjunto de datos que respondieran a los criterios de que se buscaba. Este modelo por su parte fue modificado y mejorado durante las iteraciones.

Para completar el modelo de datos fue necesario especificar la lógica de negocio del sistema: Lógica de negocio del módulo, lógica de control de procesamiento de datos, lógica de interacción entre los objetos de las entidades del modelo y la lógica de acceso a los datos de la base.

Con el modelo completo y la lógica descrita para que el módulo sea soportado por el motor fue necesario realizar algunas modificaciones en el motor para que el módulo esté completamente acoplado.

Con las fases del análisis y diseño completas se inició la implementación de las diferentes partes que compondrían el módulo.

Se concluye con los resultados obtenidos tras todo el proceso y se detalla el trabajo necesario para completar el módulo.

ÍNDICE

Resumen	1
Índice	2
1. Introducción	4
1.1 Contexto	4
1.2 Origen	4
1.3 Objetivo.....	4
1.4 Alcance	4
1.5 Metodología	5
1.6 Tecnologías utilizadas	5
1.7 Documentación	5
1.7.1 Memoria.....	5
1.7.2 Anexos	5
2. Fases del proyecto	6
2.1 Fase de análisis.....	6
2.1.1 Toma de requisitos	7
2.1.2 Diseño de prototipos.....	9
2.1.3 Funcionalidad del módulo.....	10
2.1.4 Casos de uso.....	11
2.2 Fase de diseño	12
2.2.1 Revisión y cambios del modelo de datos	12
2.2.2 Obtención de la lógica de negocio	13
2.2.3 Obtención de la lógica de objetos	14
2.2.4 Obtención de la lógica de acceso a BD. Repositorios	15
2.2.5 Funcionalidad añadida al motor de Workflows.....	16
2.2.6 Arquitectura del módulo	18
2.3 Fase de implementación.....	20
2.3.1 Front-end	20
2.3.2 Back-end.....	21
2.3.3 Motor de Workflows	22
3. Resultados obtenidos	22
3.1 Control de Web Api.....	22
3.2 Módulo de Angular.....	23
3.2.1 Acceso al diseñador	23
3.2.2 Módulo de usuario	23
4. Conclusiones del proyecto.....	25

4.1	Conclusiones	25
4.2	Líneas futuras	25
5.	Glosario.....	27
6.	Bibliografía.....	29

1. INTRODUCCIÓN

Este documento es el resultado de los procesos de análisis, diseño y desarrollo para la implementación del módulo¹ de modelado de Workflows. En este se describen las diferentes fases desarrolladas y los resultados obtenidos para resolver la problemática que se plantea.

1.1 Contexto

Endalia es una empresa que basa su actividad en el desarrollo de software que permite la gestión de las operaciones que tienen como base los recursos humanos: cálculo de nóminas, contratación, comunicación interna, gestiones de la estructura organizativa, flujos de trabajo, entre otros. Una de las actividades de las que se encarga la empresa es el desarrollo de Workflows. Un flujo de trabajo es conjunto de formularios que gestionan datos introducidos y ejecuta operaciones con ellos, gracias a ellos se pueden automatizar algunos de esos procesos. Para conseguir automatizar dichos procesos el software que oferta la empresa, *EndaliaHR* [3], tiene un módulo que emplea un motor de ejecución de Workflows, que es el núcleo del módulo. Gracias a este motor es posible interpretar el modelo de un Workflow que es un conjunto de datos que indican los estados, participantes y la información sobre ese Workflow siguiendo un modelo de datos y en base a ello generar una interfaz para trabajar con él.

1.2 Origen

El proyecto surge de la necesidad de crear Workflows. Es necesario diseñar y modelar los Workflows pero no existe una herramienta que permita hacerlo de forma intuitiva.

En un plano técnico existe el problema de la reutilización del código entre Workflows. Dos Workflows aunque parecidos pueden no estar ejecutando el mismo código y cada cual tiene cargado el suyo propio, el módulo intenta evitar esta situación unificando el código de las operaciones a ejecutar.

El modelo que se suele transmitir para generar un Workflow suele ser un problema para el cliente externo que hace uso de ese Workflow. Generalmente este cliente, que necesita el Workflow a desarrollar, no tiene una forma adecuada y reglada para comunicar el proceso que busca. Este proyecto busca analizar la forma más adecuada de responder a esa necesidad.

1.3 Objetivo

Lo que se plantea para este proyecto es conseguir un módulo que esté acoplado dentro del software que oferta la empresa. Este módulo debe estar contenido dentro de la web que compone dicho producto. El módulo debe tener lo necesario para modelar un Workflow, entendiendo por modelado:

- La capacidad de definir sus fases³.
- Definir los formularios y responsables asociados a cada fase.
- Definir las acciones que se deben ejecutar durante el avance y la comunicación asociada a dicho avance.

En cuanto al proyecto se busca desarrollar una lógica capaz de cumplir con todos los requisitos, pero sin estar plenamente ligado al sistema en el cual va a estar integrado, es decir, una lógica basada en un modelo de tres capas⁴ que puede ser implementado con independencia de la tecnología que se usara.

1.4 Alcance

Hay que definir hasta qué punto se va a llegar del objetivo propuesto. Se busca estudiar primeramente un modelo conceptual de lo que se va a desarrollar, explicar la lógica que sigue ese modelo conceptual con respecto al usuario, la lógica propia del módulo y la lógica que de los datos introducidos. Además, también se

busca establecer el diseño arquitectónico del módulo para ser acoplado al producto. Con la parte fundamental de análisis y diseño finalizada, se inicia el proceso de implementación, sin completar el desarrollo del módulo al no estar comprendido dentro del alcance de este proyecto, pero sentando las bases necesarias para dar continuidad a la implementación del mismo en etapas posteriores.

1.5 Metodología

El desarrollo se llevó a cabo siguiendo metodología Ágil. Los cambios en el diseño del módulo, datos y funcionalidad no están exentos de cambio en cualquier momento, con un sustento lógico obviamente. Cada lote de cambios realizados tras las revisiones genera una nueva versión hasta conseguir un proyecto estable. Las fases por desarrollar en el tiempo se especificaron en un diagrama de Gantt, recogido en el apartado 2, Fases del proyecto, con tres secciones: análisis, diseño e implementación resumidos en este proyecto.

1.6 Tecnologías utilizadas

Se han utilizado las siguientes tecnologías para el desarrollo del sistema.

- Microsoft .NET Framework 4 [13] y Angular 2 en su versión 5 como plataformas y soportes tecnológico.
- C# y TypeScript como lenguajes de programación.
- Microsoft SQL Server 2012 [15] como sistema gestor de base de datos
- NHibernate como librería para el mapeo de las entidades y relaciones de base de datos para .NET
- Microsoft Team Foundation Server como herramienta de gestión y control de versiones del proyecto.
- KendoUI [14] como librería para Angular 2 que contiene controles UI y utilidades para el desarrollo de la interfaz.
- Windows 8 como sistema operativo usado en las fases de implementación, pruebas e instalación.
- Microsoft Office 2013 para generación de documentación

1.7 Documentación

En este apartado se describe el contenido que se desarrolló para documentar todo el proceso de análisis del proyecto.

1.7.1 Memoria

La documentación principal que contiene un resumen de todo el análisis hecho además de contener especificaciones del proyecto.

1.7.2 Anexos

Los siguientes anexos amplían y muestran el desarrollo de algunas fases del proyecto. Se encuentran referenciados a lo largo del proyecto:

- I. **Prototipos:** Diseños conceptuales de las vistas que componen el módulo de Workflows.
- II. **Casos de uso:** Casos de uso para la funcionalidad del módulo.
- III. **Diagramas de secuencia:** Diagramas que especifican la lógica de negocio del módulo.
- IV. **Lógica de control:** Especificación de la lógica que controla el módulo.

- V. **Especificación del modelo de datos:** Especificación de los atributos y su tipo de cada objeto que intervienen en las operaciones lógicas.
- VI. **Lógica de objetos:** Especificación de la lógica de la interacción entre objetos del modelo de datos.
- VII. **Lógica de acceso a datos:** Especificación de la lógica de las clases que intervienen directamente con la BD².

2. FASES DEL PROYECTO

En esta sección se detalla el desarrollo de las tres fases de análisis, diseño y desarrollo del proyecto, cada una de ellas siguió un proceso cíclico de revisiones constantes hasta conseguir un resultado aceptable.

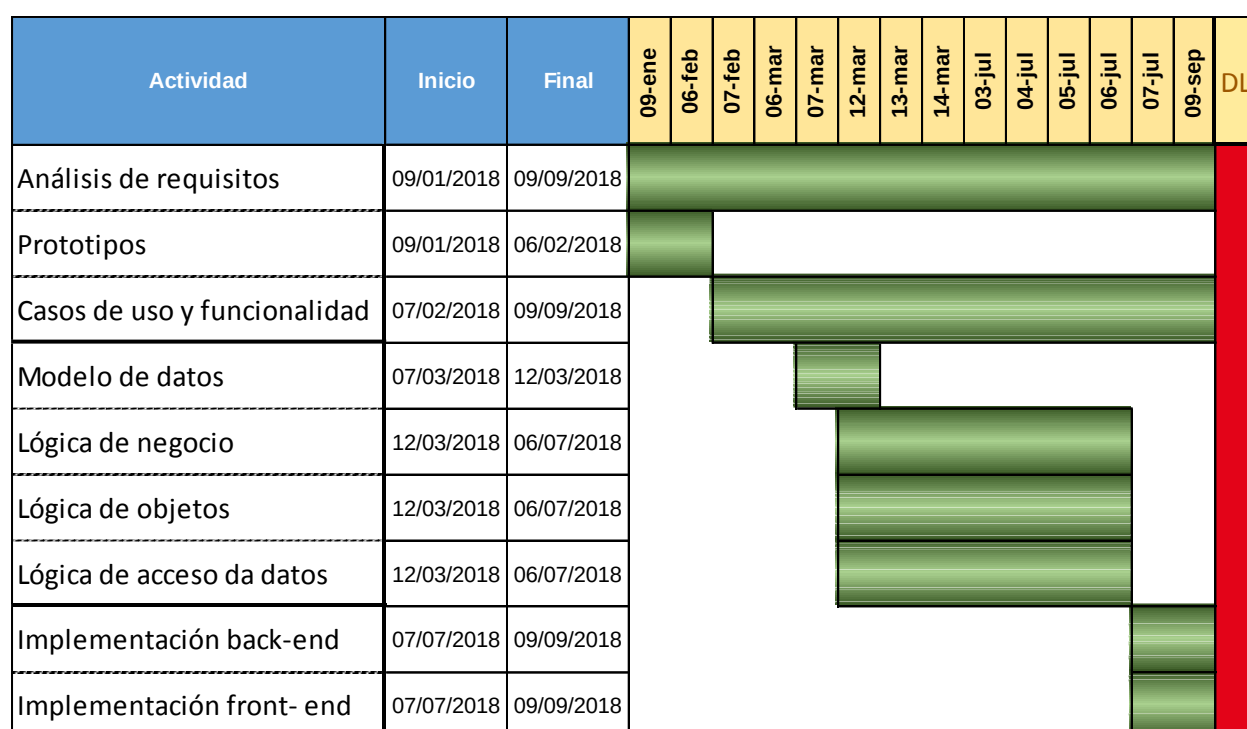


Figura 1. Diagrama de Gantt de las fases del proyecto

El diagrama de la *Figura 1* muestra el proceso que se proyectó para el desarrollo a lo largo del tiempo de las fases del proyecto. Cada una de ellas siguiendo la metodología que consta de diferentes revisiones, el esquema refleja la obtención de versiones estables de las fases, es decir, el momento en los que se pretendía tener un sistema que ya no sufriera cambios bruscos. Se hace notar que algunas de las actividades desarrolladas fueron proyectadas a lo largo de todo el tiempo de desarrollo o en periodos que no terminan hasta el fin del proyecto; esto es así porque constan de una continua revisión y modificación.

2.1 Fase de análisis

Se describe el proceso de estudio del funcionamiento del módulo con respecto al usuario.

2.1.1 Toma de requisitos

Para la toma de requisitos se organizaron reuniones con el equipo de desarrollo encargado de analizar y diseñar los Workflow, es decir, el equipo que se encarga de establecer que fases son necesarias, que campos son necesarios, que secuencia siguen los pasos y si es necesario alguna modificación, obviamente siguiendo los requisitos de cliente. A través de las diferentes reuniones con el equipo se determinaron los requisitos que es necesario que cumpla el módulo, algunos de ellos fueron añadidos, modificados o eliminados conforme el avance del proyecto continuó. Otros se proyectan a futuro y no constan en este proyecto por alcance.

Se determinaron los siguientes requisitos como los principales del sistema a abarcar:

RF - 1. Visualizar Workflows existentes.

Listar todos los Workflow a los que se tienen acceso.

RF - 2. Registrar los Workflow que no se puedan modificar por el módulo.

Debido a su complejidad⁴ es posible que algunos Workflows no puedan ser modificados por el módulo. Son solo modelables los Workflows lineales (Workflows cuyos pasos no tengan múltiples pasos para continuar), Workflows cuyo responsable sea el propio usuario o tenga el permiso para acceder a él.

RF - 3. Crear Workflow.

Poder crear Workflows nuevos. Dándole un nombre, descripción, objetivo y asignándolo a un área.

RF - 4. Asignar el Workflow a un área.

Se puede cambiar el área a la que pertenece el Workflow.

RF - 5. Modificar Workflows.

Sobre un Workflow existente es posible obtener su modelo: características principales, fases, campos, responsables de los pasos, formularios y comunicaciones para poder editarlo.

RF - 6. Crear pasos para el Workflow.

Poder crear pasos para el Workflow. Es posible definir cada fase (paso) del Workflow. Los pasos del Workflow vienen definidos por el nombre del paso, objetivos, descripción. Cada paso se corresponde con un formulario del Workflow. Cuando se crea un paso se crea un formulario para el paso.

RF - 7. Definir el orden de los pasos del Workflow.

Poder determinar y modificar la secuencia de pasos para el Workflow. Dado que los Workflow son siempre lineales (el alcance del proyecto establece que son lineales) y están definidos para realizar solo modificaciones que cumplen este modelo.

RF - 8. Poder crear nuevos campos.

Poder crear nuevos campos con nombre, tipo y descripción si es necesaria.

RF - 9. Todos los campos deben estar disponibles en los formularios.

Cada formulario del Workflows debe tener accesibles todos los campos y poder configurarlos.

RF - 10. Poder editar las características de los campos. Modificar formularios.

Para cada formulario debe ser posible determinar las características del campo. Las características de los campos deben ser editables con el módulo.

RF - 11. Poder seleccionar las acciones que se van a ejecutar.

Para cada paso es posible determinar qué acciones se van a ejecutar cuando se avance de fase. Cada paso puede ejecutar diferentes acciones. Las acciones pueden encadenarse para poder crear acciones más complejas, unas acciones son la entrada de otras.

RF - 12. Catálogo de acciones seleccionables.

Debe estar disponible un catálogo de acciones seleccionables y parametrizables para cada paso que se ejecuta cuando se avance de fase.

RF - 13. Parametrizar acciones.

El usuario debe poder usar los campos que se definieron antes para parametrizar las acciones con los valores de dichos campos.

RF - 14. Encadenar acciones.

Las acciones son simples. Es decir, sólo ejecutan instrucciones básicas. El resultado de una acción se puede usar en otra de tal forma que se pueden construir otras acciones más complejas a partir de las más básicas.

RF - 15. Poder seleccionar los usuarios responsables.

Se puede asignar los responsables según el tipo: Responsables según permiso de acceso o responsable.

RF - 16. Crear los correos que se envían en cada paso.

Cada paso debe poder enviar uno o varios correos a uno o varios destinatarios, deber ser posible elegir a quien va dirigido cada mail del paso del Workflow, además de componer su cuerpo y añadir un asunto. Los correos son de cancelación, confirmación o rechazo de la fase en la que se encuentra el Workflow.

RF - 17. Validar el Workflow.

El modelo de Workflow modificado con el diseñador no debe contener errores antes de poder guardar.

RF - 18. Eliminar Workflow.

Se debe poder eliminar un Workflow. Todo su modelo debe poderse eliminar, al menos los creados por el mismo módulo deben ser eliminables. Al menos se debe advertir que se tienen instancias abiertas. Se traduce por eliminar poner el Workflow en un estado de sólo lectura.

RF - 19. Eliminar pasos.

Se debe poder eliminar fases, pasos. La eliminación del paso pasa por la validación de la eliminación. En caso de que haya instancias del Workflow abiertas se avisa de esta eliminación y se deja hacer. Se deja el análisis de la situación al usuario.

RF - 20. Eliminar campos.

Se puede eliminar campos del Workflow.

2.1.2 Diseño de prototipos

Teniendo en cuenta los requisitos funcionales se realizaron los respectivos diseños de la vista que cumplen con ellos. La especificación y diseño de estas vistas se generaron siguiendo el patrón de trabajo descrito en el apartado 1.5, Metodología. Siguiendo un proceso iterativo que añadía a cada revisión los cambios convenidos en anteriores encuentros con el equipo desarrollador de Workflows. Esto ayudó a acotar la funcionalidad y a orientar el modelo conceptual de la herramienta.

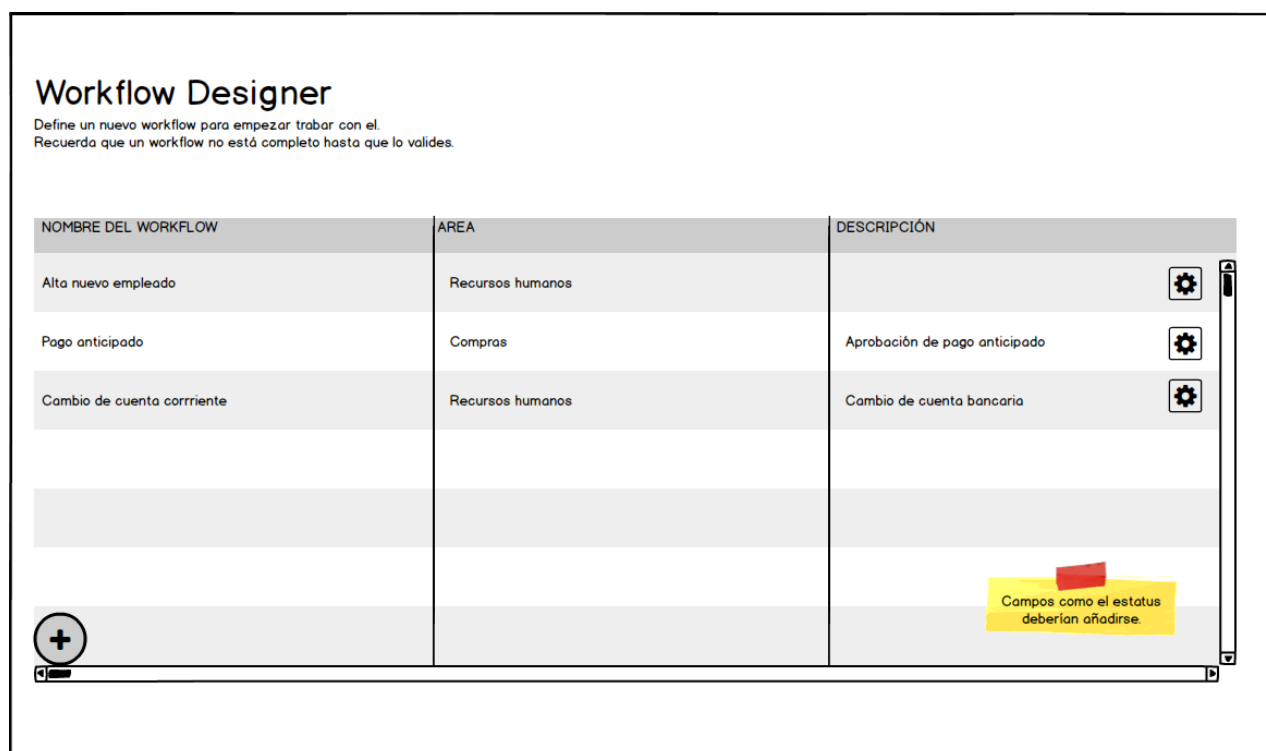


Figura 2. Prototipo de la vista principal del diseñador, lista de Workflows.

Pasos
Campos
Acciones
Responsable
Formularios
Correos
Finalizar

Formularios

Indica qué campos se van a solicitar al usuario en cada paso. Los campos definidos previamente están disponibles en todas las fases del proceso de workflow pero no necesariamente serán visibles. El orden, visibilidad, obligatoriedad, edición, etc. también son editables.

Nombre_del_paso_1	Nombre_del_paso_2	Nombre_del_paso_3
Nombre	Nombre	Nombre
Apellido	DNI	Apellido
DNI	Apellido	Fotografía
Salario	Salario	Salario
Fotografía	Fotografía	DNI
Destinatario correo	Destinatario correo	Destinatario correo

La visibilidad y algunas características como "Read only" se puede gestionar a primer nivel

Siguiente

Figura 3. Prototipo de la vista de formularios, gestión de formularios.

Lo que se buscaba con esta tarea era realizar un diseño para el equipo interesado, siempre teniendo en cuenta las limitaciones del proyecto además de orientar el resto del trabajo, pasando de una imagen mental a una conceptual que cumpliera con los patrones de diseño. Los diseños producidos tras las diferentes revisiones se muestran en el ANEXO I, Prototipos.

Considerando las propuestas del usuario para el diseño se tuvieron en cuenta dos patrones importantes a incluir:

El diseñador tiene una estructura siguiendo el patrón Wizard (Asistente). Sobre el patrón: *“El objetivo de este patrón es guiar al usuario paso a paso por medio de la interfaz gráfica para realizar tareas en un orden prescrito”* [1]. Acentuamos el hecho de que el diseñador es una ayuda que conoce lo que se debe hacer mejor que el usuario.

Se utiliza también el patrón pestañas: *“Este patrón es de utilidad cuando queremos mostrar contenido agrupado ya que permite encontrar información sin cambiar la vista actual dividiendo así los datos en diferentes secciones”* [2] en este caso las diferentes partes del diseñador pertenecen a partes del Workflow pero puede ser necesario volver atrás en el avance, en cualquier momento del modelado, para poder realizar cambios en el diseño o arreglar errores.

2.1.3 Funcionalidad del módulo

Con los prototipos validados y puesto en común con el equipo desarrollador de Workflows es posible describir la funcionalidad del módulo para el proceso de crear y modelar un Workflow. Se recuerda que es posible seguir la secuencia de vistas (wizard), una detrás de otra, pero que permite retroceder y avanzar a cualquiera de ellas en cualquier momento. La funcionalidad por vista es la siguiente:

- **Vista Workflows**

El primer paso es seleccionar un Workflow a modificar o se crea uno nuevo. Si el Workflow es un nuevo Workflow se crea también una secuencia con un primer paso obligatorio como flujo básico.

- **Vista Steps**

Se visualizan los pasos que componen el Workflow. Los pasos son secuenciales y es posible modificar su orden dentro del esquema, además de añadir nuevos, para cada uno de los pasos hay asociado un formulario más adelante.

- **Vista Fields**

Una vez que se tengan listos los pasos que sigue el Workflow se continúa con el modelado añadiendo los campos donde se introducirá la información a la hora de la ejecución del paso, los campos creados en esta vista estarán disponibles en cada uno de los formularios asociados a los pasos para su uso.

- **Vista Forms**

Después de que se definan los pasos y los campos hay que gestionar los formularios asociados a los pasos. Los formularios se generan para cada paso y tienen disponibles todos los campos, se puede gestionar su visibilidad, su obligatoriedad y si son campos editables o no.

- **Vista Actions**

Ahora es necesario asociar al avance entre los pasos del Workflow las acciones que se ejecutan en cada momento del avance. Las acciones son parametrizables: esto quiere decir que cada acción asociada puede hacer uso de los campos antes definidos como parámetros para su ejecución, no todos los parámetros son capaces de recibir todos los tipos de parámetros. Cada acción genera un resultado o no, esto depende de la acción que se seleccione. El producto de la acción puede introducirse dentro de un campo previamente definido. Toda esta funcionalidad permite encadenar diferentes acciones conformando unas más complejas. Las acciones provienen de un catálogo de acciones definidos previamente y externos a la lógica de la aplicación.

- **Vista Responsibles**

Cuando se hayan definido las acciones se elige el tipo de responsable al que se le otorgan privilegios de editar el formulario del paso del Workflow además de otorgarle permisos de cancelar, guardar manualmente los datos del paso o finalizar el Workflow.

- **Vista Mails**

La comunicación del avance de la fase del Workflow se realiza mediante correo. Para cada paso se pueden asociar varios correos. Cada correo tiene un asunto y un destinatario y puede ser de cancelación o rechazo. Los correos son editables: usando los campos definidos anteriormente se puede dar un destinatario, un asunto y editar el cuerpo.

- **Vista Finalize**

Finalmente, el Workflow hasta este momento modelado se valida para poder guardarse. Si se ha producido un error se muestra el error y si es posible se recomienda una solución. Si el Workflow es válido se guardará.

2.1.4 Casos de uso

Conforme se estudiaba la funcionalidad del sistema, para completar los requisitos, se extrajeron los diferentes casos a los que se enfrenta el usuario durante el proceso de modelado del Workflow. Los casos de uso que se determinaron son los necesarios para que se complete el modelado del Workflow.

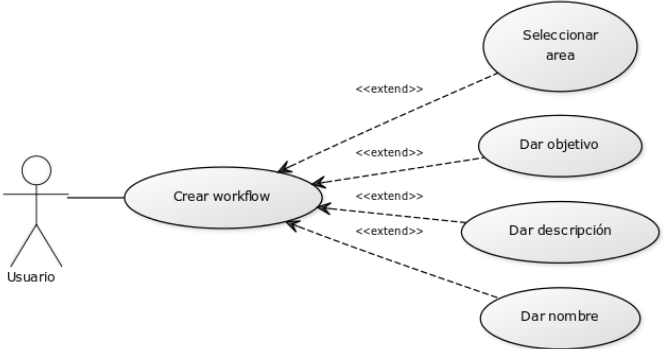
CU2. Crear un Workflows	
Descripción	El usuario crea un nuevo Workflow: proporciona un nombre, propósito del Workflow, un área y una descripción. El sistema corrobora si no existe el Workflow en el área y se añade como nuevo siendo posible modelarlo.
Flujo	• Añade nuevo Workflow: se proporciona nombre, descripción, área y objetivo. • El usuario puede cancelar la creación. • Se crea un nuevo Workflow.

Figura 4. Caso de uso para la creación de un nuevo Workflow.

Los casos de uso se encuentran descritos en el ANEXO II, Casos de uso.

2.2 Fase de diseño

El objetivo es obtener un diagrama de clases junto a sus métodos y atributo. Para ello se sigue el flujo de los casos de uso definidos y el prototipo. Se busca obtener:

- Modelo de datos.
- Lógica que controla el comportamiento del módulo, lógica de negocio.
- Lógica que sigue los objetos del modelo de datos
- Lógica que accede a la persistencia de los datos.

La obtención de las lógicas se realizó en un proceso simultáneo como se puede ver en la Figura 1 del apartado 2, Fases del proyecto. Se especificaron los procesos conforme se seguía el diálogo entre clases. Todo este proceso fue realizado siguiendo la metodología, realizando diferentes revisiones para detallar algún aspecto interesante del proceso.

Con esto se consigue una lógica independiente de la arquitectura o tecnologías que se usarán más adelante.

2.2.1 Revisión y cambios del modelo de datos

Antes de la obtención de la lógica es necesario establecer que entidades intervienen y cuáles de ellas son las que cubren las operaciones del módulo. Se revisó el modelo de datos ya existente en la base de datos que contiene la información sobre los Workflow y siguiendo los casos de uso y prototipos diseñados se extrajo un

grupo de entidades y sus relaciones que componen el modelo de datos representado en el siguiente diagrama, Figura 5.

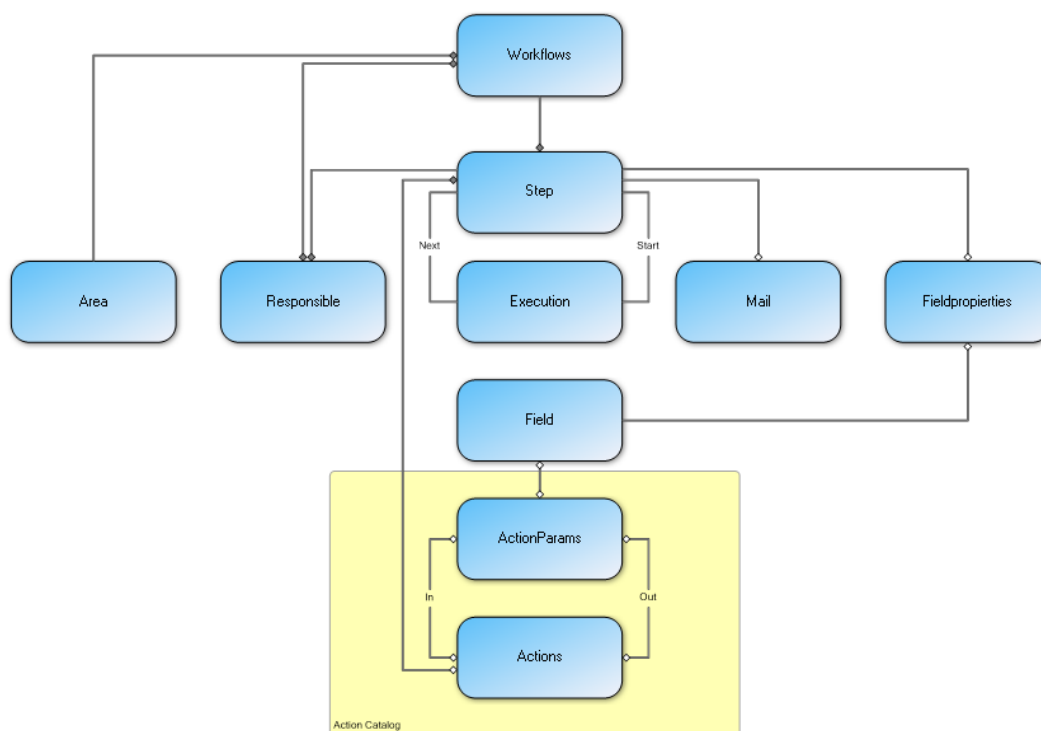


Figura 5. Esquema de entidades y sus relaciones que conforman el modelo de datos del sistema.

Gracias a este modelo se pudo descubrir que es necesario un catálogo de acciones. A forma de lista de acciones, el catálogo es un menú que debe contener la especificación, más no la implementación, de las diferentes acciones que va a ejecutar el Workflow. En el diagrama, resaltado, se muestra la parte del modelo de datos que es necesario añadir para que el motor pueda interpretar y ejecutar las operaciones.

La implementación del catálogo se trata más adelante, en el apartado 2.2.5, Funcionalidad añadida al motor de Workflows.

2.2.2 Obtención de la lógica de negocio

Para estudiar este apartado se planteó un modelo clásico de tres capas, que no ha tenido que estar completo pues la parte visual no se consideró necesaria. Lo que se buscaba ahora es separar la vista del control y del modelo de datos; para la división en capas se utilizaron como forma de ayuda diagramas de robustez [4], que no constan y que fueron solo auxiliares en el proceso de obtención de los diagramas de secuencia donde se refleja de verdad la lógica del módulo.

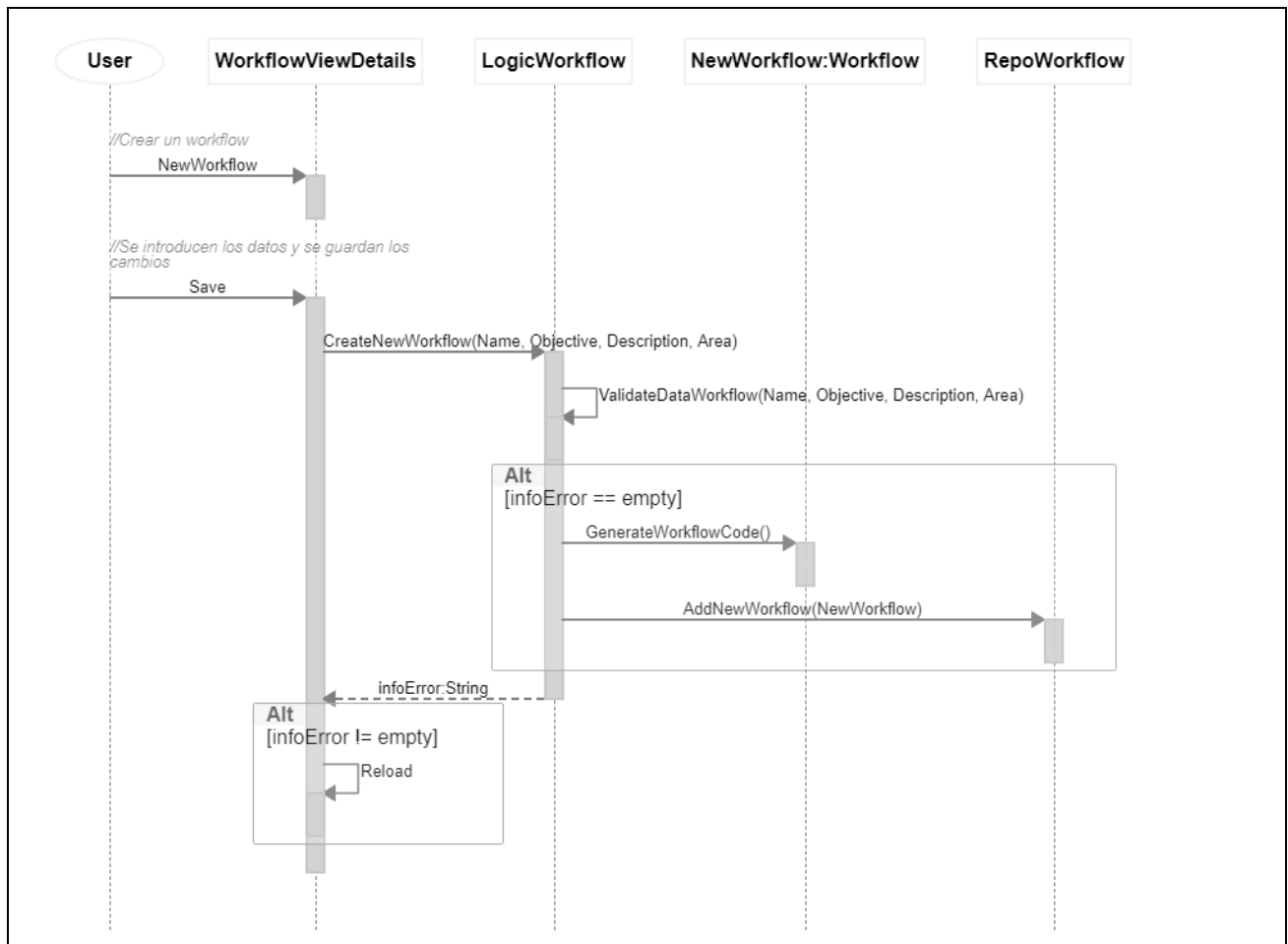


Figura 6. Diagrama de secuencia para el CU 1. Creación de un nuevo Workflow

El diagrama de la Figura 6 muestra el diálogo entre las diferentes clases; peticiones de datos y alguna lógica adicional. El origen de algunas clases del diagrama se verá posteriormente, apartado 2.2.4, Obtención de la lógica de acceso a BD. Repositorios.

En este punto ya es posible seguir el flujo de la lógica interna del módulo: La interacción de un usuario, la interacción entre los objetos, la interacción con la lógica controladora de los procesos del módulo y su persistencia. Para cada caso de uso se obtuvo un diagrama de secuencia recogidos en el ANEXO III, Diagramas de secuencia.

Con todo esto se pretende distinguir las clases "Logic"⁵ que conforman la lógica de control, que son aquellas que realizan operaciones meramente procedimentales y que por lo tanto no involucran lógica de los objetos. Los procesos de esta lógica se encuentran especificados en el ANEXO IV. Lógica de control.

2.2.3 Obtención de la lógica de objetos

La lógica de los objetos, la que involucra solo la interacción de los objetos, se muestra en los diagramas de secuencia del módulo, gracias a esto es posible obtener los atributos que completan los objetos. Los atributos están especificados en el ANEXO V. Donde se describe para cada una de las clases obtenidas en el diagrama de la Figura 5 del apartado 2.2.1, Obtención del modelo de datos, sus atributos y para cada uno de los atributos su tipo.

Con el tipo de datos especificados y el diálogo entre las clases se conoce la lógica que siguen los objetos con respecto a los controles procedimentales y procesos propios del objeto. Para cada método obtenido se especificó la lógica que debe seguir, los parámetros de entrada de los que debe hacer uso y lo que se obtiene tras el proceso. La especificación se puede encontrar en el ANEXO VI, Lógica de objetos.

2.2.4 Obtención de la lógica de acceso a BD. Repositorios

Al igual que el resto de la lógica del módulo descrita, se sigue el diálogo de clases que describen los procesos. Este tipo de lógica es la que tocaría directamente la capa que da acceso a datos. Mientras todo lo anterior actuaba sobre el modelo de datos, o sea el TAD⁶ Workflow, que se mantenía en memoria, aquí se accede a los datos para persistirlos o leer la persistencia.

La lógica de estos accesos está contenida en sus respectivos DAO⁷ y los tipos de datos utilizados son los de la base de datos. A estas persistencias nos referiremos como repositorio. Estos repositorios pueden contener grandes cantidades de datos y es necesario filtrarlos u obtener datos que no son accesibles desde otro punto del módulo.

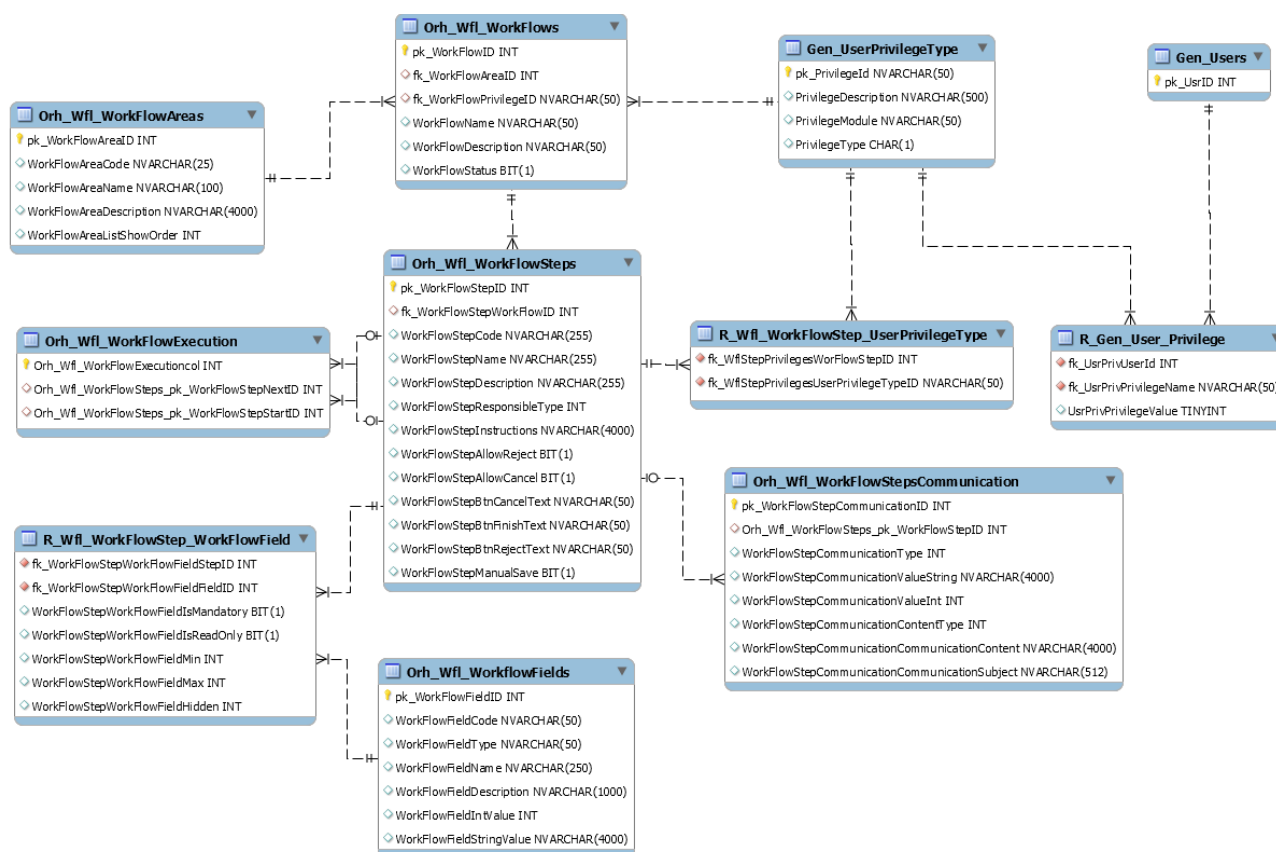


Figura 7. Esquema de datos E-R de la base de datos que mapean los repositorios del Workflow.

La Figura 7 muestra el esquema relacional que se ha extraído de la BD y con el que trabajan los repositorios de acceso a datos. La especificación de las lógicas añadidas al acceso a datos se encuentra descrita en el ANEXO VII, Lógica de repositorios, donde los tipos de datos referidos son los de la persistencia de datos.

2.2.5 Funcionalidad añadida al motor de Workflows

Hasta este punto se hablaba de datos abstractos y lógica de control de los mismos todo este proceso se desarrolló considerando el motor de Workflows del sistema. En este momento se consideró las modificaciones que serían necesarias para que dicho motor soportase los nuevos modelos creados por el diseñador. Durante todos los procesos se intentó mantener el motor como una guía. El motor marcó las pautas, en todo momento se intentó mantener una estructura que el motor leyera sin mayores problemas, sin embargo y atendiendo a las necesidades del equipo de desarrollo de Workflows y requisitos del sistema, fue necesario añadir una modificación al motor para que pudiese interpretar los campos de tipo “ACTION”.

Los campos de tipo “ACTION”, en adelante “acciones”, representan aquellas operaciones necesarias para llevar a cabo los diversos procesos que puede llevar a cabo un Workflow, en el contexto operaciones como: registrar usuarios, añadir solicitudes de ausencia etc. Haciendo uso de los datos introducidos en alguno de los formularios del Workflow.

A continuación, se describen las modificaciones del modelo de datos de la base relacional y del motor para la interpretación de acciones:

- **Catálogo de acciones**

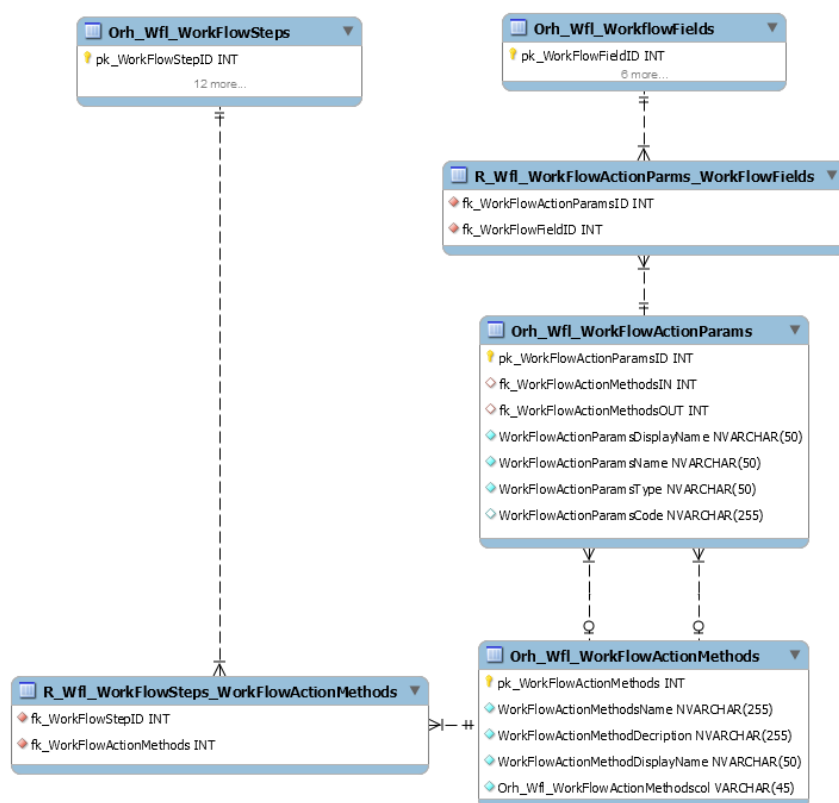


Figura 8. Modelo E-R de la persistencia del catálogo de acciones.

La Figura 8 muestra el modelo E-R del catálogo de acciones resaltado en la Figura 5 del apartado 2.2.1 Obtención del modelo de datos. Este catálogo de acciones contiene la especificación de una acción que se puede ejecutar al avanzar un paso del flujo. La especificación de la acción contiene lo necesario para

distinguirla de otra, es decir, la información que podría necesitar para el usuario que use el módulo sin tener que recurrir al código, así que cada una de las acciones debería contener al menos una descripción que la especifique su uso y un nombre distintivo que mostrar.

Las acciones de los pasos hacen uso de la información introducida en el formulario asociado al paso del Workflow y devuelven información a partir de lo introducido. Para conseguir esta funcionalidad es necesario parametrizar las acciones del Workflow, hay que aclarar entonces que las acciones pueden, o no, tener que ser parametrizadas para su uso.

Para la parametrización de una acción es necesario ligar los parámetros de entrada y salida a la acción si es necesario. En caso de usar una acción del Workflow que requiera parametrización sería necesario proporcionar un valor de entrada al parámetro, no todos los valores deben ser permitidos pues solo valores del mismo tipo son tomados por la acción y si se usa una misma acción en diferentes pasos para cada paso la acción puede tomar valores diferentes. El valor de los parámetros es una referencia al campo del formulario del paso.

Al nivel de motor las acciones son sólo un tipo de campo no visible, de solo lectura y obligatorio del tipo "ACTION". Dicho campo contiene la referencia como valor a alguna de las acciones que se va a ejecutar. Si la acción está parametrizada el motor se encarga de obtener el valor a partir de la referencia al campo del formulario. El contenido dentro de los campos (el valor) de los campos son los valores que toman los parámetros a la hora de que la acción se ejecute. El motor se encarga en este punto de obtener también la implementación de la acción a partir de la referencia al método y ejecutarlo. Así pues, una acción puede tener muchos campos de entrada y es capaz de devolver muchas salidas. Por el alcance del proyecto solo se tienen en cuenta las acciones de tipo funcional, aquellas que solo tienen un parámetro de salida y las procedimentales, aquellas que no tienen campos de salida.

Cuando el motor obtiene los valores de los parámetros puede encontrar que uno de los valores referencia a otra acción y busca completarla al igual que lo hizo con la primera consiguiendo así encadenar las acciones, conformando una más grande.

Toda esta lógica debe estar contenida en el motor para cuándo se ejecute un campo de tipo "ACTION".

- **Filtro de acciones**

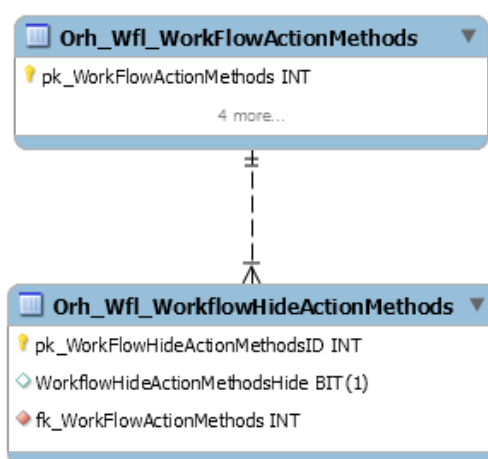


Figura 9 Tabla para el filtro de acciones.

Se necesitaba que la ejecución de las acciones fuese configurable, pues no queríamos que todos los usuarios del módulo tuviesen acceso a todas las acciones. Se desarrolló para ello un filtro para el diseñador que sirviera para configurar el listado de acciones disponibles en la herramienta. El funcionamiento es muy sencillo: se incluyen en la tabla del filtro una referencia a la acción que se va a ocultar y se oculta haciendo uso de un bit a forma de *flag*. En caso de que no esté referenciada simplemente se muestra en el catálogo. En caso de que este referenciada, pero con el *flag* desactivado, se muestra en el catálogo. Finalmente, en caso de que esté referenciada y con el *flag* activado, no se muestra en el catálogo.

La interacción de este filtro se puede observar en el ANEXO III, Diagramas de secuencia, para el caso de uso 1.4.1, Visualizar catálogo de acciones.

2.2.6 Arquitectura del módulo

El desarrollo de esta fase de diseño hasta este momento no atendía a ninguna particularidad y gozaba de independencia de implementación pues nunca fue necesario enmarcarla dentro de una arquitectura para que respondiera satisfactoriamente a los requisitos, pero una de las necesidades vistas era la de encuadrar la herramienta dentro del producto, *EndaliaHR*. Teniendo en mente lo anterior se planteó una distribución de la aplicación que consiste en un par básico de **cliente-servidor** para poder integrarlo en la versión web del producto.

- Cliente

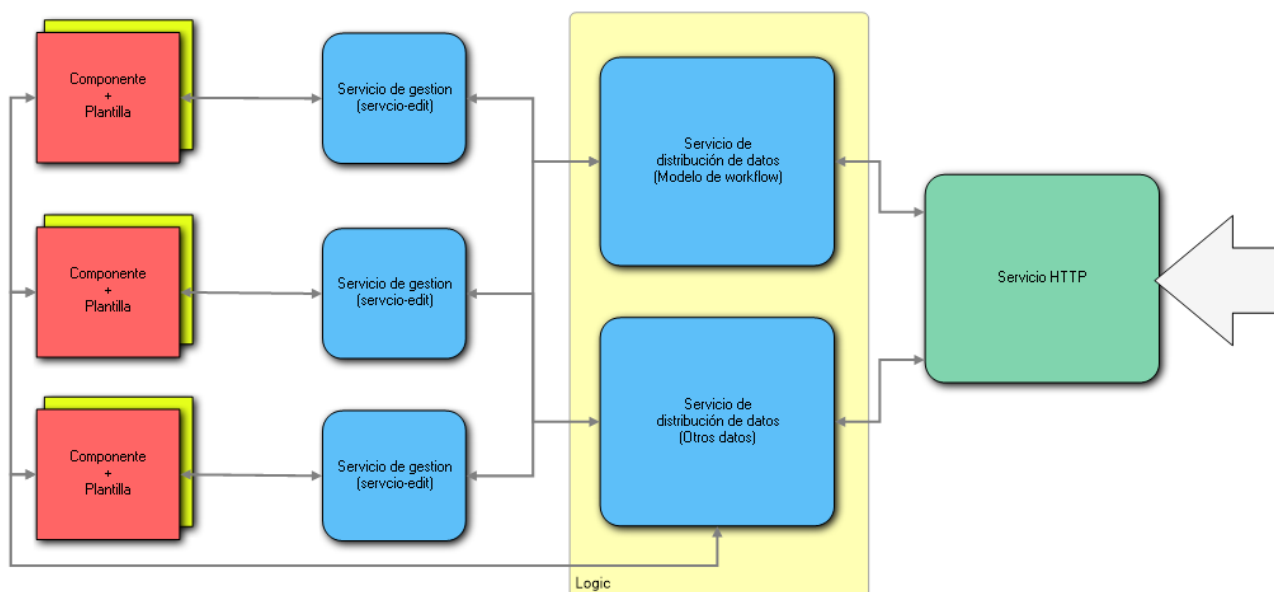


Figura 10. Arquitectura del cliente del módulo

Se necesitaba que el cliente de la aplicación fuese accesible desde navegador web. Por esta razón y teniendo en cuenta las capacidades reusables del sistema que había y la capacidad de acoplar la herramienta solo añadiendo los archivos de compilación se optó finalmente por Angular⁸ como la tecnología. Para ser preciso se usa una arquitectura MVC basada en el mainframe de Angular 2.

Con el desarrollo del módulo en Angular 2 en mente se estableció por lo tanto que el acceso a datos del cliente se realizaría a través de servicios web que leen y escriben datos formateados mediante llamadas al servidor.

La introducción de datos y su visualización se produce en la dupla Componente-Template (o plantilla), mediante el paradigma de programación reactiva. El Componente recibe los datos de los servicios y los sirve para la plantilla. A su vez, la plantilla recibe las interacciones del usuario, modificaciones en los datos, que recoge el componente que son cambios que detecta el servicio de gestión (servicio-edit).

La gestión de los datos se realiza a través de controles, servicios en Angular, que son capaces de recibir datos desde el servidor y desde el servicio de gestión del componente y los distribuyen a lo largo de la aplicación, todo esto de lado del cliente.

El servicio que contiene el modelo distribuye los cambios que se produzcan en él (actualizando el modelo). Esto se consigue manteniendo suscrito dicho modelo a las modificaciones producidas por la vista, se actualiza cuando detecta cambios.

Mediante los servicios de conexión http se comunican los datos del modelo a almacenar en BD que finalmente recoge el servidor.

- Servidor

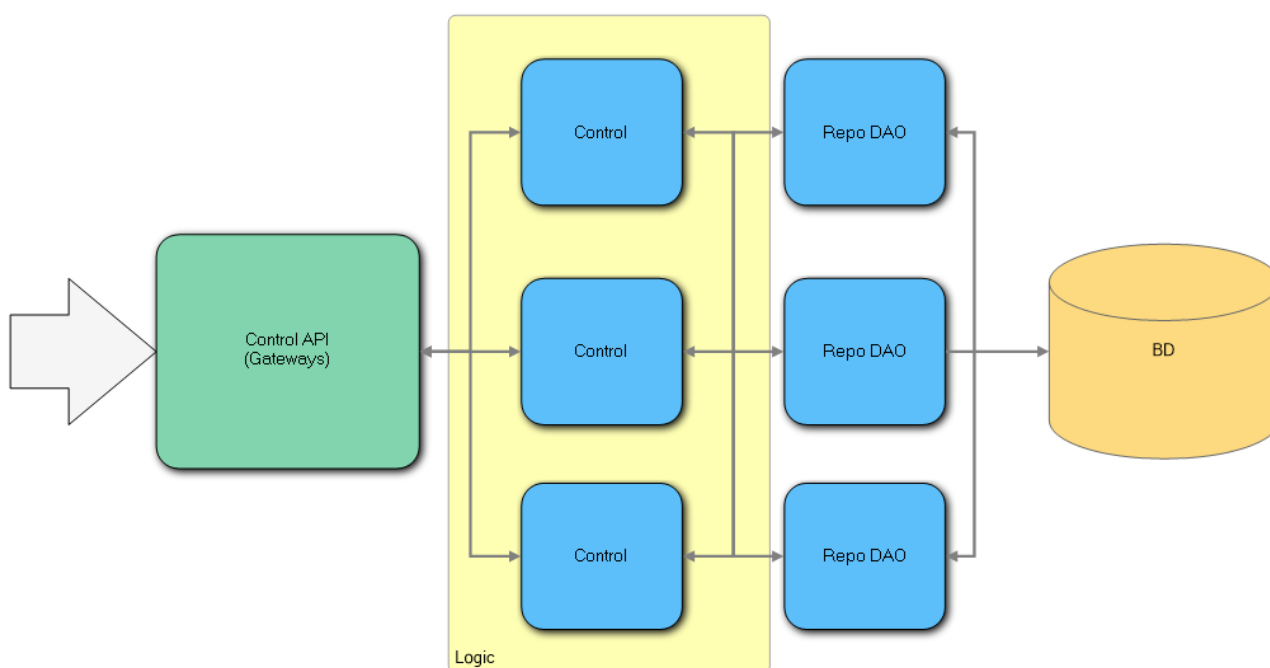


Figura 11. Arquitectura del servidor del módulo

Vista la arquitectura del cliente queda claro que no tiene acceso a los datos persistidos en BD, pues sólo usa el TAD para trabajar. Sería necesario por tanto servir los datos para que sean usados. Para realizar esto último ya se ha dicho que sería necesario el uso de servicios web, que proporcionarían el acceso a los datos.

La lógica que gestiona estos servicios está definida en uno o varios controles que implementan las *Gateway* o puertas de entrada a los que recurre la parte del cliente para obtener los datos (GET⁹). Estos controles también

reciben los datos del cliente cuando este último lo requiera y da acceso a la persistencia de datos (POST¹⁰, PUT¹¹).

El acceso a datos del servidor se realiza directamente desde BD cuyos datos han sido previamente mapeados y son accesibles desde cualquier proyecto permitido.

La idea principal de esta arquitectura era desligar la parte del cliente de la del servidor para no aumentar la carga del servidor. El servidor, la WebApi, es solo atacado en las ocasiones necesarias en la que la parte del cliente no tenga acceso o no pueda hacerse cargo de la lógica.

Vista la arquitectura que se va a usar, que responde a las necesidades que buscábamos cubrir queda un último punto que tratar: La distribución de la lógica de control dentro del sistema. Tanto en la Figura 10 como en la 11 se ha resaltado dos secciones: En el cliente, la sección de los servicios de distribución de datos y en el servidor las clases de control, estas clases conforman lo ya mencionado en el apartado 2.2.2, Obtención de la lógica de negocio, en el párrafo cuarto: "...clases 'Logic'...". Por lo tanto, según la arquitectura, la lógica descrita para estas clases debe estar distribuida de forma lógica dentro de la aplicación.

Observando el diálogo de la aplicación de los diagramas de secuencia anteriormente estudiados se observa un común entre algunos de los métodos de la lógica, la interacción con las persistencias de la BD. Para que las persistencias de la BD sean accesibles es necesario estar de lado del servidor. Gracias a esto último podemos distinguir siete métodos de la lógica que finalmente confirmarían las Gateway del servidor. El resto de la lógica se desarrolla en el lado del cliente.

2.3 Fase de implementación

Finalmente, con un diseño concluido y una especificación precisa de lo que se requiere se pasa a la fase de implementación que consiste en la programación del back-end y un front-end, servidor y cliente respectivamente. El back-end sería un control dentro de la compilación de la Web Api de la empresa y el front-end una compilación de un proyecto de Angular 2 accesible desde la versión web del producto, *EndaliaHR*. Completando de este modo el acoplamiento del módulo al sistema de la empresa.

Durante la implementación de la lógica se realizaron pruebas funcionales de cada uno de los métodos. En ocasiones siendo necesario realizar modificaciones para mejorar su funcionamiento además de comprobar el acople de aplicación. El código sigue las reglas de desarrollo de la empresa y se realizaron dentro de un entorno local.

2.3.1 Front-end

Para el front-end se inició un desarrollo de un proyecto de Angular 2, actualmente el proyecto usa Angular en su versión 5, Compilado con Angular Cli¹⁰ y que sigue las pautas de una arquitectura MVC. Se pretende mantener el módulo en una url del estilo:

<https://{dominio del cliente}/endaliaweb/workflows/designer/>

Y dentro de este apartado se acopla la compilación de angular que se ha traducido en este punto ya a Javascript y solo es necesario referenciarla en el cuerpo de la página.

La programación se realiza en el lenguaje Typescript¹² propio de Angular 2 junto a la librería de Telerik Kendo UI que proporciona los componentes desarrollados en Javascript necesarios para la vista de esta parte del módulo, siendo necesario comprender la integración con la platilla de html para poder usarlos.

Cada vista de fase de es un conjunto Componente-Template y servicio de gestión (servicio-edit) proporcionado generalmente por Telerik para KendoUI, pero siendo necesaria la modificación del componente para el funcionamiento correcto de lo que se quiere hacer.

La capacidad asíncrona de los servicios http que conectan con el cliente a través de los *Gateway* hace necesario el desarrollo del módulo a través de programación reactiva que fuerza a suscribir el modelo de datos a la vista mediante observadores, propio de la arquitectura MVC.

Los servicios, tanto de edición como los de distribución de datos se inyectan mediante programación dependencias de inyección bien directa al módulo (lo que proporciona un acople fuerte, hace que el componente dependa del servicio y el servicio no tiene sentido sin el componente) o bien desde su definición en el módulo llegando al componente a través del constructor.

2.3.2 Back-end

El control del servidor está definido dentro de un proyecto ASP.NET Core MVC¹³ en lenguaje C# como parte de la implementación de una Web API que sirve los datos mediante accesos a través de las *Gateway* que son definiciones de métodos GET, POST y PUT e identificados por medio de *end points*, URIs¹⁴. Las *Gateway* que son necesarias para que el módulo funcione son las siguientes:

URI	Tipo	Descripción
/workflows/	GET	Devuelve un listado de flujos disponibles.
/workflows/{code}	GET	Devuelve un modelo de Workflow que cumple el modelo de datos identificado con el código. En el cuerpo devuelto se incluyen los pasos, los campos, los formularios, las acciones a ejecutar y su parametrización, los responsables de los pasos y las comunicaciones que se envían.
/workflows/create	POST	Valida los datos de creación y crea un nuevo Workflow.
/workflows/delete	POST	Valida el dato de eliminación y elimina el Workflow con el código de entrada.
/workflows/update	POST	Recibe el modelo del Workflow y vuelca el modelo de datos del Workflows en BD.
/areas	GET	Devuelve un listado de áreas a las que pertenece el Workflow.
/actions	GET	Devuelve el catálogo de acciones que es posible ejecutar.

Estas *Gateway* están definidas en una interfaz que implementa el control que soporta la lógica interna.

Los datos transmitidos por los *Gateway* son datos formateados por el estándar JSON [9], que se reciben o envían según convenga en el cuerpo de la llamada del *end point*. Para formatear los datos a dicho estándar fue necesario mapear los datos DAO del servidor hacia el cliente, más no del cliente al servidor pues Angular 2 usa el estándar de forma predefinida, para ello la Web API es capaz de convertir los datos DAO de las

persistencias de la BD a un DTO¹⁴, que mapea el estándar JSON. El servidor es capaz de realizar esta operación de conversión de forma inversa, es decir de DTO a DAO, pero tanto en lectura como escritura es necesario implementar el DTO para su traducción. Para el módulo fue necesario crear todos los DTO del modelo de datos.

2.3.3 Motor de Workflows

Lo que se explica a continuación es la proyección de la implementación de la modificación del motor de Workflows para que sea posible funcionalidad del apartado 2.2.5, funcionalidad añadida al motor de Workflows.

Hasta aquí sobre la acción se ha hablado de especificación, pero cada acción a ejecutar consta de código, necesario para su evidente ejecución, haciendo caso a la necesidad de reutilizar el código de los Workflow se concibió mantener el código de las acciones dentro de una interfaz y distinguir las acciones por el nombre del método, cumple la función de ser distintivo al ser único. La interfaz está al igual que el back-end implementada en C#. Para que el motor pueda interpretar lo que la acción ejecuta es necesario extraer de la interfaz el método junto con los parámetros de entrada, para esto se usa una técnica de *reflection* [10] todos esos parámetros de entrada se asocian a los valores de la parametrización de la acción completándola así para su ejecución.

La implementación de las acciones del catálogo es trabajo del desarrollador del Workflow por el momento. El catálogo está en desarrollo y se pretende ampliar para liberar de esta labor al programador en el tiempo.

3. RESULTADOS OBTENIDOS

El proyecto, a forma de análisis, se quiere aclarar y describir el proceso que se siguió para conseguir un módulo que pudiera diseñar Workflows de forma más sencilla. Se consiguió distinguir gracias a este análisis el funcionamiento del módulo y especificar cada apartado de la implementación además de iniciar el desarrollo de este.

Arquitectónicamente se ha conseguido un módulo compatible e integrado dentro del producto.

3.1 Control de Web Api

Para la Web Api se implementó gran parte del control y toda la definición de los DTO además de la lógica que haría falta en los DAOs del sistema para el funcionamiento junto a las persistencias. En torno a un 80% del trabajo.

El control está preparado para realizar las acciones que la lógica necesite, siendo posible acceder a los métodos que la implementan mediante las *Getaways* en el control, y que generan la respuesta pertinente. Las Gateways son métodos de retorno y consumo de información, como ya se ha dicho. Específicamente son métodos que retornan respuestas formateadas *IHttpActionResult* que devuelven a su vez información en forma de mensaje mediante protocolo HTTP que pueden ser capturados. Las Getaways siempre devuelven un valor que puede ser el valor solicitado o información de un error producido durante el tratamiento de datos.

Algunas de la Getaways, las de tipo POST, reciben un cuerpo del tipo *FromBodyAttribute* que contiene los atributos que son trasladados a DTO al llegar al servidor para poder usarse y posteriormente son traducidos a los correspondiente DAOs para poder persistirse. En este apartado se puede destacar la lógica de volcado del modelo del Workflow que es parcial. Es decir, permite realizar volcados parciales para guardar el trabajo de forma temporal y continuarlo en otro momento.

Para la persistencia de las acciones se crearon los scripts en SQL que crean las tablas en BD además de su equivalente para el acceso en forma de DAO y DTO.

3.2 Módulo de Angular

La parte del módulo en Angular aún en construcción realiza algunas de las funcionalidades más relevantes, aunque se encuentra en una fase temprana de desarrollo. Sin embargo, en este apartado se describen algunos de los puntos funcionales del diseñador.

3.2.1 Acceso al diseñador



El acceso al diseñador se realiza a través del apartado “Workflows” del menú principal del módulo de EndaliaHR.

Figura 12. Menú con acceso al diseñador.

3.2.2 Módulo de usuario

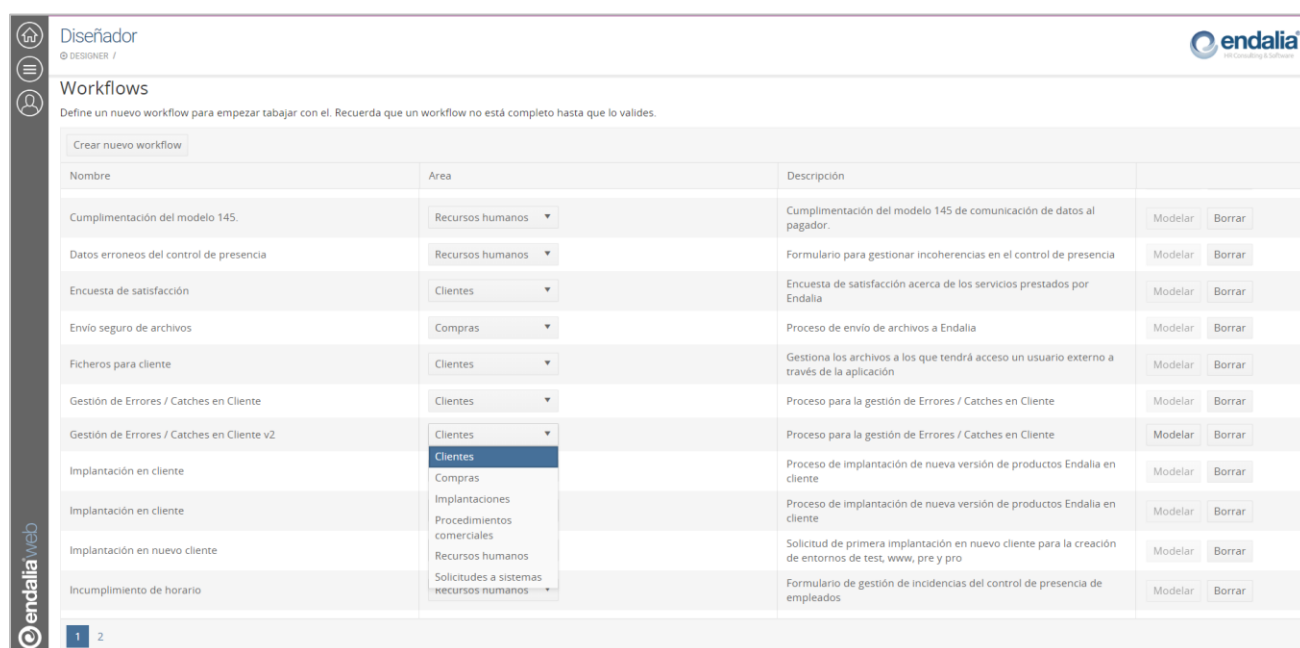


Figura 13. Vista de creación y modificación del Diseñador

Cuando el usuario accede al componente tiene disponible la lista de Workflows, mas no se han cargado los componentes restantes del módulo para el modelado. Realizando la construcción del módulo de esta forma se consigue aliviar la carga del módulo para conexiones lentas mejorando la velocidad de inicio del módulo (*Lazy load*) [17]. El resto de la carga de los componentes del módulo se hace cuando el usuario acceda al modelo del Workflow de este modo se consigue además solo tener los módulos necesarios en cada momento.

The image shows two UI components. On the left is a 'DropDown' menu with a list of items: 'Implantaciones', 'Recursos humanos', 'Recursos humanos', 'Clientes', 'Compras', 'Implantaciones', 'Procedimientos comerciales', 'Recursos humanos' (highlighted), and 'Solicitudes a sistemas'. On the right is a 'PopOut' dialog titled 'Nuevo workflow'. It contains a text input for 'NOMBRE DEL WORKFLOW' with the value 'Aprobación de pago anticipado', a dropdown for 'AREA' with the value 'Procedimientos comerciales', and a large text area for 'DESCRIPCIÓN DEL WORKFLOW'. A red error message 'La descripción del Workflow es obligatoria.' is displayed below the text area. At the bottom are 'Cancelar' and 'Crear' buttons.

Figura 14. Componente de “DropDown”, Izquierda y “PopOut” emergente a la derecha.

Se implementaron también componentes auxiliares que ayudan en la navegación y a la selección de datos, que son validados usando las directivas que proporciona Kendo UI, aunque para datos compuestos es necesario crear directivas propias para su validación.

The image is a screenshot of the 'Formularios' screen in the Workflow Designer module. It features a tabbed interface with 'Pasos', 'Campos', and 'Formularios' tabs. The 'Formularios' tab is active, showing a list of forms. A callout points to the tabs, stating: 'Pestañas que sirven tanto para retroceder en el proceso como para indicar en que punto se encuentra el proceso'. Another callout points to the 'Formulario de solicitud' form, stating: 'Información e instrucciones del funcionamiento del proceso'. The form itself has a table with columns 'Nombre del campo' and 'Propiedades del campo'. The table lists fields like 'Proyecto', 'Empleado solicitante', 'Fecha planificada devolución', 'Fecha solicitud', 'Motivo para la utilización', 'Material solicitado', and 'Registro de entrega de material'. Each field has properties: 'Esconder', 'Solo lectura', and 'Obligatorio'. A callout points to the 'Fecha solicitud' row, stating: 'Cuerpo del la pestaña. Solo cargado en el momento de seleccionar la pestaña'. At the bottom, there are buttons for 'Resolución', 'Validación de la resolución', and 'Siguiendo', with a callout pointing to the 'Siguiendo' button, stating: 'El componente actuaría siguiendo el patrón Wizard'.

Figura 15. Pantalla del módulo del diseñador para la creación de formularios.

Cuando el resto del módulo es cargado se puede ver las diferentes partes del modelador, Figura 15. En este punto es posible seguir el proceso de creación a través de las pestañas o el botón de “siguiente”, siguiendo el patrón de Wizard. La carga del cuerpo de las pestañas es dinámica y solo se produce cuando la URL apunte al componente.



Figura 16. Ejemplo de redireccionamiento a la gestión de formularios en el proceso de modelado de un Workflow.

El enrutado de las diferentes partes del diseñador es lo que consigue la carga dinámica de los componentes. El componente principal (el que contiene las pestañas y tiene acceso al módulo de enrutado) solo tiene acceso al identificador del componente hijo de tal forma que sólo es cargado cuando se cambia la URL, que sólo cambia al seleccionar una pestaña.

4. CONCLUSIONES DEL PROYECTO

4.1 Conclusiones

En base a los objetivos y el alcance planteados en el inicio del proyecto hay que valorar hasta qué punto se alcanzó el desarrollo del módulo. El objetivo principal del proyecto se consiguió puesto que era marcar de forma precisa como debe desarrollarse el módulo para que sea funcional y que el trabajo que queda por hacer no suponga mayores problemas.

Desde un primer momento se buscó crear un diseño conceptual de lo que se iba a hacer y gracias a ello se consiguió respetar dicho diseño a lo largo del proyecto. Queda clara la visión general y apartados específicos que componen el módulo. Se consiguió crear una funcionalidad lógica para el módulo que iba siempre ligada al modelo de datos que se originó y a la lógica que ahora especificada deja claro un camino a seguir. La manipulación de los datos y su interacción fue también una parte importante del desarrollo, nuevamente está marcado el camino a seguir, pero sobre todo se consiguió generar un módulo completo sólo con los datos necesarios y accesibles sólo en el momento adecuado.

Acoplar el módulo al producto creando una arquitectura soportada por el mismo era una de las necesidades planteada como objetivo del análisis y se consiguió satisfactoriamente.

Con respecto al proyecto en sí, seguir la metodología y respetar las fases del proyecto supuso encuadrar las diferentes actividades de forma coherente para que no hiciese falta retroceder en cada ocasión que se realizara una revisión, sin embargo, ha tenido que hacerse para asegurar la consistencia del módulo.

En lo personal, los problemas encontrados en cada apartado, desde nuevos conceptos, técnicas, o entender el trabajo ya ejecutado (Web Api ya existía antes del proyecto) para poder usarlo e incluso entender el trabajo más allá del personal y del proyecto, con las reuniones del equipo de desarrollo de Workflows, supusieron un reto que creo se consiguió superar.

Destacar lo aprendido, nuevas tecnologías, nuevas técnicas y derivados que no constan dentro del proyecto pero que supusieron una gran ayuda.

4.2 Líneas futuras

Si bien el proyecto está completo, el módulo aún está en una fase de implementación. Además el rumbo que seguir es muy claro. Algunos de los puntos a tener en cuenta en el futuro son:

- Seguir la lógica descrita para conseguir culminar el desarrollo.

- Aunque la estructura del catálogo está desarrollada, las acciones que lo componen deben ser desarrolladas más tarde en el tiempo.
- El motor de Workflow debería ser revisado para descubrir casuísticas no probadas aún.
- Adecuar las interfaces del módulo que, aunque planificadas, es necesario trabajar los estilos y analizar posibles fallos de usabilidad y diseño responsable.

5. GLOSARIO

A

Angular ⁸	<i>“Angular es una plataforma que facilita la creación de aplicaciones con la web. Angular combina plantillas declarativas, inyección de dependencia, herramientas de extremo a extremo y mejores prácticas integradas para resolver los desafíos de desarrollo. Angular permite a los desarrolladores crear aplicaciones que se ejecutan en la web, dispositivo móvil o escritorio”</i> (Traducción libre) [6]
----------------------	--

ASP.NET Core MVC ¹³	<i>“ASP.NET Core es un marco de código abierto multiplataforma, de alto rendimiento para la construcción de aplicaciones modernas, basadas en la nube, conectadas a Internet”</i> (Traducción libre) [11]
--------------------------------	---

B

BD ²	Base de datos.
-----------------	----------------

C

Complejidad ⁴	Aplicado a un modelo de Workflow dentro de este proyecto significa que no es un Workflow fácilmente interpretable por el diseñador dadas sus características.
--------------------------	---

D

DAO ⁷	Data Acces Object, patrón de diseño. [8]
------------------	--

F

Fases ³	En el modelo más conceptual de un flujo de trabajo. Las fases son los pasos del Workflow.
--------------------	---

G

GET ⁹	<i>“GET se usa para pedir información de un recurso específico”</i> (Traducción libre) [7]
------------------	--

I

Instancia¹⁵ Una instancia es una ocurrencia concreta del modelo. Es decir los “documentos” en los que se solicitan los datos del usuario que está trabajando en el Workflow.

M

Módulo¹ Sección de la web que gestiona algún tipo de actividad automatizada por el software, algunos ejemplos: peticiones de ausencias y vacaciones, formación, viajes, etc.

Modelo de tres capas⁴ Patrón de arquitectura en la que el software se divide en tres capas bien diferenciadas: capa de interfaz de usuarios, capa de lógica de negocio y capa de acceso a datos. [5]

P

POST¹⁰ “*POST se usa para enviar datos a un servidor para crear / actualizar un recurso.*” (Traducción libre)[7]

PUT¹¹ “*PUT se usa para enviar datos a un servidor para crear / actualizar un recurso de forma idempotente.*” (Traducción libre) [7]

L

Logic⁵ En este contexto, nombre que se le ha dado a las diferentes clases procedimentales que no pertenecen a un objeto sino al diseñador, que sería el TAD. En la implementación se tradujo como una región del servicio principal. Ejemplo: *LogicWorkflow*, *LogicStep*...

T

TAD⁶ Tipo abstracto de dato.

Typescript¹² “TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tapado estático y objetos basados en clases.” [16]

U

URI¹⁴ Uniform Resource Identifier, “*Es la URL de un servicio con acceso externo que es accedido por un servicio.*” (Traducción libre) [12]

6. BIBLIOGRAFÍA

- [1] J. L. «sawbona.blogspot.com,» 28 09 2014. [En línea]. Available: <http://sawbona.blogspot.com/2014/09/patron-de-diseno-wizard-asistente.html>. [Último acceso: 17 9 2018].
- [2] L. Hernan, «medium.com,» 22 04 2016. [En línea]. Available: <https://medium.com/mercadolibre-ux/soy-hernan-lucio-ux-mobile-designer-en-mercado-libre-y-profesor-ux-en-coderhouse-argentina-a486483760fd>. [Último acceso: 17 09 2018].
- [3] EndaliaHR, «endalia.com,» [En línea]. Available: <http://www.endalia.com/software>. [Último acceso: 11 09 2018].
- [4] C. R. P. de San Martin Oliva, «portalhuarpe.com.ar,» [En línea]. Available: <http://www.portalhuarpe.com.ar/Seminario09/archivos/MetodologiaI CONIX.pdf>. [Último acceso: 17 09 2018].
- [5] raps mk (seudonimo), «net-informations.com,» [En línea]. Available: <http://net-informations.com/q/mis/3tier.html>. [Último acceso: 2018 09 18].
- [6] Google, «angular.io,» [En línea]. Available: <https://angular.io/docs>. [Último acceso: 2018 9 19].
- [7] Refsnes Data, «w3schools.com,» [En línea]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Último acceso: 2018 09 20].
- [8] J. Jenkov, «jenkov.com,» 21 05 2014. [En línea]. Available: <http://tutorials.jenkov.com/java-persistence/dao-design-pattern.html>. [Último acceso: 2018 09 20].
- [9] Ecma, «www.ecma-international.org,» 2017. [En línea]. Available: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. [Último acceso: 21 09 2018].
- [10] D. Alexander, «stackoverflow.com,» 25 01 2013. [En línea]. Available: <https://stackoverflow.com/questions/14505297/what-is-reflection-property-of-a-programming-language>. [Último acceso: 2018 09 21].
- [11] D. Roth, R. Anderson y S. Luttin, «docs.microsoft.com,» Microsoft, 28 02 2018. [En línea]. Available: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1>. [Último acceso: 2018 09 24].
- [12] Oracle, «docs.oracle.com,» [En línea]. Available: https://docs.oracle.com/cd/E13171_01/alsb/docs30/operations/endpointurimgmt.html. [Último acceso: 2018 09 24].
- [13] T. L. T. Hoang Lam, .NET Framework Essentials, 3 ed., O'Reilly, Ed., 2003.
- [14] Progress Software Corporation, «Kendo UI for Angular,» 2018. [En línea]. Available: <https://www.telerik.com/kendo-angular-ui>. [Último acceso: 25 10 2018].
- [15] Wikimedia Foundation, «en.wikipedia.org,» 18 10 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Microsoft_SQL_Server. [Último acceso: 25 10 2018].
- [16] Wikimedia Foundation, «en.wikipedia.org,» [En línea]. Available: <https://es.wikipedia.org/wiki/TypeScript>. [Último acceso: 29 10 2018].
- [17] Rangle Inc, «Rangle.io,» [En línea]. Available: <https://angular-2-training-book.rangle.io/handout/modules/lazy-loading-module.html>. [Último acceso: 03 11 2018].



Universidad
Zaragoza

ANEXO 1

Prototipos

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 1

Diagramas conceptuales basados en componentes que se han seguido durante el desarrollo del proyecto. Se incluye el mapa de navegación entre vistas.

ÍNDICE

Índice	3
1. Prototipado	4
1.1 Mapa de navegación	4
1.2 Vista Workflows	4
1.3 Vista Steps	5
1.4 Vista Fields	6
1.5 Vista Actions	7
1.6 Vista Responsibles	8
1.7 Vista Forms	8
1.8 Vista Mails	9
1.9 Vista Finalize	10

1. PROTOTIPADO

1.1 Mapa de navegación

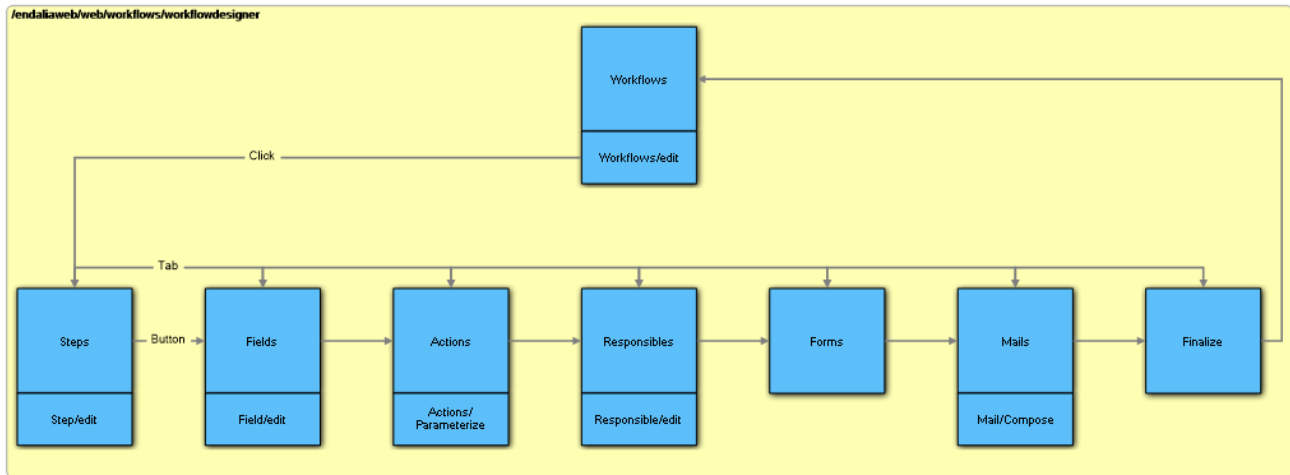


Figura 1. Mapa de navegación del diseñador de Workflows

1.2 Vista Workflows

Workflow Designer
Define un nuevo workflow para empezar a trabajar con él.
Recuerda que un workflow no está completo hasta que lo valides.

NOMBRE DEL WORKFLOW	AREA	DESCRIPCIÓN
Alta nuevo empleado	Recursos humanos	
Pago anticipado	Compras	Aprobación de pago anticipado
Cambio de cuenta corriente	Recursos humanos	Cambio de cuenta bancaria

Campos como el estatus deberían añadirse.

Figura 2. Vista Workflows

La Figura 2 representa la vista que contiene la lista de todos los modelos de los Workflows. En ella el usuario es capaz de cambiar algunas características, las principales del Workflow. Es la vista principal del diseñador y desde aquí inicia el modelado.

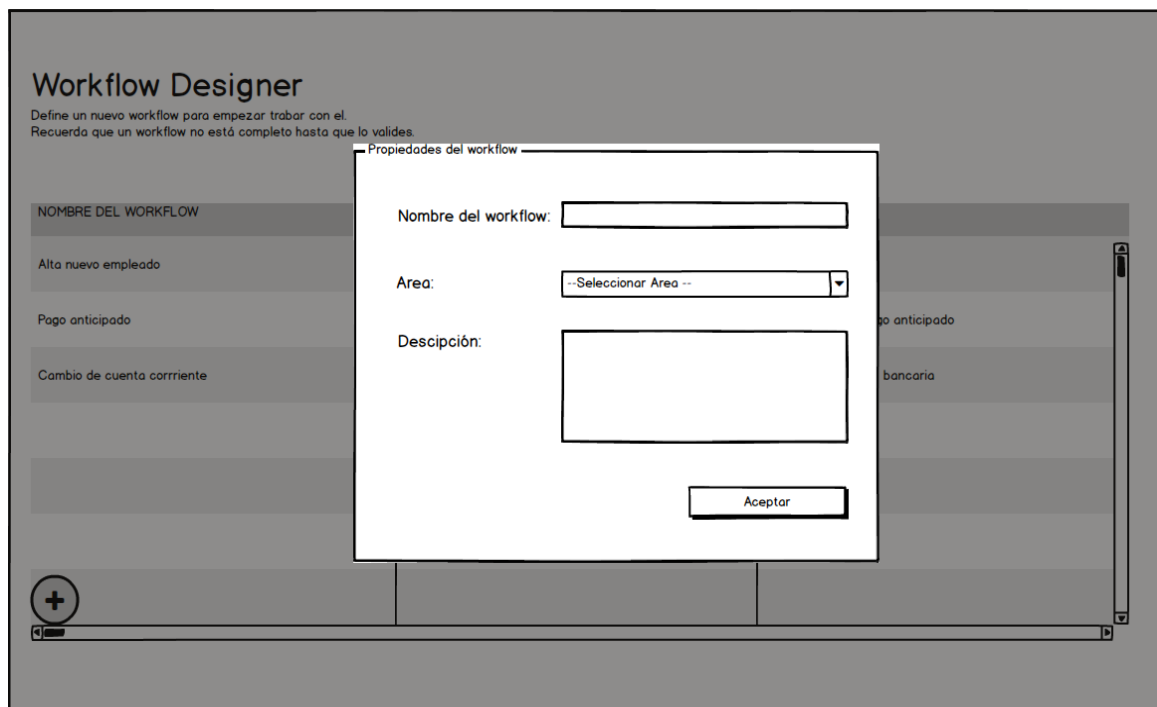


Figura 3. Vista detalle de un Workflow

La Figura 3 representa la vista en detalle de un Workflow. Se usaría como vista de creación y edición del Workflow justo al iniciar el resto del módulo.

1.3 Vista Steps

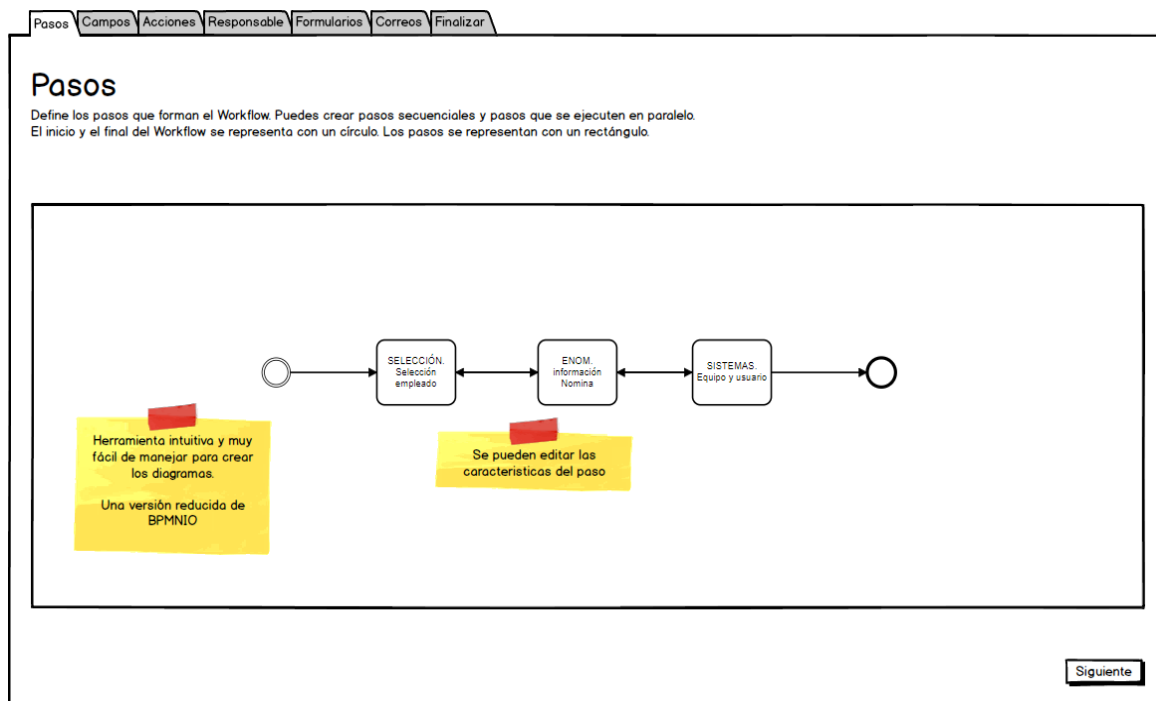


Figura 4. Vista Pasos

La Figura 4 representa la vista de la secuencia de pasos del Workflow. En ella es posible cambiar el orden de los pasos y editar alguna característica a primer nivel.

Pasos

Define los pasos que forman el Workflow. Puedes crear pasos secuenciales y pasos que se ejecuten en paralelo. El inicio y el final del Workflow se representa con un círculo. Los pasos se representan con un rectángulo.

Propiedades del workflow

Define las principales características de la fase

Nombre del paso

Objetivos

Descripción:

Aceptar

Siguiente

Figura 5. Vista detalle de un Paso

La Figura 5 representa la vista en detalle de un Paso. Se usaría como vista de creación y edición del Paso. El paso se añadiría al final de la cola de Pasos.

1.4 Vista Fields

Campos

Define los campos de información necesarios para el Workflow. Cada campo se compone de un nombre, descripción y tipo. Algunos tipos de campos requieren parámetros adicionales que la aplicación te solicitará. Recuerda que puedes crear campos ocultos, que no se mostrarán al usuario, pero que pueden contener información utilizada en los correos o en las acciones del Workflow.

NOMBRE DEL CAMPO	DESCRIPCIÓN DEL CAMPO	TIPO DEL CAMPO
Nombre	Nombre del empleado	TEXT
DNI		DNI
Fotografía		FILE
Salario	Salario del empleado a 100% de jornada	CURRENCY
Destinatario correo	Empleado que recibirá el email de notificación	CALCULATED

En este ejemplo, se ha añadido un campo que se usa en el workflow aunque no se muestre en los formularios. Este campo contiene el destinatario de los correos del workflow

Algunos tipos, como el "CALCULATED" abrirán un popup con las opciones particulares de ese tipo de campo

Siguiente

Figura 6. Vista Campos

La Figura 6. Representa la vista de la lista de los Campos que se podrán usar en los formularios. Se representa también el botón de configuración del campo (según el tipo) pero que no se ha contemplado en el proyecto por alcance.

1.5 Vista Actions

Acciones

Define que acciones se tendrán que producir en cada uno de los pasos. Puedes usar los campos de la columna izquierda para rellenar los parámetros de cada acción. Puedes definir mas campos en la pestaña "Campos", no olvides también asignar los al paso correspondiente en la pestaña "Formularios"

PASO 1	PASO 2	PASO 3
Acciones	Acciones	Acciones
Campo Resultado	Campo Resultado	Campo Resultado
Crear empleado	Dar permisos	--Seleccionar--
EmpID	Lista permisc	--Seleccionar--
Crear Usuario	Confirmación RRI	
UsrID	Mensaje de c	
--Seleccionar--	--Seleccionar--	
--Seleccionar--	--Seleccionar--	

El resultado de las acciones se guardará en el "campo resultado"

Se podrá configurar las acciones para definir su funcionamiento

Siguiente

Figura 7. Vista Acciones

La Figura 7. Es ella el usuario es paz de crear secuencia de acciones. Es posible añadir nuevas acciones, seleccionar que acción se ejecutará y hacia qué campo desembocará el resultado.

EDITOR DE ACCIONES

Acción: ALTA EMPLEADO ejecutada en el paso 2: Aqui puedes editar los parametros de entrada de la acción usando los parametros de regreso de acciones pasadas

PARAMETRO	CAMPO
NOMBRE	Nombre
APELLIDO	--Seleccionar--
DNI	Dni
EMP_ID	EmpID

Aceptar

Siguiente

Figura 8. Parametrización de una Acción

La Figura 8. En esta vista el usuario debe seleccionar que campos se usaran como entrada para la ejecución de la acción.

1.6 Vista Responsibles

Pasos
Campos
Acciones
Responsable
Formularios
Correos
Finalizar

Responsable

Define los responsables de información necesarios para el Workflow.

Cada responsable lo es de un paso, cada responsable va definido por un tipo que puede ser configurable y los privilegios que tiene en cada paso.

PASOS	TIPO DEL RESPONSABLE	PERMITIR CANCELAR	PERMITIR RECHAZAR
NOMBRE_DEL_PASO	Creator ⚙	☐	☒
PASO2	Empleado ⚙	☐	☐
PASO3	JobID ⚙	☐	☐
PASO4	CalculatedPriv ⚙	☒	☒
PASO5	Calculated ⚙	☐	☒

Responsable INT,

Responsables string1 y

string 2

Siguiente

Figura 9. Vista Responsables

La Figura 9 representa la selección del responsable para un campo. En esta vista el usuario debe seleccionar el tipo de responsable que quiere y que permisos se le quiere dar.

1.7 Vista Forms

PasosCamposAccionesResponsableFormulariosCorreosFinalizar

Formularios

Indica qué campos se van a solicitar al usuario en cada paso. Los campos definidos previamente están disponibles en todas las fases del proceso de workflow pero no necesariamente serán visibles. El orden, visibilidad, obligatoriedad, edición, etc. también son editables.

Nombre_del_paso_1	Nombre_del_paso_2	Nombre_del_paso_3
Nombre	Nombre	Nombre
Apellido	DNI	Apellido
DNI	Apellido	Fotografia
Salario	Salario	Salario
Fotografia	Fotografia	DNI
Destinatario correo	Destinatario correo	Destinatario correo

La visibilidad y algunas características como "Read only" se puede gestionar a primer nivel

Siguiente

Figura 10. Vista Formularios

En esta vista el usuario debe configurar con todos campos para todos los paso el diseño delos formularios; cambiando el orden, la visibilidad, la obligatoriedad de cada campo. En esta vista solo se representa la vista.

1.8 Vista Mails

Correos

Configurara los correos que se quieren enviar. Se puede definir el destinatario, el asunto y el contenido del correo. Puedes seleccionar el paso donde quieres que se envíe. Puedes usar los campos a la izquierda para añadir su información al correo, sólo tienes que arrastrar desde la columna de la izquierda hacia el sitio que quieras del correo.

PASO	ASUNTO	DESTINATARIO	EMAIL DE CANCELACIÓN	EMAIL DE RECHAZO
Nombre_del_paso_3	Confirmación	RRHH	☐	☒
Nombre_del_paso_2	Alta		☐	☐
--Seleccionar--			☒	☒

Añadir algun campo mas interesante al grid

+

Siguiente

Figura 11. Vista Correos

En la vista 11. Se muestra la lista de correo que se envían en los diferentes momentos de las fases del Workflow. El usuario puede añadir nuevos correos en esta vista y asignarle un paso al correo.

EDITOR DE CORREO

Correo enviado en el Paso 3: Aquí puedes editar el destinatario, asunto y cuerpo del correo.

CAMPOS

- Nombre
- Apellido
- DNI
- Salario
- Fotografia
- Destinatario correo

Destinatario: Destinatario correo

Asunto:

B I U S style

Nombre Apellido

DNI

Aceptar

Figura 12. Vista composición de un Correo de comunicación.

En la Figura 12 cuando el usuario ha creado un nuevo correo es posible componer su contenido haciendo además uso de los campos arrastrándolos al cuerpo.

1.9 Vista Finalize

Pasos
Campos
Acciones
Responsable
Formularios
Correos
Finalizar

Finalizar

Resumen del workflow con su diagrama:

Esta pantalla es necesaria para mostrar datos como la identificación por keyname del workflow que no se podrá saber hasta aquí.

NO editable.

Si hubiera errores se mostrarían aquí.

Validar y finalizar

Figura 13. Vista Finalizar

La Figura 13 básicamente aportará información al usuario sobre posibles problemas en el modelado. Cuando todo sea correcto se permite a l usuario guardar.



Universidad
Zaragoza

ANEXO 2

Casos de uso

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

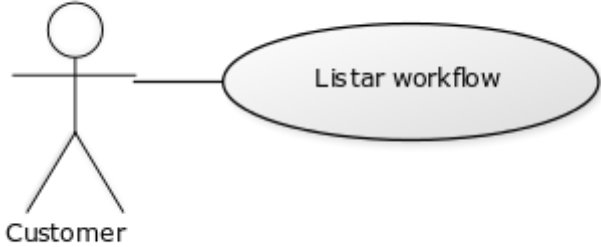
ANEXO 2

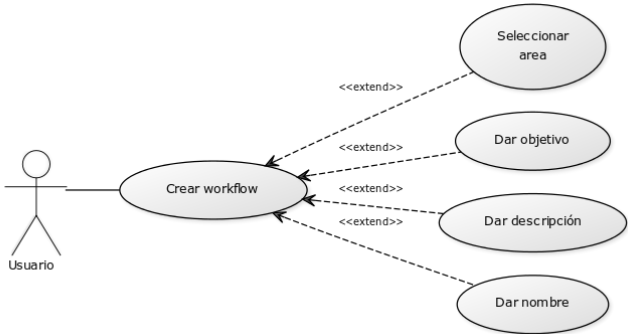
Casos de uso que describen la funcionalidad del módulo. Para cada funcionalidad se estudiaron los principales casos de uso del sistema:

ÍNDICE

Índice	2
1. Casos de uso	3

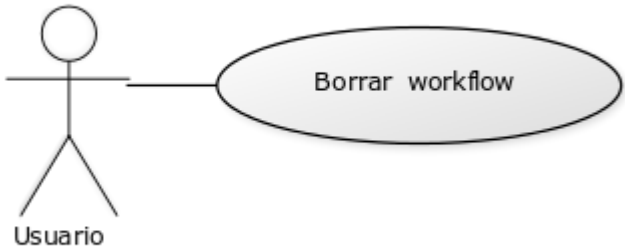
1. CASOS DE USO

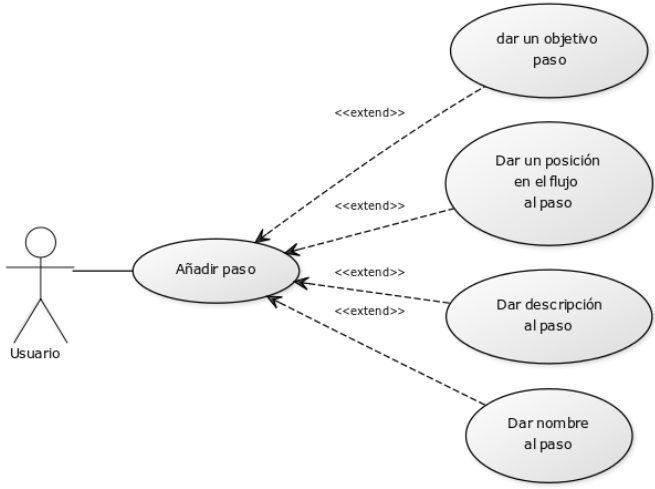
CU 1. Visualizar Workflows	
Descripción	El usuario carga la vista donde podrá ver la lista de Workflows a los que tiene acceso, sean modelables y el área al que pertenecen.

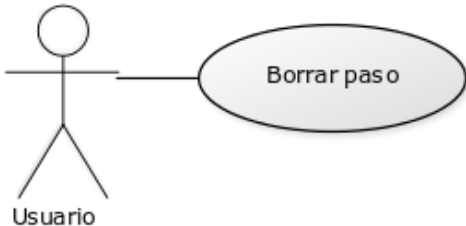
CU 2. Crear Workflows	
Descripción	El usuario crea un nuevo Workflow: proporciona un nombre, propósito del Workflow, un área y una descripción. El sistema corrobora si no existe el Workflow en el área y se añade como nuevo siendo posible modelarlo.
Flujo	• Añade nuevo Workflow, se proporciona nombre, descripción, área y objetivo. • El usuario puede cancelar la creación. • Se crea un nuevo Workflow.

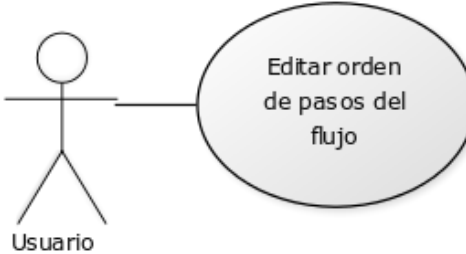
CU 3. Cambiar área del Workflow	
Precondición	Debe existir al menos un Workflow
Descripción	Para un Workflow se cambia el área del workflow.

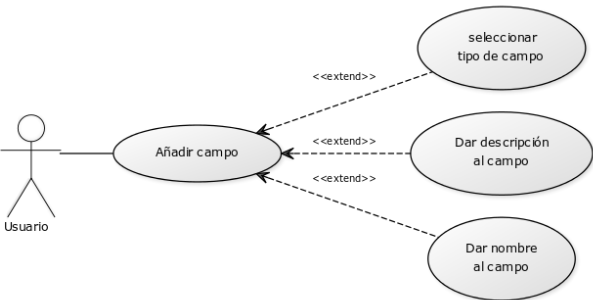
CU 4. Editar Workflow (Cargar el modelo del Workflow)	
Precondición	Es obligatorio escoger un Workflow que modelar.
Descripción	Se carga el modelo del Workflow y se inicia el modelado de este.
Flujo	• Se selecciona el Workflow. • Se inicia el diseñador.

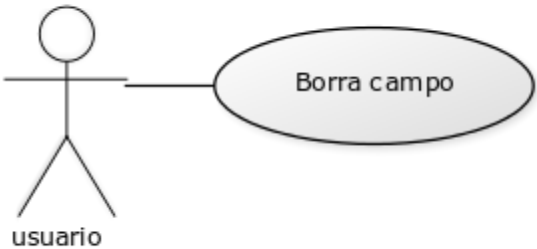
CU 5. Borrar un Workflow	
Precondición	Debe existir al menos un Workflow.
Descripción	El usuario selecciona un Workflow y lo elimina. Se deberá validar la eliminación: La ausencia de entidades abiertas o en proceso de completarse no permite la eliminación.
Flujo	• Se borra el Workflow. • Se puede cancelar el borrado.

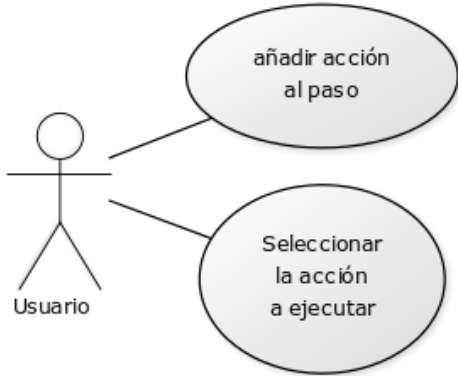
CU 6. Añadir un nuevo paso al Workflow.	
Descripción	Se añade un nuevo paso al Workflow para ello se debe proporcionar para el nuevo paso un nombre, objetivos, orden y descripción.
Flujo	• Se añade un nuevo paso al Workflow con nombre, objetivo y descripción del paso. • Se liga el paso al orden de pasos. • Se puede cancelar el guardado. • En caso de guardar los cambios se enlazará un nuevo paso al flujo de pasos.

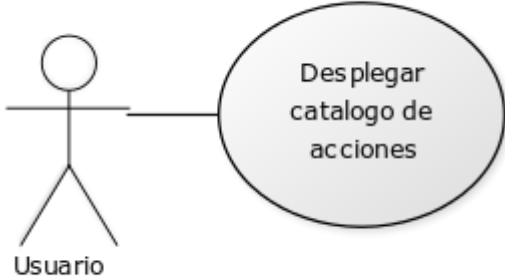
CU 7. Borrar un paso.	
Precondición	Debe existir al menos un paso
Descripción	Se modifica el Workflow quitando un paso del flujo de pasos. Esto provoca la actualización del orden de pasos del Workflow.
Flujo	• Se borra el paso • Se podrá cancelar el borrado del paso. • Se elimina el paso del flujo y se reordena le flujo.

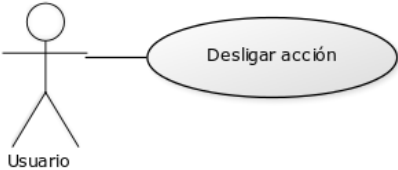
CU 8. Modificar orden de los pasos.	
Precondición	Debe existir al menos dos pasos en el flujo.
Descripción	Se modificará la lista de ejecuciones en función de cómo se esté modelando el orden de ejecución de los pasos.
Flujo	• El usuario modifica el flujo de ejecución. • Si alguno de los cambios del flujo no tiene coherencia se advierte.

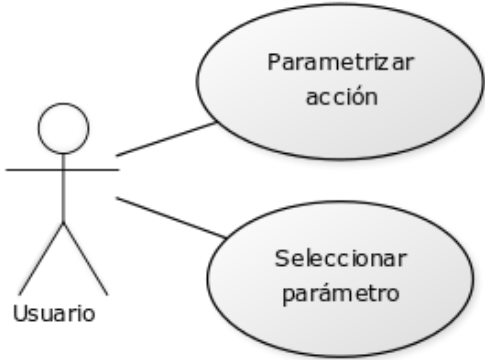
CU 9. Añadir/modificar nuevo campo al Workflow	
Descripción	Se añade un paso para usar en los formularios del Workflow. Para cada paso hay que proporcionar un nombre, seleccionar el tipo, dar una descripción al campo y un nombre con el que se visualizará el campo. Este campo será por definición solo de lectura en los formularios, aunque es posible configurarlo.
Flujo	• Se añade un nuevo campo con sus datos: Se da un nombre, seleccionar tipo de campo, descripción y nombre. • Si no se cancela el guardado.

CU 10. Borra un campo del Workflow.	
Precondición	Existe al menos un campo.
Descripción	Se selecciona un campo y se borra. El campo deja de estar disponible en los formularios de los pasos.
Flujo	• El usuario selecciona un campo de la lista de campos. • El usuario borra el campo

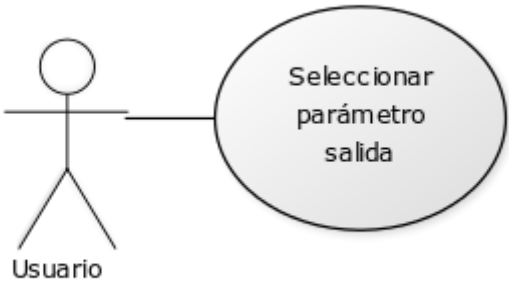
CU 11. Ligar una acción al paso.	 <pre> graph LR Usuario((Usuario)) --- UC1(añadir acción al paso) Usuario --- UC2(Seleccionar la acción a ejecutar) </pre>
Descripción	Se quiere que un paso ejecute una acción. Se añade una acción a la lista de acciones del paso. Esto generará un campo de tipo “acción” para el paso, el campo no será visible y se ejecutará al finalizar la fase.
Flujo	• Se añade una nueva acción • Se selecciona del catálogo una acción. • Si la acción permite parametrización se advierte, dependiendo del tipo de acción, si necesita ser parametrizada de entrada o/y salida.

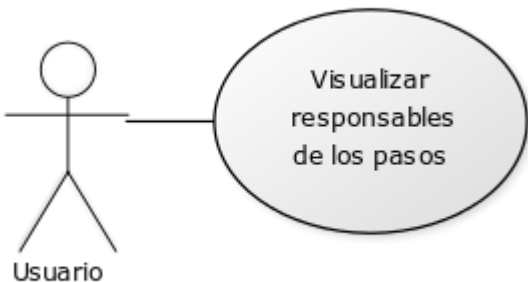
CU 12. Visualizar catálogo de acciones	 <pre> graph LR Usuario((Usuario)) --- UC1(Desplegar catalogo de acciones) </pre>
Descripción	Se despliega el catálogo de acciones que puede ejecutar un paso en al final de su ejecución

CU 13. Desligar acción del paso.	
Precondición	La salida de la ejecución no puede estar ligada a otras acciones como parámetro.
Descripción	Se desea impedir que un paso ejecute una acción para ello se desliga la acción del paso eliminándola de la ejecución.
Flujo	• Se elimina la acción la acción del paso. • Se advierte si esta acción tiene un parámetro de salida seleccionado. Se advierte de que es un valor con salida y se especifica si hay otras acciones que están usando este valor. Si no existe el parámetro de salida se elimina. • Se puede cancelar.

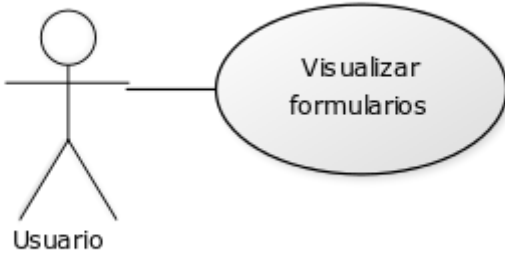
CU 14. Parametrizar acciones (parámetros de entrada).	
Precondición	Acción del catálogo seleccionada.
Descripción	Si la acción lo permite debe poderse para los parámetros de entrada que serán los campos (placeholder) del Workflow. Si la acción permite parametrización se muestra los parámetros de entrada de la acción. Cada acción tendrá los suyos propios. Se selecciona un campo como valor del parámetro. Solo tendrá disponibles aquellos cuyo tipo sea compatible con el tipo.

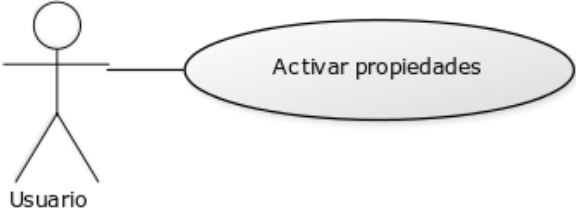
Flujo	• Se seleccionan los campos para cada parámetro. • Se puede cancelar la parametrización. • Se deja guardar si la validación es correcta. No se deja guardar si la parametrización es incoherente.
--------------	---

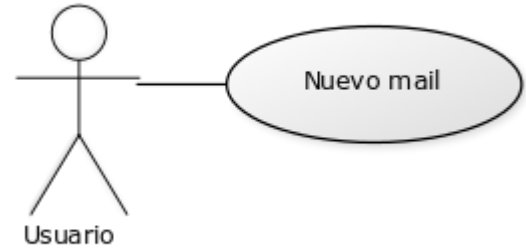
CU 15. Parametrizar acciones (parámetros de salida).	 <pre> graph LR Usuario[Usuario] --- Seleccionar((Seleccionar parámetro salida)) </pre>
Precondición	Acción del catálogo seleccionada.
Descripción	Se busca dar valores a los parámetros de entrada de la acción. Si la acción lo permite debe poderse seleccionar el campo de salida.
Flujo	• Se selecciona un campo de salida para la acción. • Se advierte si el campo es de un tipo no necesariamente compatible.

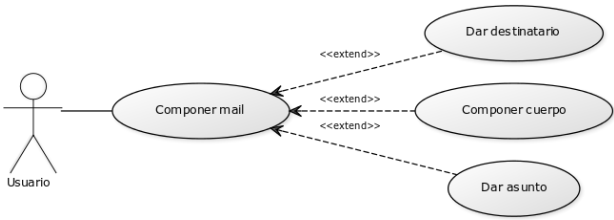
CU 16. Visualizar responsables de los pasos	 <pre> graph LR Usuario[Usuario] --- Visualizar((Visualizar responsables de los pasos)) </pre>
Precondición	Debe existir al menos un paso
Descripción	Muestra cada paso con responsable.

CU 17. Definir un responsable del paso.	
Precondición	El paso debe tener seleccionada la acción que se ejecutará
Descripción	Debe ser posible seleccionar un responsable para cada paso. Se debe poder configurar algunas de las características principales que el responsable pueda realizar.
Flujo	• Para cada paso seleccionar un tipo de responsable. • Se da los permisos de permitir cancelar y permitir el rechazo.

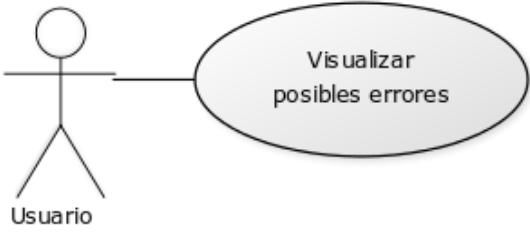
CU 18. Visualizar formularios	
Precondición	Debe existir al menos un paso
Descripción	Se cargarán las propiedades de los campos en los pasos y se podrá visualizar estas propiedades.

CU 19. Crear formulario asociado a un step.	
Precondición	Al menos un campo en el Workflow
Descripción	Cada formulario debe mostrar los campos. Estos campos poseen una característica para cada campo. El objetivo es configurar esas características. Por definición el campo es solo de lectura e invisible, esto es modificable además de establecer si el campo es obligatorio.

CU 20. Añadir mail	
Precondición	Al menos un paso en el Workflow
Descripción	Es posible definir varios mails en la misma fase del Workflow. Se debe elegir el paso al que se quiere ligar el mail. además de especificar de qué tipo de mail se trata (de cancelación, confirmación o finalización).

CU 21. Componer mail	 <pre> graph LR Usuario[Usuario] --- ComponerMail(Componer mail) DarDestinatario(Dar destinatario) -.-> <<extend>> ComponerMail ComponerCuerpo(Componer cuerpo) -.-> <<extend>> ComponerMail DarAsunto(Dar asunto) -.-> <<extend>> ComponerMail </pre>
Precondición	Mail asignado a un paso
Descripción	Se debe dar un destinatario, asunto y componer el cuerpo del mail.

CU 22. Actualizar paso en el que se enviará el mail	 <pre> graph LR Usuario[Usuario] --- SeleccionarNuevoPaso(Seleccionar nuevo paso) </pre>
Precondición	Un mail está definido para enviarse en una fase el Workflow
Descripción	Se cambia la fase en la que se enviará el mail.

CU 23. Visualizar posibles errores	
Descripción	Una vez terminado de modelar el Workflow el usuario visualizará los posibles errores de su moldeo. Solo advertencias, no errores, pues los errores habrían impedido avanzar en la construcción del modelo así que entendemos que la validación se hizo paso a paso al modelar.

CU 24. Guardar Workflow	
Precondición	El modelo no debe tener ningún error.
Descripción	Una vez completado el modelo es posible guardarlo en BD. En este punto el Workflow debe estar preparado para ser usable.



Universidad
Zaragoza

ANEXO 3

Diagramas de secuencia

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 3

Este documento se agrupan los diferentes diagramas de secuencia que describen el dialogo entre las diferentes clases que componen el modulo, *Workflow Designer. Lógica de negocio*. Para cada uno de los casos de uso se desarrolló un diagrama. Los casos de usos se agrupan por pantalla.

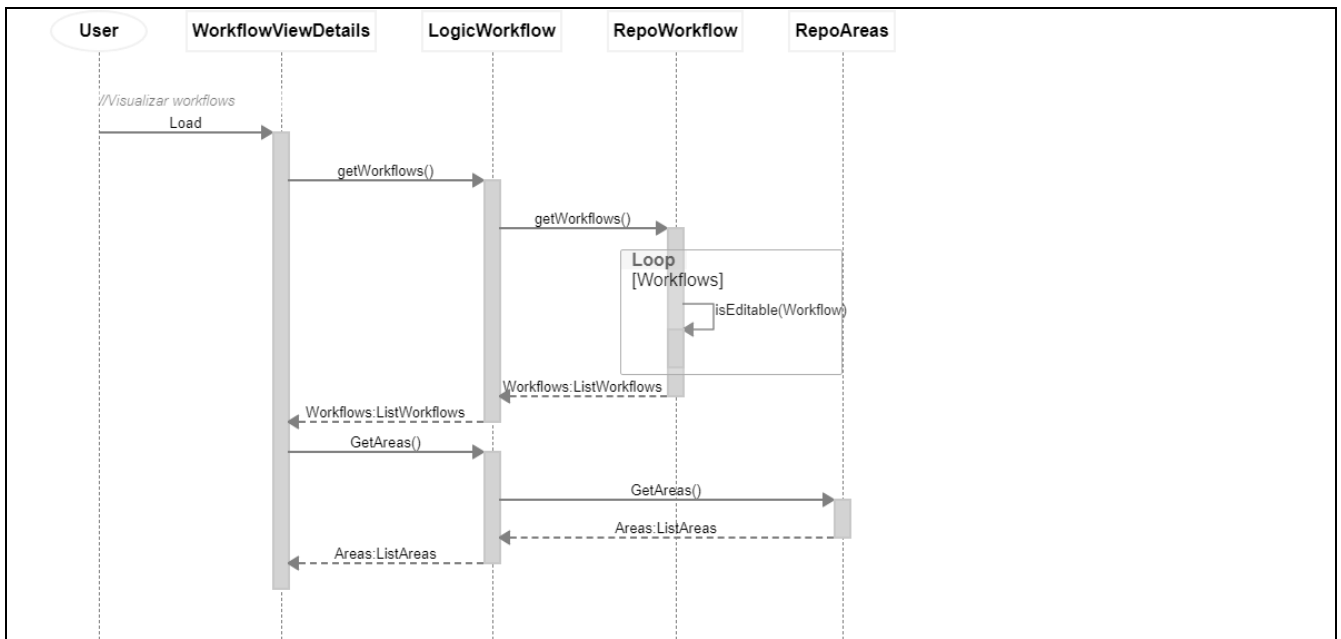
ÍNDICE

Índice	3
1. Diagramas de secuencia	4
1.1 Vista Workflows	4
1.1.1 Visualizar workflows.	4
1.1.2 Cambio de área	4
1.1.3 Crear un nuevo workflow	5
1.1.4 Borrar workflow	6
1.1.5 Obtener un modelo de workflow.	7
1.2 Vista Steps.....	8
1.2.1 Añadir paso	8
1.2.2 Modificar flujo de pasos	9
1.2.3 Borrar paso	10
1.3 Vista Fields	11
1.3.1 Crear nuevo campo	11
1.3.2 Borrar campo	12
1.4 Vista Actions	13
1.4.1 Visualizar catálogo de acciones	13
1.4.2 Ligar acción al paso al paso	13
1.4.3 Parametrizar acción (entrada)	14
1.4.4 Parametrizar acción (salida)	15
1.4.5 Desligar una acción del paso.....	16
1.5 Vista Forms.....	16
1.5.1 Visualizar formularios de las fases	16
1.5.2 Actualizar propiedades del formulario	17
1.6 Vista Responsibles	17
1.6.1 Visualizar responsables de los pasos	17
1.6.2 Actualizar tipo de responsable.....	18
1.7 Vista Mails	18
1.7.1 Cambio de paso en el que se enviará el mail.....	18
1.7.2 Añadir un nuevo mail	19
1.7.3 Borra un mail del paso	19
1.8 Vista Finalize	20
1.8.1 Guardar Workflow.....	20
1.8.2 Visualizar posibles errores.....	20

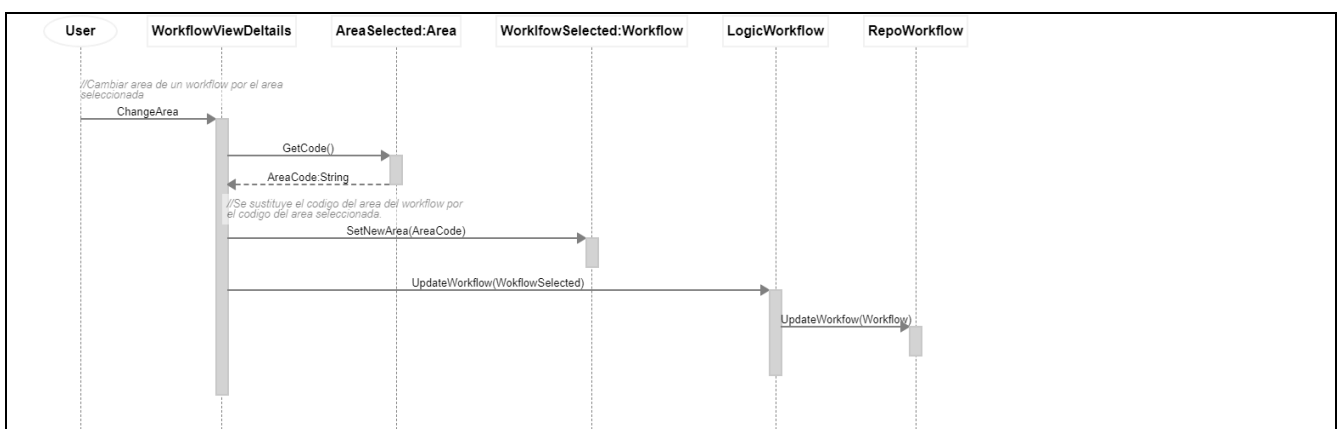
1. DIAGRAMAS DE SECUENCIA

1.1 Vista Workflows

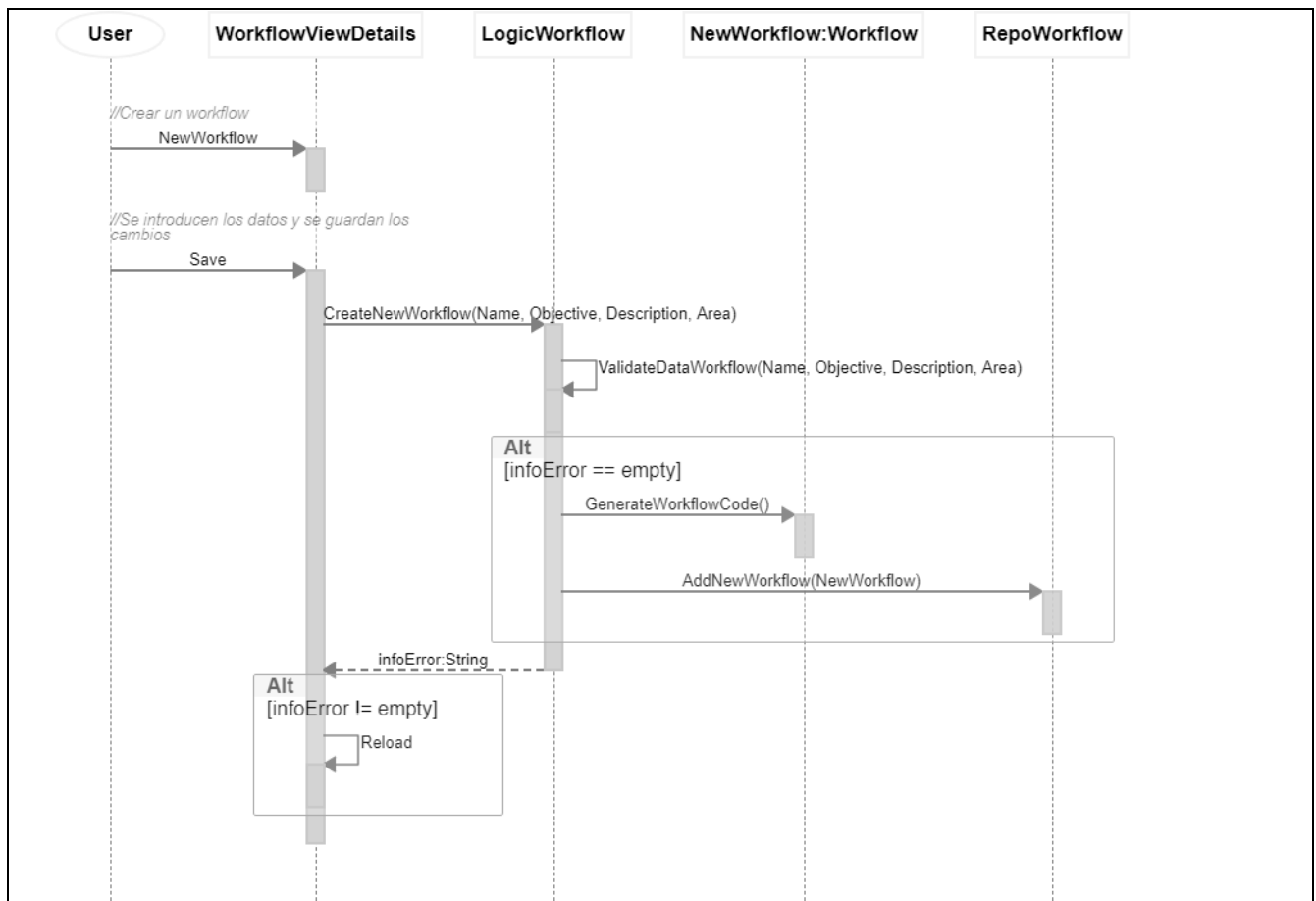
1.1.1 Visualizar workflows.



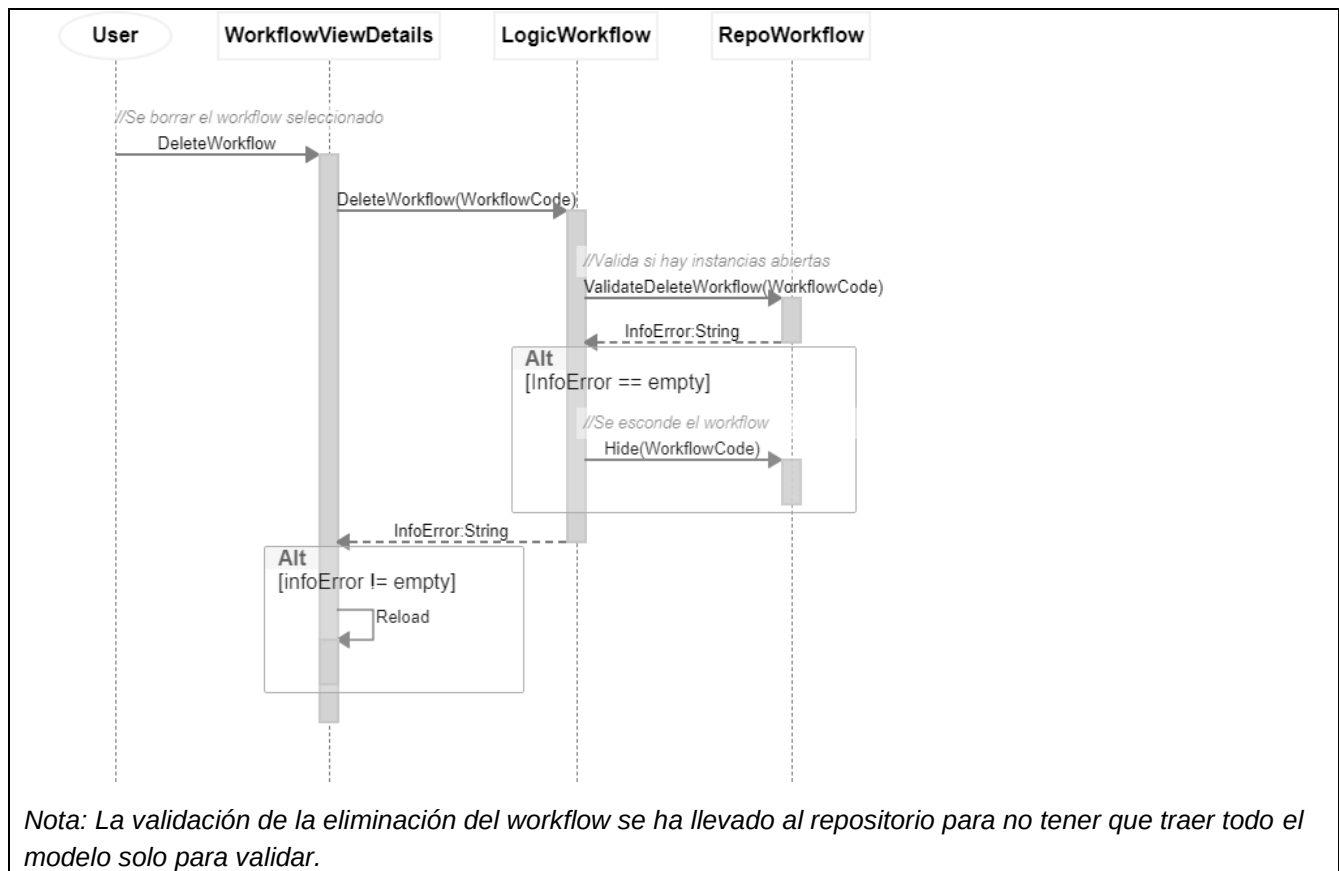
1.1.2 Cambio de área



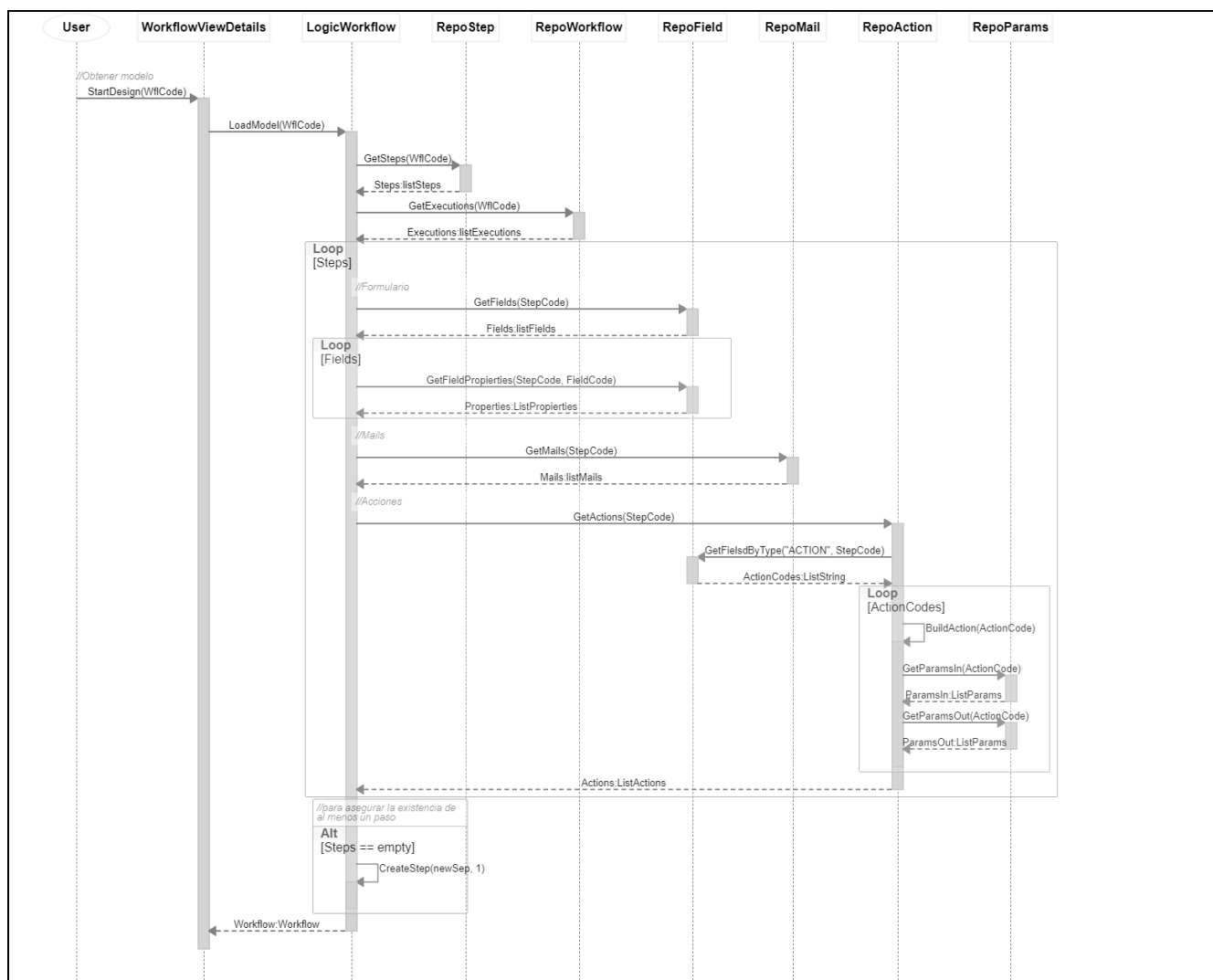
1.1.3 Crear un nuevo workflow



1.1.4 Borrar workflow

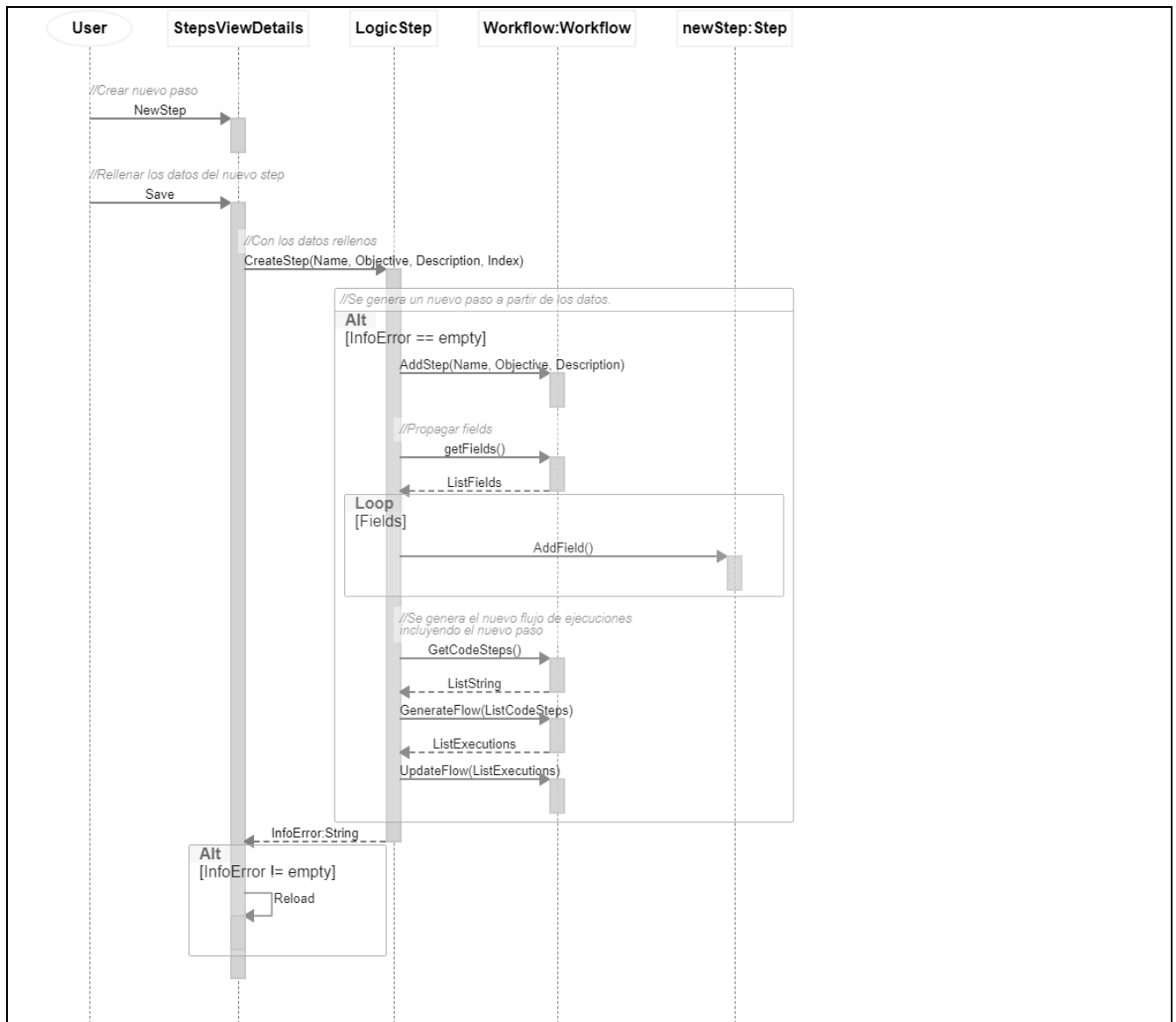


1.1.5 Obtener un modelo de workflow.

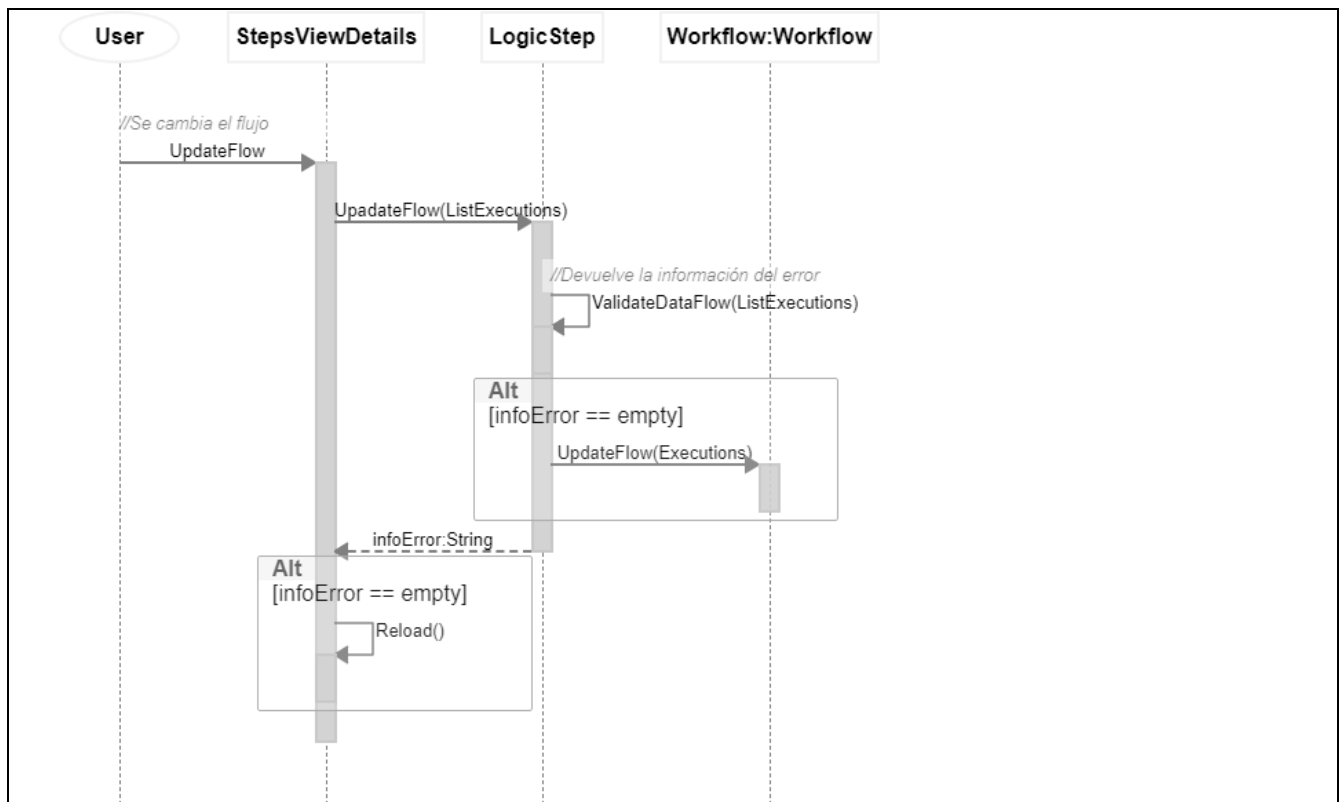


1.2 Vista Steps

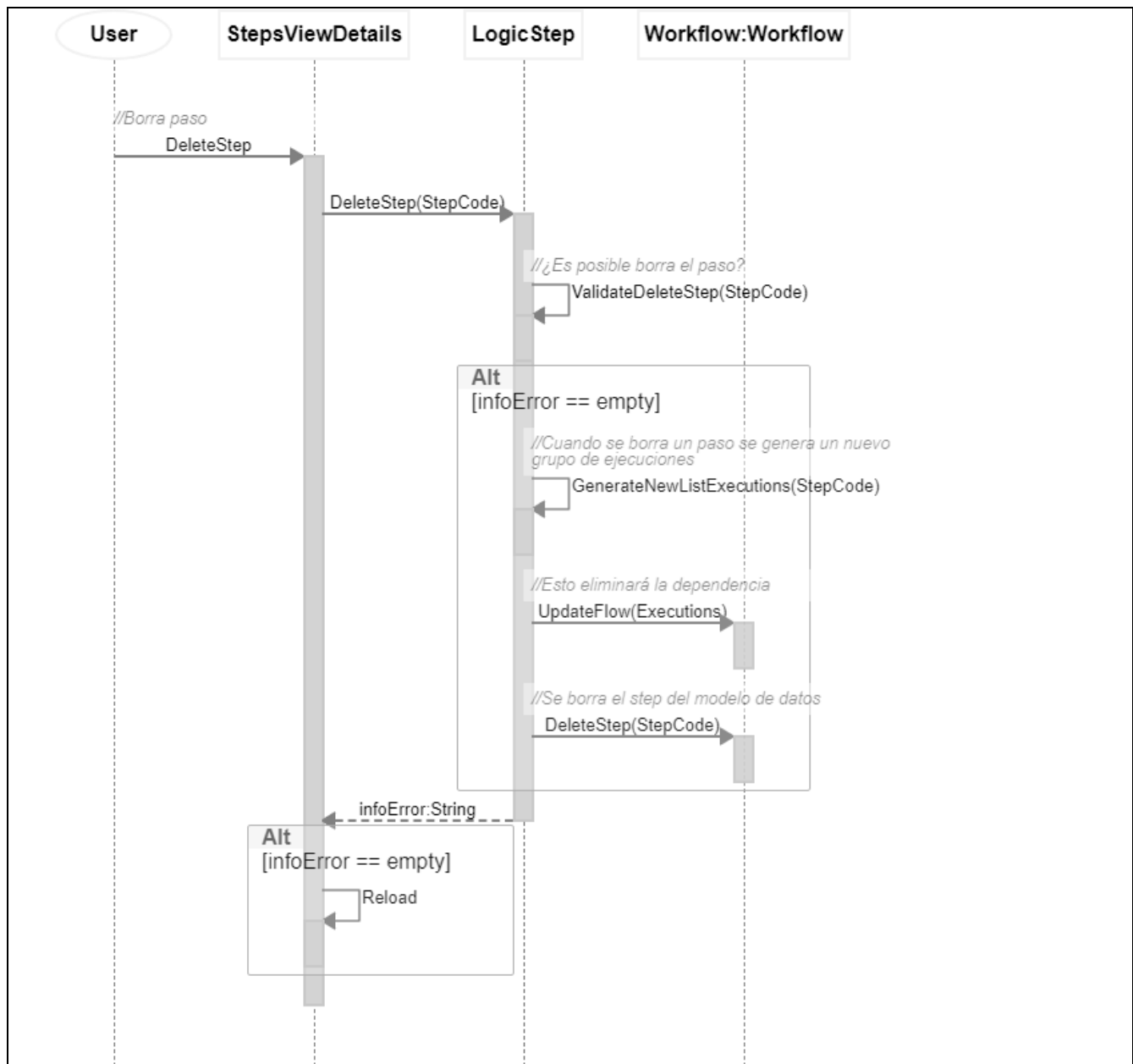
1.2.1 Añadir paso



1.2.2 Modificar flujo de pasos

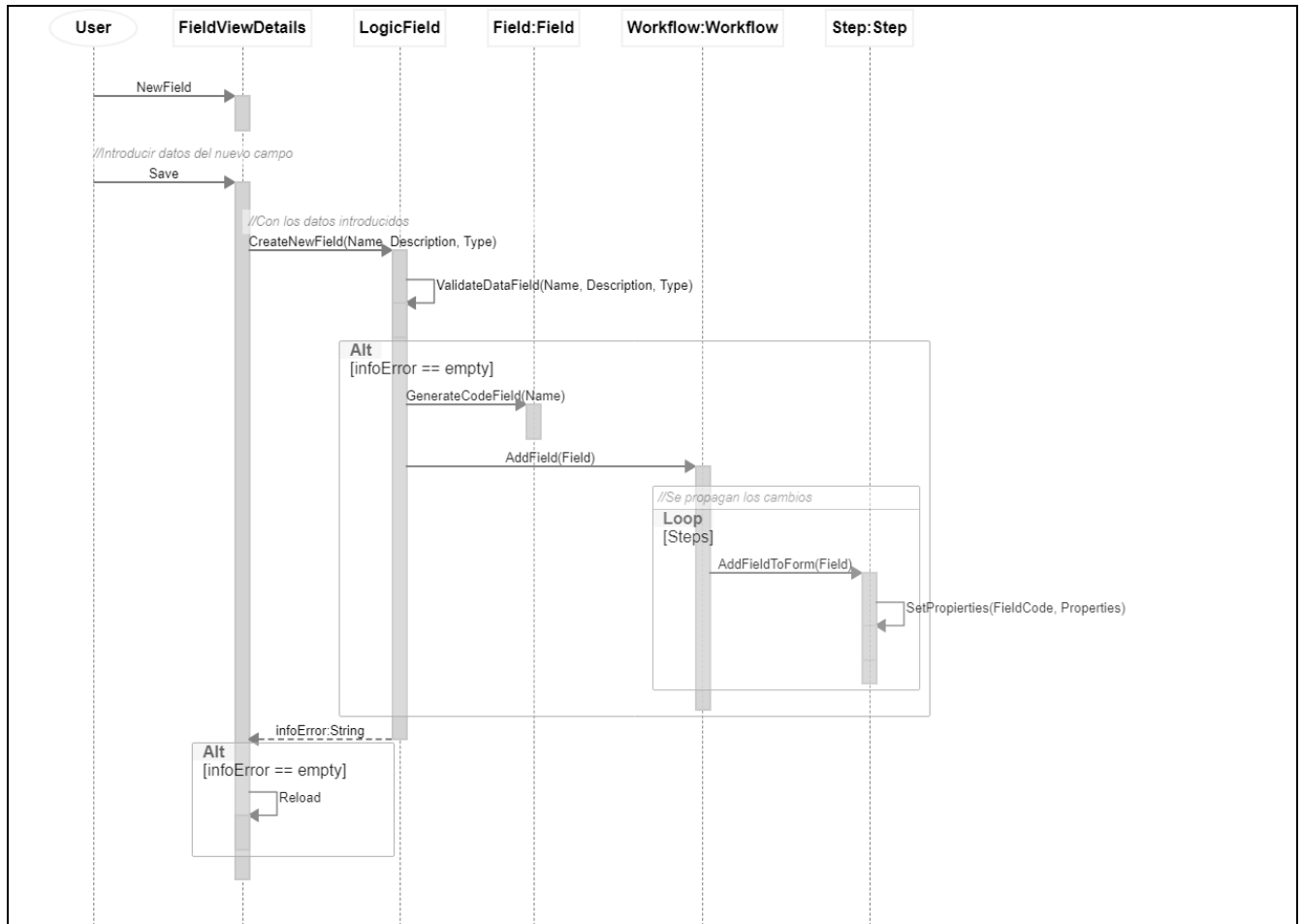


1.2.3 Borrar paso

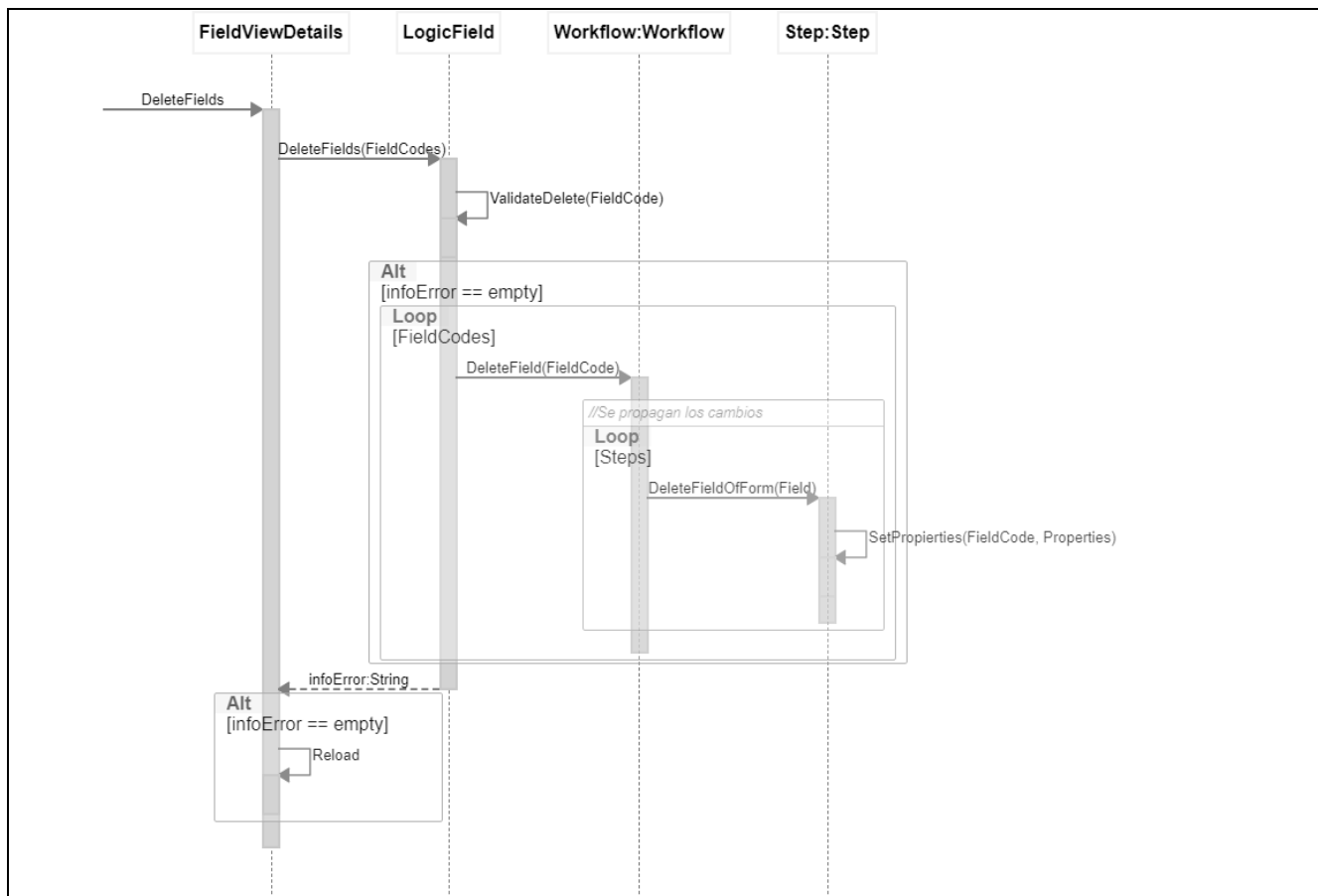


1.3 Vista Fields

1.3.1 Crear nuevo campo

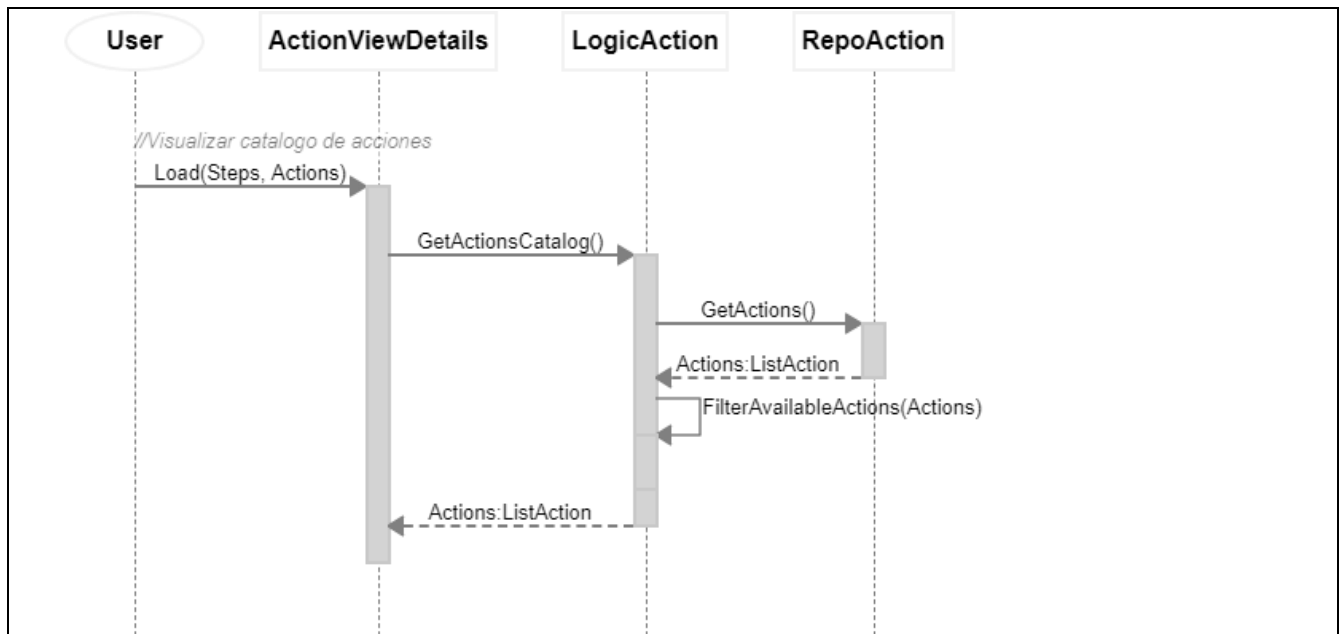


1.3.2 Borrar campo

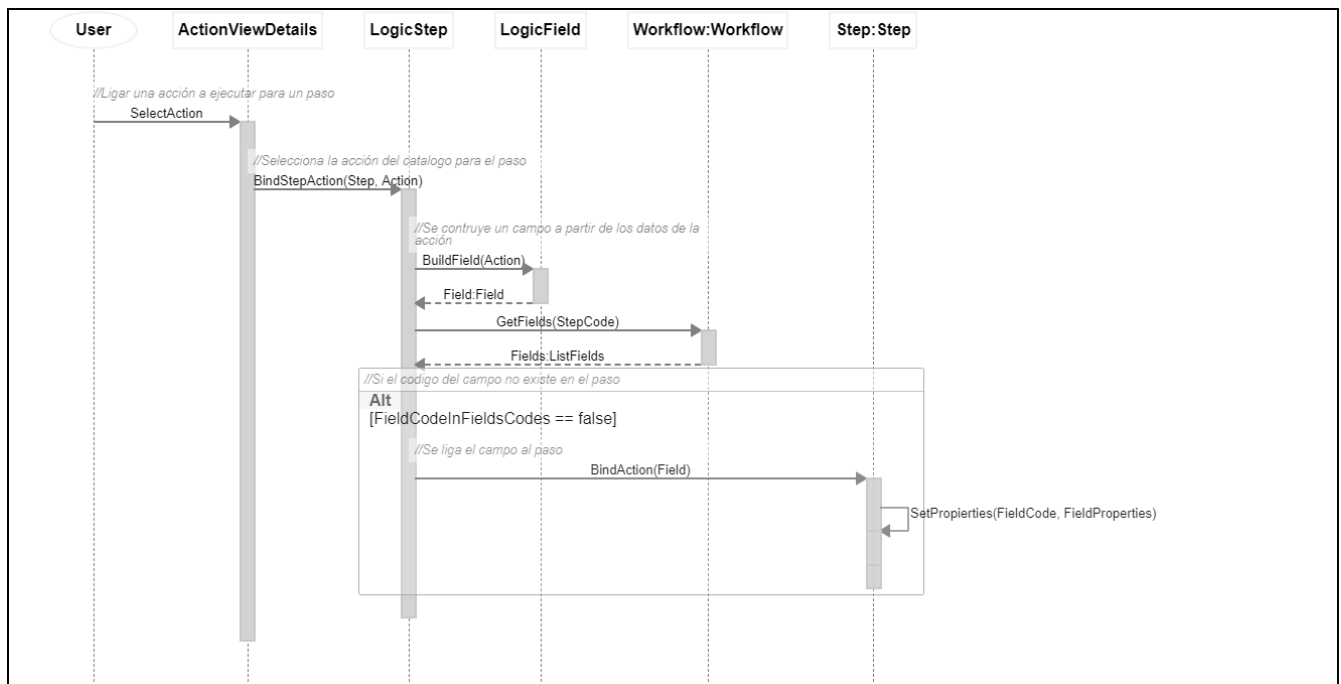


1.4 Vista Actions

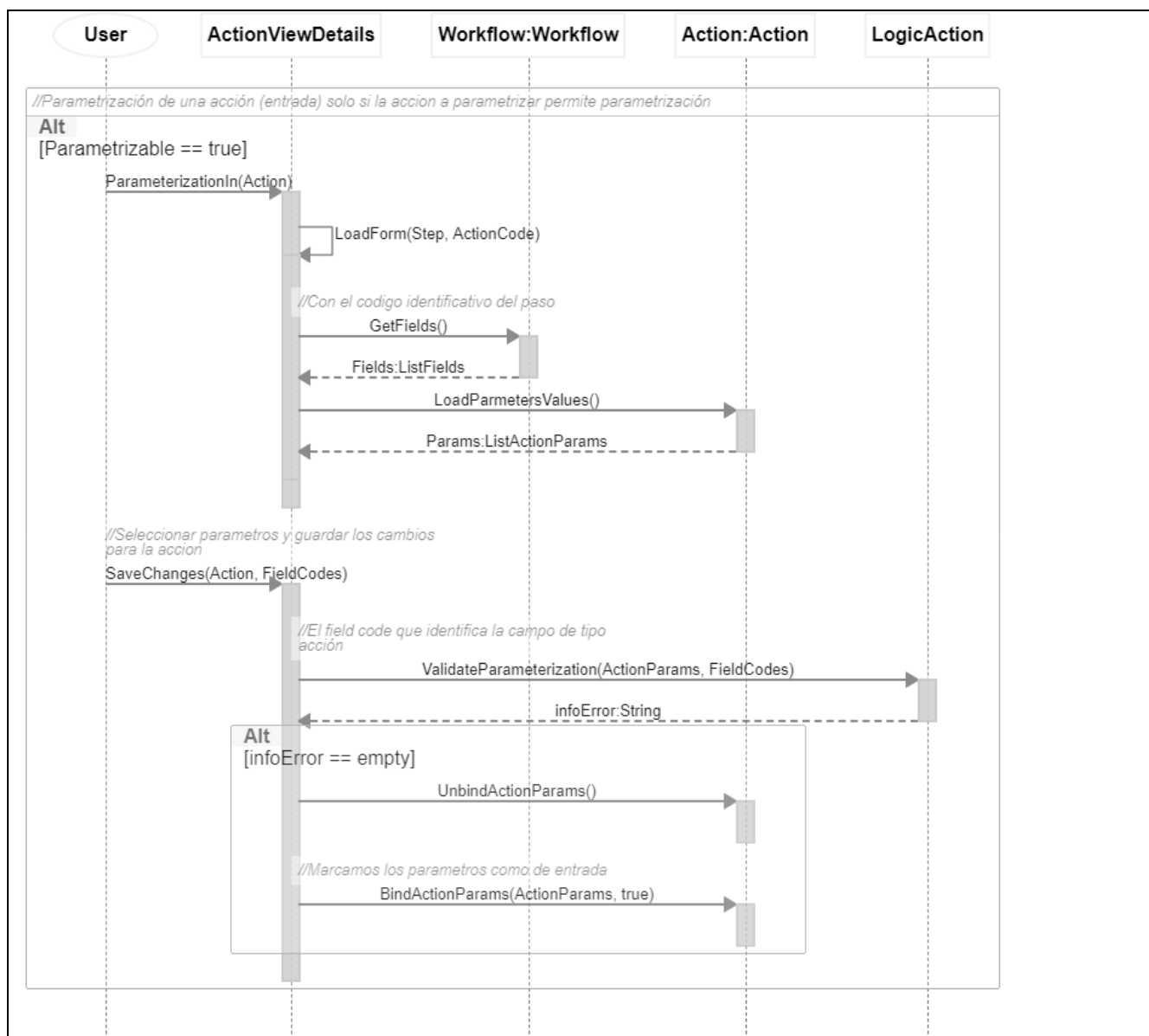
1.4.1 Visualizar catálogo de acciones



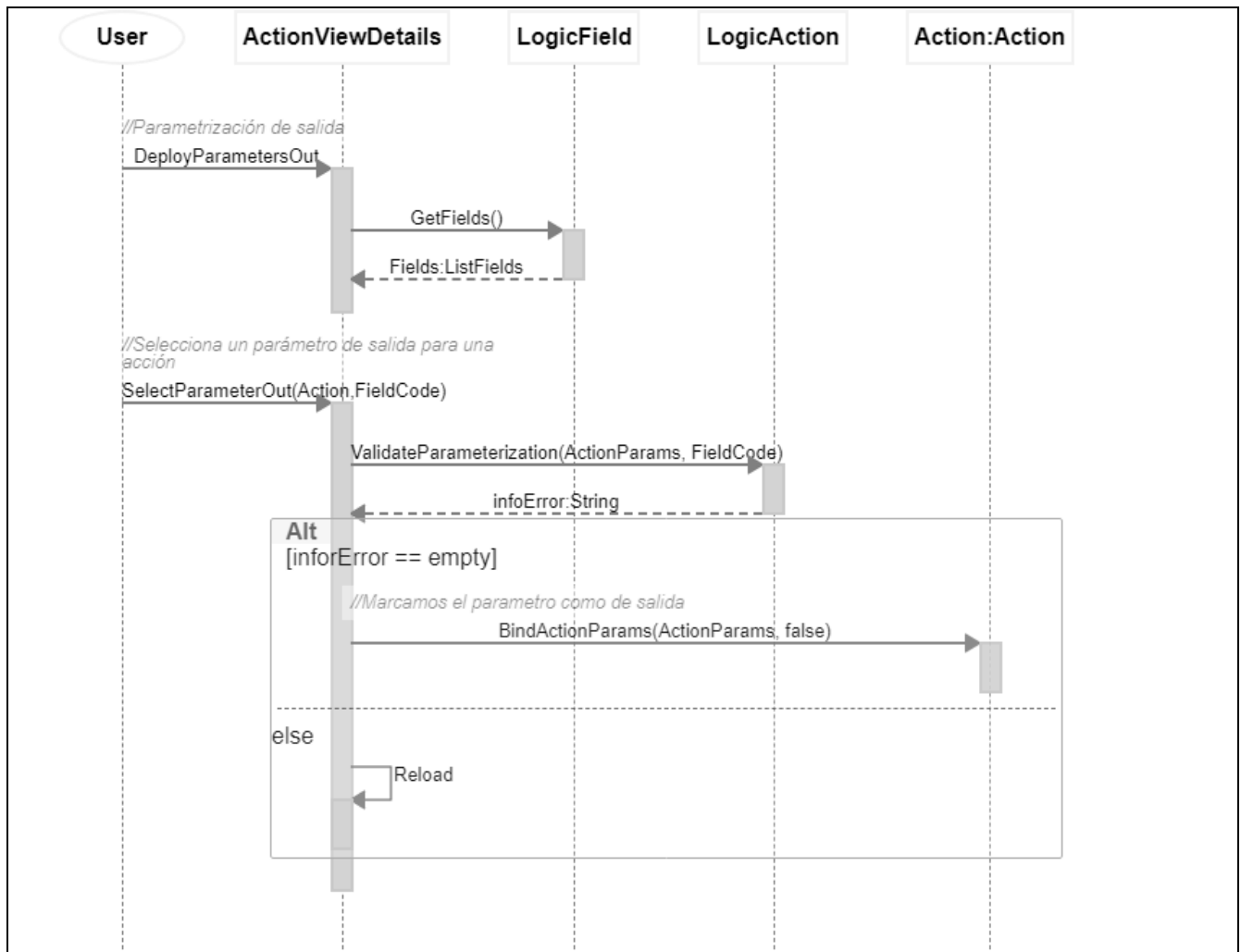
1.4.2 Ligar acción al paso al paso



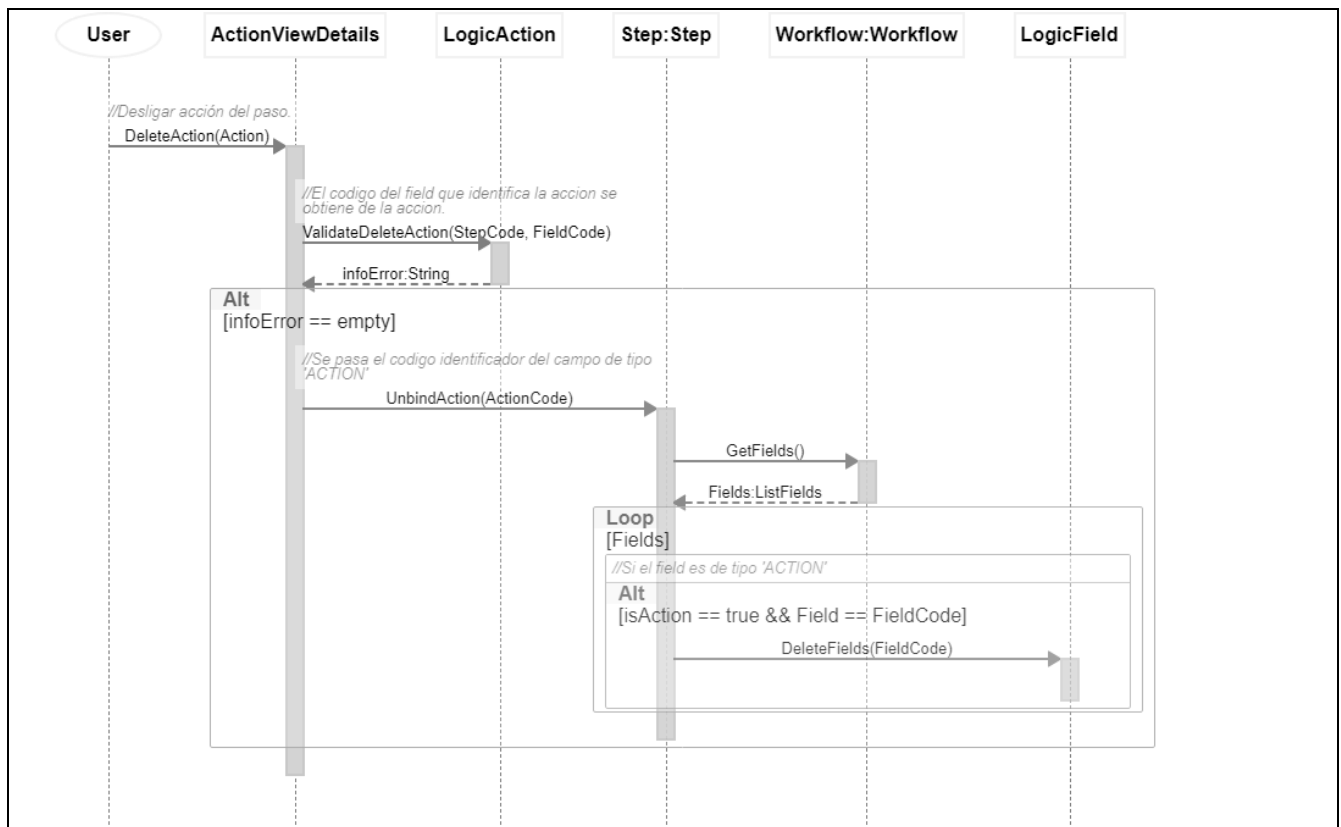
1.4.3 Parametrizar acción (entrada)



1.4.4 Parametrizar acción (salida)

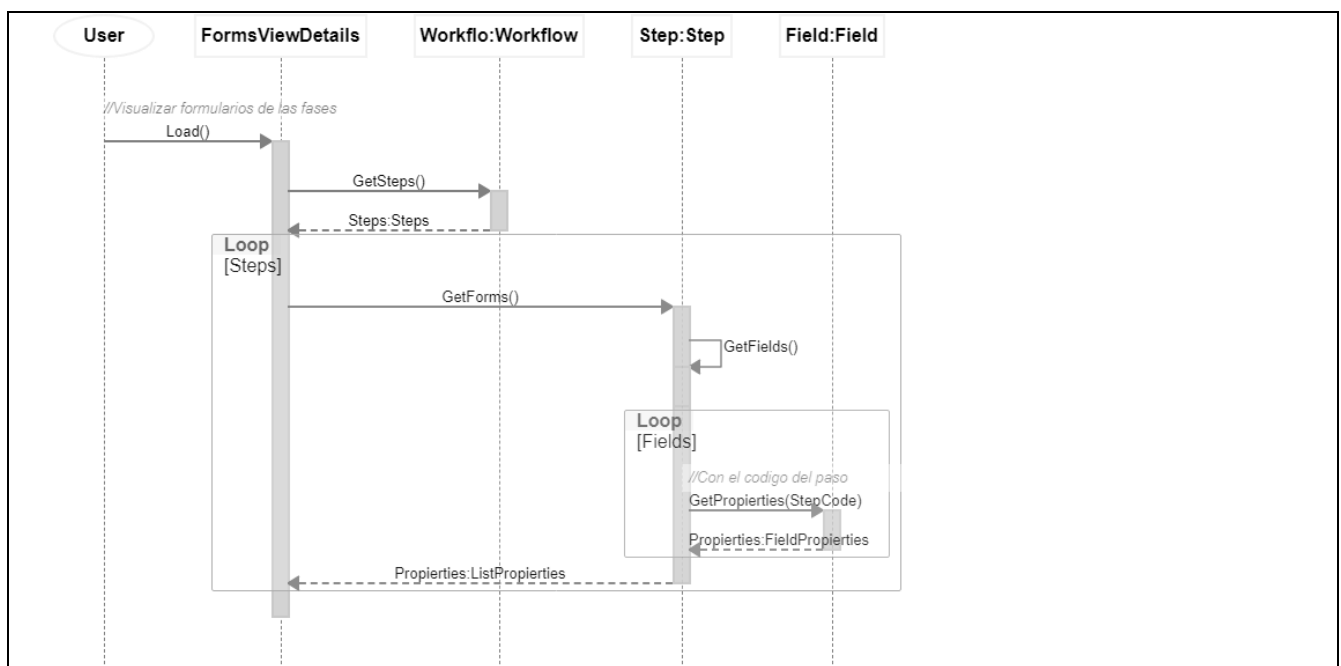


1.4.5 Desligar una acción del paso.

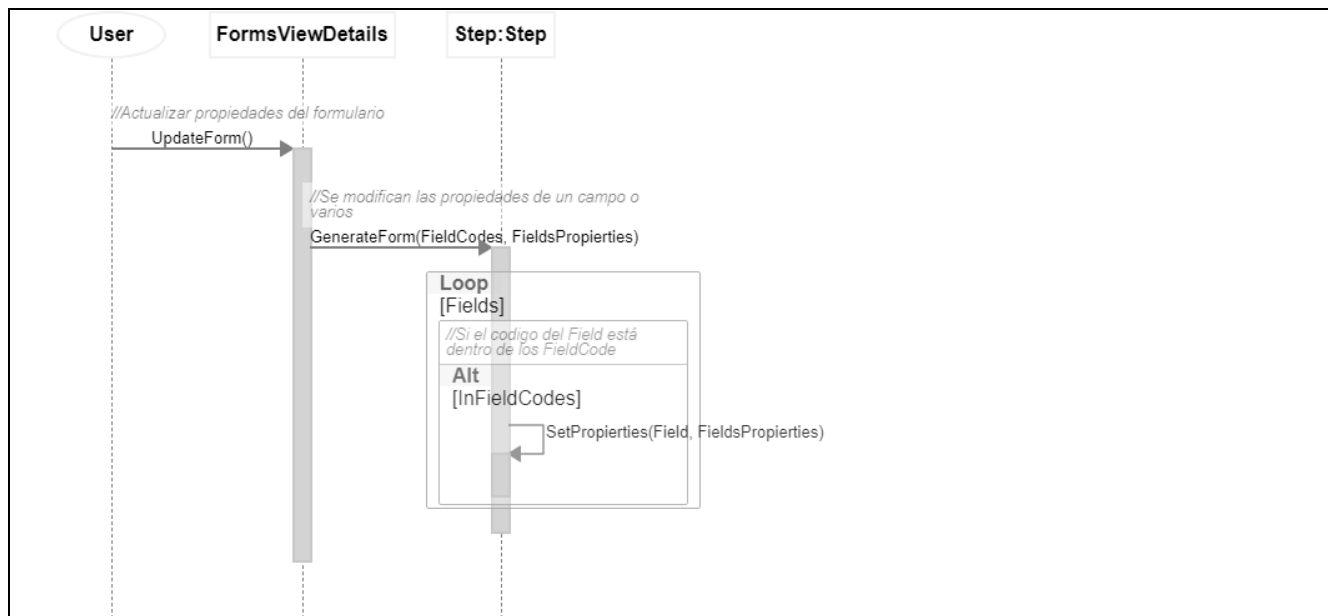


1.5 Vista Forms

1.5.1 Visualizar formularios de las fases

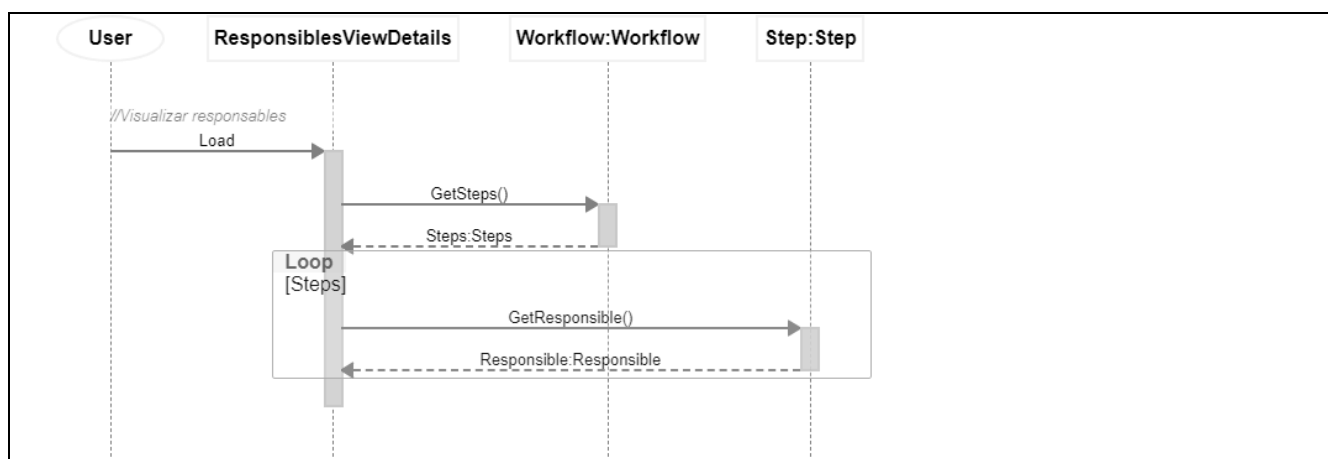


1.5.2 Actualizar propiedades del formulario



1.6 Vista Responsibles

1.6.1 Visualizar responsables de los pasos

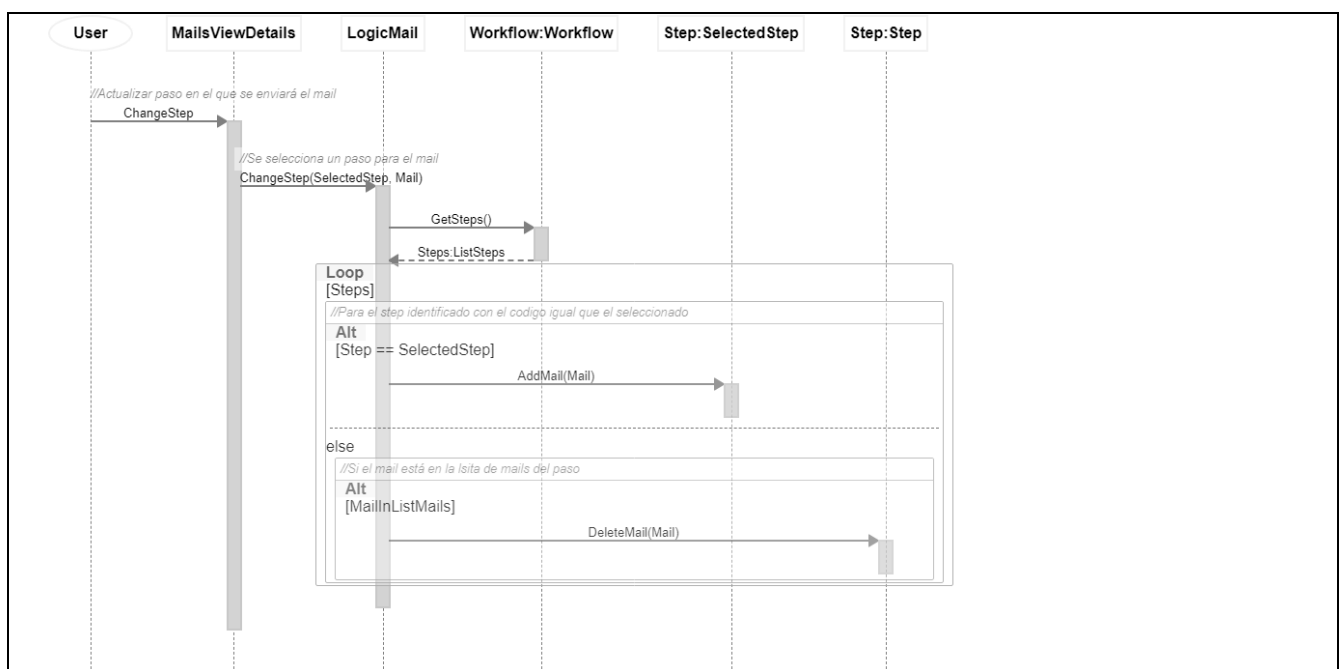


1.6.2 Actualizar tipo de responsable

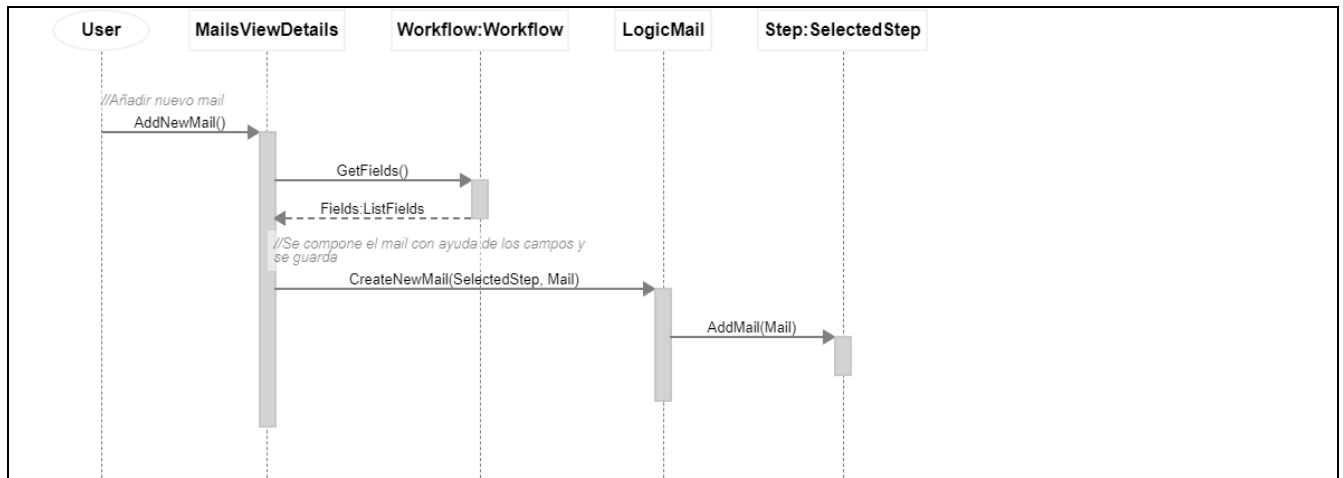


1.7 Vista Mails

1.7.1 Cambio de paso en el que se enviará el mail



1.7.2 Añadir un nuevo mail

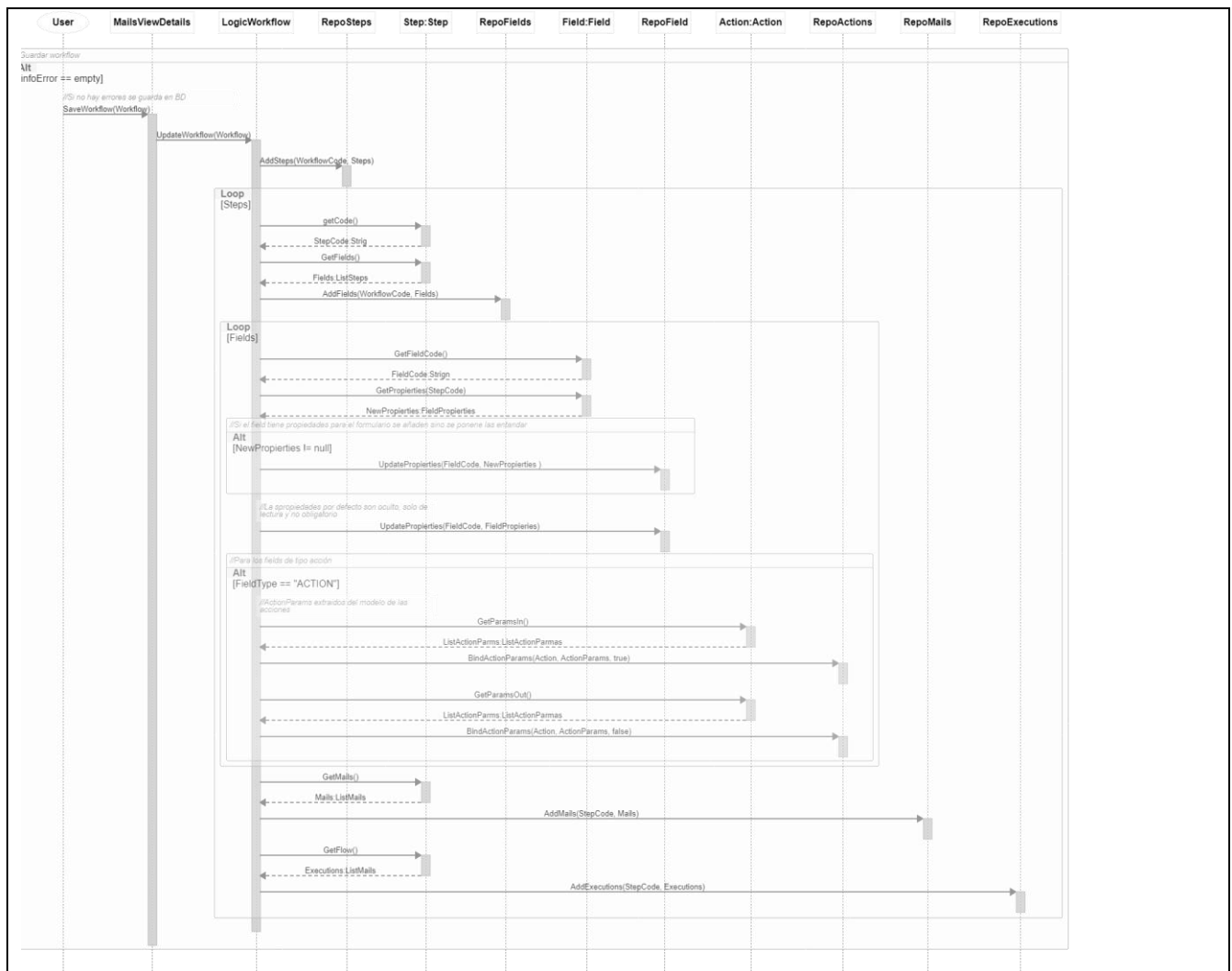


1.7.3 Borra un mail del paso

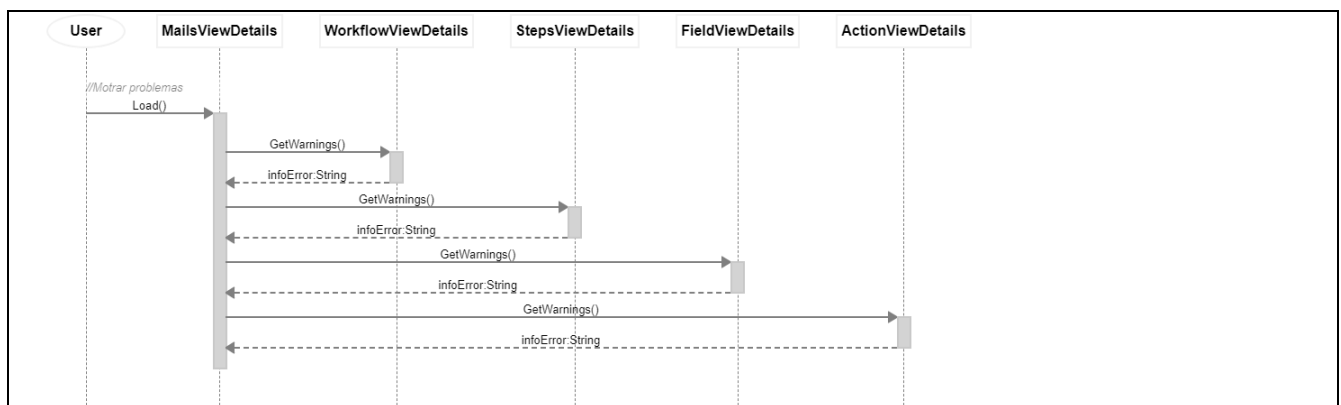


1.8 Vista Finalize

1.8.1 Guardar Workflow



1.8.2 Visualizar posibles errores





Universidad
Zaragoza

ANEXO 4

Lógica de control

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 4

Este anexo contiene la especificación de la lógica de control del módulo, *Workflow Designer*. Solo lógica procedimental.

ÍNDICE

índice	2
1. Lógica de control	3
1.1 LOGICWORKFLOW	3
1.2 LOGICSTEP	5
1.3 LOGICACTION	7
1.4 LOGICFIELD	7
1.5 LOGICMAIL	9

1. Lógica de control

1.1 LOGICWORKFLOW

Nombre método	Pre	Post
GetWorkflows		Workflow[]
Descripción: Devuelve una lista de workflows editables por el módulo.		
Obtener la lista de workflows. Mientras haya workflows en la lista - Verificar si el workflows está soportado. - Verificar si el workflow está oculto Devolvemos la lista de workflows, con los soportados marcados y sin los ocultos.		
Nombre método	Pre	Post
CreateNewWorkflow	Name: String, AreaID: String, Description: String	refError; string
Descripción: Crea un nuevo workflow.		
Si los datos del formulario son validados - Se genera el código identificativo del workflow - Se añade el nuevo workflow al repositorio Se devuelve el error con la información de la validación del form.		
Nombre método	Pre	Post
ValidateDataWorkflow	Name: String, AreaID: Int, Description: String, Error: String	String
Descripción: Valida los principales datos el workflow.		
Si el nombre está vacío o tiene caracteres raros es invalido. Si la descripción está vacía no es válido. Si el área asignada no es válida En caso de producirse se devuelve la información del error		
Nombre método	Pre	Post
LoadModel	WorkflowCode: String	Workflow

Descripción: Devuelve un modelo de workflow que cumple el modelo funcional. En el modelo se incluirán los pasos, los campos, los formularios, las acciones que realiza, los responsables de los pasos y las comunicaciones enviadas.

Para el workflow cuyo código identificador es WorkflowCode

Se obtienen los pasos del workflow

Se obtiene el flujo de ejecuciones

Para cada paso

- Se obtienen los pasos
- Para cada campo
 - o Obtener sus propiedades
- Obtener sus mails de comunicación
- Para cada acción del paso
 - o Obtener los sus parámetros de entrada y salida

Se devuelve el modelo del workflow

//En la obtención del workflow si el número de pasos es cero se generará uno para asegurarnos que
//siempre haya un paso

Nombre método	Pre	Post
ValidateUpdateWorkflow	Workflow: Workflow	Boolean, refError: String

Descripción: Valida los datos del modelo del workflow para su actualización.

El workflow debe existir en el área para ser válido.

Si tiene pasos.

Mientras el workflow tenga pasos.

- Los pasos deben tener un nombre valido, una descripción válida y un objetivo valido.
- El responsable de cada paso debe ser tipo acotado
- Un paso no debe tener más de un padre dentro del flujo de paso.
- Validar que se haya parametrizado las características del formulario en el paso.

Mientras el workflow tenga campos.

- Se valida que los campos sean correctos. Se obtiene su error si fallan.
- Si tienen campos del tipo acción
 - o Se actualizan/insertan la acción

Si tiene mails de comunicación

- Que tenga asignado un destinatario, que tenga cuerpo y un asunto válido.

Nombre método	Pre	Post
UpdateWorkflow	Workflow: Workflow	refError: String

Descripción: Actualiza los datos del workflow con los del formulario de entrada.

Se valida el workflow

Se actualiza el nombre, área, descripción, estado.

Si tiene campos

- Se actualizan/insertan todos los campos.

Si tiene pasos

- Se actualizan/insertan el nombre, descripción, objetivos, responsables del paso.

- Se actualiza el flujo de pasos
- Si el paso tiene acciones asignadas.
 - o Se actualiza/inserta el campo para acción.
 - o Se actualiza/inserta la parametrización de las acciones.
- Si tiene mails para enviar
 - o Se crean los nuevos mails.

Si tiene pasos y campos

- Se crean se relacionan todos los pasos con todos los campos.
- Se crean formularios para todos los pasos.
- Si tiene mails.
- Se actualizan/insertan los mails.

Nombre método	Pre	Post
DeleteWorkflow	WorkflowCode: String	String
Descripción: Elimina un workflow		
Si el workflow es válido para su eliminación		
- Se esconde el workflow		
En caso de no ser borrado devuelve la información del problema		

1.2 LOGICSTEP

Nombre método	Pre	Post
DeleteSteps	StepCode:String	String
Descripción: Valida y borra los pasos de la lista para el workflow.		
Valida el borrado del paso cuyo StepCode sea el del parámetro		
- Si es valido genera un nuevo grupo de ejecuciones que cumpla la linealidad - Para el workflow se actualiza el flujo con el nuevo grupo de ejecuciones - Se borra el paso		
En caso de que la validación sea errónea se devuelve la información sobre el error, en caso contrario el error vacío.		
Nombre método	Pre	Post
ValidateDataFlow	Executions:Execution[]	String
Descripción: Valida que la secuencia sea valida		
Recorre la lista y verifica que la secuencia de la lista de los parámetros sea lineal.		
Si no es lineal se devuelve la información del error.		
Nombre método	Pre	Post

BindStepAction	Step: Step, ActionCode: String	String
Descripción: Se relaciona el paso con un campo de tipo "ACTION" en base de la acción seleccionada Se construye un campo a partir de los datos de la acción. Si el código del campo no existe entre los campos del paso. - Se liga el campo al paso		
Nombre método	Pre	Post
ValidateDeleteStep	StepCode: String	String
Descripción: Valida la eliminación del paso. Si el número de paso del workflow es mayor que uno - Se borra el step.		
Nombre método	Pre	Post
CreateStep	Name:String, Objective:String, Index:integer	String
Descripción: Crea un nuevo paso para el workflow y lo enlaza a la ejecución Genera un paso a partir de los datos. Genera una cadena de ejecución valida. Si la validación es correcta - Actualiza el flujo de ejecuciones.		
Nombre método	Pre	Post
GenerateNewListExecutions	StepCode: String, Index:integer	Executions:Execution[]
Descripción: Generar una nueva cadena de ejecuciones lineal sin el step Si el índice es nulo se genera una lista de ejecuciones. Si el índice tiene contenido se genera una lista de ejecuciones con el StepCode en el índice.		
Nombre método	Pre	Post
ValidateDeleteAction	StepCode: String, FieldCode: String	String
Descripción: Valida la eliminación de una acción. Si ha sido asignada como parámetro en acciones posteriores. - Se rellena la información de error. Se devuelve el error, vacío si no hay errores.		

Nombre método	Pre	Post
UpdateStep	Nombre: String, Obejtivo: String, Descripción: String	
Descripción: Actualiza con el modelo dado el step		
Para el paso cuyo código identificador sea el código del paso. - Actualizar su nombre, de objetivos, descripción. - En caso de ser un nuevo paso insertar los datos con el código. - Actualizar las propiedades del paso		

1.3 LOGICACTION

Nombre método	Pre	Post
ValidateParameterization	ActionParams:ActionParams[], FieldCodes: String[]	String
Descripción: Valida que la parametrización de la acción sea correcta		
Para cada FieldCode - Obtener el campo con su tipo - Comparar el tipo del campo con el tipo del parámetro de su mismo índice en la lista de ActionParams. Si todo son parámetros tiene el tipo igual devolver el error vacío. Si hay tipos aceptables devolver advertencia Si hay alguno que no coincide devolver información del error.		
Nombre método	Pre	Post
GetActionsCatalog		Actions
Descripción: Devuelve un catálogo de acciones disponibles para el workflow		
Devolver la lista de acciones obtenidas desde el repositorio de acciones		

1.4 LOGICFIELD

Nombre método	Pre	Post
ValidateDataField	Code: String, Type: String, Name: String, Description: String, IntValue: int, StringValue: String,	String
Descripción: Comprueba que un campo del workflow sea válido.		
Si el campo es nuevo y CodeFied existe en el repo de campo no es válido. Si el tipo es diferente de los tipos validos el campo no es válido. Los campos deben tener un nombre valido. El campo debe tener una descripción valida. Para cada campo verificar que está completa la parametrización de su tipo. En caso de cumplirse el campo es válido. Que no sea un valor usado como parámetro por alguna de las acciones Que no se use dentro del cuerpo de un mail Si algunas de las condiciones se devuelve la información del error		
Nombre método	Pre	Post
CreateNewField	Name: String, Description: String, Type: String	String
Descripción: Se inserta el nuevo campo en los formularios del workflow.		
Si el formulario es valido - Se genera un código identificativo para el nuevo campo - Se propaga el campo a cada paso Si la validación es errónea se devuelve la información del error		
Nombre método	Pre	Post
ValidateDeletesFields	FieldCodes: String[]	String
Descripción: Comprueba que un field se puede borrar.		
El campo no debe estarse usando como parámetro en alguna de las acciones.		
Nombre método	Pre	Post
DeleteFields	FieldCodes: String[],	String
Descripción: Borra los campos cuyo código este contenido en la lista de códigos		
Para cada código de FieldCodes - Validar la eliminación Si la validación es correcta - Se borran las características del campo en los pasos. Si existe algún error en la validación se devuelve la información del error		

Nombre método	Pre	Post
BuildField	Action:Action	Field
Descripción: Construye un objeto de tipo Field con los dataos de un tipo Action.		
Generar un código identificativo para el Field. Se crea un nuevo objeto de tipo Field con tipo "ACTION" como atributo de tipo, con nombre de la acción y descripción como descripción de la acción. Devuelve el nuevo campo.		

1.5 LOGICMAIL

Nombre método	Pre	Post
CreateNewMail	StepCode: String, Mail:Mail	
Descripción: Crea un nuevo mail para el paso.		
Crea un nuevo mail para el paso del código		
Nombre método	Pre	Post
ValidateDelete	StepCode: String, MailCode: String	Boolean
Descripción: Valida si se puede borrar el mail		
Si el código del step existe y está relacionado con el paso cuyo código es el indicado por el parámetro. La validación es correcta. En caso contrario la validación es incorrecta.		
Nombre método	Pre	Post
ChangeStep	Step: Step, Mail:Mail	
Descripción: Actualiza el paso en que se enviará el mail		
Se elimina el mail de los pasos que contengan mail con el mismo código identificativo Si el paso de parámetro no contiene el correo lo añade al paso.		



Universidad
Zaragoza

ANEXO 5

Especificación del modelo de datos

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 5

En este apartado se especifican los atributos y sus tipos para cada entidad del modelo de datos usado.

ÍNDICE

Índice	3
1. Especificación	4
1.1 Modelo de Datos	4
1.1.1 Workflow	4
1.1.2 Step	5
1.1.3 Field	6
1.1.4 Execution	7
1.1.5 Responsible	7
1.1.6 FieldsPropiedades	8
1.1.7 Actions	8
1.1.8 ActionParam	9
1.1.9 Mails	9

1. ESPECIFICACIÓN

1.1 Modelo de Datos

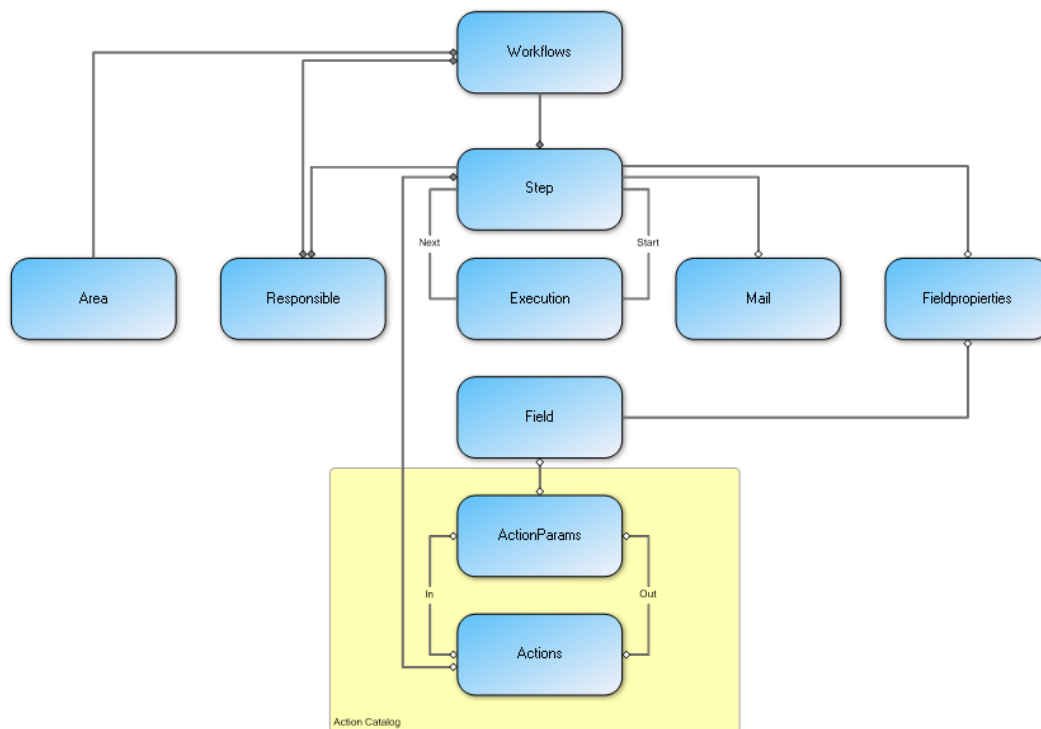


Figura 1. Esquema de entidades y sus relaciones que conforman el modelo de datos del sistema.

1.1.1 Workflow

Entidad Workflow utilizada para mantener las referencias a los Workflows que controla el cliente.

Nombre	Tipo	Descripción
ID	Integer	Código que funge como identificador en base de datos.
WorkflowCode	String	Código que será utilizado para identificar el Workflow (aun existiendo un identificador). Al ser más fácil de identificar es una ayuda a la mantenibilidad del código. Es un código único autogenerado, a partir del nombre del Workflow, cuyo valor no debe ser nulo, aunque lo permite. Su control se hace por código.
AreaCode	String	Valor que identifica el área a la que pertenece el Workflow.
Status	String	Define el estado del Workflow en el momento. Los valores contemplados son "Activo" e "Histórico". Para el módulo estableceremos por defecto un valor "Histórico" lo que hace invisible el Workflow hasta que se active su

uso. Desde la vista del administrador de la aplicación principal.

Name	String	Nombre que se mostrará al usuario. Se usa tanto en las instancias del Workflow como para mostrar el Workflow del cliente en la aplicación. Se usará para mostrar el Workflow en el módulo. El tipo se ha tratado como String por problemas de conversión del valor en los valores de primitivos de tipo char cuando se quiere recuperar del objeto DTO.
Steps	Step[]	Lista de fases que componen la secuencia de ejecución del Workflow.
StepsOrder	Execution []	Este atributo establece el flujo de datos que seguirá el Workflow fase a fase. El orden de ejecución va definido dentro como una lista porque puede haber diferentes variaciones. Ósea llegar a la misma fase paso desde diferentes otras diferentes.
IsSupported	Boolean	Valor de soporte. No indica si un Workflow es soportado para ser editado por el modulo. Su valor por definición es falso.
KeyName	String	Atributo usado por las instancias del Workflow para mostrar un identificador de la instancia al usuario. Es necesario componer el metadato hasta el final de la creación del Workflow para que pueda ser mostrado por las instancias.

1.1.2 Step

Entidad Step. Define las fases que se ejecutaran durante el recorrido del Workflow. Cada fase o paso se corresponde que un formulario que debe ser completado o validado (dependiendo de la operatividad o de como se vaya a definir el Workflow). Es posible avanzar a través de las fases una vez valido, completado, cancelado o rechazado. Se seguirá el avance a través de los Steps dependiendo del flujo definido en la propiedad del Workflow "StepOrder".

Nombre	Tipo	Descripción
Code	String	El atributo se usa como identificador dentro del módulo y la aplicación. Nuevamente aun existiendo una clave identificadora se usa para hacer mantenible el código. Incluye como parte del su código el propio PrivilegeTypeId de la entidad Workflow.

Name	String	Nombre de la etapa en la que se encuentra el Workflow paso
Description	String	Descripción del paso en que se encuentra el Workflow. Información auxiliar a las instrucciones.
Instructions	String	Puesto que estamos en un formulario puede ser necesario dar instrucciones de uso del paso al usuario.
AllowReject	Boolean	Muestra el botón de rechazo del paso en los pasos, se usan en pasos de validación para el responsable del paso en el Workflow.
AllowCancel	Boolean	Muestra el botón de cancelar el Workflow al responsable.
ManualSave	Boolean	Muestra del botón de guardado del step para este Workflow.
FieldsPropierties	FieldPropierties[]	Lista de FieldPropierties que define las propiedades de cada campo dentro del formulario.
Fields	Field[]	Lista de todos los campos en general que componen los pasos del Workflow. De esta forma todos los Workflows están disponibles en todos los pasos.

1.1.3 Field

Entidad Field. Define los campos que compondrán los formularios para cada fase del proceso del Workflow. Los “fields” (campos) son interpretados por el motor de Workflow para componer los formularios.

Nombre	Tipo	Descripción
ID	Integer	Código que funge como identificador en base de datos.
Code	String	Código que será utilizado para identificar el campo dentro del Workflow
Type	String	Define el tipo de campo que los formularios contendrán. Para el módulo se han acotado en diez tipos, algunos de los más habituales. <i>Label, Labelview, Labelsimple, Textline, Textmultipleline, Integer, Currency, Admitem, Date, Yesno, Datalistdropdown, Action.</i>
Description	String	Descripción del campo en la fase. Se suele usar para describir el uso del campo.

StringValue	String	Este atributo se usa para almacenar texto que será interpretado por el motor. Para el módulo almacenará el nombre del método que se ejecutará cuando el campo es de tipo “ <i>Action</i> ”.
IntValue	Integer	Campo que define que se hará con el contenido dentro del campo “ StringValue ”: Interpretarlo, leerlo o ejecutarlo.

1.1.4 Execution

Usado como array dentro del Workflow. Se traduce como la ejecución del flujo de trabajo. Por lo general el paso primero es un paso nulo. Al ser el primero no existe un origen, solo uno o varios pasos siguientes.

Nombre	Tipo	Descripción
StartStepCode	String	Código identificador de la fase de origen.
NextStepCode	String	Código identificador de la fase siguiente a la que avanzará el Workflow desde el origen StartStepCode

1.1.5 Responsible

Aunque para un Workflow puede existir más de un responsable para el caso del módulo se han acotado. Esto es así porque el motor de Workflow es capaz de calcular los responsable dependiendo de los valores que se le pasen en esta sección pero no se contempla para el modulo dado que no se ha pensado para establecer múltiples responsables porque sería demasiado complicado de usar para un usuario objetivo.

Nombre	Tipo	Descripción
ResponsibleType	String	Código identificador de la fase de origen.
ResponsibleValueInt	String	Valor interpretado por el motor de Workflow. Dependiendo del valor se puede establecer un responsable de la fase. Se han acotado los posibles valores dl atributo para el modulo. <i>Creator, EmployeeID, JobID, Privilege</i> . Cada uno con un valor entero asignado.
AllowReject	Boolean	Permite al responsable del paso rechazar el Workflow en

AllowCancel	Boolean	Permite al responsable del paso cancelar el avance de la fase y vuelve a la anterior. Si ese es el caso o donde se desee, dependiendo del Workflow.
--------------------	---------	---

1.1.6 FieldsProperties

Para cada fase del Workflow será necesario definir las características de los campos que componen los formularios de los pasos. Las características de cada campo difieren para cada paso de ahí que haya que relacionar sus las propiedades del campo con el paso.

Nombre	Tipo	Descripción
FieldCode	String	Código que relaciona la propiedad con el campo para poder construir el formulario.
IsMandatory	Boolean	Permite establecer si un campo es obligatorio de ser completado. Puede ser completado tanto por la interpretación del motor como por el usuario. Dependiendo del tipo de campo y de Workflow.
IsReadOnly	Boolean	Permite establecer si un campo es solo de lectura o no. Depende del Workflow y del tipo de campo que definido.
Min	Integer	Define la cantidad mínima para campos del tipo entero.
Max	Integer	Define la cantidad máxima para campos del tipo entero.
Order	Integer	Define la posición del campo del campo del formulario de la fase.
Hidden	Boolean	Permite establecer que campos serán visibles en el formulario.
ManulaSave	Booelan	Permite establecer si se desea dejar al responsable guardar el estado del campo en esa fase del Workflow en ese momento.

1.1.7 Actions

Después de cada fase es posible ejecutar diferentes acciones que manipulen los datos, hagan cálculos o simplemente se inserten en la capa de datos. Estas acciones deben ser parametrizadas específicamente para el paso en el que se han definido.

Nombre	Tipo	Descripción
FieldCode	String	Código que relaciona la propiedad con el campo para poder construir el formulario.
DisplayName	Boolean	Permite establecer si un campo es obligatorio de ser completado. Puede ser completado tanto por la interpretación del motor como por el usuario. Dependiendo del tipo de campo y de Workflow.
NameMethod	String	Nombre del método dentro de la interfaz que contiene las definiciones de las acciones que se ejecutaran.
ParamsIn	ActionParam[]	Lista de ActionParam que define los parámetros de entrada.
ParamsOut	ActionParam[]	Lista de ActionParam que define los parámetros de salida. Solo se devolverá un valor para el módulo.

1.1.8 ActionParam

Modelo de datos que funge como registro que contiene la información de los parámetros que se usarán en la acción con la que se encuentran relacionados.

Nombre	Tipo	Descripción
DisplayName	String	Atributo que indica el nombre del parámetro que verá el usuario. Se usa solo en lectura.
Name	String	Nombre del parámetro dentro de la interfaz. Se usa solo en la lectura.
Type	String	Define el tipo de dato que debe recibir el parámetro.
FieldCode	String	Código del campo del que se extraerá el valor que se usará en la ejecución de la acción.

1.1.9 Mails

Cuando el Workflow avanza en su flujo por las diferentes fases es posible configurar comunicaciones salientes cada vez que se avanza, se cancela o se finaliza el Workflow.

Nombre	Tipo	Descripción
ID	Integer	Identificador del mail dentro de la capa de datos.
Type	Integer	Determina el tipo de destinatario al que va dirigido el mail. Los valores. <i>Privilege, Emailaddress, Creator</i> . Cada uno con un valor asignado.
ContentType	Integer	Determina el tipo de contenido que irá dentro del cuerpo. Para el modulo el valor por defecto será 6.



Universidad
Zaragoza

ANEXO 6

Lógica de objetos

Autor

Anderson David Bastidas Delgado

Director

Jose Carlos Cimorra Velilla

Ponente

Jorge Delgado Gracia

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 6

Especificación de la lógica que seguirán los objetos del módulo.

ÍNDICE

Índice	3
1. LÓGICA DE OBJETOS	4
1.1 WORKFLOW	4
1.2 AREA	6
1.3 STEP.....	7
1.4 EXECUTION.....	10
1.5 FIELD.....	10
1.6 FIELDPROPIERTIES	11
1.7 ACTION	11
1.8 ACTIONPARAM	12
1.9 MAIL.....	12
1.10 RESPONSIBLE	13

1. LÓGICA DE OBJETOS

1.1 WORKFLOW

ATRIBUTOS		
• ID:Integer • WorkflowCode:String • AreaCode:String, • Status:String • Name:String • Description:String • KeyName:String • Status:String • IsSupported:Boolean, • Steps:Steps[] • StepsOrder:Execution[] • Fields:Field[]		
METODOS		
Nombre método	Pre	Post
AddField	Field	
Descripción Se inserta el campo para que esté disponible en los formularios del Workflow.		
Añade el campo a lista de campos Para cada paso del Workflow - Añadir el campo al formulario. - Pone las propiedades del campo a solo lectura y oculto.		
Nombre método	Pre	Post
DeleteExecutions		
Descripción: Elimina la cadena de ejecuciones de los pasos		
Pone a valor nulo la cadena de ejecución.		
Nombre método	Pre	Post
AddExecutions	Executions:Execution[]	
Descripción: Añade una cadena de ejecución		

Nombre método	Pre	Post
UpdateFlow	Executions:Execution[]	String
Descripción: Se actualiza el flujo con		
Si la validación del flujo es correcta		
- Se elimina todas las entradas de la cadena de ejecución - Se inserta la nueva cadena de ejecución.		
Nombre método	Pre	Post
UpdateForm	Step:Step, FieldsPropierties:FieldPropierties[], FieldCodes:String[]	
Descripción: Actualiza el formulario del paso con las propiedades.		
Para cada campo del paso cuyo que se encuentre dentro de FieldCodes		
- Actualizar con las propiedades pasadas como parámetro.		
Nombre método	Pre	Post
GenerateForm	FieldCodes: String[], FieldPropierties: FieldPorpierties[]	
Descripción: Genera un formulario para el paso		
Para cada campo cuyo código esté dentro de la lista de códigos.		
- Actualiza las propiedades del parámetro de FieldPropierties.		
Si no está dentro se mantienen las estándar		
Nombre método	Pre	Post
GetMails		Mails[]
Descripción: Obtiene una lista de correos del paso.		
Retorna la lista de mails del step.		
Nombre método	Pre	Post
ChangeArea	Area: Area	
Descripción: Cambia de área del Workflow		
Carga el Código del área		
La asigna como nueva área.		

Nombre método	Pre	Post
AddNewStep	Name: String, Objective: String, Description: String	String
Descripción: Se añade un nuevo paso al Workflow		
Se compone el nuevo paso, de tipo Step Lo añade a la lista de pasos Se genera una nueva lista de ejecuciones con el nuevo paso.		
Nombre método	Pre	Post
GenerateWorkflowCode		String
Descripción: Se genera un código identificativo del Workflow		
Partiendo del nombre del Workflow se genera un código para el Workflow. Debe ser en mayúsculas, sin caracteres raros y sin espacios.		
Nombre método	Pre	Post
GetFlow		Executions[]
Descripción: Devuelve la lista de ejecuciones		
Nombre método	Pre	Post
GetSteps	WorkflowCode: String	Step[]
Descripción: Devuelve la lista de pasos del Workflow ordenados según la secuencia de ejecución		
Nombre método	Pre	Post
GetFields	StepCode: String	Fields[]
Si StepCode está vacío devolver todos los campos del Workflow. Si el StepCode no es vacío devolver la lista de campos del paso que esté identificado con el StepCode del parámetro.		

1.2 AREA

ATRIBUTOS

- ID: Integer
- Code: String
- Name: String
- Description: String
- ParentArea: Integer

METODOS

Nombre método	Pre	Post
GetCode		String

Descripción: Devuelve el código identificativo del Área.

Devuelve el atributo Code del área.

1.3 STEP

ATRIBUTOS

- ID:Integer
- Code:String
- Name:String
- Description:String
- Instructions:String
- AllowReject:Boolean
- AllowCancel:Boolean
- ManualSave:Boolean
- Responsibles:Responsible[]
- FieldsPropierties: FieldsPropierties[]
- Actions:Action[]
- Mails:Mails[]

METODOS

Nombre método	Pre	Post
GetActions		Actions[]

Descripción: Devuelve una lista de acciones del paso.

Obtener los campos de tipo 'ACTION' del paso con su modelo.

Nombre método	Pre	Post
GetForm		FieldPropierties[]

Descripción: Devuelve las propiedades de los campos del paso

Para cada campo del paso

- Obtener sus propiedades

Devuelve una lista con todas las propiedades.

Nombre método	Pre	Post
---------------	-----	------

BindAction	Actions: Action	
-------------------	-----------------	--

Descripción: Liga una acción al paso

Añadir el nuevo campo de tipo acción al paso

Si la acción pasa es parametrizable

- Se desligan los parámetros de la acción (se eliminan las relaciones)
- Se crean las nuevas relaciones (Se ligan los parámetros de entrada y salida.)

Nombre método	Pre	Post
---------------	-----	------

UnBindAction	FiledCodes:String	String
---------------------	-------------------	--------

Descripción: Elimina la ejecución de un campo de tipo 'ACTION'

Comprobar que el FiledCodes solo contiene códigos de tipo 'ACTION'

- Eliminar los campos cuyo FieldCode esté contenido en la lista de códigos.

Nombre método	Pre	Post
---------------	-----	------

UpdateForm	Step:Step, FieldsPropiedades:FieldPropiedades[], FieldCodes:String[]	
-------------------	--	--

Descripción: Actualiza el formulario del paso con las propiedades.

Para cada campo del paso cuyo que se encuentre dentro de FieldCodes

- Actualizar con las propiedades pasadas como parámetro.

Nombre método	Pre	Post
---------------	-----	------

GetResponsible		Responsible
-----------------------	--	-------------

Descripción: Obtener el responsable del paso

Nombre método	Pre	Post
---------------	-----	------

AddFieldToForm	Field: Field	
-----------------------	--------------	--

Descripción: Se añade el campo al formulario del paso

Se añade el campo a la relación entre el paso y el campo.
Las propiedades asignadas en la inserción son por defecto oculto y solo de lectura.

Nombre método	Pre	Post
ChangeResponsible	Responsible:Responsible	String

Descripción: Actualiza los datos del responsable del paso

Actualiza el tipo de responsable del paso y sus propiedades: Cancelar y finalizar

Nombre método	Pre	Post
GetMails		Mais[]

Descripción: Obtiene una lista de correos del paso.

Obtiene una lista de mails que no sean de cancelación ni de inicio.
Retorna la lista de mails del step.

Nombre método	Pre	Post
SetPropierties	FieldCode:String, FieldPropierties: FieldPropierties[]	

Descripción: Actualiza las propiedades de los formularios. Las propiedades permiten visualizar, ocultar, volver obligatorio o establecer que sea de solo lectura.

Para el campo cuyo código identificativo sea el de FieldCode actualizar las propiedades del parámetro

Nombre método	Pre	Post
DeleteMail	MailCode:String	

Descripción: Elimina un mail para el paso

Para cada mail del paso seleccionado

- Si contiene un mail con el mismo código identificativo.
- Se borra de la lista de mails el mail.

Nombre método	Pre	Post
GenerateForm	FieldCodes: String[], FieldPropierties: FieldPorpierties[]	

Descripción: Genera un formulario para el paso		
Para cada campo cuyo código esté dentro de la lista de códigos.		
- Actualiza las propiedades del parámetro de FieldPropiedades.		
Nombre método	Pre	Post
GetCode		
Descripción: Devuelve el código del paso.		

1.4 EXECUTION

ATRIBUTOS
- StartStepCode:String - NextStepCode:String

1.5 FIELD

ATRIBUTOS

- ID:Integer
- FieldCod:String
- Type:String
- Name:String
- Description:String
- StringValue:String
- IntValue:Integer

METODOS

Nombre método	Pre	Post
GetPropierties	StepCode: String	FieldsPropierties []
Descripción: Devuelve la lista de propiedades del campo para el paso		
Buscar las propiedades de las propiedades con el campo		
Se devuelve la lista de propiedades del formulario del paso.		
Nombre método	Pre	Post

GenerateCodeField	NameMethod: String, FieldCode: String, StepCode: String	ActionParam
Descripción: Genera un código identificativo para el campo a partir del nombre		
A partir del nombre del campo se compone un código identificativo del campo.		

1.6 FIELDPROPIERTIES

ATRIBUTOS
• FieldCode:String • IsMandatory:Boolean • IsReadOnly:Boolean • Min:Ineger • Max:Integer • Order:integer • Hidden:Boolean • ManualSave:Booelan

1.7 ACTION

ATRIBUTOS

- FieldCode:String
- DisplayName: String
- NameMethod:String
- ParamsIn: ActionParamas[]
- ParamsOut:ActionParamas[]

METODOS

Nombre método	Pre	Post
GetActionParamsOut		ActionParam[]

Descripción: Devuelve los parámetros de salida de un campo de tipo acción

Nombre método	Pre	Post
GetActionParamsIn	ActionCode:String	ActionParam[]

Descripción: Devuelve ve los parámetros de entrada de un campo de tipo acción

Nombre método	Pre	Post
---------------	-----	------

BindActionParams	Params: ActionParam[], ParamsAreIn: Boolean	
-------------------------	--	--

Descripción: Relaciona una acción con sus parámetros de entrada y salida.

Para cada Parámetro

- Si el parámetro es de entrada se añade a la relación de la acción y el parámetro de entrada
- Si el parámetro es salida se añade a la relación de la acción y el parámetro de salida.

Nombre método	Pre	Post
---------------	-----	------

IsParametrizable	Action	Boolean
-------------------------	--------	---------

Descripción: Indica si una acción es parametrizable

Si la acción con uno o más parámetros es parametrizable.

Nombre método	Pre	Post
---------------	-----	------

UnbindActionParams	Params: ActionParam[],	
---------------------------	------------------------	--

Descripción: Se elimina tanto parámetros de entrada como de salida.

1.8 ACTIONPARAM

ATRIBUTOS

- DisplayName:String,
- Name:String,
- Type:String,
- Value: String //Referencia al código del campo

1.9 MAIL

ATRIBUTOS

- ID:Integer,
- Type:Integer
- ContentType:Integer=6,

- CommunicationContent:String
- CommunicationSubject:String,
- IsCancelCommunication: Boolean,
- IsRejectCommunication:Boolean,
- Code:String,
- SendToConf:Boolean

1.10 RESPONSIBLE

ATRIBUTOS

- NameTypeResponsible:String
- DescriptionTypeResponsible:String
- ResponsibleType:Integer,
- AllowReject:Boolean,
- AllowCancel:Boolean



Universidad
Zaragoza

ANEXO 7

Lógica de acceso a datos

Autor/es

Anderson David Bastidas Delgado

Director/es

Jose Carlos Cimorra

Ponente

Jorge Delgado

Escuela Universitaria Politécnica de Teruel

2018

ANEXO 7

En este anexo se especifica la lógica que siguen los repositorios en el acceso de base de datos. Esta lógica es la que hay que añadir a los DAO de acceso a datos de la BD para el correcto funcionamiento del módulo cuando se acople.

ÍNDICE

Índice	2
1. Lógica de acceso a datos	3
1.1 REPOWORKFLOW	3
1.2 REPOAREA	4
1.3 REPOACTION	5
1.4 REPOSTEP	6
1.5 REPOFIELD	6
1.6 REPOMAIL	7

1. LÓGICA DE ACCESO A DATOS

1.1 REPOWORKFLOW

METODOS		
Nombre método	Pre	Post
IsSuported		Boolean
Descripción: Indica si este workflow es apto para editarse en el WorkflowEditor.		
Obtener número de instancias abiertas del workflow. Mientras hay instancias. - Si posee instancias abiertas no es editable. Si tiene pasos, obtener los pasos del workflow y su flujo de ejecución. Mientras hay ejecuciones - Si alguno de los pasos posee más de un paso anterior el workflow no es editable. Mientras hay pasos. - Si el responsable cuyo tipo es diferente de los definidos no es editable. - Si el paso tiene mails del tipo que se envían al iniciar un paso no es editable. Si tiene campos, obtener los campos del workflow. Mientras hay campos. - Si existe dependencia entre campos del workflow no es editable. - Si existe un campo del workflow cuyo tipo difiere de los tipos asignados no es editable. Si no se cumple ninguna de las condiciones anteriores es editable.		
Nombre método	Pre	Post
ValidateDeleteWorkflow	WorkflowCode:String	String
Descripción: Comprueba si el workflow se puede borrar. La validación de la eliminación del workflow se ha llevado al repositorio para no tener que traer todo el modelo solo para validar. "ValidateDeleteWorkflow"		
Si existe el workflow - El workflow no debe tener instancias abiertas - o Se cambia a oculto el estado del workflow En caso de no haber problema se devuelve el error vacío		
Nombre método	Pre	Post
Hide	WorkflowCode	
Descripción: Comprueba si el workflow se puede borrar		
Actualiza a oculto el estado del workflow identificado con código.		

Nombre método	Pre	Post
GetWorkflows		Orh_wfl_workflows[]
Descripción: Devuelve la lista de workflow disponibles		
Nombre método	Pre	Post
AddNewWorkflow	Workflow: Orh_wfl_workflows	
Descripción: Añade o actualiza el workflow a la tabla de workflows.		
Nombre método	Pre	Post
GetExecutions	WorkflowCode:String	Orh_Wfl_WorkFlowExecution[]
Descripción: Añade o actualiza el workflow a la tabla de workflows.		
Nombre método	Pre	Post
AddExecutions	WorkflowCode:String, Executions: Orh_Wfl_WorkFlowExecution []	
Descripción: Añade una cadena de ejecución		
Elimina toda la cadena de ejecuciones del workflow y la sustituye por la nueva. No hay problemas de dependencia.		
Nombre método	Pre	Post
GetFields	StepCode:String	Orh_Wfl_WorkFlowFields[]
Descripción: Devuelve la lista de campos del workflow		

1.2 REPOAREA

METODOS		
Nombre método	Pre	Post
GetAreas		Orh_Wfl_WorkFlowArea[]
Descripción: Devuelve la lista de áreas disponibles		

1.3 REPOACTION

METODOS		
Nombre método	Pre	Post
GetParamsIn	ActionCode:String	R_Wfl_WorkFlowActionMethod _WorkFlowArctionParamsIn[]
Descripción: Devuelve la lista de Mails del paso. Devuelve los parámetros de entrada dependiendo de lo que se requiera.		
Nombre método	Pre	Post
GetParamsOut	ActionCode:String	R_Wfl_WorkFlowActionMethod _WorkFlowArctionParamsOut[]
Descripción: Devuelve la lista de Mails del paso. Devuelve los parámetros de salida.		
Nombre método	Pre	Post
BuildAction	ActionCode:String	Actions[]
Descripción: Devuelve un catálogo de acciones disponibles para el workflow. Construir los objetos de tipo 'ACTION' mediante reflection - Para cada acción - o Obtener los parámetros de entrada de la acción. - o Obtener los parámetros de salida de la acción. //Los parámetros se extraen de la relación n-n entre la acción y el step. Devolver la lista de acciones junto a los parámetros.		
Nombre método	Pre	Post
GetActions	StepCode	Orh_Wfl_WorkFlowActionMethods[]
Descripción: Construye un objeto de tipo Field con los datos de un tipo Action. Si StepCode vacío - Se filtran las acciones disponibles - Devuelve todas las acciones disponibles en la tabla de acciones. De lo contrario - Devuelve solo las acciones que estén relacionadas con el StepCode		
Nombre método	Pre	Post
BindActionParams	ActionCode: String, ActionParamsCodes: String[]	

AreIn:Boolean

Descripción: Relaciona los parámetros pasados con la acción.

Si es de entrada

- Se borran las entradas de la relación de entrada
- Se añaden las nuevas relaciones

Si es de salida se borran las entradas de salida.

- Se borran las entradas de la relación de salida
- Se añaden las nuevas relaciones

1.4 REPOSTEP

METODOS

Nombre método

Pre

Post

GetSteps

WorkflowCode: String

Orh_Wfl_WorkFlowStep[]

Descripción: Devuelve la lista de pasos del workflow

Nombre método

Pre

Post

AddSteps

WorkflowCode: String

Steps:Orh_Wfl_WorkFlowSteps[]

Orh_Wfl_WorkFlowStep[]

Descripción: Añade si es necesario de lo contrario actualiza la lista de pasos del workflow

1.5 REPOFIELD

METODOS

Nombre método

Pre

Post

GetFieldPropiedades

StepCode:String,
FieldCode:String

R_Wfl_WorkFlowStep
_WorkFlowField[]

Descripción: Devuelve la lista de áreas disponibles

Nombre método

Pre

Post

AddFields

WorkflowCode: String,

Orh_Wfl_WorkFlowFields[]

Fields: Orh_Wfl_WorkFlowInstanceFields[]		
Descripción: Añade los campos y los relaciona con todos los pasos del workflow.		
Nombre método	Pre	Post
GetFieldsByType	Type:String, StepCode:String	String[]
Descripción: Devuelve la lista de códigos de campos del tipo especificado para el paso.		

1.6 REPOMAIL

METODOS		
Nombre método	Pre	Post
GetMails	StepCode:String	Orh_Wfl_WorkFlowStepsCommunication[]
Descripción: Devuelve la lista de Mails que se envían al finalizar la fase.		
Nombre método	Pre	Post
AddMails	StepCode: String, Mails: Orh_Wfl_ WorkFlowStepsCommunication []	
Descripción: Relaciona el paso y el mail que se enviará.		
Si el mail no existe añadir como nuevo y relacionarlo con el step		
Si el mail existe actualizar la referencia.		