Informe de Análisis

Integrantes del grupo C2.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))

Fecha: Sevilla, 8 de Julio 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc170817919)

[Historial de Versiones 4](#_Toc170817920)

[Introducción 5](#_Toc170817921)

[Registro de análisis 6](#_Toc170817922)

[Requisito 217 6](#_Toc170817923)

[Descripción 6](#_Toc170817924)

[Análisis y decisiones 6](#_Toc170817925)

[Validación por el profesor 10](#_Toc170817926)

[Requisito 218 11](#_Toc170817927)

[Descripción 11](#_Toc170817928)

[Análisis y decisiones 11](#_Toc170817929)

[Validación por el profesor 12](#_Toc170817930)

[Conclusiones 13](#_Toc170817931)

[Bibliografía 14](#_Toc170817932)

# Resumen del Informe

El objetivo de este informe es proporcionar detalles sobre el análisis de todos los tipos de requisitos especificados en esta entrega del proyecto, así como las decisiones tomadas para llevarlos a cabo y enlazarlo, si procediera, a las validaciones y observaciones planteadas por el profesor encargado de la asignatura.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V1.0 | Versión inicial. Estructura y contenido del documento. | 16/02/2024 | Mª Salud Carrera Talaverón |
| V2.0 | Modificación de la plantilla para que coincida con el entregable actual. Introducción. Requisitos implementados. Conclusión. Bibliografía. | 28/06/2024 | Mª Salud Carrera Talaverón |
| V2.1 | Vista del requisito 217 corregida. | 02/07/2024 | Mª Salud Carrera Talaverón |

# Introducción

En esta entrega de la segunda convocatoria cada estudiante debe implementar aquellos requisitos obligatorios que no fueron validados en la entrega final de la primera convocatoria. En el caso del estudiante 1, estos requisitos incluyen arreglar la implementación de operaciones de managers sobre proyectos y la de operaciones de managers sobre su dashboard. Además, habrá que repetir aquellas pruebas que dejen de funcionar como resultado de estos cambios y volver a realizar el análisis de rendimiento del proyecto. Las tareas suplementarias se tratan de realizar informes de análisis y planificación y progreso que no se presentaron en su entrega correspondiente, además de añadir nuevos para esta nueva entrega.

En este informe sólo se describirán aquellas tareas que requieran modificaciones en el código del estudiante 1. La descripción de las pruebas realizadas se encuentra en el informe de testing individual.

# Registro de análisis

Los requisitos se describirán en su totalidad y no únicamente los cambios añadidos en esta entrega. Sin embargo, se especificará de qué tratan estas modificaciones y por qué se han realizado de acuerdo a la corrección indicada por el profesor tras la entrega final de la primera convocatoria.

## Requisito 217

### Descripción

Implementar las operaciones de los managers (gerentes) sobre los proyectos, que incluyen:

* Listar los proyectos que han creado.
* Mostrar los detalles de sus proyectos.
* Crear, actualizar y borrar sus proyectos. Un proyecto puede ser borrado o actualizado siempre y cuando no esté publicado. Para poder publicar un proyecto, debe tener al menos una historia de usuario asignada y todas estas deben estar publicadas. Además, no puede tener errores fatales.

### Análisis y decisiones

Para implementar este requisito, se ha creado un repositorio que define las consultas necesarias a la base de datos, un servicio por cada una de las acciones requeridas y un controlador que recoge todos los servicios implementados. Además, se han definido las vistas correspondientes y se han modificado los fragmentos necesarios para facilitar la navegación por estas funcionalidades.

Por otra parte, para controlar que un proyecto publicado no quede nunca sin historias de usuario, también hay que añadir restricciones en las funcionalidades de la entidad Assignation.

Texto

Descripción generada automáticamenteEn esta imagen podemos ver los archivos creados en el paquete “acme/features/manager/project”. El repositorio incluye la anotación “@Repository” de Spring Framework antes de la definición de la interfaz y extiende a la interfaz AbstractRepository de Acme Framework. Las consultas que define son: encontrar un proyecto por su ID, encontrar todos los proyectos por el ID de su manager, encontrar un manager por su ID, encontrar un proyecto por su código, encontrar todas las asignaciones por el ID de su proyecto y encontrar todas las historias de usuario de un proyecto a través de las asignaciones que los unan por el ID del proyecto.

El controlador tiene la anotación “@Controller” de Spring Framework y extiende a la clase AbstractController de Acme Framework, especificando que va a tratar operaciones de usuarios tipo Manager sobre entidades tipo Project. Recoge los servicios usando la anotación “@Autowired” de Spring Framework sobre la definición de cada uno de ellos, e implementa un constructor en el que especifica qué operación realiza cada servicio de la siguiente manera:

Texto

Descripción generada automáticamenteCada servicio corresponde a un tipo de operación de lectura o escritura. En el caso del servicio de publicar un proyecto, no existe una operación de “publicación”, sino que es un caso especial de actualización, por lo que se indica de forma distinta al resto de servicios. La anotación “@PostContruct” es de Javax.

Todos los servicios utilizan la anotación “@Service” de Spring Framework y extienden a la clase AbstractService de Acme Framework, especificando que va a tratar operaciones de usuarios tipo Manager sobre entidades tipo Project. Definen el repositorio utilizando la anotación “@Autowired” ya mencionada.

#### ManagerProjectListService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEl método “authorise()” controla si el usuario que intenta acceder a la URL de la funcionalidad está autorizado o no. En este caso, para listar sus proyectos un usuario debe tener el rol manager.

El método “load()” carga los datos que se necesitan para realizar la operación. En este caso, se obtiene el ID del manager a través de la petición y se obtiene la colección de proyectos creados por éste.

Por último, el método “unbind(final Project project)” crea una tupla con los parámetros de la entidad que van a ser necesarios para mostrar la información en pantalla. En este caso, el listado de proyectos sólo va a mostrar su código, título, descripción y si está en modo borrador o no. Sin embargo, se han añadido los atributos restantes (excepto el manager) al payload para poder utilizarlos en la barra de búsqueda o filtrado de la página.

#### ManagerProjectShowService.java

Este servicio es muy similar al anterior: sólo se puede mostrar la información de un proyecto si se es el manager dueño del mismo; se cargan los datos mediante el ID del proyecto, que se extrae de la petición; se incluyen todos los parámetros de la entidad normalmente, excepto el manager que tenga asociado.

#### ManagerProjectCreateService.java

En este caso, sólo se puede acceder al formulario de creación de un proyecto si se es un usuario con rol manager. El método de carga creará un nuevo proyecto vacío, que definirá en modo borrador y al que asignará como manager el usuario autenticado. El método   
“unbind” utilizará todos los parámetros del proyecto excepto su manager.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteEste servicio cuenta con otros métodos: “bind(final Project object)”, “validate(final Project object)” y “perform(final Project object)”.

El método bind se encarga de procesar los datos que se envíen en la petición. En este caso, sólo aceptará valores para los atributos código, título, descripción, errores fatales, coste y enlace.

El método validate se encarga de validar los datos recibidos por el método bind. Aparte de comprobar las restricciones simples definidas en la clase Project.java (como que el título debe tener menos de 75 caracteres y no puede estar vacío), se definen las restricciones más complejas, como que el código debe ser único.

Por último, el método perform es el que realiza la acción del servicio si el método validate no ha fallado. En este caso, guarda en la base de datos el objeto de proyecto recién creado.

#### ManagerProjectUpdateService.java

Este servicio es bastante similar al de creación. En este caso, sólo se puede actualizar un proyecto si se es el manager que lo creó. Además, debe estar en modo borrador. El proyecto que se carga se busca en la base de datos a partir del ID de la petición. Para validar que el código del proyecto no se repita, no sólo hay que comprobar que no exista ya un proyecto con ese código, sino que, si existe, debe ser el proyecto que se está modificando. Si se trata de otro distinto, la validación falla. Por lo demás, es igual al servicio anterior.

#### ManagerProjectDeleteService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEl servicio de borrado es igual al de actualización excepto en dos aspectos: en la validación no se hacen comprobaciones especiales y el método perform debe encontrar todas las asignaciones que incluyan al proyecto que se quiere borrar, eliminarlas y luego eliminar el proyecto.

#### ManagerProjectPublishService.java

De nuevo, el servicio de publicación es igual al de actualización excepto en dos aspectos: los métodos load y validate.

El método load debe establecer el atributo “draftMode” del proyecto a false; esto es lo que diferencia al método publish de un update normal.

El método validate, aparte de validar el atributo “code” igual que en la actualización, debe validar que el proyecto no tenga errores fatales, que tenga al menos una historia de usuario y que todas estas estén publicadas. Si alguna comprobación falla, se pone el atributo “draftMode” a true. Esto se hace para que el formulario de modificación del proyecto permita seguir editando sus campos tras el error para poder subsanarlo. De lo contrario, los campos del formulario se pondrán en modo de sólo lectura y el usuario tendrá que abandonar la página y volver a entrar en ella para arreglar su error.

Con estas validaciones, nos aseguramos de que no se publique ningún proyecto con errores fatales, sin historias de usuario o sin historias de usuario publicadas. Sin embargo, no se evita que se borren las asignaciones de un proyecto una vez publicado. Para ello, debemos examinar el servicio de borrado de asignaciones.

#### ManagerAssignationDeleteService.java

Este servicio se encarga de operaciones de borrado de usuarios tipo Manager sobre objetos tipo Assignation. El método que nos interesa es validate.

Texto

Descripción generada automáticamenteEste método obtiene todas las asignaciones que contienen al mismo proyecto que la que queremos borrar. Primero comprueba que el proyecto está publicado. Si no lo está, se procede con el borrado. En caso contrario, si la colección de asignaciones sólo contiene a una, que sería la que estamos evaluando, entonces no se permite el borrado, puesto que si se borrase se dejaría al proyecto publicado sin historias de usuario, lo que no debe pasar de acuerdo a la definición del requisito.

En cuanto a las vistas de la aplicación, los fragmentos creados han sido los siguientes:

#### project/list.jsp

Texto

Descripción generada automáticamente

Este fragmento define una tabla con una columna por cada uno de los atributos especificados en el método unbind del servicio de listado. Cada fila de la tabla corresponderá con un proyecto distinto y, si se pulsa en ella, conducirá al formulario que muestra los detalles del proyecto. Se añade también el payload a la tabla. Al final de la página a la izquierda, aparecerá un botón que conducirá al formulario de creación de un proyecto.

#### project/form.jsp

Texto, Carta

Descripción generada automáticamente

Este fragmento servirá para las funcionalidades de show, update, publish, create y delete. Hay un campo por cada atributo modificable del proyecto. Estos campos se podrán modificar si el atributo “draftMode” que le envía el método unbind del servicio correspondiente tiene un valor verdadero. Además, si “draftMode” es verdadero y no se trata de una petición de creación, aparecerán unos botones para actualizar, borrar o publicar el proyecto.

Independientemente de si el proyecto está publicado o no, si la petición no es de creación también se mostrará un botón para listar las historias de usuario relacionadas con el proyecto. Esta funcionalidad depende de un servicio de UserStory, por lo que no se va a profundizar en ella.

Por último, si lo que se pretende es crear un proyecto nuevo, los campos del formulario se podrán modificar y bajo éste únicamente aparecerá un botón para enviar el formulario y crear el proyecto.

Los mensajes en español e inglés de estas dos vistas están definidos en sus correspondientes archivos.

### Validación por el profesor

Tras la entrega final de la primera convocatoria, este requisito no fue validado por el profesor por el siguiente motivo: “Corrección: Este requisito no puede darse por válido. Puedo tener un proyecto sin historias de usuario asociadas en estado publicado. Para reproducir el error: popular con sample data > hacer login con administrator y convertirte en manager > crear un proyecto > crear una historia de usuario y publicarla > asociar la historia de usuario al proyecto > publicar el proyecto > listar las asignaciones > borrar la asignación realizada. Obtengo como resultado el error mencionado.”

Para corregir el requisito, se implementó la validación comentada en el servicio de borrado de asignaciones.

## Requisito 218

### Descripción

Implementar las operaciones que realizan los managers sobre sus dashboards: mostrarlos.

### Análisis y decisiones

Para implementar este requisito, se debe tener en cuenta la implementación del formulario ManagerDashboard.java que se hizo en los primeros entregables de la primera convocatoria.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteLa estructura es similar a la del requisito anterior. Se ha creado un repositorio, un servicio de show y un controlador. Los tres se encuentran en el paquete “acme/features/manager/dashboard”.

Las consultas que se definen en el repositorio se hacen sobre entidades publicadas y son las siguientes:

Texto, Carta

Descripción generada automáticamente

En el servicio, el método authorise comprueba que el usuario tenga el rol manager. El método unbind incluye a todos los atributos definidos en el formulario del dashboard. Es el método load el que es más intricado:

Texto

Descripción generada automáticamenteDebe hacer una llamada al repositorio para cada una de las estadísticas que se va a mostrar. Vemos que estas incluyen contar cuántas historias de usuario hay con cada tipo de prioridad y una serie de estadísticas sobre el coste estimado de las historias de usuario y el coste de los proyectos. A continuación, crea un objeto de tipo ManagerDashboard vacío y rellena sus atributos con los datos que acaba de obtener del repositorio.

Para las vistas, se ha creado un fragmento “form.jsp” que contiene una tabla para las cuentas de UserStory, otra para las estadísticas de UserStory y otra para las estadísticas de Project. Las tres tabñas tienen dos columnas, una para la medida estadística que corresponda y otra para el valor de esta. Cada fila de la tabla es una medida estadística.

Para los atributos de tipo media, desviación típica, mínimo y máximo, cuando estos atributos tienen un valor nulo en lugar de numérico, se muestra el mensaje “No data” en la columna correspondiente de la tabla:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Los mensajes en español e inglés de esta vista están definidos en sus correspondientes archivos.

### Validación por el profesor

Tras la entrega final de la primera convocatoria, este requisito no fue validado por el profesor por el siguiente motivo: “Corrección: Este requisito no puede dares por válido, se realizan estadísticas sobre objetos que están en modo borrador. Para reproducir el error: popular con sample data > hacer login con administrator y convertirte en manager > crear una historia de usuario > acceder al dashboard. Obtengo como resultado el error mencionado.”

Para corregir el requisito, se modificaron todas las consultas del repositorio para añadir la sentencia “AND x.draftMode = false”, como se puede observar en la imagen que ilustra el contenido del repositorio.

# Conclusiones

Tras este análisis, concluimos que la implementación de todos los requisitos no ha supuesto mayor complicación, dado que la corrección del profesor indicaba con claridad cuál había sido el error en la primera entrega. Analizando el resto de requisitos del estudiante 1 implementados, no se han encontrado otros fallos que corregir, por lo que los dos requisitos de código sin validar de la primera convocatoria han sido los únicos a abarcar en esta segunda convocatoria.

Consideramos exitoso el desarrollo de las tareas individuales de este entregable.

# Bibliografía

Transparencias de DP2 – L03, S01 – Display features.

Transparencias de DP2 – L03, S02 – Edition features.