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](#_Toc170817864)

[Historial de Versiones 4](#_Toc170817865)

[Introducción 5](#_Toc170817866)

[Registro de análisis 6](#_Toc170817867)

[Requisito 074 6](#_Toc170817868)

[Descripción 6](#_Toc170817869)

[Análisis y decisiones 6](#_Toc170817870)

[Validación por el profesor 10](#_Toc170817871)

[Requisito 075 11](#_Toc170817872)

[Descripción 11](#_Toc170817873)

[Análisis y decisiones 11](#_Toc170817874)

[Validación por el profesor 15](#_Toc170817875)

[Funcionalidades de Assignation 15](#_Toc170817876)

[Descripción 15](#_Toc170817877)

[Análisis y decisiones 15](#_Toc170817878)

[Validación por el profesor 18](#_Toc170817879)

[Requisito 076 19](#_Toc170817880)

[Descripción 19](#_Toc170817881)

[Análisis y decisiones 19](#_Toc170817882)

[Validación por el profesor 20](#_Toc170817883)

[Requisitos 093 y 094 21](#_Toc170817884)

[Descripción 21](#_Toc170817885)

[Análisis y decisiones 21](#_Toc170817886)

[Validación por el profesor 22](#_Toc170817887)

[Requisito 095 23](#_Toc170817888)

[Descripción 23](#_Toc170817889)

[Análisis y decisiones 23](#_Toc170817890)

[Validación por el profesor 24](#_Toc170817891)

[Conclusiones 25](#_Toc170817892)

[Bibliografía 26](#_Toc170817893)

# Resumen del Informe

El objetivo de este informe es proporcionar detalles sobre el análisis de todos los tipos de requisitos especificados en esta tercera 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. | 02/07/2024 | Mª Salud Carrera Talaverón |

# Introducción

En esta tercera entrega cada estudiante debe implementar obligatoriamente una serie de operaciones que puede realizar un rol sobre las entidades que gestiona el mismo. Entre las tareas suplementarias se encuentran las operaciones que pueden realizar otros usuarios y personas no autenticadas sobre las entidades del sistema, además de la producción de tres nuevos informes.

En este informe sólo se describirán aquellas tareas que requieran modificaciones en el código del estudiante 1. Para analizar la producción de los informes individuales se deberá consultar cada uno de ellos por separado.

# Registro de análisis

Estos requisitos se implementaron para el tercer entregable de la convocatoria de junio. Sin embargo, no se describirá su implementación original, sino la versión final tras las modificaciones en entregables posteriores, incluyendo la segunda convocatoria.

## Requisito 074

### 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. Estas funcionalidades se discutirán tras la descripción de las funcionalidades de UserStory.

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, dejando a este sin historias de usuario. Para ello, deberemos examinar el servicio de borrado de asignaciones, que se describe posteriormente como ya se ha mencionado.

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

#### project/list.jsp



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 se profundizará en ella más adelante.

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 una validación en el servicio de borrado de asignaciones, que se verá posteriormente en este informe.

## Requisito 075

### Descripción

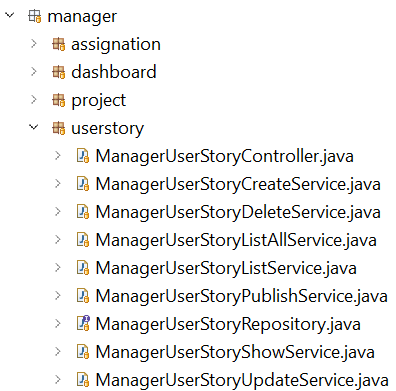
Implementar las operaciones de los managers sobre las historias de usuario (HU, en adelante), que incluyen:

* Listar las HUs de sus proyectos.
* Mostrar los detalles de HUs.
* Crear y publicar una HU.
* Actualizar o borrar una HU siempre y cuando esta no esté publicada.

Aparte de estas operaciones, se ha añadido la operación de listar todas sus HUs y no sólo las de un proyecto concreto.

### 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. Estos tres tipos de archivos se construyen igual que los ya mencionados de Project, por lo que no se va a volver a explicar su estructura. Además, se han definido las vistas correspondientes y se han modificado los fragmentos necesarios para facilitar la navegación por estas funcionalidades.

En esta imagen podemos ver los archivos creados en el paquete “acme/features/manager/userstory”. Las consultas que definidas en el repositorio son: encontrar una HU por su ID, encontrar todas las HUs 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 HU, encontrar todas las HUs de un proyecto a través de las asignaciones que los unan por el ID del proyecto y encontrar un proyecto por su ID.

El controlador implementa un constructor en el que especifica qué operación realiza cada servicio de la siguiente manera:

Cada servicio corresponde a un tipo de operación de lectura o escritura. En el caso del servicio de publicar una HU, 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. Lo mismo ocurre con el servicio de listar todas las HUs, que se trata de una operación especial de listado.

Se van a describir a continuación los servicios implementados.

#### ManagerUserStoryListAllService.java

El método authorise controla que para listar sus HUs un usuario debe tener el rol manager.

El método load obtiene el ID del manager a través de la petición y se obtiene la colección de HUs creados por éste. Además, crea un nuevo parámetro llamado “showCreate” que le indicará a la vista correspondiente si debe mostrar o no el botón de crear una nueva HU. En este caso, sí debe mostrarse.

Por último, el método unbind crea una tupla con todos los atributos de la HU excepto su manager. No es necesario añadir atributos al payload porque todos se mostrarán en pantalla.

#### ManagerUserStoryListService.java

Este servicio es esencialmente igual al de listar todas las HUs. Las diferencias son: el manager debe ser el dueño del proyecto del cual se quieren mostrar las HUs (y este proyecto debe existir); la colección de HUs se obtiene a partir del ID del proyecto y no del manager, el cual también se obtiene de la petición; no se mostrará el botón de crear una nueva HU.

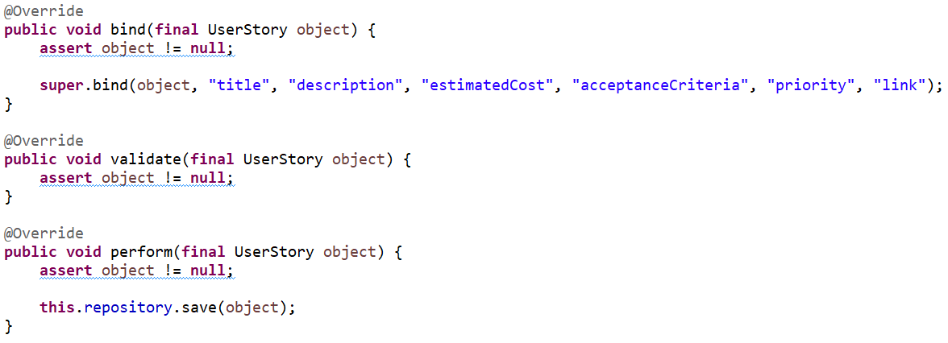
#### ManagerUserStoryShowService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEste servicio es muy similar a los anteriores: sólo se puede mostrar la información de una HU si se es el manager dueño del mismo; se cargan los datos mediante el ID de la HU, que se extrae de la petición; se incluyen todos los parámetros de la entidad normalmente, excepto el manager que tenga asociado. Para incluir los posibles valores del atributo priority, se utiliza la clase SelectChoices de Acme Framework.

#### ManagerUserStoryCreateService.java

En este caso, sólo se puede acceder al formulario de creación de una HU si se es un usuario con rol manager. El método de carga creará una nueva HU vacía, que definirá en modo borrador y a la que asignará como manager el usuario autenticado. El método unbind será igual que el de show.

En cuanto al resto de métodos: el método bind aceptará valores para todos los atributos excepto el manager y el modo borrador.

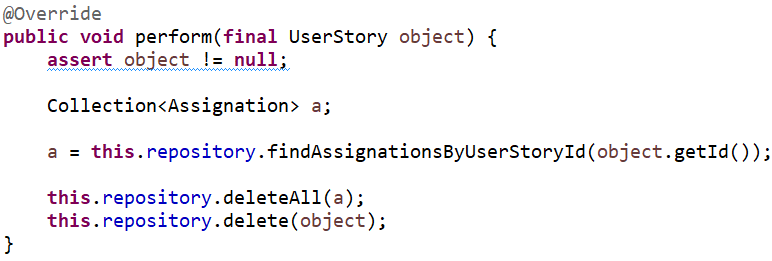
El método validate, aparte de comprobar las restricciones simples definidas en la clase UserStory.java (como que el coste debe valer como mínimo 1), no define ninguna restricción compleja puesto que esta entidad no requiere de ninguna.

Por último, el método perform guarda en la base de datos el objeto de historia de usuario recién creado.

#### ManagerUserStoryUpdateService.java

Este servicio es bastante similar al de creación. En este caso, sólo se puede actualizar una HU si se es el manager que la creó. Además, esta debe estar en modo borrador. La HU que se carga se busca en la base de datos a partir del ID extraído de la petición. Por lo demás, es igual al servicio anterior.

#### ManagerUserStoryDeleteService.java

El servicio de borrado es igual al de actualización excepto en un aspecto: el método perform debe encontrar todas las asignaciones que incluyan a la HU que se quiere borrar, eliminarlas y luego eliminar la HU.

#### ManagerUserStoryPublishService.java

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” de la HU a false; esto es lo que diferencia al método publish de un update normal.

El método validate no hace ninguna comprobación especial. Sin embargo, si algún campo del formulario no cumple las restricciones simples definidas en la clase UserStory.java, se pone el atributo “draftMode” a true. Esto se hace para que el formulario de modificación de la HU 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.

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

#### user-story/list.jsp



Este fragmento define una tabla con una columna por cada uno de los atributos especificados en el método unbind de los servicios de listado. Cada fila de la tabla corresponderá con una historia de usuario distinta y, si se pulsa en ella, conducirá al formulario que muestra los detalles de la HU. Al final de la página a la izquierda, si el parámetro showCreate es verdadero (ya se ha indicado cuándo ocurre esto y cuándo no), aparecerá un botón que conducirá al formulario de creación de una HU.

#### user-story/form.jsp



Este fragmento servirá para las funcionalidades de show, update, publish, create y delete. Hay un campo por cada atributo modificable de la HU. 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 la HU.

Si lo que se pretende es crear una HU nueva, los campos del formulario se podrán modificar y bajo éste únicamente aparecerá un botón para enviar el formulario y crear la HU.

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 fue validado por el profesor; o, al menos, no se indicó lo contrario.

## Funcionalidades de Assignation

### Descripción

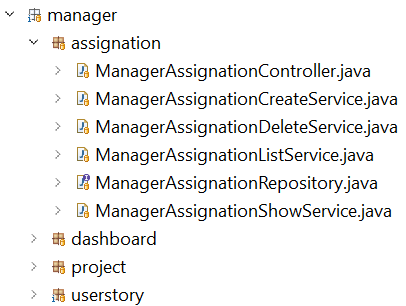
Este no es un requisito pedido por el cliente explícitamente; sin embargo, debe implementarse para el correcto funcionamiento de la aplicación. Las operaciones de managers sobre asignaciones son:

* Listar sus asignaciones y mostrar sus detalles.
* Crear una asignación nueva. A un proyecto publicado sólo se le pueden asignar HUs publicadas y no se puede crear la misma asignación entre un proyecto y una HU más de una vez. La asignación debe relacionar proyectos e historias de usuario del mismo manager.
* Eliminar asignaciones. No se puede borrar la última asignación de un proyecto publicado.

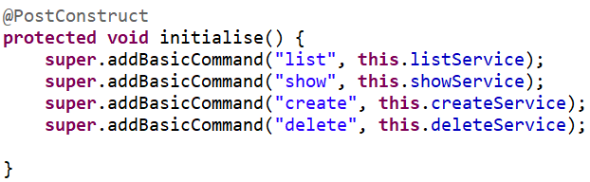
Se ha decidido no implementar la operación de actualización ya que, si se desea cambiar una asignación por otra, se puede en su lugar crear la nueva asignación que se desee y borrar la anterior.

### 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. Estos tres tipos de archivos se construyen igual que los ya mencionados de Project y UserStory, por lo que no se va a volver a explicar su estructura. Además, se han definido las vistas correspondientes y se han modificado los fragmentos necesarios para facilitar la navegación por estas funcionalidades.

En esta imagen podemos ver los archivos creados en el paquete “acme/features/manager/assignation”. Las consultas que definidas en el repositorio son: encontrar una asignación por su ID, encontrar todas las asignaciones por el ID del manager que creó las entidades que une, encontrar un manager por su ID, encontrar todas las asignaciones por el ID de su proyecto, encontrar todas las asignaciones por el ID de su HU, encontrar una asignación por el ID de su proyecto y el de su HU, encontrar todas las HUs por el ID de su manager, encontrar todos los proyectos por el ID de su manager, encontrar el proyecto de una asignación por el ID de la misma, encontrar la HU de una asignación por el ID de la misma y encontrar el manager que creó las entidades de una asignación por el ID de la misma.

El controlador implementa un constructor en el que especifica qué operación realiza cada servicio de la siguiente manera:

Cada servicio corresponde a un tipo de operación de lectura o escritura. Todas las operaciones se indican normalmente, ya que no existe ninguna “especial”.

Se van a describir a continuación los servicios implementados.

#### ManagerAssignationListService.java

El método authorise controla que para listar sus asignaciones un usuario debe tener el rol manager.

El método load obtiene el ID del manager a través de la petición y se obtiene la colección de asignaciones creadas por éste.

Por último, el método unbind crea una tupla con todos los atributos de la HU excepto su manager. Como los dos atributos de la asignación son relaciones, una con un objeto Project y otra con un objeto UserStory, estos dos objetos no se van a mostrar enteros en pantalla; sólo se mostrará el código del proyecto y el título de la HU. No es necesario añadir atributos al payload.

#### ManagerAssignationShowService.java

Este servicio es muy similar a los anteriores: sólo se puede mostrar la información de una asignación si se es el manager dueño de la misma y se cargan los datos mediante el ID de la asignación, que se extrae de la petición.

Texto

Descripción generada automáticamenteEn el método unbind, para incluir el proyecto primero se debe buscar la colección de proyectos creados por el manager que está haciendo la petición. Luego, se debe utilizar la clase SelectChoices para indicar que cada proyecto, a través de su código, será una posible opción para el campo Project. Luego, se indicará cuál de todas las opciones es la que se ha elegido. Para incluir la historia de usuario se sigue el mismo procedimiento.

#### ManagerAssignationCreateService.java

En este caso, sólo se puede acceder al formulario de creación de una asignación si se es un usuario con rol manager. El método de carga creará una nueva asignación vacía. El método unbind será igual que el de show.

En cuanto al resto de métodos: el método bind aceptará valores para todos los atributos.

El método validate debe comprobar que la asignación que se quiere crear no exista ya y que, si el proyecto está publicado, también lo esté la historia de usuario.

Por último, el método perform guarda en la base de datos el objeto de asignación recién creado.

#### ManagerAssignationDeleteService.java

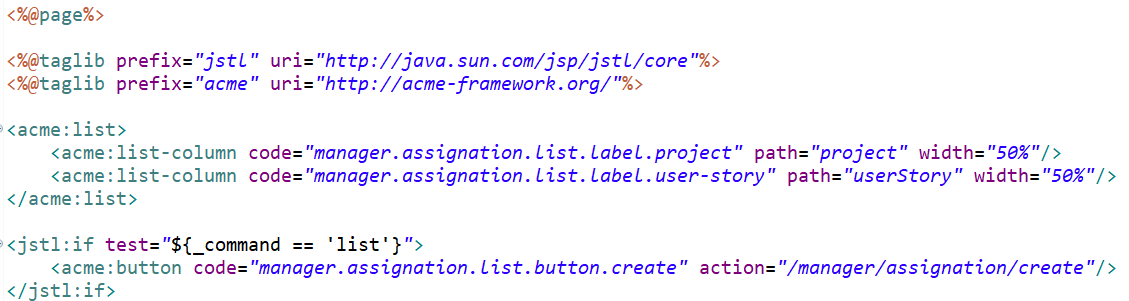
El servicio de borrado es bastante similar al de creación. En este caso, sólo se puede borrar una asignación si se es el manager que la creó (el que creó las entidades que une). La asignación que se carga se busca en la base de datos a partir del ID extraído de la petición. El método perform debe simplemente eliminar el objeto de asignación especificado.

El más complejo es el método 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:

#### assignation/list.jsp



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 una asignación distinta y, si se pulsa en ella, conducirá al formulario que muestra los detalles de la misma. Al final de la página a la izquierda, aparecerá un botón que conducirá al formulario de creación de una asignación.

#### assignation/form.jsp



Este fragmento servirá para las funcionalidades de show, create y delete. Hay un campo de tipo desplegable por cada atributo de la asignación que, salvo en el caso de la operación de creación, no se podrán modificar.

Si lo que se pretende es crear una asignación nueva, los campos del formulario sí se podrán modificar para seleccionar el proyecto y la HU deseados y bajo éste únicamente aparecerá un botón para enviar el formulario y crear la asignación.

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

### Validación por el profesor

Aunque no se trate de un requisito explícito, se entiende que este está validado por el profesor si se da el visto bueno a los requisitos 074 y 075. Dado que tras la primera convocatoria no se pudo validar la implementación de operaciones de managers sobre proyectos, este requisito tuvo que ser modificado. Se implementó la corrección pedida por el profesor, así que se da por terminado el desarrollo de las funcionalidades correspondientes a Assignation.

## Requisito 076

### 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.

## Requisitos 093 y 094

### Descripción

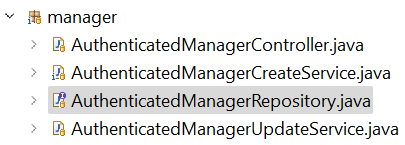
Estos requisitos se van a analizar juntos ya que comparten controlador y repositorio.

093: Implementar las operaciones de usuarios anónimos sobre cuentas de usuario: registrarse en el sistema y convertirse en manager.

094: Implementar la operación de actualización del perfil de un manager.

### 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.

En esta imagen podemos ver los archivos creados en el paquete “acme/features/authenticated/manager”. Las consultas que definidas en el repositorio son: encontrar un manager por el ID de su cuenta de usuario y encontrar una cuenta de usuario por su ID.

El controlador implementa un constructor en el que especifica qué operación realiza cada servicio análogamente a cómo ya se ha descrito para otros controladores. No existe ninguna operación especial.

Se van a describir a continuación los servicios implementados.

#### Texto Descripción generada automáticamenteAuthenticatedManagerCreateService.java

El método authorise controla que para para convertirse en manager el usuario autenticado no puede tener ya el rol de manager.

El método load obtiene el ID de la cuenta de usuario a través de la petición y obtiene dicha cuenta. A continuación, crea un nuevo objeto de tipo Manager y le asigna la cuenta de usuario obtenida.

El método bind aceptará valores para todos los atributos del rol manager.

El método validate no comprobará restricciones complejas, sino que sólo habrá que validar las restricciones simples definidas en la clase Manager.java.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEl método perform guardará el objeto recién creado con la información indicada en los campos del formulario.

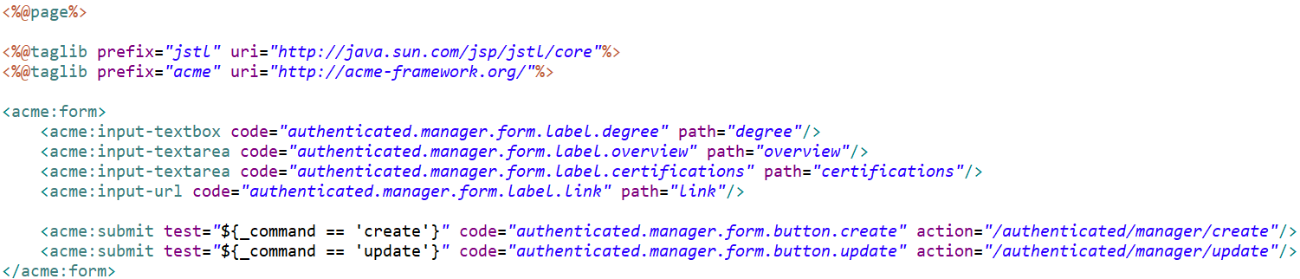
El método unbind crea una tupla con todos los atributos del rol manager.

Por último, vemos el método “onSuccess()”. Este método hace que se actualice la información de la cuenta de usuario del usuario autenticado haciendo uso del helper PrincipalHelper. Se hace necesario el uso de este método para que no se generen conflictos, debido a que el objeto que se está actualizando es el que se está utilizando para ejecutar la operación y podrían generarse inconsistencias.

#### AuthenticatedManagerUpdateService.java

Este servicio es esencialmente igual al de creación. La única diferencia es que para poder actualizar su perfil de manager el usuario autenticado debe tener el rol manager, y que la información que se carga se hace buscando el manager mediante el ID de la cuenta de usuario extraído de la petición.

En cuanto a las vistas de la aplicación, sólo se ha creado el fragmento form.jsp:



Este fragmento servirá para las funcionalidades de update y create. Hay un campo por cada atributo modificable del rol. Estos campos se podrán modificar, en cualquier caso.

Bajo el formulario aparecerá un botón para crear o actualizar, dependiendo del tipo de operación que se especifique en la petición.

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, estos requisitos fueron validados por el profesor; o, al menos, no se indicó lo contrario.

## Requisito 095

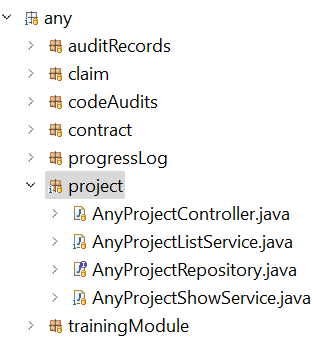
### Descripción

Implementar las operaciones de cualquier usuario sobre los proyectos, que incluyen:

* Listar los proyectos publicados en el sistema.
* Mostrar los detalles de estos proyectos publicados (excepto sus HUs).

### 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.

En esta imagen podemos ver los archivos creados en el paquete “acme/features/any/project”. Las consultas que se definen en el repositorio son: encontrar un proyecto por su ID y encontrar todos los proyectos publicados.

El controlador implementa un constructor en el que especifica qué operación realiza cada servicio análogamente a cómo ya se ha descrito para otros controladores. No existe ninguna operación especial.

Se van a describir a continuación los servicios implementados.

#### AnyProjectListService.java

Este servicio es prácticamente igual a ManagerProjectListService.java, con ciertos matices.

El método authorise controla si el usuario que intenta acceder a la URL de listado de proyectos publicados tiene o no el rol any (es decir, cualquier usuario puede acceder).

El método load, en lugar de buscar todos los proyectos de un manager, busca todos los proyectos publicados.

Por último, el método unbind crea una tupla con los parámetros código, título y descripción. Se han añadido los atributos restantes (excepto el manager) al payload para poder utilizarlos en la barra de búsqueda o filtrado de la página.

#### AnyProjectShowService.java

De nuevo, este servicio es prácticamente igual al análogo para managers. La única diferencia esta vez es que el método authorise, aparte de comprobar que el usuario tenga el rol any, debe comprobar que el proyecto no esté en modo borrador (es decir, que esté publicado).

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

#### project/list.jsp

Este fragmento es igual al análogo para managers, la única diferencia siendo que no aparecerá un botón para crear un proyecto nuevo.

#### project/form.jsp

De nuevo, este fragmento es igual al análogo para managers. La diferencia de nuevo es que no aparecerán los botones que sí aparecían bajo el formulario para managers, y que esta vez los campos del formulario nunca podrán ser modificados.

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, estos requisitos fueron validados por el profesor; o, al menos, no se indicó lo contrario.

# Conclusiones

En resumen, la implementación de los requisitos para esta entrega se ha completado de manera exitosa. La integración de las funcionalidades necesarias para la gestión de proyectos y asignaciones, así como la implementación de operaciones para usuarios anónimos y managers, se ha llevado a cabo eficazmente. La validación por parte del profesor ha confirmado la corrección de la mayoría de los requisitos, destacando únicamente pequeñas modificaciones necesarias en algunos casos. En general, consideramos que el desarrollo de las tareas individuales ha sido exitoso y ha contribuido significativamente al avance del proyecto.

# Bibliografía

Transparencias de DP2 – L03, S01 – Display features.

Transparencias de DP2 – L03, S02 – Edition features.

Transparencias de DP2 – L03, S03 – Advanced topics.