Analysis report

Diseño y pruebas II

Sprint 4

Versión 3.0

Fecha 25/05/2024

Preparado por:
Miguel Hernández Sánchez (C1.028)

Repositorio:
DP2-c1-028/Acme-SF-D04 (github.com)

Índice

D02

1. Entidad Project

A **project** aggregates several **user stories** elicited by the same **manager**. The system must store the following data about them: a code (pattern “[A-Z]{3}-[0-9]{4}”, not blank, unique), a **title** (not blank, shorter than 76 characters), an **abstract** (not blank, shorter than 101 characters), an **indication** on whether it has fatal errors, e.g., panics, a **cost** (positive or nought), and an **optional link** with further information. Projects containing fatal errors must be rejected by the system.

Para llevar a cabo este requisito, es necesario crear una clase **Project** dentro del paquete *entities.projects*. Esta clase heredará de **AbstractEntity** y tendrá un atributo ***serialVersion***, además de todos los especificados en el enunciado con sus respectivas restricciones. En cuanto a las relaciones, tiene una relación de tipo **ManyToOne** hacia **Manager**, ya que un **Manager** puede crear varios proyectos.

2. Entidad User Story

A **user story** is a document that a **manager** uses to represent the smallest unit of work in a project. The system must store the following data about them: a **title** (not blank, shorter than 76 characters), a **description** (not blank, shorter than 101 characters), an **estimated cost** (in hours, positive, not nought), the **acceptance criteria** (not blank, shorter than 101 characters), a priority (“Must”, “Should”, “Could”, or “Won’t”), and an **optional link** with further information.

Para llevar a cabo este requisito, es necesario crear una clase **UserStory** dentro del paquete *acme.entities.userStories*. Esta clase heredará de **AbstractEntity** y tendrá un atributo ***serialVersion***, además de todos los especificados en el enunciado con sus respectivas restricciones. Para el atributo priority, se crea un enumerado **Priority** en el que contendrá los valores posibles. En cuanto a las relaciones, tiene una relación de tipo **ManyToOne** hacia **Manager**, ya que un **Manager** puede crear varias user stories. Además es necesario hacer un **ManyToMany** con **Project** pero para ahorrarnos problemas con la tabla oculta que se crea automáticamente, se crea manualmente en la entidad **UserStoryProject** que contendrá dos relaciones **ManyToOne**, una hacia **Project** y otra hacia **UserStory**.

3. Manager Dashboard

The system must handle **manager** dashboards with the following data: total number of “must”, “should”, “could”, and “won’t” **user stories**; average, deviation, minimum, and maximum **estimated cost** of the **user stories**; average, deviation, minimum, and maximum **cost** of the **projects**.

Para llevar a cabo este requisito, es necesario crear una clase **ManagerDashboard** dentro del paquete *acme.forms*. Esta clase heredará de **AbstractForm** y tendrá un atributo ***serialVersion***, además de todos los especificados en el enunciado con sus respectivas restricciones.

En el dashboard se puede ver las estadísticas asociadas al dinero convertidas en las divisas que el gestor usa en sus proyectos, es decir, si un gestor usa como divisas el euro y el dólar, le aparecerán las estadísticas asociadas al dinero convertidas en euros y en dólares. Por ello mismo, las estadísticas monetarias se guardan en un mapa el cual la clave es la unidad monetaria y el valor son las estadísticas monetarias convertidas en esa divisa.

4. Testing Requirements

Produce assorted sample data to test your application informally. The data must include two **manager** accounts with credentials "**manager1**/**manager1**" and "**manager2**/**manager2**".

Para realizar este requisito, es necesario crear los csv a las respectivas entidades creadas anteriormente en la carpeta *src/main/webapp/WEB-INF/resources/sample-data.* Por cada csv se ha de crear columnas correspondientes a los atributos de la entidad en cuestión y en cuanto a las filas, cada una corresponde a un caso de prueba para un atributo, escogiendo, dentro del rango de posibles valores aceptables, los extremos, un valor cercano a los extremos y un valor medio.

5. UML

Para realizar este requisito, es necesario fijarse en los atributos de las entidades y las relaciones que hay entre estas, además de las reglas de negocio asociadas.

6. Manager Role

There is a new project-specific role called **manager**, which has the following profile data: **degree** (not blank, shorter than 76 characters), an **overview** (not blank, shorter than 101 characters), list of **certifications** (not blank, shorter than 101 characters), and an **optional link** with further information.

Para llevar a cabo este requisito, es necesario crear una clase **Manager** dentro del paquete *acme.roles.* Esta clase heredará de **AbstractRole** y tendrá un atributo ***serialVersion***, además de todos los especificados en el enunciado con sus respectivas restricciones. Esta entidad ya ha sido relacionada con las demás y por lo tanto no hay que realizar relaciones en su clase.

D03

1. Operations by managers on projects

- List the **projects** that they have created:

Para llevar a cabo este requisito fue necesario crear un paquete que contendrá todo las features de la entidad *Project* con el rol *Manager* llamado *acme.features.manager.project*. Dentro de este paquete creé un controlador (*ManagerProjectController.java*) donde se especificará qué servicio será el encargado del listado de los proyectos. También creé un servicio de listado junto a un repositorio para acceder a la base de datos (*ManagerProjectListService.java* y *ManagerProjectRepository.java*).

Luego de crear las funciones necesarias dentro del Repositorio para llamar a los proyectos de un manager en específico y las funciones que requieren del servicio para la correcta integración de AcmeFramework comprobé manualmente que no daba un error 500.

- Show the details of their **projects**.

Para llevar a cabo este requisito fue necesario crear en el paquete *acme.features.manager.project* el servicio *ManagerProjectShowService*. Se crearon las funciones necesarias en el repositorio para obtener un proyecto dados un id de un proyecto. En el *authorise* del servicio se comprobó si pertenece al manager que está intentando acceder para evitar GET Hacking. También se añadió el servicio al controlador. Más tardé comprobé manualmente que no daba un error 500.

- Create, update, or delete their **projects**. **Projects** can be updated or deleted as long as they have not been published. For a **project** to be published, it must have at least one **user story**, and all its **user stories** must have been published. Moreover, it must not have any fatal errors.

Para llevar a cabo este requisito creé varios archivos en el paquete *acme.features.manager.project.* Estos archivos fueron *ManagerProjectCreateService, ManagerProjectUpdateService* y *ManagerProjectPublishService*. Se crearon teniendo en cuenta el POST Hacking en el authorise y se añadieron al controlador siendo el servicio de *publish* un *customCommand* que hace la función de un *update*. Más tarde comprobé manualmente que no existía ningún error 500.

Cuando un proyecto es borrado, se elimina la relación de la tabla intermedia con user story.

2. Operations by **managers** on **user stories:**

- List the **user stories** in their **projects**.

Para llevar a cabo este requisito, creé un paquete *acme.features.manager.userStory* donde creé un repositorio (*ManagerUserStoryRepository.java)*, un controlador *(ManagerUserStoryController.java)* y el servicio *(ManagerUserStoryListService.java)*. En ellos se realizaron las funciones pertinentes para su correcto funcionamiento y se comprobó que funcionasen correctamente.

Para realizar las operaciones de las User Stories, se pueden acceder a sus features desde el menú de navegación para listar tus user stories y a través de un proyecto en concreto para listar las user stories en relación con un proyecto.

Para saber si estas listando las User Stories dentro de un proyecto o desde el menú de navegación añadí un título que pone explícitamente dónde te encuentras.

- Show the details of their **user stories**.

Para llevar a cabo este requisito, creé un archivo en el paquete *acme.features.manager.userStory* para realizar el servicio (*ManagerUserStoryShowService.java)*. Realicé las funciones pertinentes en el repositorio y en el servicio para evitar el GET Hacking y lo añadí al controlador.

- Create and publish a **user story**.

Para llevar a cabo este requisito, creé los archivos en el paquete *acme.features.manager.userStory* respectivos (*ManagerUserStoryCreateService.java* y *ManagerUserStoryUpdateService.java*). Realicé las funciones pertinentes en el repositorio y en el servicio para evitar el GET y POST hacking en el caso del *create* y los añadí en el controlador, siendo *publish* un *customCommand*.

Si se accede al servicio del *create* desde un listado de user stories con relación a un proyecto, en el *create* se verá reflejado en la url el proyecto al que está relacionado para que cuando se cree esta nueva user story se creé automáticamente en relación a este proyecto.

- Update or delete a **user story** as long as it is not published.

Para llevar a cabo este requisito, creé los archivos en el paquete *acme.features.manager.userStory* respectivos (*ManagerUserStoryUpdateService.java* y *MangerUserStoryDeleteService.java*). Realicé las funciones pertinentes en el repositorio y en el servicio para evitar POST Hacking y se comprobó su correcto funcionamiento.

Cuando una user story es borrada, se elimina la relación en la tabla intermedia con su proyecto.

3. Operaciones con la tabla intermedia

Se crearon las features pertinentes para poder interactuar directamente con la tabla intermedia para crear nuevas relaciones entre los proyectos y las user stories de manera que si a un proyecto se le desea que tenga una user story ya existente sea fácil su adición.

4. Operations by **managers** on **manager** dashboards:

Para la realización de este requisito, se creó una vista siguiendo como ejemplo el dashboard del proyecto Starter Acme Jobs. Posteriormente para poder mostrar las estadísticas monetarias en cada divisa se realizó un *foreach* de JSTL que iterase el mapa el cual contiene los datos monetarios.

5. Operations by anonymous principals on user accounts:

Para la realización de este requisito, se modificaron las vistas del menú para que apareciese la opción de convertirse en gestor a los usuarios registrados. Además, se crearon los archivos del controlador, el repositorio y el servicio de crear con la vista del formulario.

6. Operations by managers on user accounts:

Para la realización de este requisito, se creó el servicio de actualizar y se añadió al controlador.

7. Operations by any principals on projects:

Para llevar a cabo este requisito, creé los archivos en el paquete *acme.features.any.project* el controlador, el servicio de listado y mostrado de detalles y el repositorio teniendo en cuenta de que solo aparezcan proyectos publicados. Realicé las funciones pertinentes en el repositorio y en el servicio para evitar POST Hacking y se comprobó su correcto funcionamiento.

D04

1. Produce a test suite for Requirements #6 y #7

Para la realización de los tests hubo una serie de problemas. Tuve que rehacer varias veces por completo los tests ya que si se cambian el número de líneas de los CSVs la id en la base de datos cambian y el resultado ya no es el mismo resultando en fallos. Una vez pusimos en conjunto los CSVs de todos los estudiantes en cada rama de trabajo hicimos los tests finales.

2. Produce a testing report

Para la realización del testing report, se realizó el coverage y el analyse y se se analizaron en el documento. Para el performance testing, analicé los datos en Excel siguiendo la metodología dada en clase y comparé los resultados de ejecutar mis tests en mi equipo y en el equipo de Gonzalo Navas Remmers (Estudiante 2).

No se vió necesario la adición de índices ya que no mejora lo suficiente los resultados.