WIS REPORT

Grupo E4.01

02/06/2022

Integrantes:

Daniel Díaz Nogales
Luis Miguel Bellido Zancarrón
Diego González Quintanilla
Eloy Moreno Dominguez
José Mª García Quijada
Juan Antonio Mena Vargas

En este documento queda planteada la base de conocimientos adquirida a lo largo de los diferentes entregables con respecto a la arquitectura y el testeo del sistema de información web realizado en la asignatura.

Al comienzo de la asignatura, en el primer entregable, quedaron destacadas diferentes arquitecturas web que ya se habían empleado en grupo en otros proyectos como por ejemplo:

- Spring cómo framework Java con lenguaje de servidor, destacando en este:
    - Acceso/Integración de datos y transacciones con JPA.
    - Arquitectura web con patrón de diseño MVC (modelo - vista - controlador) y uso de servlets.
    - Core del framework y abstracción de información en *beans.*
    - Implementación de sistema para tests unitarios de funcionalidades, vistas e integración.

También, teniendo en cuenta los conocimientos en otras arquitecturas de sistemas de información también expuestos, se desarrolla la nueva base de conocimientos adquirida en este proyecto

Arquitectura WIS de un sistema web con AcmeFramework

Manejo de entidades y modelos de datos

El sistema de información creado con AcmeFramework permite de manera simple crear nuevos componentes o entidades con los que ha operado la aplicación. Las entidades permitían establecer atributos en clases a los que asignar restricciones que eran respetadas en todo el sistema de diferente manera, como por ejemplo con errores de validación.

Esto ha facilitado en la mayoría de los casos establecer las especificaciones de control de datos que los formularios requerían, además de facilitar las relaciones que quedaron establecidas inicialmente en el modelo UML.

Integración de servidores y aplicaciones cliente externas

Dentro del framework para la integración de datos se requiere el uso de una base de datos MySQL externa que queda ejecutada de manera local en el propio equipo. Esta base de datos es autogestionada por el framework, y crea las tablas según las entidades de la aplicación, sus atributos y restricciones.

Con las opciones de lanzamiento de la aplicación, se puede poblar la base de datos con información inicial (initial data), e información de ejemplo (sample data).

Roles de usuario

El framework a través de la extensión de clases permitía crear entidades de tipo usuario que funcionaban como nuevos roles. El uso de los roles garantiza y permite el acceso a diferentes zonas de la aplicación, tener el control y el uso de peticiones de datos, entre otras más funciones dependientes del rol en cada caso.

Controladores, vistas y servicios con autenticación por rol de usuario

A partir de controladores se enlazan las peticiones creadas por las visitas de la aplicación web y se gestionan las respuestas según si su existencia. Si existen, la respuesta y los datos se producen según servicios y quedan expuestas en una vista que renderiza el JSP y sus variables en una respuesta en HTML. Si el path de la vista no existe, devuelve un error de acceso.

Además, el framework permite restringir el uso de los controladores a través de roles de usuario sin necesidad de programarlo específicamente.

Test Suite

Hemos implementado casos de prueba para cada entidad creada con sus respectivos controladores y servicios. Esto nos permite probar cada funcionalidad creada, tanto para casos positivos, es decir, pasándole datos correctos de formulario para que se cree, edite o borre según la funcionalidad, y para casos negativos, obligando al test a que falle para comprobar las diferentes validaciones.

Librerías JAR externas y API

Tras haber integrado validaciones en la aplicación para la comprobación de SPAM en la introducción de información en los formularios, se han exportado las clases como librerías externas y han sido incorporadas como librerías JAR.

Además, se han introducido funcionalidades de microservicios externos para la conversión de las cantidades según las divisas de la clase Money que tenía por defecto el propio AcmeFramework.

Desarrollo en Eclipse y GitHub

Durante el desarrollo del proyecto a lo largo del cuatrimestre hemos seguido una política de trabajo estricto en GitHub llamada GitFlow, que nos ha ayudado a simplificar el trabajo a la hora de resolver conflictos de código al realizar *commits* ya que cada uno ha trabajado por su cuenta cada tarea en una rama diferente. Creando una rama paralela a *master*, llamada *develop* donde han sido puestas en común todas las tareas que cada uno de nosotros ha desarrollado.

Continuando con GitHub cabe destacar el uso de un tablero de trabajo para cada Sprint en el que las tareas a realizar han sido asignadas a cada integrante, teniendo estas diferentes estados; *To do*, *In progress*, *Awaiting* y *Done* según su nivel de maduración y contando con etiquetas, siendo estas; *Documentation*, *Bug*, *Test Case* y *Enhancement* según el tipo de tarea.

A pesar de las numerosas actualizaciones que ha tenido el framework a lo largo de los meses, su sencillo mecanismo de actualización en local nos ha permitido un uso fluido del mismo en el entorno de desarrollo Eclipse. Por lo que no ha sido un impedimento en nuestro caso a la hora de trabajar en el proyecto.

Conclusiones

Para finalizar nos gustaría resaltar y poner cierto énfasis en la independencia que tienen los diversos artefactos de este sistema de información con las funcionalidades a desarrollar ya que, en gran medida, esta característica permitida por el framework ha permitido que la cumplimentación de este proyecto sea un éxito debido a un desarrollo ágil.