Reporte de la Arquitectura WIS

Deliverable-05

Miembros del equipo

Apellidos, Nombre	Correo electrónico	Roles
Gallardo Roco, Raúl	raugalroc@alum.us.es	Manager Desarrollador
Gómez Arias, Nuria	nurgomari@alum.us.es	Desarrollador
Ortiz Guerra, Juan Antonio	juaortgue@alum.us.es	Desarrollador
Páez Páez, Jesús	jespaepae@alum.us.es	Desarrollador Operador
Piñero Calera, Borja	borpincal@alum.us.es	Desarrollador
Ramos Berciano, Pablo	pabramber@alum.us.es	Desarrollador

Repositorio: https://github.com/raugalroc/Acme-Toolkits

Grupo E2.06

Fecha: 28 de Mayo de 2022

Índice

Resumen Ejecutivo

En nuestro grupo E2.06 tenemos conocimientos sobre este tipo de proyectos debido a asignaturas como Diseño y Pruebas I e Introducción a la ingeniería del Software y Sistemas de la Información 1 y 2. Gracias a ello podemos desarrollar sistemas con características parecidas a las que se requieren para este proyecto.

Nuestra intención es completar todas las características de cada Sprint para intentar obtener la máxima calificación posible en la asignatura.

Tabla de Revisión

Versión	Fecha	Descripción
1	28/05/2022	Primera versión

Introducción

En este documento se tratarán los aspectos que el equipo ha aprendido de cómo funciona y cómo está compuesta una arquitectura WIS tras el paso por la asignatura de Diseño y Pruebas II y tras la realización del proyecto Acme Toolkits.

Contenidos

En primer lugar podemos hablar de cómo es el flujo de peticiones en este tipo de arquitecturas. Tras realizar el usuario una petición al sistema esta llega al servidor HTTP que la pasa al servidor de Servlet que determina qué controlador debe de servir dicha petición. Tras esto el controlador apropiado abrirá una transacción para dicha petición gracias al Transaction Manager. Después el controlador entregará la petición al servicio que realizará las acciones oportunas que se hayan definido y que, normalmente, terminarán en peticiones al repositorio de la entidad con la que el usuario quiere trabajar. Estas peticiones las recogerá el repositorio y las enviará al Query Manager, que se encarga de conectarse con la base de datos. Pero antes de conectarse a esta, le pasará la petición al Transaction Manager, para que este trabaje con la Staging Area y no directamente con los datos. Esta forma de trabajar es una de las claves de las transacciones ya que si algo falla en medio de todas las peticiones, no se habrán modificado los datos reales de la base de datos algo que mejorará la fiabilidad del sistema. Tras terminar con las peticiones, el controlador decidirá hacer commit de la transacción, por lo que los cambios realizados pasarán a la base de datos para persistir. Tras obtener el resultado de la petición del usuario está pasará al renderizador de vistas que creará de forma dinámica la vista que se mostrará en pantalla.

Además, para cada tipo de petición tenemos distintos workflows de trabajo en cuanto al desarrollo de la misma y su transacción asociada. En primer lugar tenemos las peticiones

GET. Estas, al llegar al controlador se crea su transacción, tras esto se comprueba que el usuario que ha hecho la petición, por sus permisos, puede hacerla, tras esto se prepara la entidad que debe ser devuelta al usuario, se embebe en la vista y se cierra la transacción haciéndole commit. Por otra parte las peticiones POST empiezan de la misma forma, creándose su transacción, comprobándose los permisos del usuario y preparando la entidad para ser enviada al usuario. Tras esto a la entidad se le inicializan sus atributos con los que ha proporcionado el usuario y que viajan en el modelo de la petición. Después se comprueban que los datos de la entidad sean los correctos con restricciones personalizables para validaciones algo más complejas que las que puedan surgir en la entidad. Tras esto se pasan los datos de la entidad al modelo y se comprueban si ha habido errores. Si los hay a la transacción se le hace un rollback, es decir, se cierra sin persistir los cambios y si todo ha ido correctamente, se hace commit a toda la transacción persistiendo los cambios y devolviendo la respuesta al usuario.

En nuestro caso, las tecnologías que usamos en cada caso son: para el servidor HTTP usamos Coyote, para el servidor de Servlet Catalina, para el renderizador de vistas Jasper y ya en la propia aplicación usamos Acme-Framework que está soportado por Spring Boot, incluyendo tecnologías como Java, JPA o JSP.

Conclusiones

Como hemos podido comprobar, hemos aprendido las principales características de una arquitectura WIS, su funcionamiento general, sus elementos más importantes y las tecnologías que pueden ser usadas en ellos y las interacciones que pueden darse durante el proceso.

Bibliografía

- The starter project (Theory)
- Diving into features (Theory)
- Feature design & testing (Theory)