Programación de Objetos Distribuidos
Entrega #4:
Implementación de Casos de Uso

Grupo 3.
46099 - Abramowicz, Pablo Federico
46281 - Cabral, Martín Esteban
47031 - Gomez Vidal, Darío Maximiliano
46383 - Goñi, Juan Ignacio
46233 - Palombo, Martín
46069 - Sessa, Carlos Manuel

1. Introducción

En este documento se detallan algunas de las decisiones tomadas para la implementación de los casos de uso de la cuarta entrega. Asimismo, se comentan agregados que se realizaron para la mejor escalabilidad del proyecto y un más eficiente y rápido desarrollo.

2. ConnectionManager

Módulo que carga el driver de oracle y genera la conexión con la base de datos. Mantiene una conexión abierta a la cual acceden los distintos threads para mostrar resultados.

3. ConnectionPool

Al empezar a levantar información de la base de datos usando un único *ConnectionManager* notamos que el sitio funcionaba muy lento. Para resolver este conflicto creamos un *listener* que al iniciar el servidor genere un pool de *ConnectionManager*. Los distintos threads piden un *ConnectionManager* al pool y cuando se cierra el servidor libera todas las conexiones realizadas.

4. Precio de las reservas

En un principio, comenzamos utilizando *double* para representar el precio de las reservas. Ésto trajo una serie de problemas a la hora de calcular si un pago causaba que una reserva pase al estado de señada o pagada debido a que la aritmética de punto flotante no ofrece una representación exacta. Se decidió utilizar la clase *BigDecimal* provista por el paquete *java.util.math* para representar de manera exacta los precios de las reservas. Esto fue debido a que el pago de una reserva es una operación sumamente sensible para el usuario y un mal cómputo del importe podría causar problemas.

5. ACL

Realizamos un módulo de ACL, imponiendo roles y recursos. De esta manera limitamos el acceso.

6. FormHandler

Desde la última entrega el FormHandler se actualizó para que entienda de *JavaScript*. Para realizar esto creamos la librería *J2Query* que se explicará a continuación.

6.1. J2Query

El siguiente módulo permite manejo de JS con JQuery dentro del *FormHandler*.

Mejoras ofrecidas por esta librería:

6.1.1. Campo múltiple

La idea de este campo es que se pueda agregar inputs en el formulario a medida que el usuario lo requiera. Por ejemplo, cuando se agrega un complejo se requiere ingresar más de un teléfono.

6.1.2. Time Picker

En un principio para la selección de fechas utilizamos combos pero notamos que era molesto a la hora de una inserción masiva. Para solucionar esto, decidimos que los campos de fechas sean texto, pero cuenten con un widget de js para facilitar la inserción.

6.1.3. Tooltips

Notamos que alguno de los campos de los fórmularios no tenían labels representativos, entonces utilizamos tooltips para facilitar el entedimiento de los datos a cargar por el usuario.

6.1.4. Futuras mejoras

Por falta de tiempo no pudimos terminar todas las ideas que teníamos para este módulo. Una de ellas, que creemos fundamental, es utilizar la información de los validadores realizados para que se agregue verificación de los campos del lado del cliente.

7. Log4J

Se decidió utilizar un SMTPAppender para enviar un correo electrónico a los administradores del sitio ante un error fatal. Se mantuvo además la funcionalidad presente en la entrega anterior.

8. Selenium

Para realizar el testeo de los casos de uso utilizamos una herramienta que nos permitiera automatizar la carga de datos y simular la navegación del usuario. Se adjunta al código fuente una carpeta con todos los tests realizados, separados según el caso de uso en cuestión.