Facultad de Ingeniería
Escuela de Ingeniería Civil en Informática

DESARROLLO DE UNA PLATAFORMA PARA LA SOLICITUD Y GESTIÓN DE REQUERIMIENTOS Y SCM

Por

Alejandro Alvarez Ahumada

Trabajo realizado para optar al Título de
INGENIERO CIVIL EN INFORMÁTICA
Prof. Guía: Carlos Becerra Castro
Prof. Co-Referente: Nombre Profesor Correferente
Septiembre 2012

Certifico que he leído este documento y que, en mi opinión, es adecuado en ámbito y calidad como trabajo para optar al título de Ingeniero Civil en Informática.

Carlos Becerra Castro Profesor Guía

Certifico que he leído este documento y que, en mi opinión, es adecuado en ámbito y calidad como trabajo para optar al título de Ingeniero Civil en Informática.

Nombre Profesor Correferente Profesor Co-Referente

Certifico que he leído este documento y que, en mi opinión, es adecuado en ámbito y calidad como trabajo para optar al título de Ingeniero Civil en Informática.

Nombre Profesor Informante 1 Profesor Informante

Aprobado por la Escuela de Ingeniería Civil en Informática, UNIVERSIDAD DE VALPARAÍSO.

Resumen

La Dirección de Servicios de Información y Computación (DISICO) de la Universidad de Valparaíso durante los últimos años ha estado en constante crecimiento y en busca de mejoras que le permitan brindar un mejor servicio. Aunque en este poco tiempo son muchas las mejoras que se han hecho, aún quedan aspectos por mejorar, algunos de estos son los procesos relacionados a las solicitudes de requerimientos y solicitudes de cambios, para las cuales ya se han diseñado procedimientos y metodologías, sin embargo se carece de una herramienta que permita la automatización de estas. El propósito de este trabajo de título es dar solución a dicho problema mediante el desarrollo de una plataforma que permita automatizar los procedimientos actuales de solicitud de requerimientos y SCM. Los principales resultados que se esperan son disminuir el tiempo y esfuerzo invertido en la aplicación de las metodologías que existen actualmente.

Agradecimientos

Aquí pueden colocar sus agradecimientos. Si han estudiado con becas es recomendable colocar los agradecimientos a las instituciones que les otorgaron las becas.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Pruebas

En este capitulo se detallan las pruebas realizadas, junto con los resultados obtenidos durante la realización de estas. Las pruebas realizadas se dividen en:

- Pruebas Unitarias.
- Pruebas de Integración.
- Pruebas de Rendimiento.
- Pruebas de Aceptación.
- Pruebas Beta.

El principal enfoque de las pruebas es la detección de errores.

1.1. Pruebas Unitarias

Para llevar a cabo las pruebas unitarias se diseño un set de pruebas automatizadas, las cuales fueron implementadas aciendo uso del framework de pruebas JUnit y de Glassfish Embedded (dentro del cual se despliegan los EJB para ser utilizados durante la ejecución de las pruebas).

Hacer pruebas JUnit en clases java corrientes, es bastante simple y no presenta mayor dificultad, sin embargo cuando se desea realizar pruebas, ya sea unitarias o de integración, en clases java que son Enterprise Java Beans (EJBs), se presentan algunas dificultades las cuales radican principalmente en el echo que los EJB son componentes gestionados por en servidor de aplicaciones y por ende no pueden funcionar fuera de este, esto significa que para poder probar los EJB debemos desplegar los componentes dentro de un servidor de aplicaciones, junto con las configuraciones de persistencia y datasources correspondientes,

para solucionar este inconveniente es que se utiliza Glassfish Embedded, que actúa como contenedor embebido el cual es mas rápido y ligero.

El procedimiento utilizado para ejecutar los test dentro Glassfish Embedded es el siguiente:

- Copiar los archivos del directorio *build/classes* a un directorio temporal *build/embedded* para el despliegue de los EJB.
- Ademas se copia el archivo test-percistence.xml (que contiene una configuración diferente para la ejecución de las pruebas en una base de datos diferente a la de producción) dentro del directorio *build/embedded/META-INF/* y se renombra como *percistence.xml*.
- Antes de levantar Glassfish Embedded se crea un properties de configuración que le indica a Glassfish donde se encuentran los módulos a desplegar y el archivo de configuración *domain.xml* que debe utilizar (en este caso es un archivo aparte que contiene la configuración necesaria para la ejecución del test).
- Luego se inicia el Glassfish Embedded.
- Se ejecutan los test necesarios.
- Por ultimo se cierra Glassfish Embedded y se borra el directorio temporal *build/embedded*.

Para esto gestionar todo ese proceso se implemento una nueva clase denominada BaseTestEJB de la cual extienden todos los test que hacen uso de los EJB.

De acuerdo a lo que se especifico en la fase de diseño de pruebas, se opto por no realizar una documentación extensiva de los test unitarios ya que estos deben ser auto-explicativos y de manera tal que el mismo código documenta la prueba, sus entradas y salidas esperadas.

Sin embargo a continuación en la Tabla 1.1 se presenta un resumen las pruebas unitarias diseñadas y ejecutadas, con su nombre y propósito.

Clase	Test	Propósito
Resources	testGetValue	Verificar que el método getValue es capas de recuperar la cadena ”ABCD” desde un archivo de propiedades.
	testGetValueConEspacios	Verificar que el método getValue es capas de recuperar la cadena ”A B C D” desde un archivo de propiedades sin verse afectado por la cantidad de espacios entre los caracteres.
	testGetValueShort	Verificar que el método getValueShort es capas de recuperar cadena desde un archivo de propiedades y convertirla a Short siempre que cumpla con el formato de este.
	testGetValueShortNegativo	Verificar que el método getValueShort es capas de recuperar cadena desde un archivo de propiedades y convertirla a Short aunque este sea negativo.
	testGetValueShortErrorEnString	Verificar que el método getValueShort dispara la excepción NumberFormatException al leer un String desde el archivo de propiedades.
	testGetValueShortErrorValorMayorAShort	Verificar que el método getValueShort dispara la expeción NumberFormatException al leer un numero entero que excede el valor máximo de un Short.
	testGetValueShortErrorValorDecima	Verificar que el método getValueShort dispara la excepción NumberFormatException al leer un valor con decimales desde el archivo de propiedades.
	testGetValueInteger	Verificar que el método getValueInteger es capas de recuperar cadena desde un archivo de propiedades y convertirla a Integer siempre que cumpla con el formato de este.
	testGetValueIntegerNegativo	Verificar que el método getValueInteger es capas de recuperar cadena desde un archivo de propiedades y convertirla a Integer aunque este sea negativo.
	testGetValueIntegerErrorValorMayorAInteger	Verificar que el método getValueInteger dispara la expeción NumberFormatException al leer un numero entero que excede el valor máximo de un Integer.

Clase	Test	Propósito
Resources	testGetValueIntegerErrorValorMayorAInteger	Verificar que el método getValueInteger dispara la expeción NumberFormatException al leer un numero entero que excede el valor máximo de un Integer.
	testGetValueIntegerErrorValorDecimal	Verificar que el método getValueInteger dispara la excepción NumberFormatException al leer un valor con decimales desde el archivo de propiedades.
	testGetValueLong	Verificar que el método getValueLong es capas de recuperar cadena desde un archivo de propiedades y convertirla a Long siempre que cumpla con el formato de este.
	testGetValueLongNegativo	Verificar que el método getValueLong es capas de recuperar cadena desde un archivo de propiedades y convertirla a Long aunque este sea negativo.
	testGetValueLongErrorValorMayorALong	Verificar que el método getValueLong dispara la excepción NumberFormatException al leer un numero entero que excede el valor máximo de un Long.
	testGetValueLongErrorValorDecimal	Verificar que el método getValueLong dispara la excepción NumberFormatException al leer un valor con decimales desde el archivo de propiedades.
	testGetPropertiesPath	Verifica que el metodo getPropertiesPath obtenga la ruta de un archivo properties correctamente.
	testGetPropertiesPathNotFound	Verfica que el metodo getPropertiesPath dispare la excepcion MissingResourceException al intentar obtener la ruta de un propertie que no existe.
	testGetPageList	Verifica que el metodo getPageList obtenga una lista con todas las propiedades contenidas en el archivo propertie especificado.
	testGetMapPageList	Verifica que el metodo getMapPageList obtenga un Map con todas las propiedades contenidas en el archivo propertie especificado.

Clase	Test	Propósito
MathUtils	testCalcularPorcentaje Redondeado	Verifica que el calculo del porcentaje entero sin decimal.
	testCalcularPorcentaje RedondeadoParaArriba	Verifica que el porcentaje sea redondeado hacia arriba cuando el decimal es mayor o igual a 5.
	testCalcularPorcentaje RedondeadoParaAbajo	Verifica que el porcentaje sea redondeado hacia abajo cuando el decimal es menor a 5.
	testCalcularRegla DeTres	Verifica el calculo de una regla de tres.
	testCalcularReglaDe TresParaDenominadorCero	Verifica que se devuelva un 0 en caso que el denominador sea 0.
	testCalcularReglaDe TresParaNumeradorCero	Verifica que se devuelva un 0 en caso que el numerador sea 0.
	testCalcularReglaDe TresParaCien	Verifica el caso en que se debe devolver 100.
	testCalcularReglaDe TresParaPorcentaje MayorACien	Verifica que funcione cuando el numerador es mayor a cien.
	testRedondearCero	Verifica la funcion redondear cuando el numero es cero.
	testRedondearPara ArribaSinDecimal	Verifica que el numero sea redondeado hacia arriba cuando el decimal es mayor o igual a 5 y el resultado no debe tener ningún decimal.
	testRedondearPara AbajoSinDecimal	Verifica que el numero sea redondeado hacia abajo cuando el decimal es menor a 5 y el resultado no debe tener ningún decimal.
	testRedondearPara ArribaConUnDecimal	Verifica que el numero sea redondeado hacia arriba cuando el decimal es mayor o igual a 5 y el resultado no debe tener un decimal.
	testRedondearPara AbajoConUnDecimal	Verifica que el numero sea redondeado hacia abajo cuando el decimal es menor a 5 y el resultado no debe tener un decimal.
	testRedondearPara ArribaConDecimal5	Verifica el caso en que el decimal es 5.
	testRedondearFloat Cero	Verifica la función redondear cuando el numero es cero. Cuando el resultado es convertido de BigDecimal a Float.

Clase	Test	Propósito
MathUtils	testRedondearFloat ParaArribaSinDeci-mal	Verifica que el numero sea redondeado hacia arriba cuando el decimal es mayor o igual a 5 y el resultado no debe tener ningún decimal. Cuando el resultado es convertido de BigDeci-mal a Float.
	testRedondearFloat ParaAbajoSinDeci-mal	Verifica que el numero sea redondeado hacia abajo cuando el decimal es menor a 5 y el resul-tado no debe tener ningún decimal. Cuando el resultado es convertido de BigDecimal a Float.
	testRedondearFloa tParaArribaConUn Decimal	Verifica que el numero sea redondeado hacia arriba cuando el decimal es mayor o igual a 5 y el resultado no debe tener un decimal. Cuan-do el resultado es convertido de BigDecimal a Float.
	testRedondearFloat ParaAbajoConUn Decimal	Verifica que el numero sea redondeado hacia abajo cuando el decimal es menor a 5 y el re-sultado no debe tener un decimal. Cuando el resultado es convertido de BigDecimal a Float.
	testRedondearFloat ParaArribaCon Decimal5	Verifica el caso en que el decimal es 5. Cuan-do el resultado es convertido de BigDecimal a Float.
TimerSolicitud Requerimientos UpdateTest	testBuscarSolicitudes VencidasVerificar Cambio	Verifica que el método buscarSolicitudesVenci-das haya cambiado el estado de una solicitud enviada cuya fecha de vencimiento ya fue supe-rada.
	testBuscarSolicitudes VencidasVerificar NoCambioVencida	Verifica que el método buscarSolicitudesVenci-das no haya cambiado el estado de solicitudes que ya se encontraban en estado vencida.
	testBuscarSolicitudes VencidasVerificar NoCambioCerrada	Verifica que el método buscarSolicitudesVenci-das no haya cambiado el estado de solicitudes que se Cerraron después de la fecha de venci-miento.
	testBuscarSolicitudes VencidasVerificar NoCambioFinalizada SinRespuesta	Verifica que el método buscarSolicitudesVenci-das no haya cambiado el estado de solicitudes que se Finalizaron sin respuesta después de la fecha de vencimiento.
	testBuscarSolicitudes VencidasVerificar NoCambioSolicitud SinFechaVencimien-to	Verifica que el método buscarSolicitudesVenci-das no haya cambiado el estado de solicitudes que no poseen fecha de vencimiento

Clase	Test	Propósito
SolicitudRequerimientoEJB	testGenerarCodigo NumCero	Verifica que el código es vacío cuando el numero es cero.
	testGenerarCodigo NumNegativo	Verifica que el código es vacío cuando el numero es negativo.
	testGenerarCodigo NumUno	Verifica que el código es ”q” cuando el numero es 1.
	testGenerarCodigo MaxLong	Verifica que el código es ”2teCogGBXee” cuando el numero es el máximo valor de un Long.
	testGenerarCodigo ConsultaNoDuplicados	Verifica que el código no se repite aunque sea generado en instantes de tiempo casi iguales y con ruts similares.
	testValidarCodigo ConsultaExistente	Verifica que un código de consulta sea invalido cuando ya existe.
	testValidarCodigo ConsultaInexistente	Verifica que un código de consulta es invalido cuando no existe.

Tabla 1.1: Test unitarios

1.1.1. Análisis de Resultados

Todas las pruebas han sido implementadas y ejecutadas satisfactoriamente, y todas han sido superadas, es decir luz verde. Todos aquellos que no fueron superados, fueron corregidos y solucionados a la brevedad. Y ante cualquier modificación del código fuente en el futuro, debe volver a ser probadas con este mismo set de pruebas y corregido hasta volver a lograr un 100 % de aprobación de los tests.

1.1.2. Problemas detectados

Durante el desarrollo de los tests, los principales problemas que existieron fueron principalmente debido a la incorrecta configuración de los frameworks de pruebas, pero una vez superado aquello, no se encontraron demasiados problemas. Los principales problemas que se detectaron y corrigieron.

- Problemas en el calculo de porcentajes, cuando existía un denominador 0.
- Duplicación de códigos de consulta en lapsos de tiempos muy cercanos, al no existir variación en la semilla del generador de números aleatorios utilizado en una parte del algoritmo.

1.2. Pruebas Integración

Una vez completada la fase de pruebas unitarias, se realizaron las pruebas de integración, las cuales a diferencia de los especificado en la fase de diseño, se opto por no utilizar Arquillian ya que solo era necesario para utilizar inyección de dependencia en los test. Y presenta algunos problemas en proyectos que no gestionan sus dependencias con maven.

Las pruebas realizadas en esta fase se separaron en 2 tipos, las que prueban directamente la capa de persistencia (ya que probarlas requiere modificar el estado de la base de datos y normalmente de a lo menos las funciones crear, buscar y eliminar, para poder agrega nuevos datos al inicio del test, buscar para verificar la correcta creación o modificación de algo y la eliminación para restaurar el estado de la bd, a su estado original). Y ademas un conjunto de pruebas acorde al esquema de integración definido.