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
Agosto 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

Resumen III

Agradecimientos IV

1. Pruebas **1**

1.1. Pruebas Unitarias 1

Bibliografía **3**

Índice de tablas

1.1. Test unitarios 2

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

Las pruebas unitarias están enfocadas en probar unidades pequeñas de código. Para esto se diseñaron test automatizados, los cuales fueron desarrollados haciendo uso de la JUnit y de Glassfih Embedded, dentro del cual se despliegan los EJB para ser utilizados durante la ejecución de las pruebas.

En la Tabla 1.1 se encuentran documentadas las pruebas realizadas junto con el propósito de cada una.

Modulo	Test	Proposito del Test
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 excepción NumberFormatException al leer un numero entero que excede el valor máximo de un Short.
	testGetValueShortErrorValorDecimal	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

Modulo	**Test**	**Proposito del Test**
Resources	testGetValueIntegerNegativo	Verificar que el método getValueInteger es capas de recuperar cadena desde un archivo de propiedades y convertirla a Integer aunque este sea negativo.
	testGetValueIntegerErrorEnString	Verificar que el método getValueInteger dispara la excepción NumberFormatException al leer un String desde el archivo de propiedades.
	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.
	testGetValueLongErrorEnString	Verificar que el método getValueLong dispara la excepción NumberFormatException al leer un String desde el archivo de propiedades.
	testGetValueLongErrorValorMayorALong	Verificar que el método getValueLong dispara la expeción NumberFormatException al leer un numero entero

Bibliografía

[1] Oracle, "The java ee 6 tutorial." Último acceso: 1 Junio 2012, http://docs.oracle.com/javaee/6/tutorial/doc/.

[2] I. Sommerville, *Ingenieria de Software*. Pearson Addison Wesley, 2005. 7ma Edición.

[3] Rectoría de la Universidad de Valparaíso, "Decreto exento nº 03301." Último acceso: 27 Junio 2012, http://uv.cl/universidad/descargas/archivos/decreto_03301.pdf.

[4] Dirección de Extensión y Comunicaciones, "Manual de normas gráficas de la universidad de valparaíso." Último acceso: 27 Junio 2012, http://uv.cl/universidad/descargas/archivos/manualdenormasUV.pdf.

[5] P. Méndez, "Desarrollo de Metodologías de SQA y SCM para la Dirección de Servicios de Información y Computación," Trabajo de Título, Universidad de Valparaíso, 2011.

[6] Roger Pressman, *Ingeniería del Software: Un enfoque practico*. McGraw-Hill, 2005.

[7] C. Jurado, *Diseñoo Ágil con TDD*. Lulu, 2010.

[8] Object Mentor, "Junit.org resources for test driven development." Último acceso: 24 Junio 2012, http://www.junit.org/.

[9] JBoss Community, "Arquillian - write real tests!." Último acceso: 24 Junio 2012, http://www.jboss.org/arquillian.html.

[10] The Apache Software Fundation, "Apache jmeter." Último acceso: 27 Junio 2012, http://jmeter.apache.org/.

[11] Oracle, "Visualvm all-in-one java troubleshooting tool." Último acceso: 27 Junio 2012, http://visualvm.java.net/.

[12] M. I. M. Estellés, M. C. Bria, P. L. Torres, and F. S. Grueso, "Gestión de requisitos basada en pruebas de aceptación: Test-driven en su máxima expresión.," pp. 61–72, 2010.