Universidad de las Ciencias Informáticas

Facultad 3

“Informatización de métodos para la evaluación ergonómica de puestos de trabajo”.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Anitza Sánchez Alonso.

Yesmely Maite Cuellar Bermudez.

Tutor(es): Ing. Yasser León Montes de Oca.

Dr. Yordán Rodríguez Ruíz.

"LA HABANA, JUNIO 2013"

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2013.

Anitza Sánchez Alonso

Yesmely Maite Cuellar Bermudez

_____

Yasser León Montes de Oca

Yordán Rodríguez Ruiz

_____

AGRADECIMIENTOS

Agradezco a todos **mis amigos**, **familiares**, **tutores** y **demás personas** que de una forma u otra me han ayudado y apoyado en esta tesis, en estos cinco años de curso y en la vida de forma general.

Agradezco especialmente a toda **mi familia** en especial a **mis padres (Elvira y José Miguel)** y a mis **hermanas (Mailín y Aliuska)** que día a día estuvieron preocupados y al tanto de mí, estando desde lejos llamándome en todo momento para saber cómo estaba que si había aprobado las pruebas, exigiéndome que tenía que estudiar en todo momento, y bueno en fin aunque estaban lejos de mí, yo los tenía en mis pensamientos a toda hora y sé que yo en ellos también que gracias a ellos es que aguanté estos 5 años tan lejos porque sabía que un regalo muy grande para ellos sería mi título de ingeniera.

Agradezco especialmente a **Humberto**, que gracias al apoyo de mis padres y a él estoy ahora aquí, por haber dedicado sus vacaciones de 2do año repasándome incondicionalmente para que yo pudiera pasar al 3er año. También a seis amigos míos que aparte de su gran amistad, me ayudaron en el transcurso de estos cinco años en cada una de las asignaturas: **A Martha, a Daniel, a Juan David, a Humberto, a Aylin, a Maricela** y **demás compañeros del aula**.

Agradezco especialmente a mi **novio Yasser** que aparte de haber sido mi tutor y mi amigo fue un ejemplo a seguir como persona. Le agradezco además por haberme ayudado en todo lo que pudo y más, por no haber dudado en ningún momento en dejar sus cosas a un lado para ayudarme y darme su apoyo incondicional. También le agradezco a su mamá Rosita y a **su familia** que me hicieron sentir en todo momento parte de ellos.

Agradezco especialmente a mis **tutores Yasser** y **Yordán** por habernos ofrecido ese tema para investigar tan importante y novedoso y de ahí por haberme brindado su ayuda y dedicación incondicional, gracias también a **Elizabeth** que también nos apoyó mucho en la investigación y nos brindó siempre sus conocimientos, dudas y sugerencias.

Agradezco especialmente al piquete explosivo que gracias a su amistad y alegría constante pasé muy bien y contenta estos últimos 3 años, A **Yesmely (rum pi pi), a Yessenia (la repartera), a Gladis (la malcría), a Martha (la perrilla), a Doris (la morena) y a Yoanzy**.

Agradezco especialmente a la **familia de Yesmely y Yessenia** por quererme como una más de la familia**.**

Agradezco especialmente a mi compañera de tesis por estar siempre a mi lado en el desarrollo de esta investigación.

Les agradezco a los **profesores de la facultad** por sus enseñanzas, exigencias y conocimientos trasmitidos durante estos 5 años.

Anitza Sánchez Alonso

Agradezco a toda mi **familia, amigos, tutores** y **demás personas** que de una forma u otra me han ayudado y apoyado en esta tesis y en estos cinco años de curso.

Agradezco especialmente a mi abuela **Raquel** por ser esa luz al final del camino que me guiaba siempre, a mi papá **Eduardo** por confiar en mí, por ser tan preocupado, por saber cuándo lo necesitaba sin tener que hablar ni decir nada, a mi mamá **Yeslaine** por apoyarme siempre, por estar día tras día preocupada por mí, a mi tías (**Yudelsy y Danory**) porque cada día estaban al tanto de mí, llamándome desde lejos en todo momento para saber cómo estaba, a mi hermanita

Araimis por llamarme cada día, porque a pesar de ser tan pequeña siempre se preocupó por mí, a mis primas (**Magdalena y Lilismay**), Magdy gracias por estar ahí cuando necesite de ti, y bueno en fin aunque estaban lejos de mí, yo los tenía siempre presente y sé que ellos también, que gracias a la confianza que depositaron en mi es que aguanté estos 5 años porque sabía que este título de ingeniera sería un regalo enorme para ellos.

Agradezco especialmente a mis **tutores Yasser** y **Yordán** por habernos ofrecido ese tema tan importante para investigar, por haberme brindado su ayuda y dedicación incondicional, gracias también a **Elizabeth** que nos apoyó mucho en la investigación y nos brindó siempre sus conocimientos, dudas y sugerencias.

Gracias a mi amigo **Luis** (**Pintico**) por soportar mis malcriadeces, por ser tan comprensivo, por cada regaño que me daba, por cada consejo que no acepte, por cada cumpleaños que pasamos juntos, por las comelatas con Vaca (**Irenely),** Toty (**Carlos**), el Negro (**Yeikel**), la Pimpo (**Aylin**), gracias Pintico por estar a mi lado en momentos súper difíciles para mí.

Agradezco a mis amistades de primer año **Lidia, Yaima, Lizandra, Miry, Adiany, Irenely, Ailyn, Yordanka** por pasar momentos tan maravillosos, por confiar en mí y por ayudarme siempre que los necesite en ese año.

Bueno gracias al piquete **La pólvora** muchos sabrán que le dicen así porque donde caigan explotaban, **Pitza, la Yesi, la Glady, la Morena, la Marthi y a Yoanzy,** porque gracias a su alegría, sus ocurrencias pasamos momentos maravillosos, porque podía contar con ellas en momentos buenos y malos, por estos 3 últimos años que vamos a extrañar y van a ser inolvidables, por los campismos juntas y hasta por las salidas a Valle Grande.

Un agradecimiento súper especial a mi **novio Raydel** por ser ante todo un amigo desde el principio, por no desistir de mí, por cada noche que lo llame y vino sin importar la hora que fuese, por apoyarme en los momentos bueno y malos, por darme todo lo que podía y mucho más, por permitirme contar de manera incondicional con él, por cada consejo que me ha dado, por dejar sus cosas para ayudarme y darme su apoyo, por ser esa persona que llegó cuando yo creía que todo estaba perdido y aun así siempre confió en mí, por comprenderme y por ayudarme a ser una mejor persona cada día. Muchas gracias por permitirme compartir cada día a tu lado.

Gracias a todos los del grupo a Carlos, Abel, Laritza, el Nasty y en especial a Chistian (La magia) por este último año que compartimos, porque a pesar de no hablar mucho en el aula en el apartamento eras completamente distinto, por permitirme contar contigo, en fin gracias por todos.

Gracias a mi tío **Toñito**, mi tía **Sonia**, mi madrina **Raiza**, a la chinita **Yanisleidy**, **Manuel**, mi tío **Francisco**, **Katia**, **Karla**, mi tío **Aramis**, **Kaky**, **Yoel**, **Sandy**, **Lisy**, **Georgina**, gracias a cada una de estas personas por confiar en mí, por apoyarme y ayudarme en todo lo que podían, por formar parte de mi familia también.

Gracias a las amistades que conocí en el camino en especial a Reynaldo, Abelito, Humberto, Carlos Yoesly, Yasser porque además de ser mi tutor ha sido un amigo y ejemplo a seguir para mí.

Les agradezco a los **profesores de la facultad** por sus enseñanzas, exigencias y conocimientos trasmitidos durante estos 5 años.

Yesmely Maite Cuellar Bermudez

DEDICATORIA

Le dedico muy pero muy especialmente mi título de ingeniera en informática a mi tío Agustín (**Pacheco**) que en paz descanse, que todavía no entiendo porque se lo llevaron tan pronto de este mundo dejándonos un gran vacío a todos los que lo quieren y lo recuerdan en todo momento. A él porque lo quiero, lo extraño y nunca lo olvidaré.

Anitza Sánchez Alonso

Les dedico esta tesis a mi abuela y mi papá, porque sé que este es el sueño de sus vidas y porque ellos anhelaban tanto este día como yo, se la dedico en especial a mi abuelo Miguel porque antes de partir de este mundo él deseaba llevarme siempre a la escuela, sin importar la edad que tuviese y sé que donde quiera que esté él está muy orgulloso de mi.

Yesmely Maite Cuellar Bermudez

RESUMEN

Los desórdenes músculo-esqueléticos (DMEs) de origen laboral constituyen la enfermedad ocupacional de mayor impacto económico social de la actualidad. Contradictoriamente en la actualidad las herramientas informáticas construidas para agilizar el proceso de evaluación ergonómica no contribuyen al análisis integral de los puestos de trabajo, ni facilitan la toma de decisiones en las organizaciones.

El presente trabajo de diploma tiene como propósito desarrollar una herramienta informática para la evaluación ergonómica de puestos de trabajos, que facilite el proceso de evaluación en un breve período de tiempo y que tribute a la toma de decisiones. Para su cumplimiento se realiza una comparación de las herramientas existentes para la evaluación ergonómica en el mundo. Se definen los requisitos teniendo en cuenta las restricciones o políticas definidas por el cliente. Se presentan los artefactos generados en las etapas de diseño e implementación siguiendo un estilo arquitectónico. Al finalizar se muestran las pruebas efectuadas para detectar y corregir los errores antes de la entrega al cliente.

Con la implementación de esta propuesta se espera que las organizaciones tengan a su disposición un sistema que cumpla con sus necesidades ergonómicas, dotándolos de una aplicación que mejore el proceso de evaluación, que permita identificar los riesgos y la magnitud de estos, a fin de disminuirlos o eliminarlos, mejorando así las condiciones de los puestos de trabajo, haciéndolos más seguros y saludables.

PALABRAS CLAVE

Ergonomía, Evaluación, Métodos Ergonómicos, Desorden Músculo-Esquelético.

TABLA DE CONTENIDOS

TABLAS DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Federico Engels escribió: *"El trabajo es la condición básica y fundamental de toda la vida humana. Y lo es en tal grado que, hasta cierto punto, se debe decir que el trabajo ha creado al propio hombre"* (Engels 1961). Se comparte por los autores el criterio anterior y considera, además, que el trabajo es una actividad consciente del hombre dirigido a adaptar la sustancia de la naturaleza a las necesidades de éste, invirtiendo así, la fuerza de trabajo, energía física, nerviosa e intelectual del hombre para el proceso de creación de los productos que le son útiles. Sobre esta base los autores infieren en que el significado del trabajo para el desarrollo del hombre es extraordinario (Engels 1961).

Las nuevas formas de organizar el trabajo, desde el punto de vista de la calidad de vida laboral, ofrecen a todos los trabajadores oportunidades para la creatividad y la participación, favoreciendo al aprendizaje y la innovación. Además, está demostrado que la participación de las personas en el trabajo también contribuye al desarrollo económico de la empresa. Por lo tanto, la calidad de vida laboral es un punto central para el éxito de las empresas (Margarita Oncins de Frutos).

En el ambiente laboral el trabajador realiza su actividad, se relaciona con su objeto de trabajo, los instrumentos de producción, el puesto de trabajo, la zona de trabajo y los elementos del medio físico o natural que intervienen en el proceso productivo, entre los que se encuentran los factores de riesgo nocivos y peligrosos, que pueden alterar la salud y producir enfermedades, destacándose los desórdenes músculo-esqueléticos (DMEs) de origen laboral.

Los DMEs de origen laboral se producen debido a las deficientes condiciones ergonómicas de los puestos de trabajo. Actualmente, es el entorno el que debe adaptarse a cada trabajador, facilitando así su realización en un trabajo agradable y confortante, en el que pueda desarrollar sus capacidades (Castillo Rosal 2012). La disciplina científica y la profesión responsable de lograr esta adaptación es la Ergonomía la cual se encarga de estudiar las interacciones entre las personas y los otros elementos de un sistema y la profesión que aplica la teoría, los principios, la información y los métodos para optimizar el bienestar humano y el desempeño general del sistema (Association(IEA) 2000-2003).

El proceso de evaluación ergonómica permite identificar los riesgos y la magnitud de estos a fin de eliminarlos mejorando las condiciones de los puestos de trabajo haciéndolos más seguros y saludables (Castillo Rosal 2012).

Actualmente están disponibles varios métodos para la evaluación ergonómica relacionado con los DMEs: Rapid Upper-Limb Assessment (RULA), Rapid Entire Body Assessment (REBA), Strain Index (SI), Occupational Repetitive Actions (OCRA), Ecuación Niosh, Ovako Working Analysis System (OWAS) y Tabla de Snook. Su selección depende del tiempo y recursos disponibles, regiones corporales a evaluar y nivel de conocimientos del usuario que los emplee (personal 26/1/2013).

Existen métodos para la evaluación ergonómica agrupados o enmarcados en varias ramas de la Ergonomía que evalúan física y cognitivamente una tarea en un puesto de trabajo. El proceso de evaluación ergonómica de los puestos de trabajo en las empresas en muchos casos no se realiza y en otros no se lleva a cabo de la manera más correcta. Los períodos para la evaluación se hacen muy extensos y se incurren en errores humanos a la hora de emitir un criterio o una evaluación de un trabajador en un puesto de trabajo.

Las herramientas informáticas más confiables construidas para mitigar estos problemas en su mayoría son privadas, los reportes generados no facilitan un análisis integral y profundo de un puesto de trabajo y en muchos casos no quedan guardadas las evaluaciones de manera que se tenga un historial que facilite la toma de decisiones.

Tomando en cuenta la situación anterior se plantea el siguiente **problema**:

Los procesos automatizados para la evaluación ergonómica de puestos de trabajo no contribuyen al análisis integral de los puestos de trabajo, ni facilitan la toma de decisiones en las organizaciones.

El **objeto de estudio** se centra en los métodos ergonómicos para la evaluación de puestos de trabajo. Sobre un **campo de acción** que está constituido por procesos automatizados de métodos ergonómicos para la evaluación de puestos de trabajo.

Para dar solución al problema anteriormente planteado se ha trazado como **objetivo general** desarrollar una herramienta informática para la evaluación ergonómica de puestos de trabajos que contribuya al análisis integral de los puestos de trabajo y facilite la toma de decisiones en las organizaciones.

Para darle cumplimiento al objetivo y solucionar la problemática de esta investigación se trazaron las siguientes tareas de investigación:

1. Estudiar los métodos ergonómicos para la evaluación de puestos de trabajo.
2. Seleccionar los métodos a ser automatizados.
3. Estudiar las herramientas informáticas disponibles que automatizan los distintos métodos de evaluación ergonómica.
4. Establecer una comparación entre las herramientas disponibles a partir de los siguientes indicadores:

    - ✓ Privada.
    - ✓ Generación de reportes.
    - ✓ Persistencia de datos.
    - ✓ Confiabilidad e integridad de la información acorde a la legitimidad del método ergonómico.

5. Analizar los requisitos funcionales y no funcionales de la herramienta informática a desarrollar.
6. Estudiar las principales tecnologías y herramientas a utilizar para el desarrollo de la solución propuesta.
7. Desarrollar los artefactos definidos para la implementación del sistema:

    - ✓ Diagrama de procesos del negocio.
    - ✓ Documento de especificación de requisitos.
    - ✓ Diagrama de casos de uso del sistema.
    - ✓ Documento de especificación de casos de usos del sistema.
    - ✓ Diagrama de secuencia del diseño.

- ✓ Diagrama de clase del diseño.
- ✓ Documento de arquitectura.
- ✓ Modelo de despliegue.
- ✓ Modelo físico de datos.

8. Implementar los casos de uso necesarios para el funcionamiento de la herramienta informática.
9. Realizar pruebas a la herramienta informática desarrollada.

Este trabajo está conformado por tres capítulos que recogen lo abordado en la investigación:

Estructura del contenido.

Capítulo 1. "Fundamentación teórica". Se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema, así como aquellos que son importantes a tener en cuenta para dar la solución que se requiere.

Capítulo 2. "Análisis y diseño". Se exponen los artefactos generados con la metodología a utilizar como son: procesos de negocios, requisitos, casos de uso del sistema, diagramas de secuencias, diagramas de clases y modelo de datos.

Capítulo 3. "Implementación y prueba". Se exponen los artefactos generados durante la implementación de la solución, así como las métricas y pruebas utilizadas para la validación.

El documento cuenta además con las conclusiones del trabajo, algunas recomendaciones a tener en cuenta para la puesta en práctica del software y la bibliografía consultada.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se presenta el papel de la ergonomía en la mejora de las condiciones de los puestos y sistemas de trabajo, se analizan los procesos automatizados publicados y a su vez disponibles, y los métodos ergonómicos que fueron seleccionados para su informatización. Además, se establece una comparación entre los procesos automatizados que implementan estos métodos de acuerdo a diversos indicadores de interés. Se describe la metodología de desarrollo software a seguir y las principales tecnologías para el desarrollo de la herramienta informática.

1.2 Ergonomía

Tomando como punto de partida el objeto de estudio de la investigación: “Métodos ergonómicos para la evaluación de puestos de trabajo”. Se hace necesaria la explicación y definición de varios términos para su correcto entendimiento.

El término **Ergonomía** se deriva de las palabras griegas: ergos (trabajo) y nomos (leyes, reglas), equivalente a estudio del trabajo. La primera definición de Ergonomía se le atribuye al científico polaco W.B. Jastrzebowski en 1857. La definición contemporánea fue introducida en 1949 por el psicólogo británico K.F.H. Murrell, considerándola en esa época como una ciencia aplicada, una tecnología o ambas.

Relativamente reciente y con total vigencia en el año 2000 la Sociedad Internacional de Ergonomía definió la Ergonomía (o factores humanos) como:

…la disciplina científica que se encarga de estudiar las interacciones entre las personas y los otros elementos de un sistema y la profesión que aplica la teoría, los principios, la información y los métodos para optimizar el bienestar humano y el desempeño general del sistema (Association(IEA) 2000-2003).

1.2.1 Métodos ergonómicos

La correcta evaluación de tareas es un elemento importante para lograr una efectiva prevención de enfermedades y dolencias derivadas del trabajo (Castillo Rosal 2012). Este trabajo de diploma se enfoca

en métodos ergonómicos dirigidos a la evaluación de riesgo de DMEs los cuales son un rango de condiciones surgidas por o asociadas con el trabajo. Ellas son producidas por disconfort o dolor persistente y/u otras disfunciones en articulaciones, músculos, tendones u otros tejidos blandos del cuerpo (NIOSH 1997).

Siguiendo la clasificación realizada por Takala y sus colegas (Takala 2010), los métodos son agrupados en tres grupos: extremidades superiores, manejo manual de materiales y generales.

En el grupo para las extremidades superiores (ES) se encuentran los métodos: Rapid Upper-Limb Assessment (RULA), Strain Index (SI), Occupational Repetitive Actions (OCRA), Stetson's Checklist, Washington State Ergonomic Checklist.

En el grupo para la manipulación manual de materiales (MMM) se encuentran los métodos: NIOSH Lifling Equation, Manual Handling Assessment Charts (MAC) y Tabla de snook.

Por último en el grupo de los métodos generales se encuentran los siguientes: Método práctico para evaluar la exposición a factores de riesgo de desórdenes músculo-esquelético (ERIN), Ovako Working Analysis System (OWAS), ergonomic job Analysis Procedure (AET), Posture Targenting, Ergonomic Analysis (ERGAN), Task Recording and Analysis on computer (TRAC), Portable ergonomic Observation (PEO), Hands Relative to the Body (HARBO), Method Assigned for the Identification Profile of Items Knowledge of Work of ergonomics Hazards (PLIBEL), Quick Eposure check (QEC), Rapid Entire Body Assessment (REBA).

1.3 Selección de los métodos a ser automatizados

Para la selección de los métodos a ser automatizados se siguieron dos criterios: primero, que estén incluidos en normativas nacionales e internacionales y / o avalados por instituciones nacionales o internacionales reconocidas; segundo, según el criterio del Dr. Yordán Rodríguez Ruíz, profesor de Ergonomía del Instituto Superior Politécnico José Antonio Echeverría (CUJAE) y miembro del comité técnico de normalización número 6: "Seguridad y Salud en el Trabajo y Ergonomía" de la República de Cuba.
En la tabla # 1 se describen los métodos seleccionados:

Tabla 1. Descripción de métodos ergonómicos.

Categoría	Método	Descripción
Extremidades superiores	Strain Index (SI)	Es un método que permite valorar si los trabajadores que los ocupan están expuestos a desarrollar desórdenes traumáticos acumulativos en la parte distal de las extremidades superiores debido a movimientos repetitivos. Así pues, se implican en la valoración la mano, la muñeca, el antebrazo y el codo (MOORE 1995).
Manejo manual de materiales	Ecuación Niosh	Este método proporciona una guía para los lugares de trabajo en aceptable límites de peso para el levantamiento de las tareas que, de acuerdo con los desarrolladores del método, protegería a casi todos trabajadores de los trastornos relacionados con el trabajo de espalda asociado con la elevación y el descenso. (McAtamney 2000).
Generales	ERIN	Desarrollado para que personal no experto evalúe individuos expuestos a factores de riesgo de desórdenes músculo-esqueléticos (DMEs) de origen laboral. Evalúa la postura del tronco, brazo, muñeca, cuello y su frecuencia de movimiento; el ritmo, la intensidad del esfuerzo y la autovaloración. Recomienda niveles de acción ergonómica según el nivel de riesgo global (Ruíz 2011)

1.4 Procesos automatizados de métodos ergonómicos para la evaluación de puestos de trabajo.

En la actualidad existen herramientas informáticas para agilizar el proceso de evaluación ergonómica, muchas de estas herramientas son privadas mientras que otras no cumplen con requisitos básicos como la confiabilidad e integridad de la información acorde a la legitimidad de los métodos ergonómicos

utilizados. Además, la mayoría no generan reportes que tributen a un correcto análisis de la información recogida y en muchos casos no quedan guardadas las evaluaciones de manera que se tenga un historial que apoye a la toma de decisiones.

A continuación se muestra en la tabla # 2 las herramientas analizadas de acuerdo a un conjunto de criterios de comparación:

Tabla 2. Comparación entre las herramientas para la evaluación ergonómica.

						Lenguaje				Reporte				Portabilidad		Usabilidad				
Método	**Software**	**Extremidades Superiores**	**Manejo Manual de Materiales**	**Generales**	**Libre**	**Ingles**	**Español**	**Otros**	**Ficha**	**General**	**Exportar**	**Manual de usuario**	**Persistencia de Datos**	**Plataforma**	**Instalación**	**Entendible**	**Fácil de aprender**	**Operable**	**Atractivo**	**Referencia**
Rula	RULAv04 Revised	X			X	X			X		.xls	X		W		X	X	X	X	1
	e_Rula	X			X		X		X					W		X	X	X		2
	MIRTH RULA	X			X	X			X		.txt, .mdb	X	X	W		X	X	X		
Strain Index	MIRTH Strain Index	X			X	X			X		.txt, .mdb	X	X	W		X	X	X		

Ocra	MIRTH OCRA Procedure	X			X	X			X		.txt, .mdb	X	X	W		X	X	X		
Niosh	LTAS.NET 3.0		X		X	X			X	X	X	X	X	W	X	X	X	X	X	3
	e-Niosh		X		X		X		X		.QRP			W					X	2
	NIOSH BViewer		X			X			X	X			X	W	X	X	X	X	X	4
Reba	Mirth REBA Procedure			X	X	X			X		.txt, .mdb	X	X	W		X	X	X		
OWAS	WinOWAS			X	X		X				.ows	X	X	W			X			
Rula, SI, OWAS, OCRA	ErgoMet 2.0	X		X			X		X			X	X	W	X	X	X	X	X	5
Rula, OWAS, Reba, OCRA, Niosh, Snook, Job SI	ergonautas	X	X	X		X			X		.rtf, .pdf		X	W	X	X	X	X	X	6
OWAS, Reba, OCRA, NIOSH, Snook	Ergo/IBV 11.0	X	X	X			X			X			X	W	X	X	X	X	X	7

OCRA, REBA, NIOSH	HADA 2.0	X	X				X		X		.xls			W	X	X	X	X	X	8
REBA, NIOSH, OCRA, SI, Rula	ERGOEASY Profesional	X	X	X		X								W		X	X	X	X	9
NIOSH, OWAS, RULA, REBA	ErgoFellow	X	X	X		X							X	W	X	X	X	X	X	

De acuerdo al contenido en la tabla # 2 se deduce que la totalidad de las herramientas mostradas no cumplen con la mayoría de los parámetros a comparar. La mayor diferencia radica en la licencia bajo la cual estos productos están publicados, el soporte multi-plataformas, la generación de reportes que tribute a la toma de decisiones, la presencia de un instalador para su despliegue y en su mayoría no generan acciones ni recomendaciones al usuario final.

Tanto el cliente de este producto de software, como sus desarrolladores en la UCI, están interesados en una aplicación completamente de código abierto. Por otra parte, el despliegue de la herramienta se debe hacer sobre varias plataformas (Linux y Windows) indicador que ninguna de las analizadas cumple para el caso de la plataforma Linux. Además, los reportes que se generan son fundamentalmente fichas técnicas y no permiten establecer una visión global de la entidad que se evalúa en cuestión.

Es necesario resaltar, que la selección de una de estas herramientas para dar solución al problema que ocupa la investigación en curso, queda descartada por no cumplir en su totalidad con esos indicadores; lo cual justifica el desarrollo de una herramienta que sea libre, que implemente varios métodos para garantizar una evaluación ergonómica integral, que exista persistencia de datos, que genere reportes para facilitar la toma de decisiones y que sea multiplataforma.

1.5 Proceso de desarrollo de software

Un proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software es una estructura aplicada al desarrollo de un producto de software. Tiene como propósito la producción eficaz y eficiente de un producto de software que reúna los requisitos del cliente. Dicho proceso, en términos globales se muestra en la Figura 1 (Jacaboson 2000).

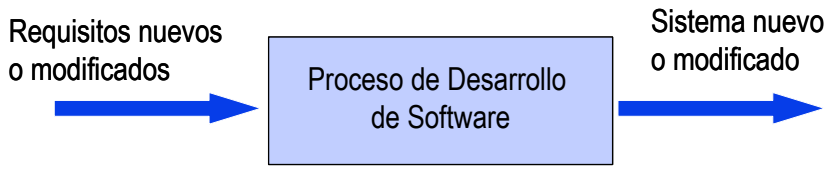


Figura 1. Proceso de desarrollo de software.

Existen varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque distinto para diferentes actividades que tienen lugar durante el proceso.

Una gran cantidad de organizaciones de desarrollo de software implementan metodologías para el proceso de desarrollo a continuación se muestran algunas de estas.

1.6 Metodología de desarrollo de software

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo (INTECO 2009, Marzo).

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente, donde predecir no significa perder la capacidad adaptativa, no significa evitar la introducción de cambios en los requisitos, ni evitar que nuevos requisitos surjan sino definir un camino reproducible para obtener resultados confiables. Definen, además, una representación que permite facilitar la manipulación de modelos y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema (Gacita 2003).

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, se tienen aquellas propuestas más tradicionales como son RUP (Rational Unified Procces), MSF (Microsoft Solution Framework) y MOF (Microsoft operation framework), que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Por otra parte, existen las metodologías ágiles como son: Extreme Programming (XP), Scrum, Agile Unified Process (AUP), Agile Modeling Adaptive Software Development (ASD), Crystal Clear, Agile Documentation, Agile Data Method, LeanCMMI entre otras, que se centran en otras dimensiones, como por ejemplo el factor humano o el producto software, dando mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas (Abrahamsson 2002).

A continuación, en la tabla # 3 se muestra una comparación entre las metodologías ágiles y tradicionales que justifican la selección de la metodología a utilizarse. (Abrahamsson 2002).

Tabla 3. Comparación entre metodologías ágiles y tradicionales.

Metodologías tradicionales	Metodologías ágiles
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software

Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Para el desarrollo de la herramienta informática se utilizó el **Proceso Unificado Ágil** de Scott Ambler o en inglés **Agile Unified Process (AUP)**, ya que este tipo de metodología se adapta al entorno de trabajo en el cual se desarrollará el software. Además, propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo.

AUP es una versión simplificada del Proceso Unificado de Rational (RUP). Esta metodología se describe de una manera simple y fácil de entender, como la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Establece un Modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina: Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de las disciplinas: Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno, coinciden con las restantes de RUP (Roberth G. Figueroa).

Usando la metodología AUP para el desarrollo de software se generaran los siguientes artefactos al pasar por cada una de las disciplinas de Modelado, Implementación y Prueba:

Modelado

- ✓ Diagrama de procesos de negocio.
- ✓ Documento de especificación de requisitos.
- ✓ Casos de uso del sistema (CUS).
- ✓ Documento de descripción de CUS.
- ✓ Diagramas de secuencias.
- ✓ Documento de arquitectura.
- ✓ Diagrama de clase del diseño.
- ✓ Modelo de datos.

Implementación

- ✓ Diagrama de componente.

- ✓ Diagrama de despliegue.

Prueba

- ✓ Diagrama de control de flujo.
- ✓ Casos de prueba de caja negra por cada camino del diagrama de control de flujo.
- ✓ Casos de prueba funcionales.

La metodología AUP reúne características como:

Iterativo e incremental

- ✓ Descomposición de un proyecto grande en mini-proyectos.
- ✓ Cada mini-proyecto es una iteración.
- ✓ Las iteraciones deben estar controladas.
- ✓ Cada iteración trata un conjunto de casos de uso.

Ventajas del enfoque iterativo

- ✓ Detección temprana de riesgos.
- ✓ Administración adecuada del cambio.
- ✓ Mayor grado de reutilización.
- ✓ Mayor experiencia para el grupo de desarrollo.

Dirigido por casos de uso

- ✓ Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo o dispositivo) que interactúa con él.
- ✓ Casos de uso como el hilo conductor que orienta las actividades de desarrollo.

Centrado en la arquitectura

- ✓ Concepto similar a la arquitectura de un edificio.
  - Varios planos con diferentes aspectos del edificio.

- Tener una imagen completa del edificio antes de comenzar la construcción.

✓ Arquitectura en software.

- Diferentes vistas del sistema: estructural, funcional, dinámico, etc.
- Plataforma en la que va a operar.
- Determina la forma del sistema.

✓ Arquitectura: determina la forma del sistema.

✓ Casos de uso: determinan la función del sistema.

En la figura # 2 se muestra el ciclo de vida de la metodología AUP, con solo las disciplinas por la que pasará la aplicación en desarrollo.

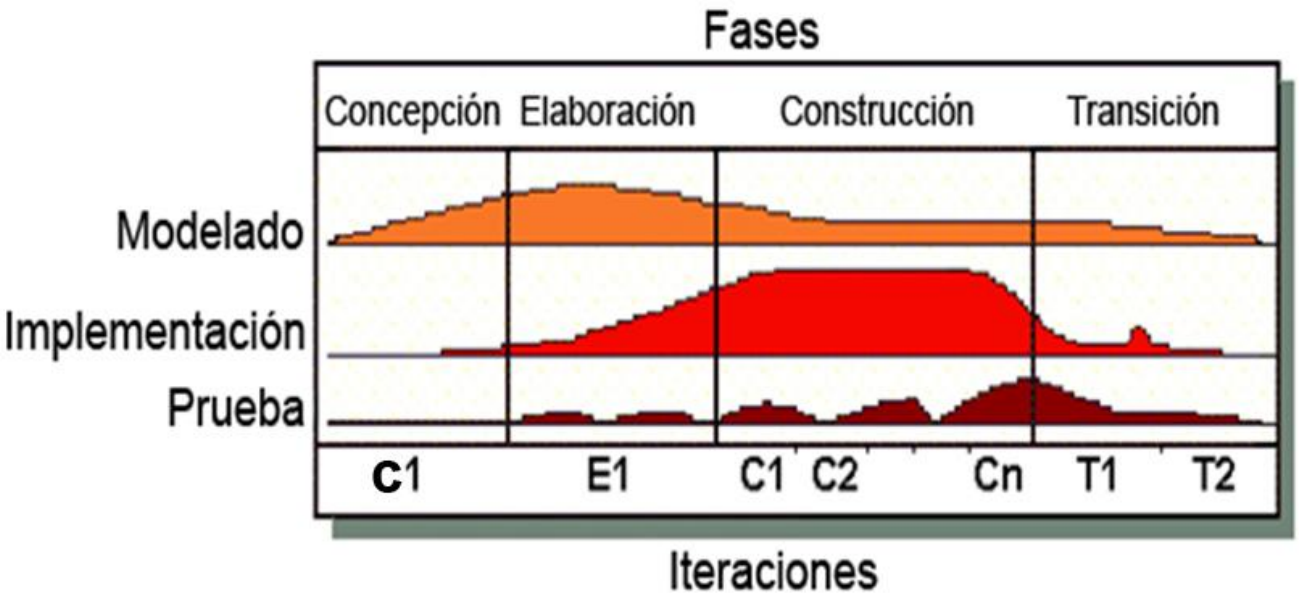


Figura 2. Ciclo de vida del proceso unificado ágil (AUP).

Ventajas

✓ El personal sabe lo que está haciendo: no obliga a conocer detalles.

✓ Simplicidad: apuntes concisos.

- ✓ Agilidad: procesos simplificados del RUP.
- ✓ Centrarse en actividades de alto valor: esenciales para el desarrollo.
- ✓ Herramientas independientes: a disposición del usuario.
- ✓ Fácil adaptación de este producto: de fácil acomodo (HTML).

Desventajas

- ✓ El AUP es un producto muy pesado en relación al RUP.
- ✓ Como es un proceso simplificado, muchos desarrolladores eligen trabajar con el RUP, por tener a disposición más detalles en el proceso.

1.7 Lenguajes y herramientas

Se hace necesario utilizar un lenguaje que permita desarrollar la herramienta informática de manera que sea multiplataforma y que permita satisfacer todas las funcionalidades requeridas. Para lograr la portabilidad del sistema que se quiere desarrollar y la independencia de la plataforma, debe ser una herramienta de escritorio. Teniendo en cuenta las características expuestas anteriormente se muestran a continuación los lenguajes y herramientas seleccionados por el equipo de desarrollo:

1.7.1 Lenguaje unificado de modelado

El éxito de los proyectos de desarrollo de aplicaciones o sistemas se debe al enlace entre quien tiene la idea y el desarrollador. La herramienta encargada de este enlace es el Lenguaje Unificado de Modelado (UML) por sus siglas en inglés, ayuda a capturar la idea de un sistema para comunicarla posteriormente a quien esté involucrado en su proceso de desarrollo; permitiendo así la representación conceptual y física de un sistema (Mora 2002).

UML es un lenguaje visual de modelado para visualizar, especificar, construir y documentar los artefactos de un sistema software (Group September 2001) es decir, UML es ante todo un lenguaje gráfico que estandariza la forma de crear diagramas, el significado preciso de los mismos, y las relaciones existentes entre ellos.

La popularidad que ha ido adquiriendo el modelado de sistemas software con diagramas UML se encuentra vinculada a la existencia de herramientas CASE para UML que faciliten su gestión. Existen varias herramientas CASE para UML ejemplo de ellas Visual Paradigm.

Visual Paradigm

Considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones, tales como NetBeans IDE (Integrate Development Environment). Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación.

Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además, apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Sommerville 2005).

Notación para modelado de procesos de negocio

El objetivo esencial del lenguaje estándar BPMN (por sus siglas en inglés Business Process Modeling Notation), es proveer una notación que sea legible y entendible para todos los usuarios de negocios, desde los analistas que realizan el diseño inicial de los procesos y los responsables de desarrollar la tecnología que ejecutará estos procesos, hasta los gerentes de negocios encargados de administrar y realizar el monitoreo de los procesos. BPMN define un modelo de procesos de negocio basándose en diagramas de flujo.

Un modelo de procesos de negocio, es una red de objetos gráficos que representan las actividades (por ejemplo tareas) y los controles de flujo que definen su orden de ejecución. Hasta la aparición de BPMN no existía un estándar específico sobre técnicas de modelado desarrollado para estos fines. BPMN ha sido desarrollado para proveer una notación estándar a los usuarios, de forma análoga a como UML estandarizó el mundo del modelado en la Ingeniería de Software (Giandini 2010).

1.7.2 Lenguaje de programación

Los lenguajes de programación son idiomas diseñados para expresar cálculos y procesos que serán llevados a cabo por ordenadores. Un lenguaje de programación está formado por un conjunto de palabras reservadas, símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. El proceso de programación consiste en la escritura, compilación y verificación del código fuente de un programa (Guevara).

Java

El lenguaje de programación Java presenta ciertas características de interés para el equipo de desarrollo de esta investigación. Entre ellas se encuentran: uso del paradigma orientado a objetos, independencia de la plataforma, fácil de aprender, de sintaxis sencilla, verboso y con gran cantidad de recursos disponibles, tanto en librerías como en documentación y comunidad (Guevara).

La versión del JDK por sus siglas en inglés Java Development Kit utilizada es JDK 1.7, la cual se integra con el IDE (Integrate Development Environment) de desarrollo NetBeans que se presenta en la siguiente sección.

Su utilización en el desarrollo de la solución responde a un requisito no funcional identificado con el cliente. Además, de la familiarización con el mismo del equipo de desarrollo y la buena integración con las otras herramientas y tecnologías que en el presente capítulo se presentan.

IDE de desarrollo

El IDE de desarrollo que se utiliza es el NetBeans v6.9, este es con el que más familiarizado está el equipo de desarrollo, además es un entorno de desarrollo integrado libre. Esta plataforma es una base modular y extensible usada como una estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones (Domínguez-Dorado Noviembre, 2005). En el desarrollo de la solución esto se evidencia en la integración con los plugins iReport-4.5.0 y Jasperreports-components-plugin-4.5.0.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están (Domínguez-Dorado Noviembre, 2005) (web):

- ✓ Administración de las interfaces de usuario tales como: menús y barras de herramientas.
- ✓ Administración de las configuraciones del usuario.
- ✓ Administración del almacenamiento, guardando y cargando cualquier tipo de dato.
- ✓ Administración de ventanas.
- ✓ Framework basado en asistentes de tipo diálogos paso a paso.

1.7.3 Herramientas para la generación de reportes

Existen numerosas herramientas para la generación de reportes, algunas de código abierto y otras no, con abundante bibliografía y fáciles de integrar al entorno de desarrollo. Tres de estas que cumplen con las mejores de cada una de estas características, cuya integración las hacen fuertes y potentes en cuanto a sus alcances y manejabilidad y suficientes para cubrir cualquier necesidad, son JasperReport, Ireport y JFreeChart, además, del notable hecho que son de código abierto (Franklin Rivero Duharte 2005-2006).

En el desarrollo de la solución se emplean los plugins iReport-4.5.0.nbm y Jasperreports-components-plugin-4.5.0.nbm para el IDE NetBeans el cual permite el diseño, la configuración y generación de los reportes que se generarán en la herramienta propuesta.

1.7.4 Gestor de base de datos

SQLite es una pequeña librería en C (Lenguaje de programación) que contiene en ella un completo sistema de gestión de base de datos. A diferencia de los motores de base de datos convencionales con la arquitectura cliente-servidor, SQLite es independiente, ya que no se comunica con un motor de base de datos, sino que las librerías de SQLite pasan a integrar la aplicación. La misma utiliza las funcionalidades de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre

procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar, en la máquina local (Gerardo Antonio Cabero).

A continuación se presentan las características fundamentales que justifican el uso de esta librería como gestor de base de datos para la herramienta a desarrollar (Gerardo Antonio Cabero):

Tamaño: SQLite tiene una pequeña memoria y una única biblioteca, es necesaria para acceder a base de datos, lo que lo hace ideal para aplicaciones de base de datos incorporadas.

Portabilidad: Se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.

Estabilidad: Es compatible con ACID, reunión de los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad (Gerardo Antonio Cabero).

SQL: Implementa un gran sub-conjunto de la ANSI - 92 SQL estándar, incluyendo sub-consultas, generación de usuarios, vistas y triggers.

Interfaces: Cuenta con diferentes interfaces del API, las cuales permiten trabajar con Java, C++, PHP, Perl, Python, Tcl, groovy, etcétera.

Costo: Es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

Estas características junto al hecho de que la herramienta es una aplicación no distribuida y de escritorio hacen que SQLite sea el gestor más apropiado a utilizar. Para su integración con el lenguaje Java se utiliza el driver JDBC para SQLite en su versión sqlitejdbc-v056.

Cliente SQLiteManager

Para la comunicación con el gestor anteriormente descrito se utiliza el cliente SQLiteManager en su versión SQLiteManager-0.7.7, el cual constituye una extensión del navegador Web Mozilla Firefox. A continuación se mencionan sus principales características:

- ✓ Un árbol jerárquico intuitivo que muestra objetos de la base de datos.
- ✓ Ventanas útiles para gestionar las tablas, índices, vistas y disparadores.
- ✓ Se puede navegar y buscar en las tablas, así como añadir, editar, eliminar y duplicar los registros.
- ✓ Facilidad para ejecutar cualquier consulta SQL (por sus siglas en inglés structured query language).
- ✓ Un menú desplegable ayuda con la sintaxis SQL lo que hace la escritura más fácil.
- ✓ Fácil acceso a las operaciones más comunes a través de menús, barras de herramientas, botones y el menú contextual.
- ✓ Se pueden guardar las consultas.

1.8 Patrones de diseño

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que pueda utilizarse un millón de veces sin tener que hacer dos veces lo mismo. Un patrón es la solución general, fruto de la experiencia, a un problema general que puede adaptarse a un problema concreto (Larman 1999).

Durante el desarrollo de la solución se utilizarán varios de los patrones de diseño los cuales están divididos en GoF (Gang of Four) y GRASP, estos últimos más conocidos como patrones de asignación de responsabilidades. A continuación una breve descripción de los mismos:

1.8.1 Patrones de asignación de responsabilidades (GRAPS)

Los patrones a los que se hace referencia se aplican durante la elaboración de los diagramas de interacción, al asignar las responsabilidades a los objetos y al diseñar la colaboración entre ellos. A continuación se describen un conjunto de estos patrones (Gamma and Wesley 1995):

Experto: Su solución consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los

sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Creador: Su solución consiste en asignarle a la clase B la responsabilidad de crear una instancia de clase A. El propósito fundamental de este patrón es encontrar un creador que debe conectar con el objeto producido en cualquier evento.

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.

Bajo acoplamiento: Su solución consiste asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras.

Controlador: Su solución consiste en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario (IGU) operado por una persona. Si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan (Larman 1999).

1.8.2 Patrones GoF

Singleton (instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

1.9 Pruebas de software

Una gestión de calidad bien implementada, especialmente durante el proceso de desarrollo, reducen los costos, acortan el tiempo necesario para el mismo y aumentan la calidad del sistema final.

Un instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos. Se usan casos de prueba, especificados de forma estructurada.

1.9.1 Prueba unitaria

Objetivo de la prueba: Se focaliza en ejecutar cada módulo, lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores.

Busca asegurar que el código funciona de acuerdo con las especificaciones y que el módulo lógico es válido.

Descripción de la prueba:

- ✓ Particionar los módulos en unidades lógicas fáciles de probar.
- ✓ Por cada unidad hay que definir los casos de prueba (pruebas de caja blanca).

Pruebas de caja blanca o estructural

Se denomina pruebas de caja blanca (Figura # 3) a un tipo de prueba de software que se realiza sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las pruebas de caja blanca están dirigidas a las funciones internas. Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las pruebas de caja negra sobre varios subsistemas (Pressman 2010). La técnica que se utilizó fue:

Prueba del camino básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de

control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

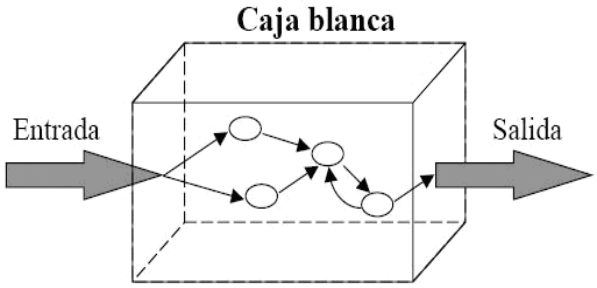


Figura 3. Flujo de caja blanca.

1.9.2 Prueba funcional

Objetivo de la prueba: Se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Descripción de la prueba:

Las pruebas funcionales deben enfocarse en los requisitos funcionales, las pruebas pueden estar basadas directamente en los casos de uso (o funciones de negocio), y las reglas del negocio. Las metas de estas pruebas son:

- ✓ Verificar la apropiada aceptación de datos.
- ✓ Verificar el procesamiento, recuperación y la implementación adecuada de las reglas del negocio.

Este tipo de pruebas están basadas en el método de caja negra, que es verificar la aplicación (y sus procesos internos) mediante la interacción con la aplicación vía GUI (por sus siglas en inglés graphical user interface), y analizar la salida (resultados). A continuación se muestra un diseño preliminar (Pressman 2010).

Método de Caja Negra: Los casos de prueba se basan sólo en el comportamiento de entrada/salida.

- ✓ Orientadas a los casos de uso.

Buscan asegurar que:

- ✓ Se ha ingresado toda clase de entrada.
- ✓ Que la salida observada sea igual a la esperada.

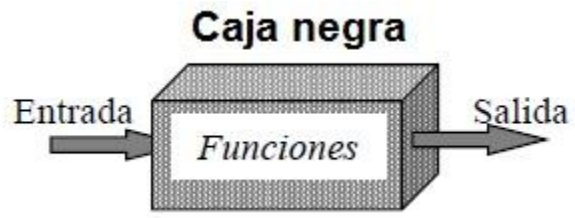


Figura 4. Flujo de caja negra.

Conclusiones

En el presente capítulo se puede concluir diciendo que en la actualidad no se cuenta con una herramienta informática para la evaluación ergonómica de puestos de trabajos, que facilite el proceso de evaluación en un breve período de tiempo y que tribute a la toma de decisiones.

Los métodos ergonómicos seleccionados por cada categoría estudiada a implementar son: SI, Ecuación Niosh y ERIN mediante el criterio de expertos.

Se propone la creación de una herramienta cuya metodología y tecnologías de construcción sean: AUP, BPMN, Visual Paradigm, lenguaje de programación Java, IDE de desarrollo NetBeans, JasperReports para la generación de reportes y para la gestión de base de datos SQLite.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA.

Se aborda una visión general del problema que se está resolviendo y las áreas claves de funcionalidad que se deben tratar en la solución. Los artefactos generados, modelarán los principales procesos del sistema propuesto a desarrollar, en un lenguaje entendible para los programadores. Del análisis de esta etapa, depende en gran medida el logro de los objetivos finales, mientras que la calidad depende de la modelación clara y específica de los artefactos generados mediante el análisis y diseño. De igual manera se definirá en el presente capítulo la arquitectura y los patrones a utilizar.

2.1 Modelado de negocio

Describe un conjunto de actividades dentro de los procesos que se desarrollan para la evaluación ergonómica de puestos de trabajos. El objetivo es comprender la estructura del mismo, así como tener una visión de la organización.

El negocio está compuesto por dos áreas de procesos las cuales son: Registro y Evaluación. El área Registro se divide en dos subáreas las cuales son Estructura de la organización y RRHH (Recursos Humanos). En la estructura de la organización se registran todos los datos de la empresa u organización y en RRHH se registra todo el personal de la empresa asociados a un puesto de trabajo dentro de un área de la empresa en una tarea específica. Una vez registrada la estructura de la empresa y el personal de esta, el evaluador puede proceder a realizar la evaluación, esta área está dividida en tres subáreas las cuales son Generales, Manejo manual de materiales y Extremidades superiores y cada una de estas con el método correspondiente.

A continuación se muestran los diagramas de proceso de negocio:

El diagrama de procesos de negocio para el registro de la empresa está dividido en dos subprocesos, el primero Registrar estructura que posee una secuencia de actividades para registrar la organización y el segundo Registrar personal este tiene algunas actividades que dependen de las anteriores (área, puesto). Para mayor entendimiento de estos, se pueden apreciar en las Figuras 5, 6 y 7.

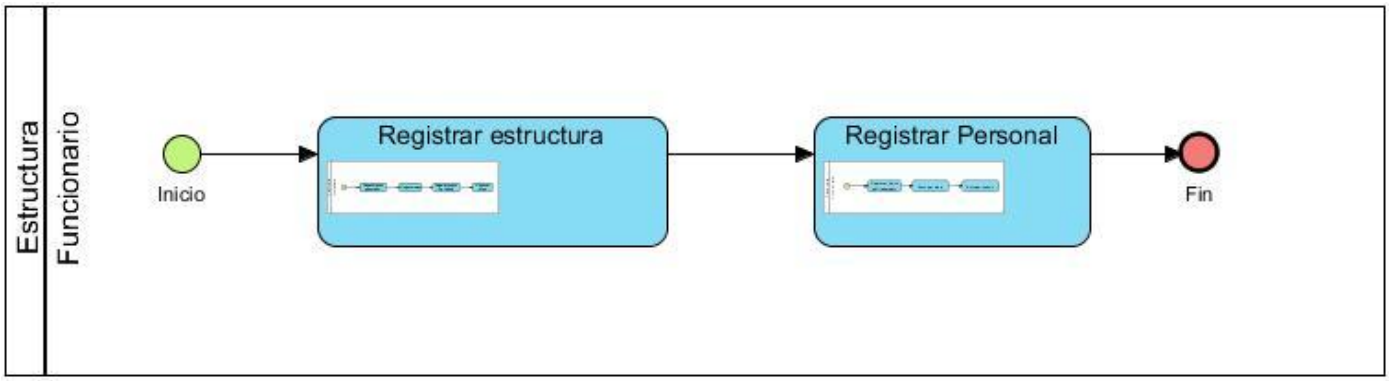


Figura 5. Diagrama de procesos de negocio para el registro de la empresa.

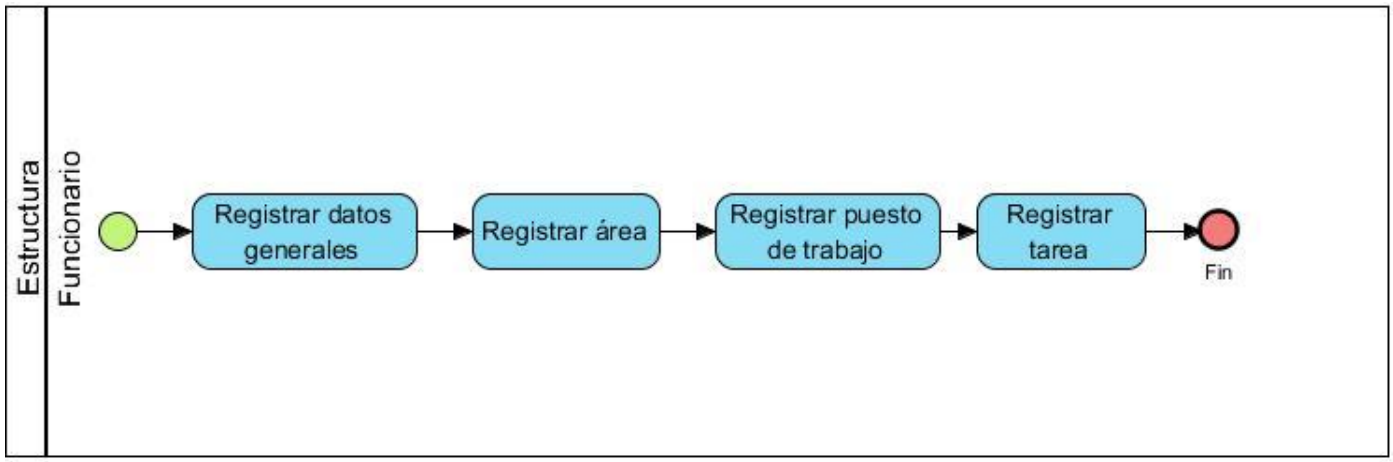


Figura 6. Diagrama de procesos para registrar estructura.

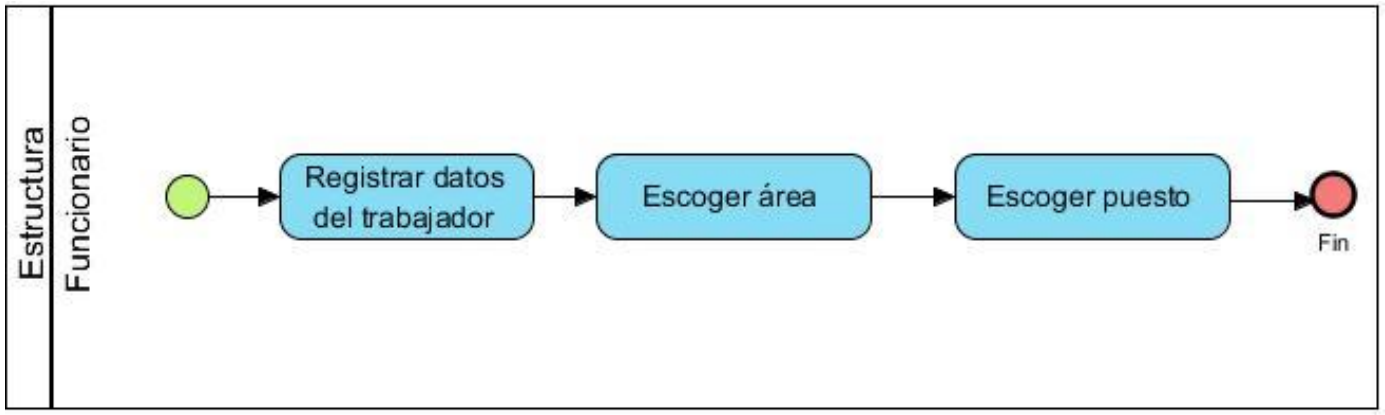


Figura 7. Diagrama de procesos para registrar trabajador o personal.

El diagrama de procesos de negocio para realizar la evaluación consiste en seleccionar dentro del puesto de trabajo la tarea que se desea evaluar, una vez seleccionada esta, se debe seleccionar la categoría de evaluación para dentro de esta escoger un método, ya terminada la evaluación se guarda y se genera una ficha, si desea evaluar con otro método debe seleccionar una de las categorías nuevamente sino puede terminar. Para mayor entendimiento de este proceso, se puede apreciar en las Figuras 8.

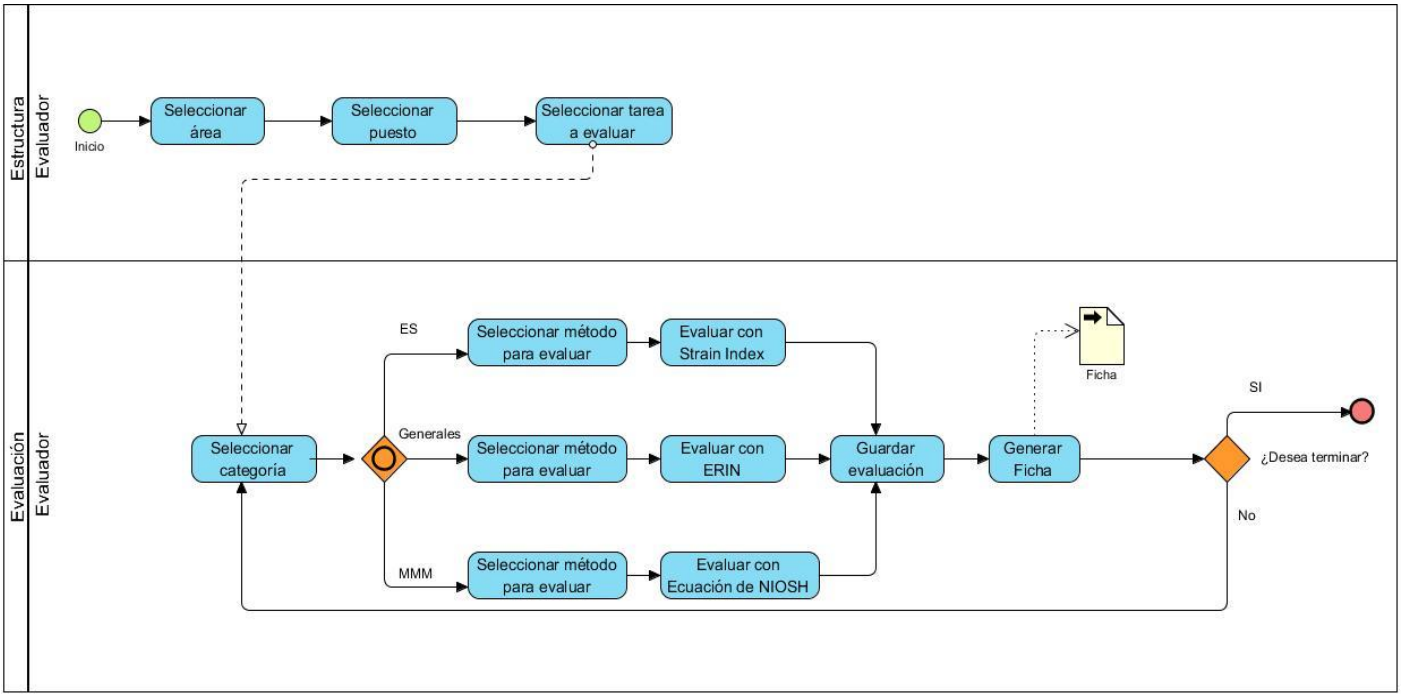


Figura 8. Diagrama de procesos para el módulo Evaluación.

2.2 Requisitos del sistema

Un requisito es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente (Ivar Jacobson 2000).

Captura de requisitos

En la captura de requisitos se aborda una visión general del problema que se está resolviendo y las áreas claves de funcionalidad que se deben tratar en la solución. Ese flujo de trabajo brinda una visión de qué es necesario hacer para dar respuesta a las solicitudes del usuario, dado que tiene como objetivo

principal describir qué hará el sistema (Sommerville 2005). En el mismo se capturan los requisitos funcionales y no funcionales que el sistema debe poseer lo que posibilita a desarrolladores y clientes un entendimiento común, además, provee una base para planificar el costo y la duración del proyecto, establecer y mantener un acuerdo entre el equipo de desarrollo y los clientes acerca de lo que el sistema debe hacer.

2.2.1 Requisitos funcionales del sistema

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir (Ivar Jacobson 2000). A continuación se enuncian los requisitos funcionales identificados a partir de los resultados de las entrevistas aplicadas a los clientes en la etapa de la captura de requisitos, ver especificación de requisitos en Anexo 1:

RF 1 Registrar estructura de la organización: Permitirá que el funcionario defina la estructura de la organización.

RF 2 Modificar estructura de la organización: Permitirá que el funcionario modifique la estructura de la organización.

RF 3 Eliminar organización: Permitirá que el funcionario elimine la estructura.

RF 4 Insertar área: Permitirá al funcionario insertar el área.

RF 5 Modificar área: Permitirá al funcionario modificar el área seleccionada.

RF 6 Eliminar área: Permitirá al funcionario eliminar el área seleccionada.

RF 7 Insertar puesto: Permitirá al funcionario insertar un puesto en correspondencia con el área especificada.

RF 8 Modificar puesto: Permitirá al funcionario modificar un puesto en correspondencia con el área especificada.

RF 9 Eliminar puesto: Permitirá al funcionario eliminar un puesto en correspondencia con el área especificada.

RF 10 Insertar tarea: Permitirá al funcionario insertar una tarea que deberá estar en correspondencia con el puesto.

RF 11 Modificar tarea: Permitirá al funcionario modificar una tarea que deberá estar en correspondencia con el puesto.

RF 12 Eliminar tarea: Permitirá al funcionario eliminar una tarea que deberá estar en correspondencia con el puesto.

RF 13 Registrar personal de la organización: Permitirá que el funcionario adicione el personal de la organización, mostrando el resultado en una tabla, en la parte inferior.

RF 14 Eliminar personal de la organización: Permitirá que el funcionario elimine la persona seleccionada, mostrando el resultado en una tabla, en la parte inferior.

RF 15 Editar personal de la organización: Permitirá que el funcionario modifique la persona seleccionada, mostrando el resultado en una tabla, en la parte inferior.

RF 16 Evaluar con el método ERIN: Permitirá al evaluador realizar la evaluación, a una tarea dentro de un puesto de trabajo con el método ERIN.

RF 17 Evaluar con el método Ecuación de Niosh: Permitirá al evaluador realizar la evaluación, a una tarea dentro de un puesto de trabajo con el método Ecuación de Niosh.

RF 18 Evaluar con método Strain Índex (SI): Permitirá al evaluador realizar la evaluación, a una tarea dentro de un puesto de trabajo con el método SI.

RF 19 Generar reporte general: Permitirá al evaluador generar un reporte general de la organización.

RF 20 Generar ficha: Permitirá al evaluador generar una ficha por cada evaluación.

2.2.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para

que los clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (Ivar Jacobson 2000).

A continuación se enuncian los requisitos no funcionales que la aplicación debe cumplir:

Hardware: La computadora que utilizará el software a desarrollar deberá tener la velocidad de la CPU (por sus siglas inglés Central Processing Unit) a 1,6 GHz como mínimo, se recomienda uno superior. La memoria RAM (por sus siglas en inglés Random-Access Memory) debe tener como mínimo 512 MB o superior.

Diseño: El sistema debe tener una apariencia profesional y un diseño gráfico sencillo, debe ser fácil de utilizar para que usuarios inexpertos puedan familiarizarse rápidamente.

Software: El sistema se podrá ejecutar sobre los sistemas operativos Linux, Windows XP/Vista/7. Para que el sistema se ejecute correctamente la versión de la máquina Virtual de Java requerida es JVM 1.6u25, para la persistencia de datos se utiliza la biblioteca SQLite que implementa un motor de base de datos SQL embebido.

Restricciones en el diseño y la implementación: El sistema se implementa usando el lenguaje de programación Java, como entorno de desarrollo Netbeans, para la comunicación con la base de datos se usa la librería SQLiteManager-0.7.7, como framework se usa JasperReport y como metodología de desarrollo se utiliza el Proceso Unificado Ágil (AUP).

Usabilidad: La solución debe contar con una interfaz gráfica atractiva a la vista del usuario. La evaluación debe ser por un trabajador que tenga al menos un conocimiento básico en ergonomía.

Manual de usuario: El sistema debe contar con un manual general que guiará al trabajador a la hora que este interactúe con el sistema, logrando así un mejor uso de la herramienta.

2.3 Modelado de casos de uso

El modelo de casos de uso permite que los desarrolladores del software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas (Rumbaugh 2000).

Un modelo de casos de uso del sistema contiene actores, casos de uso y sus relaciones. A continuación se presenta la descripción de estos y ejemplos de este.

2.3.1 Descripción de los actores del sistema

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad (UML). En el desarrollo del análisis para la propuesta de solución se identificaron los actores del sistema Funcionario y Evaluador. En la tabla 4 se hace una descripción de cada uno de estos actores.

Tabla 4. Descripción de los actores.

Actor	Descripción
Funcionario	Persona encargada de gestionar la estructura de la organización y gestionar el personal de la organización.
Evaluador	Persona encargada de realizar evaluaciones con cada uno de los métodos ergonómicos automatizados y generar reportes.

2.3.2 Diagrama casos de uso del sistema

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto (Ivar Jacobson 2000). A continuación se presenta el diagrama de caso de uso del sistema, el cual está conformado por el evaluador que es el encargado de inicializar los casos de usos correspondientes a la evaluación y los reportes; y el funcionario que es quien inicializa los casos de uso correspondientes a la estructura y el personal de la empresa.

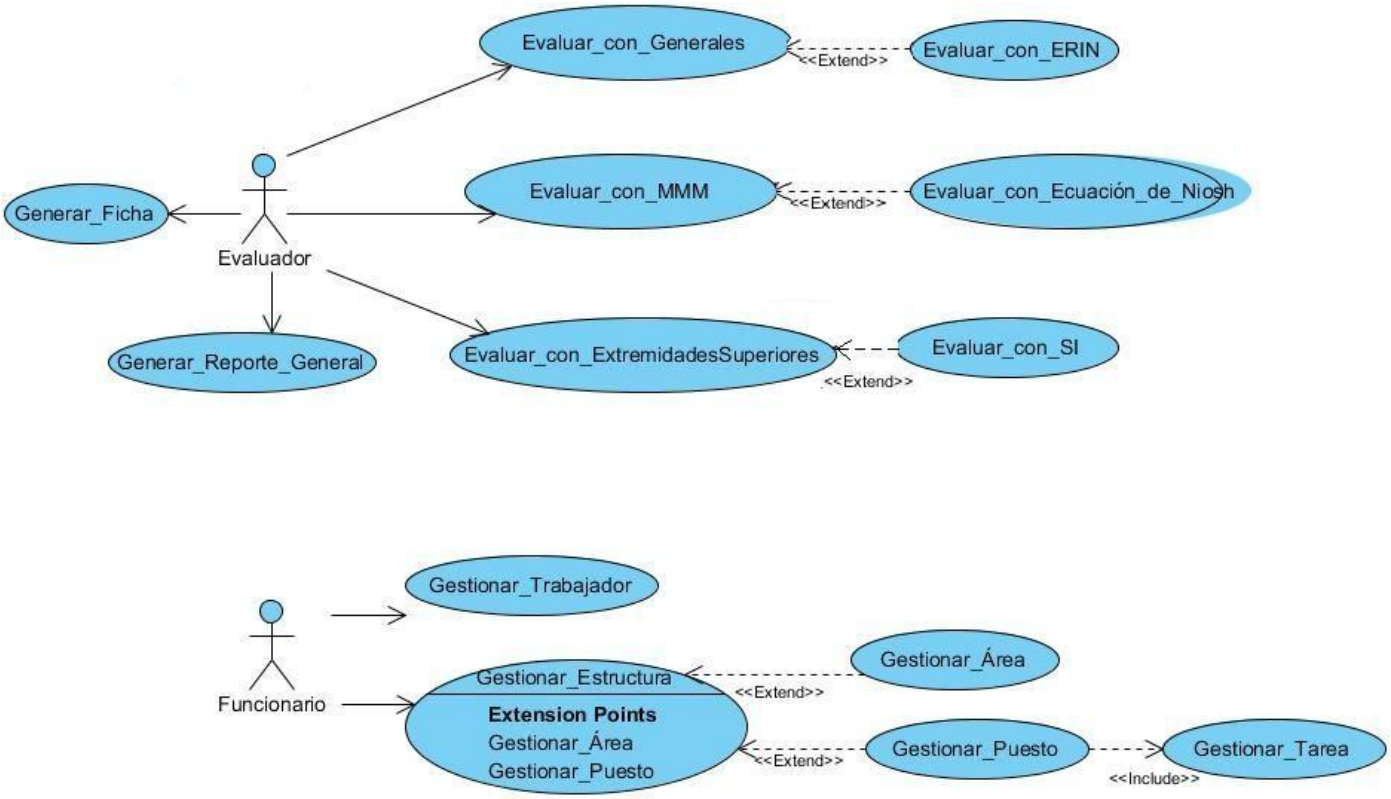


Figura 9. Diagrama de casos de uso.

2.3.3 Descripción de los casos de uso

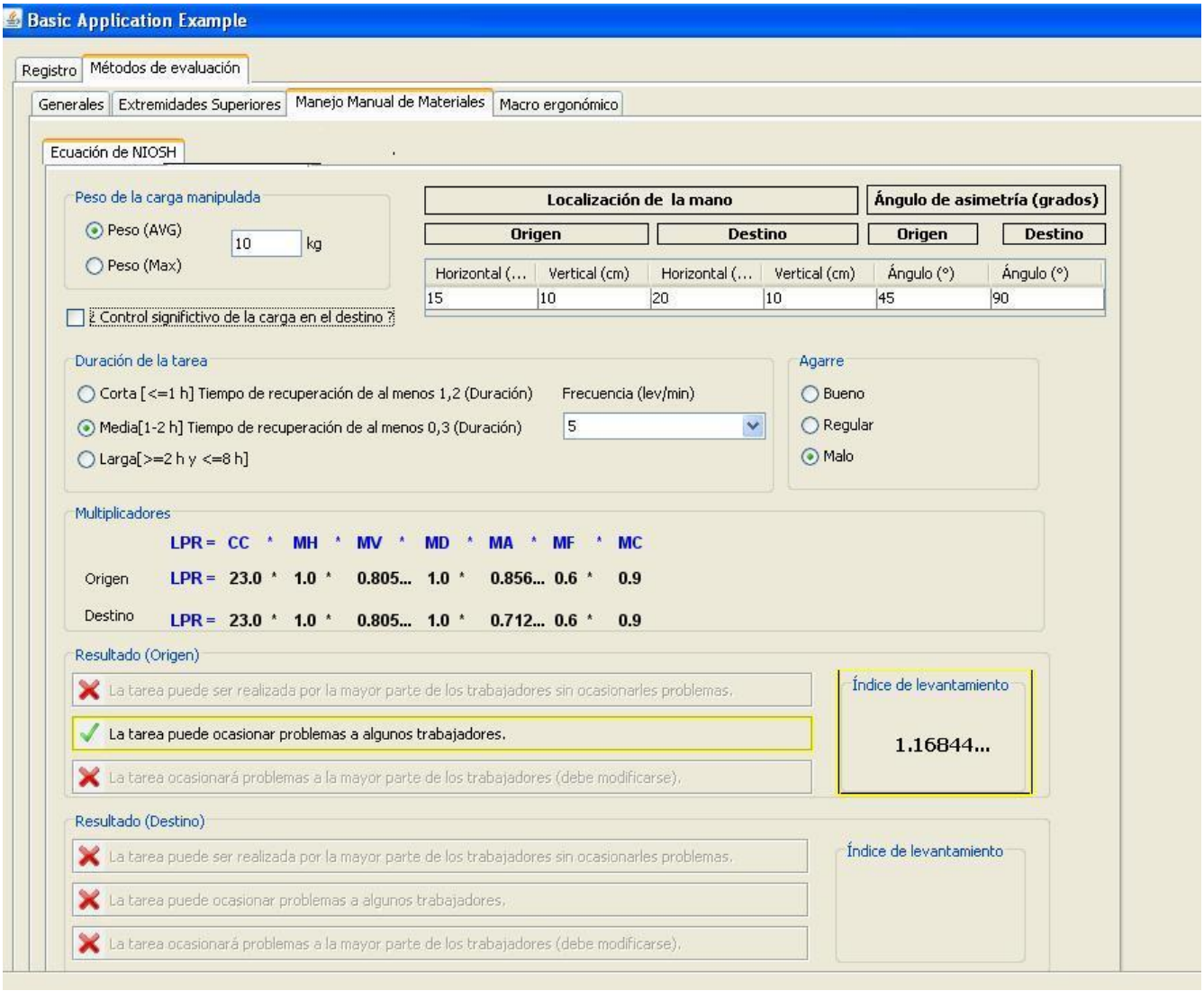
En el presente epígrafe se realizan las descripciones de los casos de uso del sistema. Estos se describen a un nivel más detallado con el fin de obtener una mejor comprensión del funcionamiento del sistema. La descripción de los casos de uso se encuentra en el artefacto: Descripción de los casos de uso. A continuación se muestra un ejemplo de la descripción del caso de uso Evaluar con manejo manual de materiales, ver descripción de casos de uso en Anexo 2.

CU 5. Evaluar con Manejo Manual de Materiales (MMM).

Tabla 5. Descripción del caso de uso Evaluar con MMM.

Caso de uso:	Evaluar con Manejo Manual de Materiales.
Actores:	Evaluador

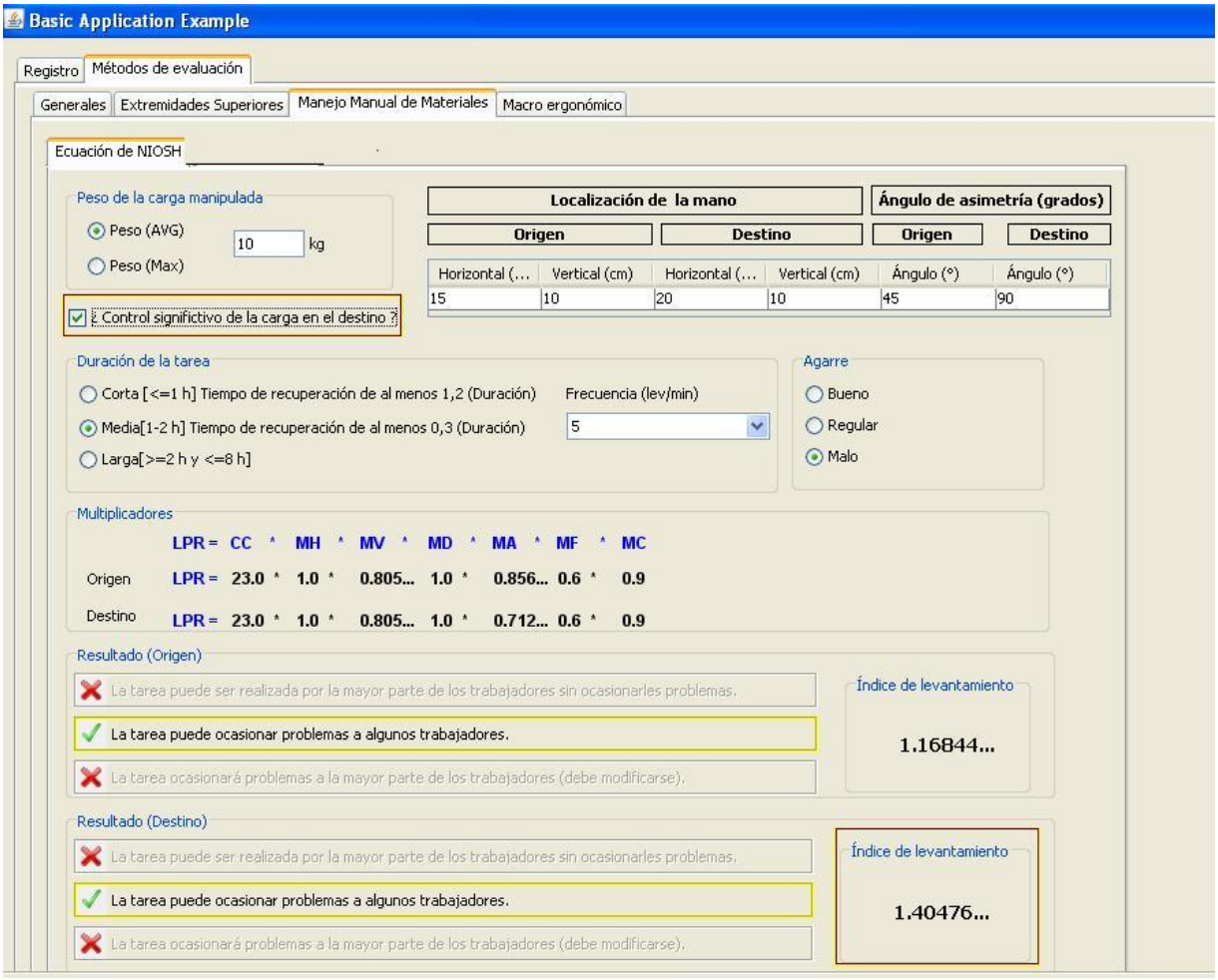
Resumen:	Permitirá al evaluador evaluar un puesto de trabajo con el método Ecuación de Niosh.
Precondiciones:	Debe ser el evaluador.
Flujo normal de eventos	
Sección "Evaluar con Ecuación de Niosh"	
Acción del actor	**Respuesta del sistema**
1. El evaluador debe seleccionar la pestaña de evaluar con Ecuación de Niosh. 3. El evaluador debe seleccionar el peso de la carga manipulada y poner el valor en kg. 4. El evaluador debe colocar los valores de la localización de la mano (origen y destino). 5. El evaluador debe colocar los valores del ángulo de la mano (origen y destino). 6. El evaluador debe seleccionar duración de la tarea. 7. El evaluador debe seleccionar la frecuencia. 8. El evaluador debe seleccionar el agarre.	2. El sistema muestra la información correspondiente al método Ecuación de Niosh. 9. El sistema incorpora la información en su base de datos. 10. El sistema muestra al usuario la información resultante en la parte inferior.
Prototipo de interfaz	



Flujo alterno

Acción del actor	Respuesta del sistema
1. El evaluador debe seleccionar la pestaña de evaluar con Ecuación de Niosh. 3. El evaluador debe seleccionar el peso de la carga manipulada y poner el valor en kg.	2. El sistema muestra la información correspondiente al método Ecuación de Niosh. 10. El sistema incorpora la información en su base de datos.

4. Si la tarea necesita un control significativo de la carga, el evaluador debe seleccionar la opción "¿Control significativo de la carga en el destino?" 5. El evaluador debe colocar los valores de la localización de la mano (origen y destino). 6. El evaluador debe colocar los valores del ángulo de la mano (origen y destino). 7. El evaluador debe seleccionar duración de la tarea. 8. El evaluador debe seleccionar la frecuencia. 9. El evaluador debe seleccionar el agarre.	11. El sistema muestra al usuario la información resultante en la parte inferior.
Prototipo de interfaz	



2.4 Diseño del sistema

El diseño del sistema ayuda a identificar objetivos de los requisitos necesarios para el estudio de la organización y de los usuarios, esto implica la propuesta de desarrollo, la recopilación y especificación de entradas y salidas, donde se detallan los reportes, documentos y prototipos de pantallas, es decir, descripción de la distribución de la interfaz del sistema (Olivares 14 mayo 2012).

2.4.1 Diagrama de secuencia

El diagrama de secuencia muestra la forma en que los objetos se comunican entre sí a través de mensajes manteniendo un orden al transcurrir el tiempo y destacan la organización de los objetos que participan en una interacción (Almazán María Octubre de 2012).

A continuación se muestra el diagrama de secuencia para evaluar con el método Ecuación de Niosh, ver demás diagramas de secuencia desde Anexos 3 a Anexos 20:

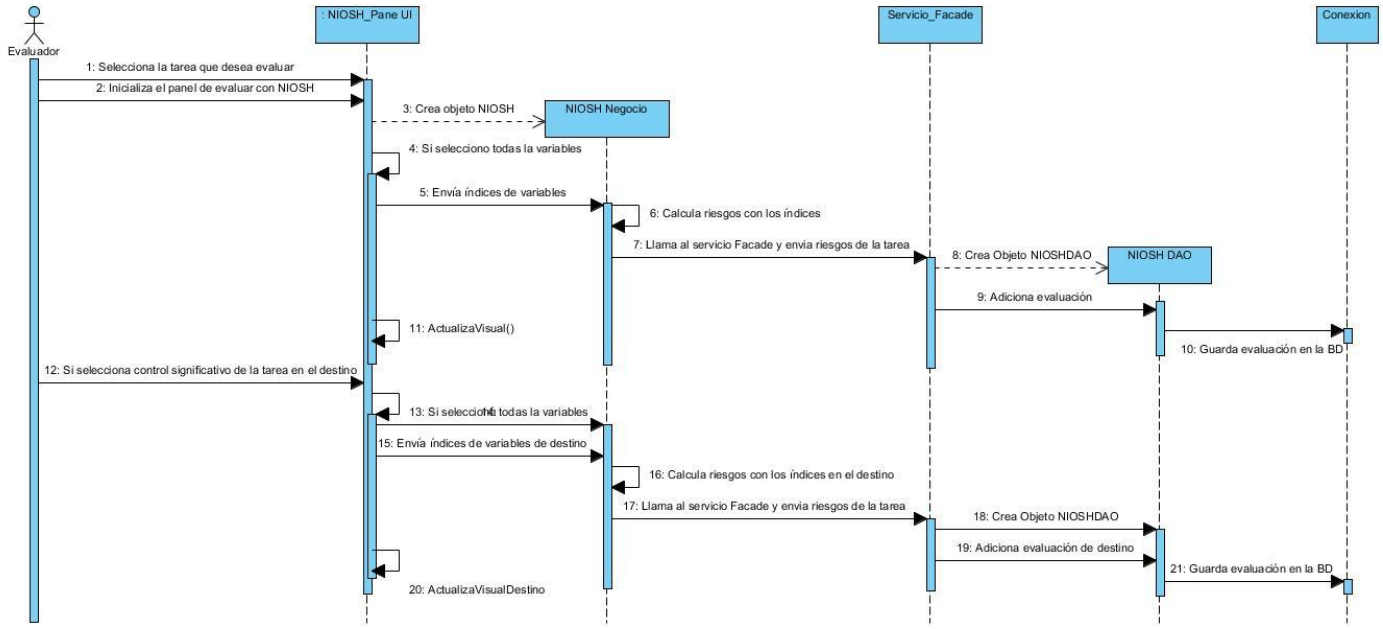


Figura 10. Diagrama de secuencia para evaluar con el método Ecuación de Niosh.

2.4.2 Arquitectura del sistema

Según el documento de la IEEE Std 1471-2000 que plantea: “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución” (IEEE 2000).

Estilo arquitectónico

Los estilos arquitectónicos como un conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer en sistema o subsistema, junto con las restricciones locales o globales de la forma en que se lleva a cabo la composición. Es en gran medida la

interacción entre los componentes, mediados por conectores, lo que confiere a los distintos estilos sus características distintivas (IEEE 2000).

Arquitectura seleccionada

La arquitectura seleccionada por el equipo de desarrollo fue la arquitectura en capas (King, 2005), de esta arquitectura se seleccionó la arquitectura cuatro capas y dos niveles, debido al diseño que posee la aplicación. El término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física.

La arquitectura en cuatro capas y dos niveles va a estar compuesta por (King 2005)**:**

Capa de presentación: Esta va a ser la capa superior en esta arquitectura, que representa la interfaz de usuario y es la encargada de interactuar con este. Esta capa es la encargada de la codificación, el control de las páginas y de las formas de navegación del usuario por las pantallas.

Capa del negocio: Conocida por la capa intermedia, es donde se localiza la lógica del negocio, esta es la responsable de implementar las reglas del negocio, las validaciones y los cálculos, esta capa recibe la petición del usuario a través de la capa de presentación y se encarga de darle respuesta mediante los repositorios de información. En algunos sistemas esta capa tiene su propia representación interna de las entidades del dominio del negocio, en otros utiliza el modelo definido por la capa de persistencia.

Capa de persistencia: No es más que un grupo de clases y de componentes responsables del almacenamiento de los datos, esta incluye necesariamente un modelo de las entidades del dominio del negocio.

Capa de datos: Es la representación real de los datos, y representa además la persistencia del estado del sistema, en esta capa se van a encontrar los datos, o sea, esta capa representa la base de datos que en este caso está desarrollada en SQLite. Cada sistema tiene una infraestructura de clases de utilidades o excepciones, que son usadas por cada capa de la aplicación, estos elementos infraestructurales no forman una capa, puesto a que no obedecen las reglas para la dependencia de la capa intermediaria de una arquitectura en capas.

El siguiente diagrama representa esta arquitectura:

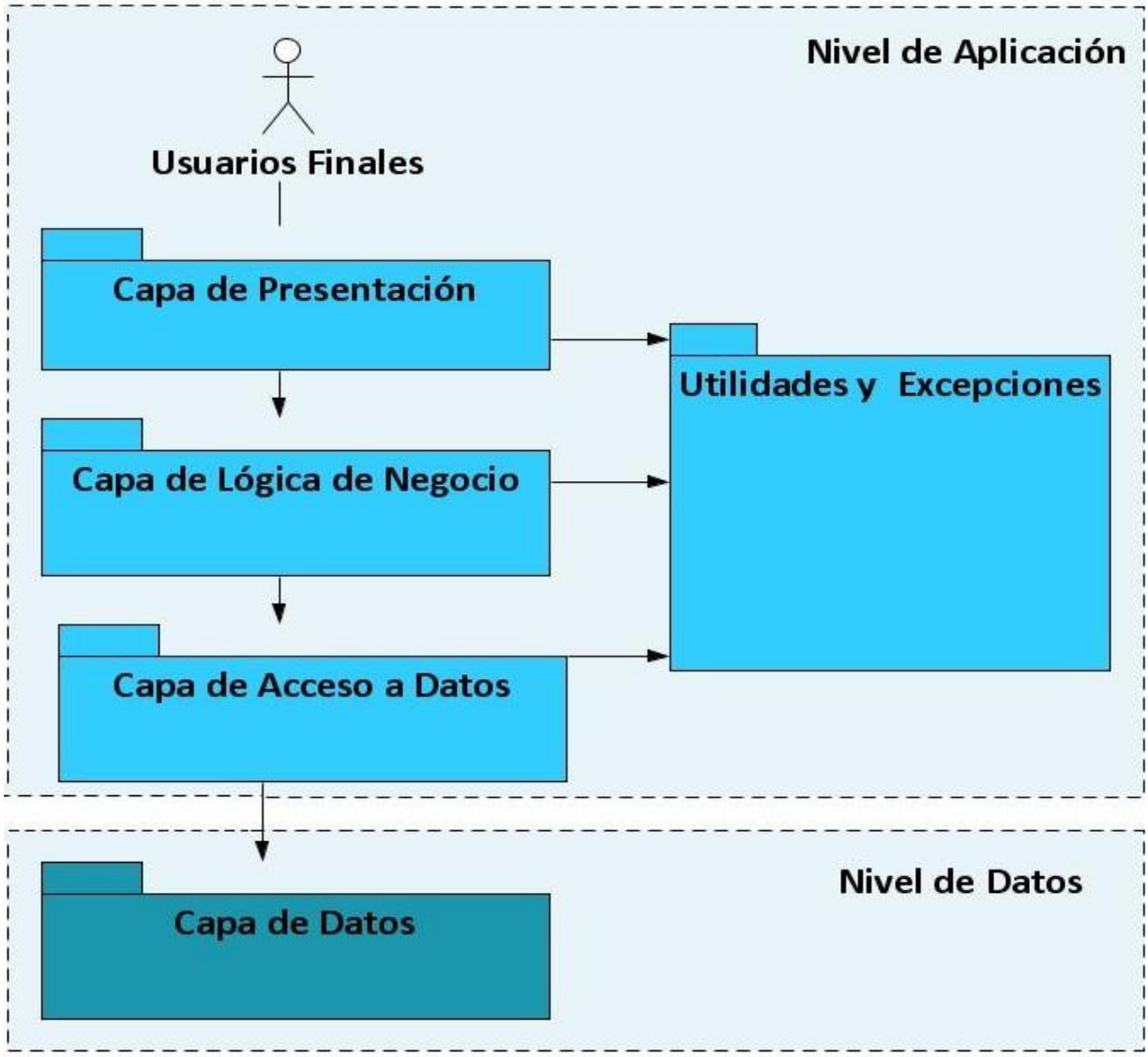


Figura 11. Arquitectura cuatro capas y dos niveles.

Las capas pueden estar formadas por paquetes y subsistemas, el uso de la arquitectura en capas es muy frecuente en nuestros días, porque permite una mayor organización del sistema y para este estilo hay una gran cantidad de patrones que se recomiendan en varias bibliografías. Este estilo se usa mucho en sistemas informáticos complejos porque simplifica la comprensión de los mismos. Los elementos de una

capa pueden interactuar con los demás elementos de esa misma capa, para mantener la flexibilidad, y el estilo nos permita eliminar una capa completa si es necesario en algún momento (King 2005).

2.4.3 Diagrama de clases del diseño

Olivares define el diagrama de clases como el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones (Olivares 2009).

Estos son diagramas de estructura estática que describen la estructura de un sistema mostrando sus clases, atributos y relaciones entre ellos.

En la Figura # 12 se muestra el diagrama de clases agrupados por paquetes a partir de la arquitectura propuesta:

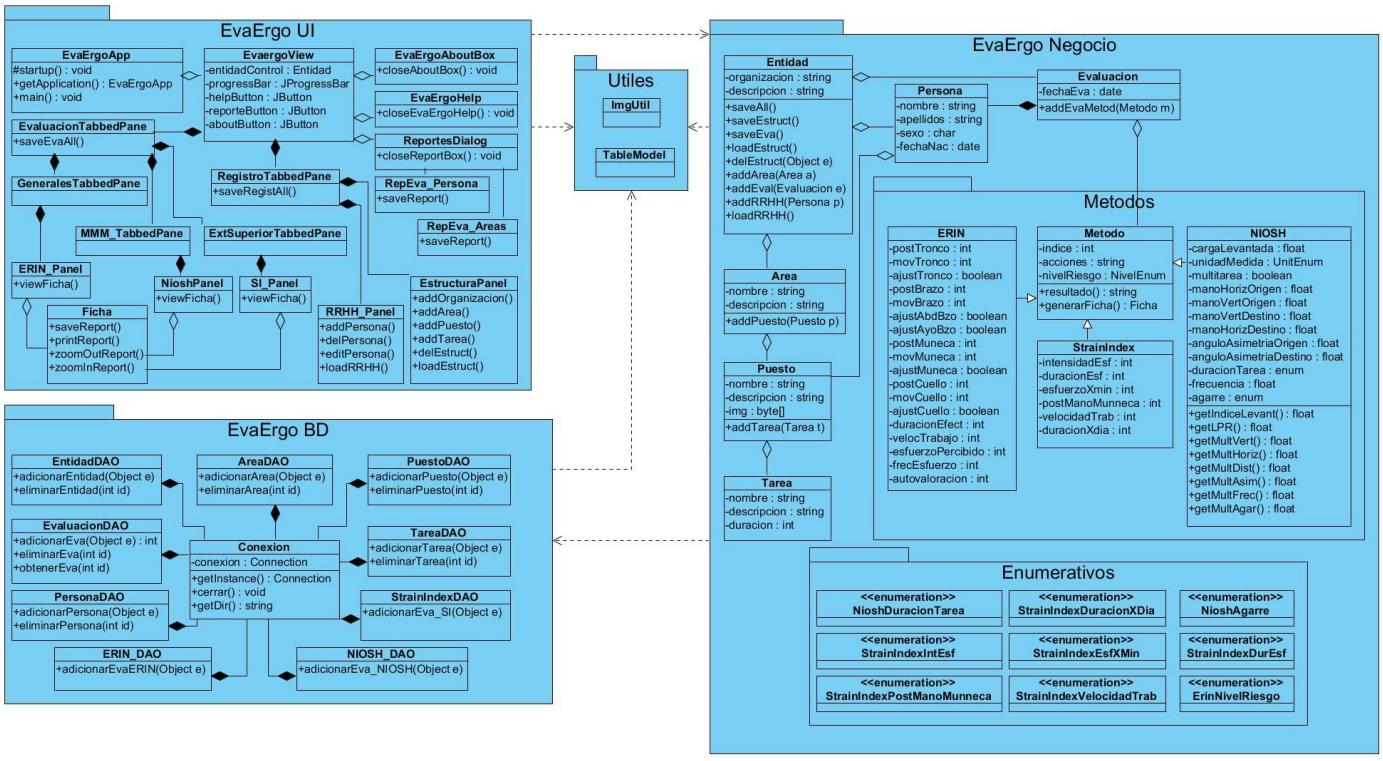


Figura 12. Diagrama de clases.

2.4.4 Patrones de diseño para casos de uso

Son comportamientos que deben existir y ayudar a describir qué es lo que se debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen como deberían ser estructurados y organizados los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso.

A continuación se muestran los patrones que se siguieron para el diseño de casos de usos:

Patrón CRUD: Se utilizó en los casos donde se quiere realizar altas, bajas, cambios y consultas a alguna entidad del sistema, en este caso: Área, Puesto, Tarea y Trabajador. Su nombre es un acrónimo de las palabras en inglés Create, Read, Update, Delete.

El patrón CRUD Completo consiste en un caso de uso para administrar la información (CRUD Información), nos permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja. Este patrón deberá ser usado cuando todas las operaciones contribuyen al mismo valor de negocio y todas son cortas y simples (Larman 1999).

Un ejemplo de su uso se evidencia en la siguiente figura:

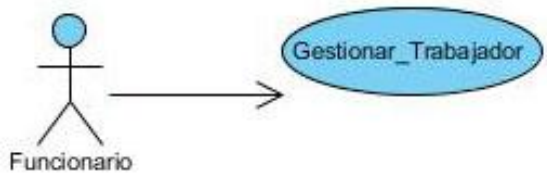


Figura 13. Ejemplo 1. Patrón CRUD Completo.

Patrón Extensión Concreta o Inclusión: Éste patrón está dividido en concreta extensión o concreta inclusión.

Extensión: Consiste en dos casos de uso y una relación extendida entre ellos. Puede ser instalado en sí mismo, así como extendido en el caso de uso base. El referente puede ser concreto o abstracto. Éste

patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo (Larman 1999).

Un ejemplo de su uso se evidencia en la siguiente figura:

Figura 14. Ejemplo 2. Patrón Extensión

Inclusión: Se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo. El caso de uso base puede ser concreto o abstracto (Larman 1999).

Como ejemplo de su aplicación se puede ver la figura que a continuación se muestra:

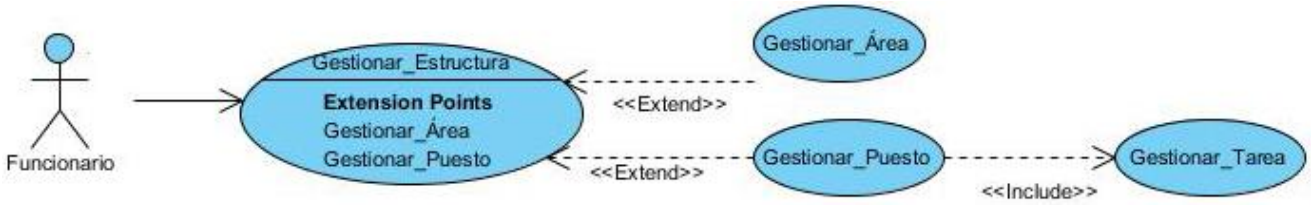


Figura 15. Ejemplo 3. Patrón Inclusión

Patrón de reportes: Este patrón de reportes se ve evidenciado porque el evaluador genera varios reportes: reporte de áreas, fichas técnicas y reportes generales. A continuación un ejemplo de este:

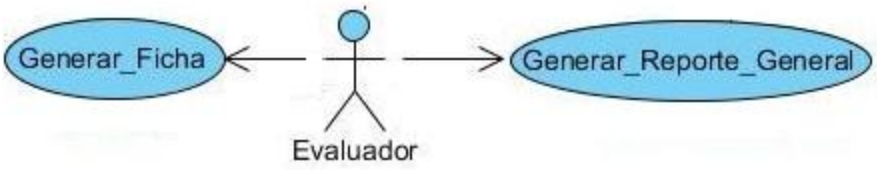


Figura 16. Ejemplo 4. Reportes

2.4.5 Modelo de datos

El modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir (Pressman 2010):

- ✓ Las estructuras de datos: tipo de datos que hay en la base de datos y la forma en que se relacionan.
- ✓ Las restricciones de integridad: conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- ✓ Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base de datos.

A continuación se muestra el modelo de datos que muestra las entidades que modelan las clases persistentes de la aplicación:

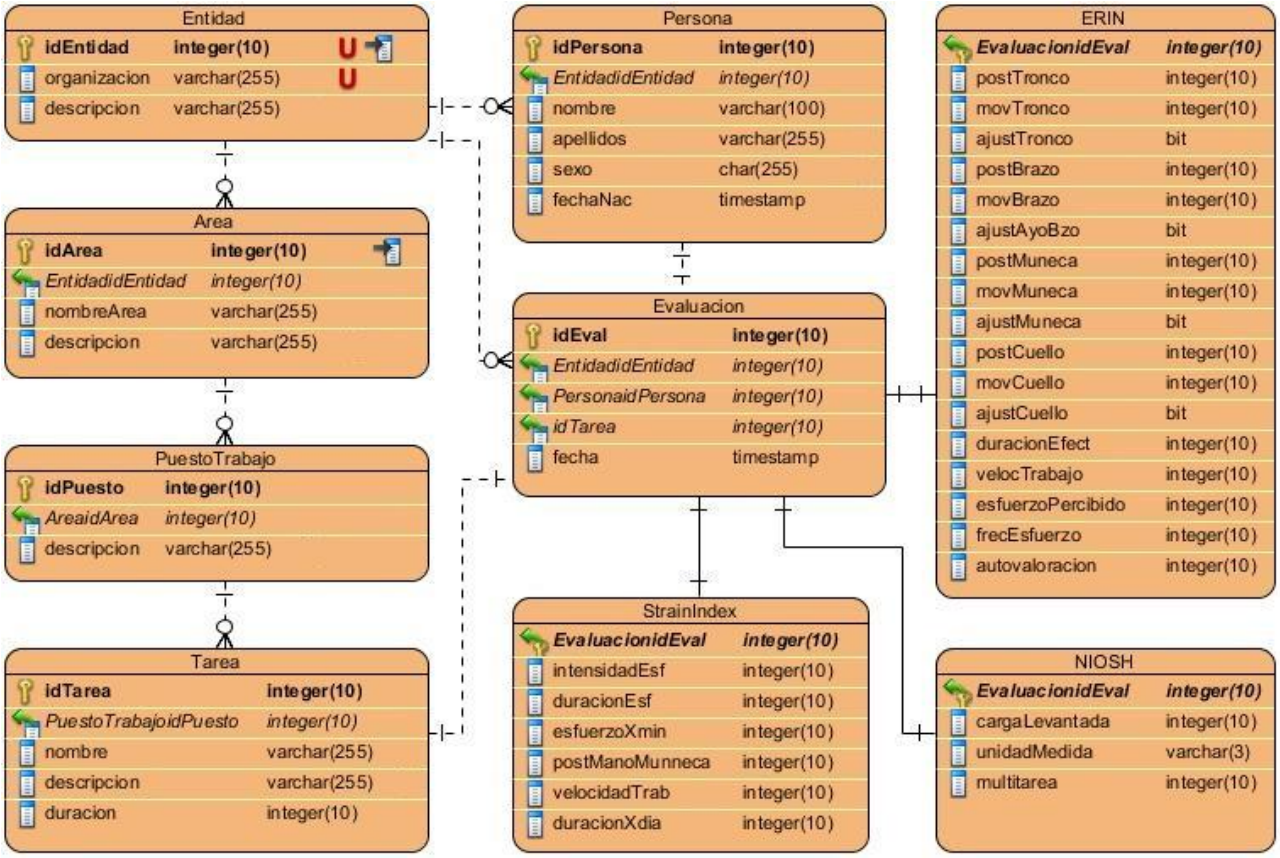


Figura 17. Modelo de datos.

Conclusiones del capítulo

Los artefactos descritos: diagrama de procesos, descripción de requisitos, diagrama de casos de uso y diagrama de secuencias, son necesarios y de vital importancia para la construcción del sistema, ya que posibilitaron definirlo con suficiente detalle para permitir su interpretación y realización física.

El modelo del diseño y modelo de dato permitieron materializar con precisión los requisitos del cliente y sirve como guía para las actividades de implementación, al producir una representación técnica del software que se está desarrollando.

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN Y VALIDACIÓN

En éste capítulo se describirán los estándares de codificación a utilizar para garantizar un buen entendimiento y legibilidad del código y los componentes necesarios para realizar la implementación y despliegue de la solución propuesta. Se presenta una descripción sobre los elementos que se han construido para su desarrollo y cómo interactúan entre sí. Finalmente se muestran los resultados obtenidos en las pruebas mediante métricas de diseño, pruebas unitarias, pruebas funcionales y métodos escogidos para validar la solución.

3.1 Implementación

3.1.1 Estándares de codificación.

Para comenzar la implementación se deben conocer los estándares de codificación. Un estándar de codificación son reglas que permiten entender de manera rápida, fácil y sencilla, el código empleado en el desarrollo del software. Además, garantiza un mantenimiento óptimo de dicho código por parte del programador, de tal manera que otros programadores se les faciliten entender el código (ejemplo: identificar las variables, las funciones o métodos, etc.) (Web).

En particular Java, tiene reglas como:

- ✓ Las clases se inician con mayúsculas.
- ✓ Los métodos se inician con minúsculas.
- ✓ Los atributos inician con minúsculas.

3.1.2 Estructuración del código fuente de la aplicación.

El código fuente de la aplicación está estructurado en un conjunto de paquetes los cuales responden a cada una de las capas del estilo arquitectónico utilizado. Su propósito se describe a continuación:

Tabla 6.Descripción de paquetes.

Nombre del paquete	Propósito
vista	Agrupa cada una de las clases que modelan visualmente los requisitos de la aplicación: *EvaErgoView, EvaErgoAboutBox.*
bd	Contiene la clase *Connection* encargada de establecer una conexión a la

	base de datos para su acceso y manipulación.
negocio	Está estructurado en los siguientes paquetes: ✓ *enumerativos:* Agrupa todos los enumerativos de la aplicación. ✓ *metodos:* Agrupa las clases que modelan la lógica del negocio de cada uno de los métodos de evaluación (*ERIN, NIOSH, StrainIndex y Metodo*) ✓ *estructura*: Agrupa las clases que modelan la lógica de negocio referente a la estructura de la organización.
recurso	En él se encuentran las imágenes, los iconos que se muestran en la aplicación.
util	Destinado a recopilar funcionalidades comunes para la reutilización de código.

Uso de patrones de diseño:

Las clases que están en el paquete de *negocio* (*NIOSH, ERIN, Strain Index*) son implementadas bajo el patrón experto ya que cada una de ellas se especializa en un método de evaluación ergonómica. Estas clases heredan los atributos y funcionalidades de una clase abstracta denominada *Métodos* la cual conceptualiza las operaciones comunes: *resultado, getIndice y es completo*. A continuación se muestra un ejemplo de la clase NIOSH donde son redefinidas estas operaciones.

```
public String resultado() {
float indice = getIndice();
        if (indice <= 1) {
        return "La tarea puede ser realizada por la mayor parte de los trabajadores sin ocasionarles proble
        mas.";
        }else if (indice > 1 && indice < 3) {
        return "La tarea puede ocasionar problemas a algunos trabajadores.";
        } else //if (indice >= 3) { {
        return "La tarea ocasionará problemas a la mayor parte de los trabajadores (debe modificarse).";
        } }
```

```
public float getIndice() {
      if (!esCompleto()) {
         return 0;
      } else {
         return cargaLevantada_CC /  getLPR();       }    }

public boolean esCompleto() {
         if (cargaLevantada_CC == 0 || durTareaEnum == null || frecuencia == -1 || agarreEnum == null) {
         return false;       }
     if(a==0)       {
      if (distanciaHorizontal_Origen_MH == -1 || distanciaHorizontal_Origen_MH == -1) {
     return false;
     }}else{
       if (distanciaVertical_Destino_MD == -1 || distanciaVertical_Origen_MD == -1) {
     return false;    }}
     return true;   }
```

En el paquete *bd* se encuentra la clase *Connection* la cual implementa el patrón Singleton donde garantiza la existencia de una única instancia para la conexión y la creación de un mecanismo de acceso global a dicha instancia. A continuación se muestra dicha clase.

```
private static class ConexionHolder
        {
              private static Connection INSTANCE = new Connection();
         }
        public static java.sql.Connection getInstance()
        {
              return ConexionHolder.INSTANCE.conexion;
        }
```

Para la comunicación entre las capas del negocio y de acceso a datos se utiliza la interfaz Servicio Facade la cual unifica las operaciones de acceso a datos. En este caso se evidencia el uso del patrón Facade quien brinda un solo punto de acceso para la manipulación y acceso a la base de datos.

```
public class Entidad
        {
          //  …
            public ServicioFacade getServicioFacade()
        {
                return servicioFacade;
        }
         //…
    }
```

3.2 Tratamiento de errores

Los lenguajes orientados a objeto han buscado la forma de facilitar la programación de las situaciones de error en un programa. Muchas situaciones pueden generar excepciones (o errores). Una excepción es un evento que ocurre durante la ejecución de un programa que rompe el flujo normal de ejecución. Cuando se habla de excepciones nos referimos a un evento excepcional.

Para el tratamiento de errores en la aplicación se utilizó las excepciones brindadas por el propio lenguaje java. A continuación se muestra el uso de tratamientos de errores en la aplicación.

```
try {
          Class.forName("org.sqlite.JDBC");
            String bddir = Connection.DirectorioActual()
       Conexion=DriverManager.getConnection("jdbc:sqlite:"+"C://Users//Yasser//Desktop//nueva//erindb");
       } catch (SQLException ex) {
            Logger.getLogger(Connection.class.getName()).log(Level.SEVERE, null, ex);
       } catch (ClassNotFoundException ex) {
            Logger.getLogger(Connection.class.getName()).log(Level.SEVERE, null, ex);    }
```

3.3 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes del hardware y el software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software (UML2 2007).

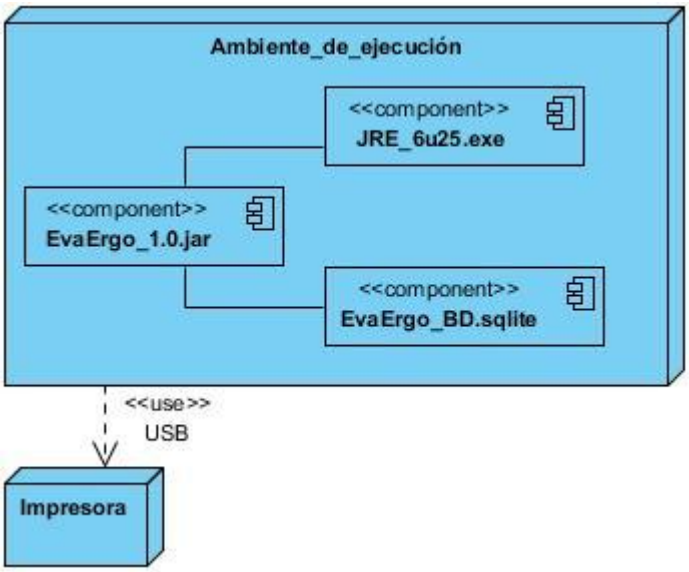


Figura 18. Diagrama de despliegue.

3.4 Diagrama de componentes

Dentro del modelo de implementación se encuentran los diagramas de componentes. El diagrama de componentes, describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (Ambler).

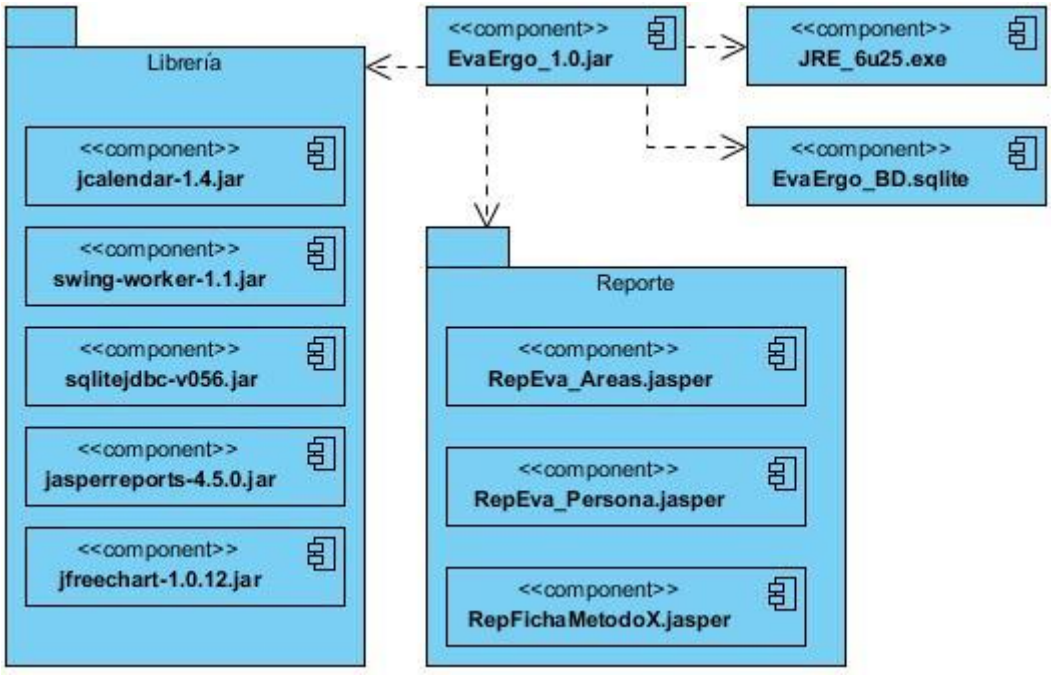


Figura 19. Diagrama de componentes.

3.5 Pruebas de software

La fase de pruebas es una de las más valiosas del ciclo de vida de un software y en ese sentido, deben evaluarse todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación.

Existen varios tipos de prueba de las cuales solo se aplicaron las pruebas unitarias, pruebas funcionales y pruebas de aceptación.

3.5.1 Prueba unitaria

El método que se utilizó para las pruebas unitarias es el de caja blanca o estructural descrita en el capítulo 1 de esta investigación.

Pasos a seguir en la aplicación de la técnica camino básico:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del camino básico.

Los casos de prueba, derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para realizar la prueba es necesario efectuar primeramente el análisis de complejidad del algoritmo sobre el que se va a realizar la prueba, con el propósito de calcular los valores de la complejidad ciclomática.

Paso 1

```
public String resultado(int a) {
    float indice = getIndice(a); (1)

    if (indice <= 1) (2)
    {
        return "La tarea puede ser realizada por la mayor parte de los trabajadores sin ocasionarles problemas."; (3)
    } else if (indice > 1 && indice < 3) (4)
    {
        return "La tarea puede ocasionar problemas a algunos trabajadores."; (5)
    } else
    {
        return "La tarea ocasionará problemas a la mayor parte de los trabajadores (debe modificarse)."; (6)
    }
} (7)
```

Figura 20. Código fuente de la funcionalidad Resultado.

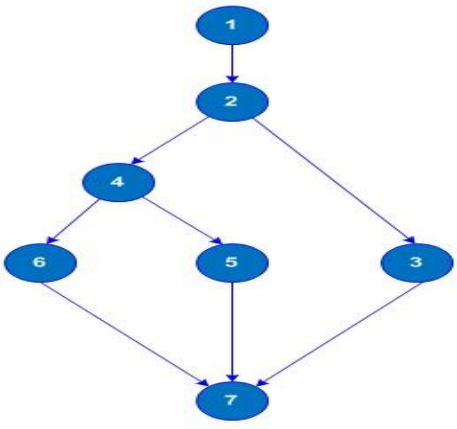


Figura 21. Grafo de flujo asociado a la funcionalidad Resultado.

Paso 2

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

$$V(G) = (A - N) + 2$$
$$V(G) = (8 - 7) + 2$$
$$V(G) = 3$$

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

$$V(G) = P + 1$$
$$V(G) = 2 + 1$$
$$V(G) = 3$$

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

$$V(G) = R$$
$$V(G) = 3$$

Paso 3

Mediante los cálculos realizados anteriormente, basados en las formas en que se puede obtener el valor de la complejidad ciclomática, se puede observar que todos arrojaron como resultado el valor de 3, lo que significa que existen tres posibles caminos por donde puede circular el flujo. Este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. A continuación se muestran los caminos básicos por los que puede recorrer el flujo.

Tabla 7. Caminos Básicos del flujo.

Número	Camino básico
1	1, 2, 3, 7
2	1, 2, 4, 5, 7
3	1, 2, 4, 6, 7

Paso 4

Para cada camino se realiza un caso de prueba, y es preciso cumplir con las siguientes exigencias:

- ✓ Descripción: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o se entre algún dato erróneo.
- ✓ Condición de ejecución*:* se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- ✓ Resultados esperados: se expone el resultado que se espera que devuelva el procedimiento.

Camino básico # 1: 1, 2, 3, 7.

Descripción: Se desea realizar la evaluación con el método Ecuación de Niosh.

Condición de ejecución: Para la ejecución del algoritmo es necesario que la variable *indice* tenga valor menor o igual que 1.

Entrada: La variable *indice* tiene valor 1.

Resultados esperados: Teniendo en cuenta los datos introducidos se espera que el resultado sea “La tarea puede ser realizada por la mayor parte de los trabajadores sin ocasionar problemas”.

Camino básico # 2: 1, 2, 4, 5, 7.

Descripción: Se desea realizar la evaluación con el método Ecuación de Niosh.

Condición de ejecución: Para la ejecución del algoritmo es necesario que la variable *indice* tenga valor mayor que 1 y menor que 3.

Entrada: La variable *indice* tiene valor 1,55.

Resultados esperados: Teniendo en cuenta los datos introducidos se espera que el resultado sea “La tarea puede ocasionar problemas a algunos trabajadores”.

Camino básico # 3: 1, 2, 4, 6, 7.

Descripción: Se desea realizar la evaluación con el método Ecuación de Niosh.

Condición de ejecución: Para la ejecución del algoritmo es necesario que la variable *indice* tenga valor mayor o igual que 3.

Entrada: La variable *indice* tiene valor 3,60.

Resultados esperados: Teniendo en cuenta los datos introducidos se espera que el resultado sea “La tarea ocasionará problemas a la mayor parte de los trabajadores (debe modificarse)”.

3.5.2 Prueba funcional

El método que se utilizó para la prueba funcional fue el de caja negra descrita en el capítulo 1.

Pasos a seguir en la aplicación de la técnica partición equivalente:

1. Se define un caso de prueba por cada caso de uso.
2. Se realizan escenarios en cada caso de prueba con las variables pertenecientes a cada uno de estos casos de uso.
3. Se introducen datos válidos e inválidos, de manera tal que cuando se introduzcan los datos válidos se ejecute el flujo correcto y cuando se introduzcan los datos inválidos se muestren mensajes de error.

A continuación se muestra un ejemplo del caso de prueba para el método Ecuación de Niosh.

Tabla 8. Descripción de variables.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
V1	Peso de la carga manipulada	Selección y valor numérico	No	Se debe seleccionar el peso si es Max o AVG y colocar el valor de este.
V2	Localización de la mano (origen y destino)	Campo numérico	No	El campo debe admitir solo números.
V3	Ángulo de la mano (origen y destino)	Campo numérico	No	El campo debe admitir solo números.
V4	Duración de la tarea	Selección	No	Debe seleccionar la duración.
V5	Frecuencia	Lista de selección	No	Debe seleccionar la frecuencia.
V6	Agarre	Selección	No	Debe seleccionar el agarre.

Caso de prueba

Tabla 9. Caso de prueba escenario ¨Tarea sin control significativo de la carga en el destino¨.

Escenario	Descripción	Var1	Var 2	Var3	Var4
EC 1.1 Escenario Tarea sin control significativo de la carga en el destino.	El evaluador no necesita seleccionar el control de la carga destino ya que esta tarea no requiere de ese valor.	V Peso (AVG) valor: 10kg	V Horizontal: 5cm Vertical: 10cm	V Ángulo origen:45 Ángulo destino: 90	V Media [1-2h]

Var5	Var6	Var7	Respuesta del sistema	Flujo central
V 5	V Malo	NA	La tarea puede ocasionar problemas a algunos de los trabajadores.	1. Seleccione la pestaña MMM 2. Seleccione cada valor en correspondencia con el puesto a evaluar. 3. En la parte inferior obtengo el Resultado (Origen) y el Índice de levantamiento de 1.1684.

[Las celdas de la tabla contienen **V**, **I**, o **N/A**. **V** indica válido, **I** indica inválido y **N/A** que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

3.6 Métricas de diseño

El glosario de estándares del IEEE define métrica como una “medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado”. Las métricas se centran en cuantificar tanto la funcionalidad, como la complejidad y eficiencia inmersa en el desarrollo de software, inclinando sus objetivos a mejorar la comprensión de la calidad del producto, estimar efectividad del proceso y mejorar la calidad del trabajo.

3.6.1 Tamaño operacional de clase (TOC)

Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado (ECURED 23 de mayo de 2013).

Tabla 10. Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento en la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 11. Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=promedio
	Media	Entre promedio y 2*promedio
	Alta	>2*promedio
Complejidad de implementación	Baja	<=promedio
	Media	Entre promedio y 2*promedio
	Alta	>2*promedio
Reutilización	Baja	>2*promedio
	Media	Entre promedio y 2*promedio
	Alta	<= promedio

Resultados obtenidos al aplicar la métrica TOC

Haciendo una interpretación de los resultados arrojados por la métrica TOC, se puede decir que son positivos y se incluyen dentro del rango de calidad aceptable. Se puede observar que los atributos responsabilidad y complejidad para la mayoría de las clases están dentro de una categoría Baja para un 64% del total, mientras que el atributo reutilización está en la categoría Alta con un 64%. Por tanto se

puede concluir que la mayoría de las clases son reutilizables, no tienen grandes responsabilidades y el sistema no tendrá una compleja implementación.

De acuerdo a los resultados obtenidos se construyeron las siguientes gráficas para un mejor entendimiento de los mismos:

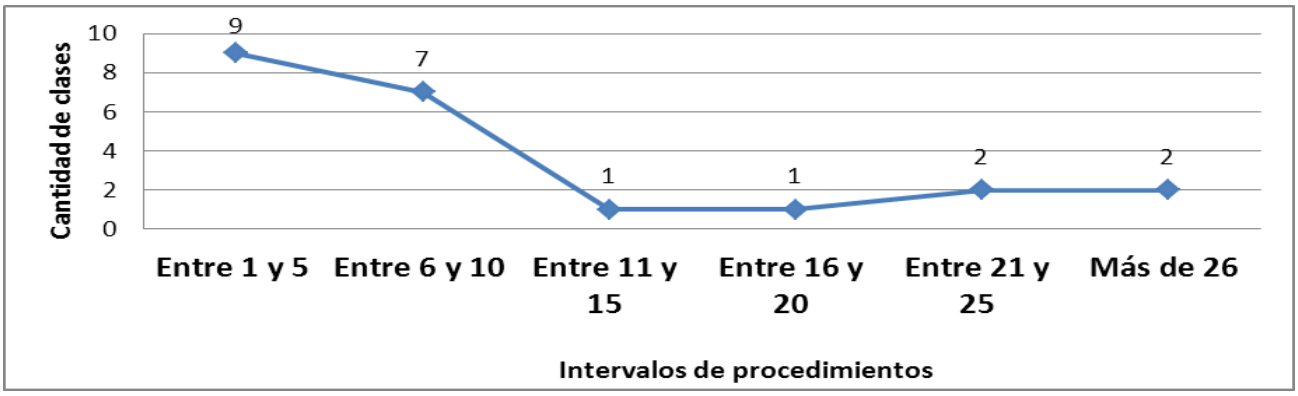


Figura 22. Cantidad de clases por intervalos de procedimientos.

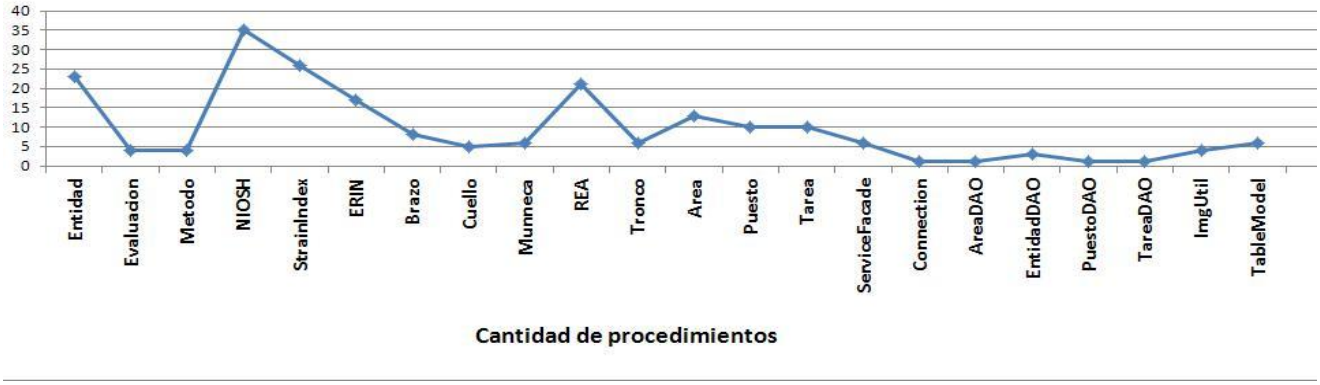


Figura 23. Cantidad de procedimientos por cada clase.

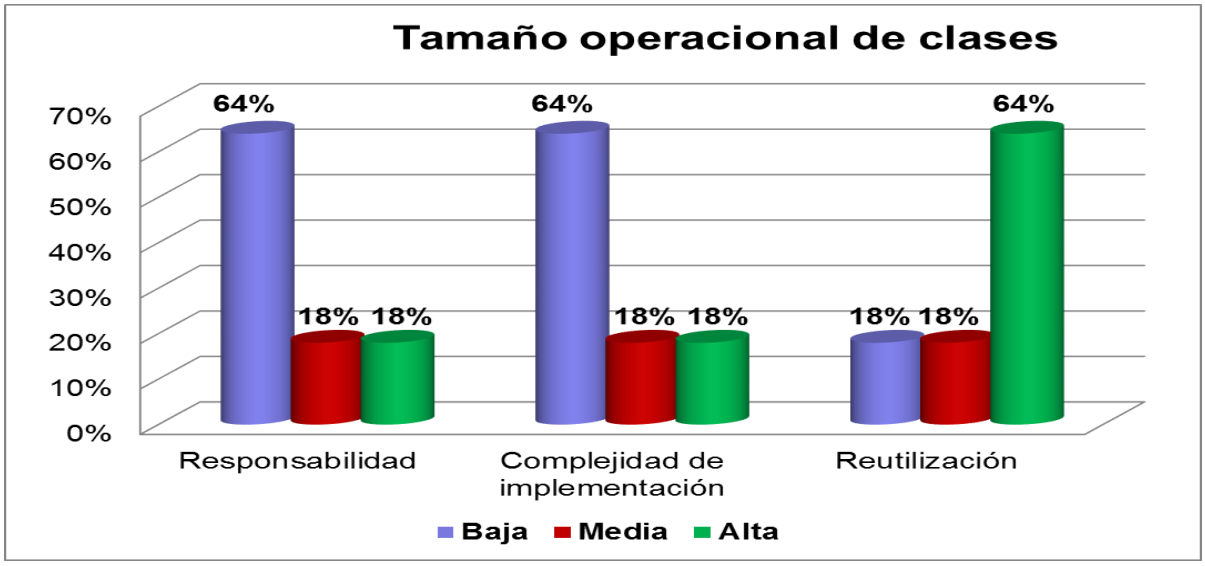


Figura 24. Resultados obtenidos en la evaluación con los atributos de calidad, en la métrica TOC.

3.6.2 Relaciones entre clases (RC)

Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado (ECURED 23 de mayo de 2013).

Tabla 12. Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento en la complejidad de mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución del grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 13. Criterios de evaluación de la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<=promedio
	Media	Entre promedio y 2*promedio
	Alta	>2*promedio
Reutilización	Baja	>2*promedio
	Media	Entre promedio y 2*promedio
	Alta	<=promedio
Cantidad de pruebas	Baja	<=promedio
	Media	Entre promedio y 2*promedio
	Alta	>2*promedio

Resultados obtenidos al aplicar la métrica RC

Al analizar los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que las clases del diseño poseen un bajo acoplamiento ya que para este atributo las categorías de Ninguno y Bajo sumaron un 73% del total. Los atributos complejidad de mantenimiento y cantidad de pruebas están dentro de la categoría Baja con un 73%, lo cual demuestra que no se necesita mucho esfuerzo para corregir o realizar cambios y las pruebas se podrán realizar de manera factible. La mayoría de las clases son reutilizables ya que este atributo posee una categoría alta para un 64% del total.

De acuerdo a los resultados mencionados anteriormente se construyeron las siguientes gráficas para un mejor entendimiento de los mismos:

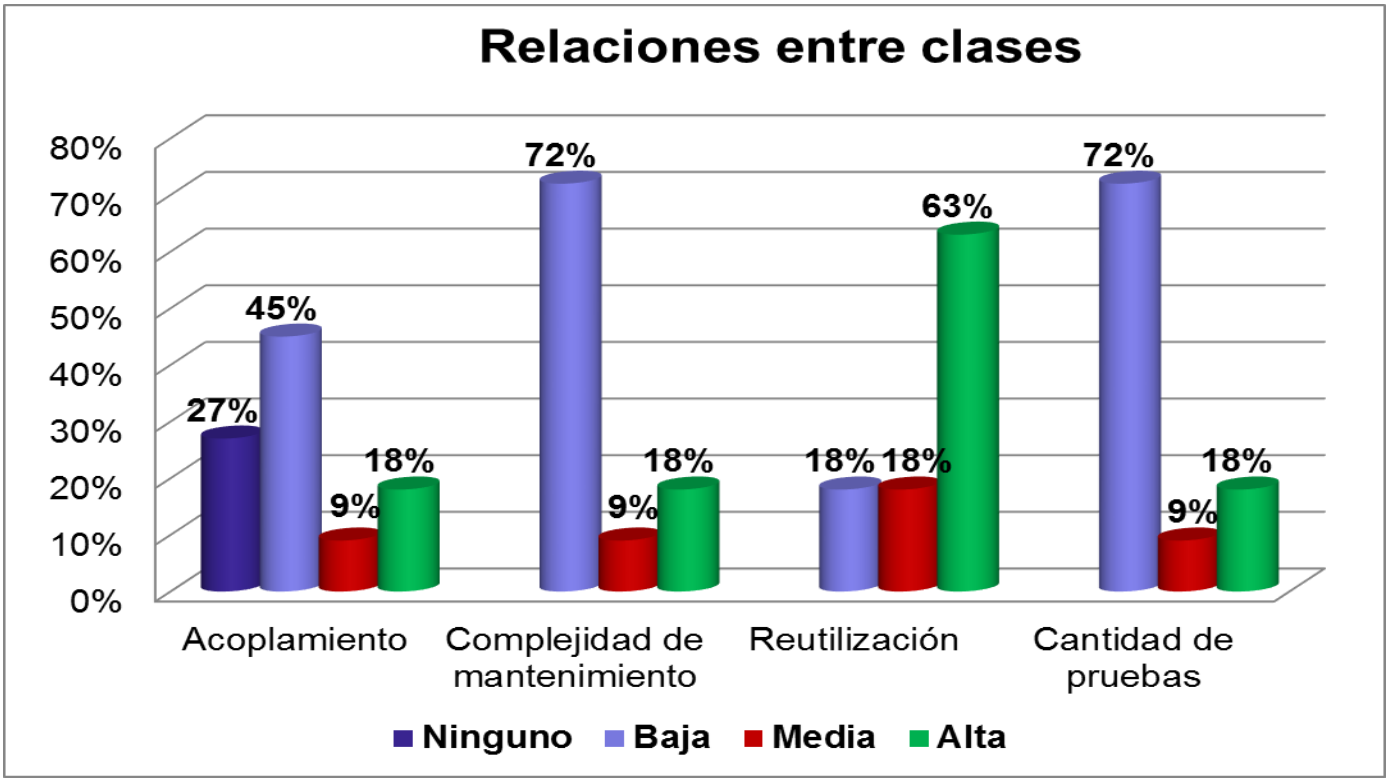


Figura 25. Resultados obtenidos en la evaluación con los atributos de calidad, en la métrica RC.

3.7 Validación de las variables del objetivo

Éste trabajo de diploma tiene como objetivo principal desarrollar una herramienta informática para la evaluación ergonómica de puestos de trabajos que facilite el proceso de evaluación en un breve período de tiempo y que tribute a la toma de decisiones.

Con el cumplimiento de este objetivo se puedo observar que la problemática que llevó a cabo éste trabajo de diploma fue solucionada, por haber cumplido con cada una de las variables que dieron paso a éste problema. El evaluador puede contar solo con un conocimiento previo en ergonomía debido a que la herramienta consta con un manual de usuario bastante explicito; la herramienta contribuye al análisis integral de los puestos de trabajo; guarda un historial de las evaluaciones y genera un reporte general agrupando las evaluaciones por área y puestos de trabajo, facilitando así la toma de decisiones en la organización.

Conclusiones del capítulo

La implementación de la solución se realizó bajo buenas prácticas de programación siguiendo una estandarización y el uso de patrones de diseño.

El modelo de despliegue y el diagrama de componentes permitieron obtener una visión general de la estructura que se requiere para la posterior fase de transición cuando el producto sea implantado en el entorno real de la organización.

Las pruebas de software permitieron detectar y corregir los errores no identificados durante la implementación, posibilitando cumplir con las especificaciones requeridas.

CONCLUSIONES

1. El estudio realizado sobre las diferentes tecnologías de desarrollo permitió seleccionar las herramientas y metodologías más adecuadas a utilizar.
2. Con el diseño e implementación de la herramienta para la evaluación ergonómica de puestos de trabajo se resolvieron las insuficiencias planteadas en el problema, ya que:
   - ✓ Quién efectúe la evaluación no tendrá que tener un conocimiento total de ergonomía.
   - ✓ La herramienta contribuye al análisis integral de los puestos de trabajo.
   - ✓ Genera un reporte general agrupando las evaluaciones por área y puestos de trabajo, facilitando así la toma de decisiones en la organización.

3. La aplicación de las pruebas tanto al código como a las interfaces permitió corregir las no conformidades de manera satisfactoria arrojando los resultados esperados por el equipo de desarrollo.

RECOMENDACIONES

- ✓ Se recomienda seguir enriqueciendo la herramienta con métodos ergonómicos por cada clasificación, para garantizar aún más una evolución integral de los puestos de trabajo.
- ✓ Se recomienda integrar a la aplicación la clasificación Macro-Ergonómica con el método Modelo de madurez para evaluar el estado actual de una organización, la madurez para implementar programas ergonómicos y lograr una cultura ergonómica, siendo este último el estado deseado.
- ✓ Realizar una lista de chequeo donde, una vez llenada, recomiende los métodos más óptimos para realizar la evaluación.
- ✓ Se recomienda el uso de esta herramienta en las organizaciones para garantizar la calidad de vida laboral de cada trabajador.

BIBLIOGRAFÍA

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2002). Agile software development methods Review and analysis.

Almazán María, B. (Octubre de 2012). Herramientas para la Interoperabilidad y Normalización de datos en RI.

Ambler, S. W. UML 2 Component Diagram.

Association(IEA), I. E. (2000-2003).

Castillo Rosal, A. P. (2012). Contribución al mejoramiento de la calidad de vida laboral a partir de la gestión ergonómica en los puestos de trabajos, mediante el análisis de los procesos, Observatorio de la Economía Latinoamericana.

Domínguez-Dorado, M. (Noviembre, 2005). Todo Programación.

ECURED (23 de mayo de 2013). Métrica de diseño.

Engels, F. (1961). Dialéctica de la Naturaleza.

Franklin Rivero Duharte, J. C. G. C., Yadira Benavides Zaila (2005-2006). Reportes con JasperReport, IReport y JFreeChart en Aplicaciones Empresariales en Java.

Gacita (2003).

Gamma, H., Jonhnson, and V. A. Wesley (1995). Design patterns, elements of reusable objectoriented software.

Gerardo Antonio Cabero, D. M. Sqlite: Rápido, ágil,liviano y robusto.

Giandini, P., y Pons (2010).

Group, O. M. (September 2001). "Unified Modeling Language Specification."

Guevara, J. M. L. d. Fundamentos de la Programación en Java. Editorial EME, ISBN 978-84-96285-36-2.

IEEE (2000).

INTECO, L. N. d. C. d. S. d. (2009, Marzo). Ingeniería del software: metodologías y ciclos de vida.

Ivar Jacobson, G. B., James Rumbaugh (2000). El proceso unificado de desarrollo de software.

Jacaboson, I., Booch, G., Rumbaugh J. (2000). El Proceso Unificado de Desarrollo de Software, Addison Wesley

King (2005). Iberneig in Action.

Larman, C. (1999). UML y PATRONES, Introducción al análisis y diseño orientado a objetos.

Margarita Oncins de Frutos, T. M. B. Nuevas formas de organizar el trabajo: la organización que aprende.

McAtamney, S. H. a. L. (2000). Rapid entire body assessment (REBA).

MOORE, J. S. Y. G., A. (1995). The Strain Index: A proposed method to analyze jobs for risk of distal upper extremity disorders. American Industrial Hygiene Association Journal.

Mora, F. (2002). DCCIA.

NIOSH (1997).

Olivares (2009).

Olivares, F. E. P. (14 mayo 2012). diseño UML: diagrama de clases.

personal, C. (26/1/2013).

Pressman, R. S. (2010). Ingeniería de Software. Un enfoque práctico.

Roberth G. Figueroa, C. J. S., Armando A. Cabrera METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.

1 Ruíz, M. I. Y. R. (2011). ERIN: Método práctico para evaluar la exposición a factores de riesgo de
2 desórdenes músculo-esqueléticos.
3 Rumbaugh, J. (2000). El Lenguaje Unificado de Modelado. Manual de Referencia. Madrid.
4 Sommerville, I. (2005). Ingeniería del Software.
5 Takala, P. (2010). Systematic evaluation of observational methods assessing biomechanical exposurses at
6 work.
7 UML2, T. d. (2007). Diagrama de Despliegue.
8 UML, O. OMG Unified Modeling Language (OMG UML).
9 Web, P. from http://www.aves.edu.co/ovaunicor/recursos/view/265.
10 web, P. "netbeans." from http://netbeans.org/community/releases/roadmap.html.